Universidad Nacional de Ingeniería (UNI)

Facultad de Ciencias

Escuela Profesional de Ciencia de la Computación

Plan Curricular 2026

– Reporte Final–

Última modificación: 2026

Este documento está basado en la propuesta internacional de ACM/IEEE-CS *Computing Curricula*, que a su vez sirvió como base de la versión en Español creada por la Sociedad Peruana de Computación (SPC)

ISBN:

Todos los derechos reservados. Ernesto Cuadros-Vargas, Universidad Nacional de Ingeniería (UNI).

Equipo de Trabajo

Dr. Ciro Nuñez Iturri
Director (e) de la Escuela de Ciencia de la Computación
email: *ciro.nunez.i@uni.edu.pe*

Dr. Carlos Javier Solano Salinas
Director del Departamento de Ciencia de la Computación
e-mail: *jsolano@uni.edu.pe*

Dr. José Luis Segovia-Juárez
PhD en Ciencia de la Computación. Wayne State University (Detroit, USA) (2001)
Profesor del Departamento de Ciencia de la Computacion. UNI
email: *jsegoviaj@uni.edu.pe*
http://www.tecnociencia9.com/jlsegovia/es

Dr. Marcos Antonio Alania Vicente
PhD en Ciencias Universidad de Amberes. Bélgica (2017)
Profesor del Departamento de Ciencia de la Computacion. UNI
Desarrollador de algoritmos de IA para procesos complejos interdisiplinarios
email: *alania.vicente@gmail.com*

MSc. Eduardo Yauri Lozano
MSc. en Ciencia de la Computación. UNI (2024)
Profesor del Departamento de Ciencia de la Computacion. UNI
email: *eduardo.yauri@uni.edu.pe*

MSc. Ronald Ricardo Martinez Chunga
MSc. Ciencias Ing Electrónica (2021)
Profesor del Departamento de Ciencia de la Computacion. UNI
email: *rmartinezch@uni.edu.pe*
https://www.linkedin.com/in/rmartinezchunga/

Cesar Martin Cruz-Salazar
Mag(c) en Automática e Instrumentación, UNI
Profesor de Ciencia de la Computación, UNI, Lima-Perú
email: *ccruz@uni.edu.pe*

Prof. Miguel Arrunategui Angulo
Docente del Departamento de Ciencia de la Computación
email: *garrunateguia@uni.edu.pe*

Mag. Yuri Ccoicca
Departmento de Ciencia de la Computación, UNI, Lima-Perú
email: *yccoicca@uni.edu.pe*

M.Sc. Raul Huarote Zegarra
Mag en Ciencia de la Computación, UNT (2014)
Profesor del Departamento de Ciencia de la Computación, UNI, Lima-Perú
email: *rhuarotez@uni.edu.pe*

M.Sc. Carlos Alberto Bazán Cabanillas
Magister en Ciencias, Mención matemáticas, UNI 1980
Licenciado en Matemáticas, UNI 1977
email: *cbazanc@uni.edu.pe*

Dr. Jose Alfredo Herrera-Quispe
Doctor en Ciencia de la Computación, UNSA (2013)
Docente del Departamento de Ciencia de la Computación

Mag. Sidney Weisser Valer Quispe
Maestro en Ciencia con mención en Telemática, FIEE-UNI (2026)
Docente del Departamento de Ciencia de la Computación
email: *svalerq@uni.edu.pe*

Mag. Johnny Quispe Obregón
Docente del Departamento de Ciencia de la Computación
email: *johnny.quispe.o@uni.edu.pe*

Dr. Ernesto Cuadros-Vargas
Dr. en Ciencia de la Computación, Universidad de Sao Paulo, Brasil (2004)
Orador distinguido para la *Association of Computing Machinery* (ACM) (2025-2027)
Miembro del Directorio de Gobernadores de la Sociedad de Computación del IEEE (2020-2023)
Miembro del *Steering Committee ACM/IEEE-CS Computing Curricula 2020 (CS2020)*
Miembro del *Steering Committee ACM/IEEE-CS Computing Curricula/Computer Science (CS2013)*
Secretario Ejecutivo del Centro Latinoamericano de Estudios en Informática (CLEI) 2008-2016
email: *ecuadros@spc.org.pe*

Resumen ejecutivo

Este documento representa el informe final de la nueva malla curricular 2026 del Escuela Profesional de Ciencia de la Computación de la Universidad Nacional de Ingeniería (UNI) (*https://www.uni.edu.pe*) en la ciudad de Lima-Perú.

Este documento presenta el diseño curricular de la carrera de Ciencia de la Computación bajo los estándares internacionales del *Computing Curricula* CC2020, marco de referencia desarrollado conjuntamente por la *Association for Computing Machinery* (ACM) y la *IEEE Computer Society* (IEEE-CS). El CC2020, disponible en http://www.acm.org/education, sintetiza tres décadas de evolución disciplinar (desde CS2001 hasta CC2020) y responde a las demandas actuales de la cuarta revolución industrial.

La propuesta adapta este referente global a nuestro contexto regional, incorporando:

- Requerimientos del mercado laboral local en tecnología
- Fortalezas institucionales en investigación aplicada
- Tendencias emergentes en educación computacional

La computación hoy en día presenta los siguientes perfiles de formación profesional para pregrado:

- Ingeniería de Computación (*Computer Engineering* – CE) [Soldan et al., 2016],
- Ciencia de la Computación (*Computer Science* – CS) [ACM/IEEE-CS/AAAI Joint Task Force, 2023],
- Sistemas de Información (*Information Systems* – IS) [The Joint ACM/AIS IS2020 Task Force, 2020],
- Ingeniería de Software (*Software Engineering* – SE), [LeBlanc et al., 2015]
- Tecnología de la Información (*Information Technology* – IT) [ACM and IEEE-CS, 2017].
- Ciberseguridad (*Cybersecurity* – CY) [ACM/IEEE-CS Joint Task Force on Cybersecurity Education, 2017].
- Ciencia de Datos (*Data Science* – DS) [ACM Data Science Task Force, 2021].

El modelo pedagógico se sustenta en tres ejes transformadores:

- **Excelencia técnica**: Dominio de fundamentos computacionales y especialización progresiva
- **Innovación disruptiva**: Metodologías ágiles, pensamiento computacional y creación de startups
- **Responsabilidad social**: Ética profesional, sostenibilidad tecnológica e impacto comunitario

Esta malla producirá profesionales con:

- Capacidad para resolver problemas complejos en entornos globales
- Competencias para liderar la transformación digital
- Sensibilidad hacia los desafíos éticos de la inteligencia artificial, privacidad y gobernanza tecnológica

Esta propuesta incorpora mecanismos de actualización continua, asegurando relevancia frente a la acelerada obsolescencia tecnológica y alineamiento con los estándares de acreditación internacional.

Índice general

Agradecimientos xxi

Abreviaturas xxiii

1. Introducción 1
1.1. Definiciones básicas . . . 1
1.2. Campo y mercado ocupacional . . . 3
1.3. Importancia de la carrera en la sociedad . . . 4
1.4. Información institutional . . . 4
1.4.1. Misión . . . 4
1.4.2. Visión . . . 4
1.5. Información de acreditacion . . . 5
1.5.1. Resultados de la carrera *Outcomes* . . . 5
1.5.2. Objetivos educacionales . . . 5
1.5.3. Aprendizaje esperado (*Learning Outcomes*) . . . 6
1.6. Perfiles . . . 6
1.6.1. Perfil del estudiante . . . 6
1.6.2. Perfil del graduado . . . 7
1.6.3. Perfil Profesional . . . 7
1.6.4. Perfil del docente . . . 8
1.7. Grados y Títulos . . . 8
1.8. Recursos para dictado de clases . . . 8
1.9. Organización del documento . . . 8

2. Cuerpo del conocimiento de Ciencia de la Computación–*Computer Science* 11
2.1. Inteligencia Artificial (AI) . . . 13
2.1.1. AI/Introducción a la IA: Conceptos y Comportamiento Inteligente (CS Core: 2 hrs, KA Core: 1 hr) . . . 15
2.1.2. AI/Características del Problema y Naturaleza de los Agentes (CS Core: 1 hr, KA Core: 1 hr) . . . 15
2.1.3. AI/Estado y Búsqueda No Informada (CS Core: 1 hr, KA Core: 2 hrs) . . . 15
2.1.4. AI/Búsqueda Heurística y CSP (CS Core: 1 hr, KA Core: 2 hrs) . . . 16
2.1.5. AI/Búsqueda Adversarial y Optimización Estocástica (CS Core: 1 hr, KA Core: 3 hrs) . . . 16
2.1.6. AI/Representación del Conocimiento y Razonamiento Probabilístico Básico (CS Core: 1 hr, KA Core: 1 hr) . . . 17
2.1.7. AI/Independencia, Modelos de Markov y Toma de Decisiones (CS Core: 1 hr, KA Core: 1 hr) . . . 17
2.1.8. AI/Fundamentos de Aprendizaje Automático (CS Core: 1 hr, KA Core: 2 hrs) 17
2.1.9. AI/Aprendizaje Automático: Datos, Representaciones y Evaluación (CS Core: 1 hr, KA Core: 2 hrs) . . . 18
2.1.10. AI/Redes Neuronales y Ética (CS Core: 1 hr, KA Core: 1 hr) . . . 19
2.1.11. AI/Formulación del Aprendizaje Automático y Evaluación (CS Core: 1 hr, KA Core: 2 hrs) . . . 19

2.1.12. AI/Aprendizaje Profundo, RL y Ética (CS Core: 1 hr, KA Core: 2 hrs) 19
2.1.13. AI/Entrenamiento y Optimización de Redes Neuronales (CS Core: 1 hr, KA Core: 1 hr) 20
2.1.14. AI/Técnicas de Regularización para Redes Profundas (CS Core: 1 hr, KA Core: 1 hr) 20
2.1.15. AI/Arquitecturas de Redes Convolucionales y Residuales (CS Core: 1 hr, KA Core: 1 hr) 20
2.1.16. AI/Arquitecturas Transformer y Modelos Generativos (CS Core: 1 hr, KA Core: 1 hr) 20
2.1.17. AI/Aplicaciones e Impacto Social de la IA (CS Core: 2 hrs, KA Core: 2 hrs) . . 21
2.1.18. AI/Aplicaciones Adicionales e Impacto Social de la IA (CS Core: 2 hrs, KA Core: 2 hrs) 21
2.1.19. AI/Representación Lógica y Razonamiento 22
2.1.20. AI/Representación Probabilística y Razonamiento 22
2.1.21. AI/Planificación 23
2.1.22. AI/Agentes y Sistemas Cognitivos 23
2.1.23. AI/Procesamiento del Lenguaje Natural 24
2.1.24. AI/Robótica 25
2.1.25. AI/Percepción y Visión por Computadora 25
2.2. Fundamentos Algorítmicos (AL) 26
2.2.1. AL/Estructuras de Datos Fundamentales (CS Core: 5 hrs, KA Core: 3 hrs) . . 27
2.2.2. AL/Algoritmos Fundamentales (CS Core: 4 hrs, KA Core: 2 hrs) 27
2.2.3. AL/Algoritmos Avanzados (CS Core: 3 hrs, KA Core: 2 hrs) 28
2.2.4. AL/Estrategias Algorítmicas (CS Core: 6 hrs) 28
2.2.5. AL/Marco de Análisis de Complejidad (CS Core: 2 hrs, KA Core: 1 hr) 29
2.2.6. AL/Notación Asintótica y Clases de Complejidad (CS Core: 1 hr, KA Core: 1 hr) 30
2.2.7. AL/Análisis de Complejidad: Recursión, Amortización y Cotas Ajustadas (CS Core: 2 hrs, KA Core: 1 hr) 30
2.2.8. AL/Teoría de la Complejidad Computacional (CS Core: 2 hrs, KA Core: 1 hr) 30
2.2.9. AL/Lenguajes Formales y Autómatas (CS Core: 4 hrs, KA Core: 10 hrs) . . . 31
2.2.10. AL/Computabilidad y Decidibilidad (CS Core: 6 hrs, KA Core: 14 hrs) 31
2.2.11. AL/Sociedad, ética y la Profesión 32
2.3. Arquitectura y Organización (AR) 33
2.3.1. AR/Lógica Digital y Sistemas Digitales (KA Core: 3 hrs) 33
2.3.2. AR/Representación de Datos a Nivel de Máquina (CS Core: 1 hr) 34
2.3.3. AR/Organización de Máquina a Nivel de Ensamblador (CS Core: 1 hr, KA Core: 2 hrs) 34
2.3.4. AR/Jerarquía de Memoria (CS Core: 6 hrs) 35
2.3.5. AR/Interconexión y Comunicación (CS Core: 1 hr) 35
2.3.6. AR/Organización Funcional (KA Core: 2 hrs) 36
2.3.7. AR/Rendimiento y Eficiencia Energética (KA Core: 3 hrs) 36
2.3.8. AR/Arquitecturas Heterogéneas (KA Core: 2 hrs) 37
2.3.9. AR/Arquitecturas de Procesador Seguras (KA Core: 2 hrs) 37
2.3.10. AR/Arquitecturas Cuánticas (KA Core: 2 hrs) 38
2.3.11. AR/Problemas de Sostenibilidad 38
2.4. Gestión de Datos (DM) 39
2.4.1. DM/El Rol de los Datos y el Ciclo de Vida de los Datos (CS Core: 2 hrs) . . . 40
2.4.2. DM/Conceptos Fundamentales de Sistemas de Bases de Datos (CS Core: 2 hrs, KA Core: 1 hr) 40
2.4.3. DM/Modelado de Datos (CS Core: 2 hrs, KA Core: 3 hrs) 41
2.4.4. DM/Bases de Datos Relacionales (CS Core: 1 hr, KA Core: 3 hrs) 42
2.4.5. DM/Construcción de Consultas (CS Core: 2 hrs, KA Core: 4 hrs) 42
2.4.6. DM/Procesamiento de Consultas (KA Core: 4 hrs) 43
2.4.7. DM/Internos de DBMS (KA Core: 4 hrs) 43
2.4.8. DM/Sistemas NoSQL (KA Core: 2 hrs) 44
2.4.9. DM/Seguridad y Privacidad de Datos (CS Core: 1 hr, KA Core: 2 hrs) 44

2.4.10. DM/Análisis de Datos (KA Core: 3 hrs) . . . 45
2.4.11. DM/Bases de Datos Distribuidas/Computación en la Nube . . . 45
2.4.12. DM/Bases de Datos Semiestructuradas y No Estructuradas . . . 45
2.4.13. DM/Sociedad, ética y Profesión . . . 46
2.5. Fundamentos de los Lenguajes de Programación (FPL) . . . 46
2.5.1. FPL/Fundamentos de Programación Orientada a Objetos (CS Core: 2 hrs, KA Core: 1 hr) . . . 47
2.5.2. FPL/Programación Orientada a Objetos: Encapsulación, Subtipado y Reflexión (CS Core: 2 hrs, KA Core: 1 hr) . . . 48
2.5.3. FPL/Programación Funcional (CS Core: 4 hrs, KA Core: 3 hrs) . . . 48
2.5.4. FPL/Programación Lógica (KA Core: 2 hrs) . . . 49
2.5.5. FPL/Scripting de Shell (CS Core: 2 hrs) . . . 50
2.5.6. FPL/Programación Dirigida por Eventos y Reactiva (CS Core: 2 hrs, KA Core: 2 hrs) . . . 50
2.5.7. FPL/Computación Paralela y Distribuida (CS Core: 2 hrs, KA Core: 2 hrs) . . . 51
2.5.8. FPL/Programación Orientada a Aspectos . . . 52
2.5.9. FPL/Fundamentos de Sistemas de Tipos (CS Core: 2 hrs, KA Core: 2 hrs) . . . 52
2.5.10. FPL/Sistemas de Tipos: Polimorfismo y Tipado Complementario (CS Core: 1 hr, KA Core: 1 hr) . . . 53
2.5.11. FPL/Teoría de Tipos Avanzada (CS Core: 2 hrs, KA Core: 2 hrs) . . . 53
2.5.12. FPL/Modelo de Ejecución y Memoria de Sistemas (CS Core: 2 hrs) . . . 54
2.5.13. FPL/Gestión de Memoria (CS Core: 1 hr) . . . 54
2.5.14. FPL/Traducción y Ejecución de Lenguajes (CS Core: 2 hrs, KA Core: 3 hrs) . . . 54
2.5.15. FPL/Abstracción y Representación de Programas (KA Core: 3 hrs) . . . 55
2.5.16. FPL/Análisis Sintáctico . . . 56
2.5.17. FPL/Análisis Semántico del Compilador . . . 56
2.5.18. FPL/Análisis de Programas y Analizadores . . . 57
2.5.19. FPL/Generación de Código . . . 57
2.5.20. FPL/Comportamiento en Tiempo de Ejecución y Sistemas . . . 58
2.5.21. FPL/Construcciones de Programación Avanzadas . . . 59
2.5.22. FPL/Pragmática del Lenguaje . . . 59
2.5.23. FPL/Semántica Formal . . . 60
2.5.24. FPL/Metodologías de Desarrollo Formal . . . 60
2.5.25. FPL/Principios de Diseño de Lenguajes de Programación . . . 61
2.5.26. FPL/Sociedad, ética y Profesión . . . 61
2.6. Gráficos y Técnicas Interactivas (GIT) . . . 61
2.6.1. GIT/Conceptos Fundamentales de Gráficos y Técnicas Interactivas (CS Core: 4 hrs, KA Core: 3 hrs) . . . 62
2.6.2. GIT/Visualización (KA Core: 6 hrs) . . . 62
2.6.3. GIT/Renderizado Aplicado y Técnicas (KA Core: 15 hrs) . . . 63
2.6.4. GIT/Modelado Geométrico (KA Core: 6 hrs) . . . 63
2.6.5. GIT/Sombreado y Renderizado Avanzado (KA Core: 6 hrs) . . . 63
2.6.6. GIT/Animación por Computadora (KA Core: 6 hrs) . . . 64
2.6.7. GIT/Simulación (KA Core: 6 hrs) . . . 64
2.6.8. GIT/Inmersión (KA Core: 6 hrs) . . . 64
2.6.9. GIT/Interacción (KA Core: 4 hrs) . . . 64
2.6.10. GIT/Procesamiento de Imágenes (KA Core: 6 hrs) . . . 65
2.6.11. GIT/Computación Tangible/Física (KA Core: 6 hrs) . . . 65
2.6.12. GIT/Sociedad, ética y Profesión . . . 65
2.7. Interacción Humano-Computadora (HCI) . . . 66
2.7.1. HCI/Comprender al Usuario: Objetivos individuales e interacciones con otros (CS Core: 2 hrs, KA Core: 5 hrs) . . . 66
2.7.2. HCI/Responsabilidad y Rendición de Cuentas en el Diseño (CS Core: 2 hrs, KA Core: 2 hrs) . . . 67
2.7.3. HCI/Accesibilidad y Diseño Inclusivo (CS Core: 2 hrs, KA Core: 2 hrs) . . . 67
2.7.4. HCI/Evaluación del Diseño (CS Core: 1 hr, KA Core: 2 hrs) . . . 67

2.7.5. HCI/Diseño de Sistemas (CS Core: 1 hr, KA Core: 5 hrs) . . . 68
2.7.6. HCI/Sociedad, ética y Profesión . . . 68
2.8. Fundamentos Matemáticos y Estadísticos (MSF) . . . 68
2.8.1. MSF/Matemáticas Discretas (CS Core: 15 hrs, KA Core: 6 hrs) . . . 69
2.8.2. MSF/Matemáticas Discretas Avanzadas (CS Core: 15 hrs, KA Core: 6 hrs) . . . 69
2.8.3. MSF/Probabilidad (CS Core: 11 hrs, KA Core: 29 hrs) . . . 70
2.8.4. MSF/Estadística (CS Core: 10 hrs, KA Core: 30 hrs) . . . 70
2.8.5. MSF/álgebra Lineal (CS Core: 5 hrs, KA Core: 35 hrs) . . . 71
2.8.6. MSF/Cálculo (KA Core: 40 hrs) . . . 71
2.9. Redes y Comunicaciones (NC) . . . 72
2.9.1. NC/Fundamentos de Redes y Comunicaciones (CS Core: 3 hrs) . . . 72
2.9.2. NC/Aplicaciones en Red (CS Core: 4 hrs) . . . 73
2.9.3. NC/Soporte de Fiabilidad (KA Core: 6 hrs) . . . 73
2.9.4. NC/Enrutamiento y Reenvío (KA Core: 4 hrs) . . . 74
2.9.5. NC/Comunicación de un solo salto (KA Core: 3 hrs) . . . 74
2.9.6. NC/Seguridad de Red (KA Core: 3 hrs) . . . 74
2.9.7. NC/Soporte de Movilidad (KA Core: 4 hrs) . . . 75
2.9.8. NC/Temas Emergentes (KA Core: 4 hrs) . . . 75
2.10. Sistemas Operativos (OS) . . . 75
2.10.1. OS/Rol y Propósito de los Sistemas Operativos (CS Core: 2 hrs) . . . 76
2.10.2. OS/Principios del Sistema Operativo (CS Core: 2 hrs) . . . 77
2.10.3. OS/Concurrencia (CS Core: 2 hrs, KA Core: 1 hr) . . . 77
2.10.4. OS/Protección y Seguridad (CS Core: 2 hrs, KA Core: 1 hr) . . . 78
2.10.5. OS/Planificación (KA Core: 2 hrs) . . . 79
2.10.6. OS/Modelo de Proceso (KA Core: 2 hrs) . . . 79
2.10.7. OS/Gestión de Memoria (KA Core: 1 hr) . . . 80
2.10.8. OS/Gestión de Dispositivos (KA Core: 1 hr) . . . 80
2.10.9. OS/API de Sistemas de Archivos e Implementación (KA Core: 2 hrs) . . . 81
2.10.10. OS/Sistemas de Archivos Avanzados (KA Core: 1 hr) . . . 81
2.10.11. OS/Virtualización (KA Core: 1 hr) . . . 82
2.10.12. OS/Sistemas en Tiempo Real y Embebidos (KA Core: 1 hr) . . . 83
2.10.13. OS/Tolerancia a fallos (KA Core: 1 hr) . . . 83
2.10.14. OS/Sociedad, ética y Profesión . . . 84
2.11. Computación Paralela y Distribuida (PDC) . . . 84
2.11.1. PDC/Programas: Paralelismo Declarativo (CS Core: 1 hr, KA Core: 1 hr) . . . 85
2.11.2. PDC/Programas: Inicio de Actividades (CS Core: 1 hr, KA Core: 1 hr) . . . 85
2.11.3. PDC/Programas: Propiedades de Ejecución (CS Core: 1 hr, KA Core: 1 hr) . . . 86
2.11.4. PDC/Programas: Distribución (CS Core: 1 hr, KA Core: 1 hr) . . . 86
2.11.5. PDC/Programas: Mapeos de Implementación (CS Core: 1 hr, KA Core: 1 hr) . . . 86
2.11.6. PDC/Programación GPU . . . 87
2.11.7. PDC/Comunicación (CS Core: 2 hrs, KA Core: 4 hrs) . . . 87
2.11.8. PDC/Comunicación: Propiedades y Extensiones (CS Core: 1 hr, KA Core: 1 hr) 88
2.11.9. PDC/Memoria y Consistencia (CS Core: 1 hr, KA Core: 1 hr) . . . 88
2.11.10. PDC/Coordinación (CS Core: 1 hr, KA Core: 3 hrs) . . . 89
2.11.11. PDC/Coordinación: Sincronización y Atomicidad (CS Core: 1 hr, KA Core: 2 hrs) . . . 89
2.11.12. PDC/Coordinación: Propiedades Avanzadas (CS Core: 1 hr, KA Core: 2 hrs) . . . 90
2.11.13. PDC/Evaluación (CS Core: 1 hr, KA Core: 3 hrs) . . . 90
2.11.14. PDC/Evaluación: Métodos Formales (CS Core: 1 hr, KA Core: 1 hr) . . . 91
2.11.15. PDC/Evaluación: Pruebas y Medición (CS Core: 1 hr, KA Core: 1 hr) . . . 91
2.11.16. PDC/Algoritmos (CS Core: 1 hr, KA Core: 3 hrs) . . . 92
2.11.17. PDC/Algoritmos: Dominios de Aplicación (CS Core: 1 hr, KA Core: 3 hrs) . . . 92
2.11.18. PDC/Algoritmos: Dominios Algorítmicos (CS Core: 1 hr, KA Core: 3 hrs) . . . 93
2.12. Fundamentos de Desarrollo de Software (SDF) . . . 93
2.12.1. SDF/Conceptos y Construcciones Básicas de Programación (CS Core: 10 hrs) . . . 94

2.12.2. SDF/Prácticas de Programación: Pruebas, Depuración y Documentación (CS Core: 10 hrs) . . . 95
2.12.3. SDF/Estructuras de Datos Fundamentales (CS Core: 12 hrs) . . . 95
2.12.4. SDF/Algoritmos (CS Core: 6 hrs) . . . 95
2.12.5. SDF/Prácticas de Desarrollo de Software (CS Core: 5 hrs) . . . 96
2.12.6. SDF/Sociedad, ética y la Profesión . . . 96
2.13. Ingeniería de Software (SE) . . . 96
2.13.1. SE/Colaboración y Dinámica de Equipo (CS Core: 2 hrs, KA Core: 2 hrs) . . . 98
2.13.2. SE/Diversidad, Stakeholders y Equipos Distribuidos (CS Core: 1 hr, KA Core: 1 hr) . . . 99
2.13.3. SE/Control de Versiones y CI/CD (CS Core: 1 hr, KA Core: 2 hrs) . . . 99
2.13.4. SE/Entorno de Desarrollo e IDE (CS Core: 1 hr, KA Core: 2 hrs) . . . 100
2.13.5. SE/Ingeniería de Requisitos (KA Core: 2 hrs) . . . 100
2.13.6. SE/Evolución y Estimación de Requisitos (KA Core: 1 hr) . . . 101
2.13.7. SE/Principios y Arquitectura de Software (CS Core: 1 hr, KA Core: 2 hrs) . . . 101
2.13.8. SE/Diseño de API y Modelado (CS Core: 1 hr, KA Core: 1 hr) . . . 102
2.13.9. SE/Calidad y Evaluación del Diseño (CS Core: 1 hr, KA Core: 2 hrs) . . . 102
2.13.10. SE/Prácticas de Codificación (CS Core: 1 hr, KA Core: 2 hrs) . . . 103
2.13.11. SE/Construcción a Escala y Proceso (CS Core: 1 hr, KA Core: 2 hrs) . . . 103
2.13.12. SE/Conceptos de verificación y validación (CS Core: 1 hr, KA Core: 1 hr) . . . 104
2.13.13. SE/Planificación y Tipos de Pruebas (CS Core: 1 hr, KA Core: 1 hr) . . . 104
2.13.14. SE/Prácticas Avanzadas de Pruebas (CS Core: 1 hr, KA Core: 1 hr) . . . 105
2.13.15. SE/Herramientas y Análisis de Pruebas (CS Core: 1 hr, KA Core: 1 hr) . . . 105
2.13.16. SE/Pruebas de Rendimiento y Evaluación Comparativa (CS Core: 1 hr, KA Core: 1 hr) . . . 106
2.13.17. SE/Compatibilidad y Versionado de API (KA Core: 2 hrs) . . . 106
2.13.18. SE/Técnicas de Refactorización (KA Core: 1 hr) . . . 107
2.13.19. SE/Conceptos de Confiabilidad (KA Core: 2 hrs) . . . 107
2.13.20. SE/Ingeniería de Confiabilidad (KA Core: 1 hr) . . . 108
2.13.21. SE/Métodos Formales . . . 108
2.14. Seguridad (SEC) . . . 109
2.14.1. SEC/Principios y Mentalidad de Seguridad (CS Core: 1 hr, KA Core: 3 hrs) . . . 110
2.14.2. SEC/Control de Acceso y Aplicaciones (CS Core: 1 hr, KA Core: 3 hrs) . . . 111
2.14.3. SEC/Seguridad en IA y Hardware (CS Core: 1 hr, KA Core: 2 hrs) . . . 111
2.14.4. SEC/Ética, Ley y Sociedad en Seguridad (CS Core: 1 hr, KA Core: 2 hrs) . . . 112
2.14.5. SEC/Privacidad y Seguridad por Diseño (CS Core: 1 hr, KA Core: 1 hr) . . . 112
2.14.6. SEC/Inyección y Validación de Entrada (CS Core: 1 hr, KA Core: 2 hrs) . . . 113
2.14.7. SEC/Seguridad de Memoria y Tipos (CS Core: 1 hr, KA Core: 1 hr) . . . 113
2.14.8. SEC/Flujo de Información y No Interferencia (CS Core: 1 hr, KA Core: 1 hr) . . . 114
2.14.9. SEC/Análisis de Malware y Seguridad Avanzada (CS Core: 1 hr, KA Core: 3 hrs) 114
2.14.10. SEC/Fundamentos de Criptografía (CS Core: 1 hr, KA Core: 2 hrs) . . . 114
2.14.11. SEC/Algoritmos Criptográficos (CS Core: 1 hr, KA Core: 2 hrs) . . . 115
2.14.12. SEC/Protocolos, Aplicaciones y Criptografía Avanzada (CS Core: 1 hr, KA Core: 2 hrs) . . . 115
2.14.13. SEC/Análisis de Amenazas e Ingeniería de Seguridad (CS Core: 1 hr, KA Core: 4 hrs) . . . 116
2.14.14. SEC/Diseño e Ingeniería de Controles de Seguridad (CS Core: 1 hr, KA Core: 3 hrs) . . . 116
2.14.15. SEC/Computación Confiable e Ingeniería de Privacidad (CS Core: 1 hr, KA Core: 3 hrs) . . . 117
2.14.16. SEC/Técnicas de Informática Forense (KA Core: 4 hrs) . . . 117
2.14.17. SEC/Aspectos Legales y Manejo de Incidentes (KA Core: 3 hrs) . . . 118
2.14.18. SEC/Gobierno, Gestión y Política de Seguridad (KA Core: 2 hrs) . . . 118
2.14.19. SEC/Gestión de Riesgos y Datos (KA Core: 1 hr) . . . 119
2.15. Sociedad, ética y la Profesión (SEP) . . . 119
2.15.1. SEP/Impacto Social e IA (CS Core: 2 hrs, KA Core: 1 hr) . . . 120

2.15.2. SEP/Equidad Digital y Accesibilidad (CS Core: 2 hrs, KA Core: 1 hr) 121
2.15.3. SEP/Fundamentos y Marcos del Análisis Ético (CS Core: 2 hrs, KA Core: 1 hr) 121
2.15.4. SEP/Métodos y Herramientas de Análisis Ético (CS Core: 1 hr, KA Core: 1 hr) 122
2.15.5. SEP/Valores, Naturaleza y Actualización Profesional (CS Core: 1 hr, KA Core: 1 hr) 122
2.15.6. SEP/Códigos de Ética y Responsabilidad Legal (CS Core: 1 hr, KA Core: 1 hr) 123
2.15.7. SEP/Responsabilidad Profesional y Política Pública (CS Core: 1 hr, KA Core: 1 hr) 123
2.15.8. SEP/Disidencia Ética y Contexto Cultural (CS Core: 1 hr, KA Core: 1 hr) . . 124
2.15.9. SEP/Ética en el Lugar de Trabajo (CS Core: 1 hr, KA Core: 1 hr) 124
2.15.10. SEP/Fundamentos de la Propiedad Intelectual (CS Core: 1 hr, KA Core: 1 hr) 124
2.15.11. SEP/Licencias, Plagio y Autoría (CS Core: 1 hr, KA Core: 1 hr) 125
2.15.12. SEP/Derechos de PI Digital y Derechos de Autor (CS Core: 1 hr, KA Core: 1 hr) 125
2.15.13. SEP/Open Source, Piratería y DRM (CS Core: 1 hr, KA Core: 1 hr) 125
2.15.14. SEP/Datos, Vigilancia y Tecnología de Privacidad (CS Core: 1 hr, KA Core: 1 hr) 126
2.15.15. SEP/Fundamentos Filosóficos y Legislación de Privacidad (CS Core: 1 hr, KA Core: 1 hr) 126
2.15.16. SEP/Libertades Civiles y Expresión (CS Core: 1 hr, KA Core: 1 hr) 127
2.15.17. SEP/Comunicación Técnica y Profesional (CS Core: 1 hr, KA Core: 1 hr) . . . 127
2.15.18. SEP/Comunicación en Equipo y Cultura (CS Core: 1 hr, KA Core: 1 hr) . . . 127
2.15.19. SEP/Huella Ambiental de los Sistemas Informáticos (CS Core: 1 hr, KA Core: 1 hr) 128
2.15.20. SEP/Efectos Sistémicos y Contexto Social (CS Core: 1 hr, KA Core: 1 hr) . . . 128
2.15.21. SEP/Diseño Sostenible y Computación Ubicua (CS Core: 1 hr, KA Core: 1 hr) 129
2.15.22. SEP/Fundamentos e Historia Temprana de la Informática (CS Core: 1 hr, KA Core: 1 hr) 129
2.15.23. SEP/Historia Moderna: PC, Internet e IA (CS Core: 1 hr, KA Core: 1 hr) . . . 130
2.15.24. SEP/Modelos y Estrategias Económicas (KA Core: 1 hr) 130
2.15.25. SEP/Automatización, IA y Ética Económica (KA Core: 1 hr) 131
2.15.26. SEP/Delitos y Ataques Informáticos (CS Core: 1 hr, KA Core: 1 hr) 131
2.15.27. SEP/Leyes, Políticas y Responsabilidad en Seguridad (CS Core: 1 hr, KA Core: 1 hr) 131
2.15.28. SEP/Identidad y Diversidad en la Informática (CS Core: 1 hr, KA Core: 1 hr) 132
2.15.29. SEP/Contexto Histórico e Intercultural de DEIA (CS Core: 1 hr, KA Core: 1 hr) 132
2.15.30. SEP/Inclusión y Prácticas en Informática (CS Core: 1 hr, KA Core: 1 hr) . . . 133
2.15.31. SEP/Diseño Accesible (CS Core: 1 hr, KA Core: 1 hr) 133
2.16. Fundamentos de Sistemas (SF) 133
2.16.1. SF/Componentes Hardware y Lógica (CS Core: 2 hrs) 134
2.16.2. SF/Capas, Abstracciones y Escalabilidad (CS Core: 2 hrs) 134
2.16.3. SF/Sistemas Digitales y Lógica (CS Core: 2 hrs) 135
2.16.4. SF/Máquinas de Estado (CS Core: 2 hrs) 135
2.16.5. SF/Procesamiento Secuencial y Paralelo (CS Core: 2 hrs) 136
2.16.6. SF/Gestión de Recursos (CS Core: 1 hr, KA Core: 1 hr) 136
2.16.7. SF/Latencia, Caché y Jerarquía de Memoria (CS Core: 2 hrs, KA Core: 2 hrs) 137
2.16.8. SF/Virtualización y Aislamiento (CS Core: 1 hr, KA Core: 1 hr) 137
2.16.9. SF/Métricas y Benchmarks de Rendimiento (CS Core: 1 hr, KA Core: 1 hr) . . 138
2.16.10. SF/Modelos Analíticos de Rendimiento (CS Core: 1 hr, KA Core: 1 hr) 138
2.16.11. SF/Eventos, Herramientas y Experimentación (CS Core: 1 hr, KA Core: 1 hr) 138
2.16.12. SF/Confiabilidad del Sistema (CS Core: 2 hrs, KA Core: 1 hr) 139
2.16.13. SF/Seguridad del Sistema (CS Core: 2 hrs, KA Core: 1 hr) 139
2.16.14. SF/Diseño del Sistema (CS Core: 2 hrs, KA Core: 1 hr) 140
2.16.15. SF/Sociedad, ética y la Profesión 140
2.17. Desarrollo de Plataformas Especializadas (SPD) 140

2.17.1. SPD/Aspectos Comunes: Plataformas, APIs y Restricciones (CS Core: 3 hrs, KA Core: 2 hrs) . . . 141
2.17.2. SPD/Aspectos Comunes: Lenguajes de Plataforma y Patrones Web (CS Core: 2 hrs, KA Core: 1 hr) . . . 142
2.17.3. SPD/Frontend, Frameworks y Estándares Web (KA Core: 3 hrs) . . . 142
2.17.4. SPD/Arquitectura Cloud y Gestión de Datos (KA Core: 3 hrs) . . . 142
2.17.5. SPD/Plataformas Móviles: Fundamentos del Desarrollo (KA Core: 2 hrs) . . . 143
2.17.6. SPD/Plataformas Móviles: Arquitectura y Restricciones (KA Core: 2 hrs) . . . 143
2.17.7. SPD/Plataformas Móviles: Capacidades y Computación Asíncrona (KA Core: 1 hr) . . . 144
2.17.8. SPD/Plataformas Robóticas (KA Core: 4 hrs) . . . 144
2.17.9. SPD/Características y Arquitecturas Embebidas (KA Core: 1 hr) . . . 144
2.17.10. SPD/Sistemas, Programación y Sensores (KA Core: 2 hrs) . . . 145
2.17.11. SPD/Tiempo Real, Memoria y Verificación (KA Core: 3 hrs) . . . 145
2.17.12. SPD/Historia, Tipos y Características de Plataformas (KA Core: 1 hr) . . . 146
2.17.13. SPD/Sensores, Controladores y Ecosistemas eSports (KA Core: 1 hr) . . . 146
2.17.14. SPD/Arquitecturas y Renderizado (KA Core: 1 hr) . . . 147
2.17.15. SPD/Algoritmos, Medios y Mundos Virtuales (KA Core: 1 hr) . . . 147
2.17.16. SPD/Lenguajes, APIs y Herramientas de Desarrollo (KA Core: 1 hr) . . . 147
2.17.17. SPD/Motores de Juego (KA Core: 1 hr) . . . 148
2.17.18. SPD/Fundamentos del Diseño de Juegos (KA Core: 1 hr) . . . 148
2.17.19. SPD/Géneros, Audiencias y Alcance Social (KA Core: 1 hr) . . . 148
2.17.20. SPD/Plataformas Interactivas de Datos y Análisis (KA Core: 2 hrs) . . . 149
2.17.21. SPD/Programación con IA y Plataformas Cuánticas (KA Core: 1 hr) . . . 149
2.17.22. SPD/SPD-SEP/Móvil . . . 150
2.17.23. SPD/SPD-SEP Web . . . 150
2.17.24. SPD/Propiedad Intelectual en los Juegos . . . 151
2.17.25. SPD/Accesibilidad e Interfaz del Juego . . . 151
2.17.26. SPD/Salud, Seguridad y Sostenibilidad . . . 151
2.17.27. SPD/Representación, Inclusión e Igualdad . . . 152
2.17.28. SPD/SPD-SEP/Robótica . . . 152
2.17.29. SPD/SPD-SEP/Interactiva . . . 153

3. Cuerpo del conocimiento de Ciencias Básicas para Computación 155
3.1. Cálculo y Análisis (CAN) . . . 156
3.1.1. CAN/Integrales de Línea y Campos Vectoriales (CS Core: 1 hr, KA Core: 1 hr) 156
3.1.2. CAN/Integrales Múltiples (CS Core: 1 hr, KA Core: 1 hr) . . . 157
3.1.3. CAN/Transformaciones Multivariables (CS Core: 1 hr, KA Core: 1 hr) . . . 157
3.1.4. CAN/Integrales de Superficie y Teoremas Fundamentales (CS Core: 1 hr, KA Core: 1 hr) . . . 157
3.1.5. CAN/Funciones Vectoriales de una Variable Real (CS Core: 1 hr, KA Core: 1 hr)158
3.1.6. CAN/Ecuaciones Paramétricas y Coordenadas Polares (CS Core: 1 hr, KA Core: 1 hr) . . . 158
3.1.7. CAN/Límites y Continuidad (CS Core: 1 hr, KA Core: 1 hr) . . . 158
3.1.8. CAN/Cálculo Diferencial (CS Core: 1 hr, KA Core: 1 hr) . . . 159
3.1.9. CAN/Antiderivadas y Técnicas de Integración (CS Core: 1 hr, KA Core: 1 hr) 159
3.1.10. CAN/Integrales Impropias (CS Core: 1 hr, KA Core: 1 hr) . . . 159
3.1.11. CAN/Sucesiones, Series y Series de Potencias (CS Core: 1 hr, KA Core: 1 hr) . 160
3.1.12. CAN/Cálculo Diferencial Multivariable (CS Core: 1 hr, KA Core: 1 hr) . . . 160
3.1.13. CAN/Análisis Armónico y Teoría de Fourier (CS Core: 1 hr, KA Core: 1 hr) . 160
3.2. Álgebra y Teoría de Números (ANT) . . . 161
3.2.1. ANT/Matrices y Sistemas de Ecuaciones Lineales (CS Core: 1 hr, KA Core: 1 hr)161
3.2.2. ANT/Espacios Vectoriales y Transformaciones Lineales (CS Core: 1 hr, KA Core: 1 hr) . . . 161
3.2.3. ANT/Valores Propios, Producto Interno y Formas Cuadráticas (CS Core: 1 hr, KA Core: 1 hr) . . . 162

3.2.4. ANT/Teoría de Grupos (CS Core: 1 hr, KA Core: 1 hr) . . . 162
3.2.5. ANT/Anillos, Cuerpos y Teoría de Galois (CS Core: 1 hr, KA Core: 1 hr) . . . 162
3.2.6. ANT/Criptografía y Teoría de Códigos (CS Core: 1 hr, KA Core: 1 hr) 163
3.3. Matemática Discreta y Combinatoria (DMC) . . . 163
3.3.1. DMC/Combinatoria Enumerativa (CS Core: 1 hr, KA Core: 1 hr) . . . 163
3.3.2. DMC/Teoría de Grafos (CS Core: 1 hr, KA Core: 1 hr) . . . 164
3.3.3. DMC/Optimización Discreta (CS Core: 1 hr, KA Core: 1 hr) . . . 164
3.3.4. DMC/Lógica Matemática y Teoría de Conjuntos (CS Core: 1 hr, KA Core: 1 hr) 164
3.4. Probabilidad y Estadística (PST) . . . 165
3.4.1. PST/Axiomas de Probabilidad y Distribuciones (CS Core: 1 hr, KA Core: 1 hr) 165
3.4.2. PST/Teoremas Límite (CS Core: 1 hr, KA Core: 1 hr) . . . 165
3.4.3. PST/Procesos Estocásticos (CS Core: 1 hr, KA Core: 1 hr) . . . 166
3.4.4. PST/Teoría de la Estimación (CS Core: 1 hr, KA Core: 1 hr) . . . 166
3.4.5. PST/Inferencia Bayesiana (CS Core: 1 hr, KA Core: 1 hr) . . . 166
3.4.6. PST/Análisis de Series Temporales (CS Core: 1 hr, KA Core: 1 hr) . . . 167
3.5. Análisis Numérico y Científico (NSA) . . . 167
3.5.1. NSA/Teoría de Aproximación e Interpolación (CS Core: 1 hr, KA Core: 1 hr) . 168
3.5.2. NSA/Integración Numérica y Cuadratura (CS Core: 1 hr, KA Core: 1 hr) . . . 168
3.5.3. NSA/Análisis de Errores y Aritmética de Punto Flotante (CS Core: 1 hr, KA Core: 1 hr) . . . 168
3.5.4. NSA/Búsqueda de Raíces de Ecuaciones No Lineales (CS Core: 1 hr, KA Core: 1 hr) . . . 169
3.5.5. NSA/Álgebra Lineal Numérica (CS Core: 1 hr, KA Core: 1 hr) . . . 169
3.5.6. NSA/Métodos Numéricos para Ecuaciones Diferenciales (CS Core: 1 hr, KA Core: 1 hr) . . . 169
3.5.7. NSA/Algoritmos de Optimización (CS Core: 1 hr, KA Core: 1 hr) . . . 170
3.5.8. NSA/Método de Elementos Finitos (CS Core: 1 hr, KA Core: 1 hr) . . . 170
3.5.9. NSA/Computación Paralela y de Alto Rendimiento (CS Core: 1 hr, KA Core: 1 hr) . . . 171
3.6. Modelado Matemático y Simulación (MMS) . . . 171
3.6.1. MMS/Ecuaciones Diferenciales Ordinarias de Primer Orden (CS Core: 1 hr, KA Core: 1 hr) . . . 171
3.6.2. MMS/Ecuaciones Diferenciales de Orden Superior y Sistemas (CS Core: 1 hr, KA Core: 1 hr) . . . 172
3.6.3. MMS/Sistemas Dinámicos y Caos (CS Core: 1 hr, KA Core: 1 hr) . . . 172
3.6.4. MMS/Teoría Matemática del Control (CS Core: 1 hr, KA Core: 1 hr) . . . 172
3.6.5. MMS/Ecuaciones Diferenciales Estocásticas (CS Core: 1 hr, KA Core: 1 hr) . . 173
3.7. Fundamentos Matemáticos de la Ciencia de Datos (FDS) . . . 173
3.7.1. FDS/Teoría del Aprendizaje Estadístico (CS Core: 1 hr, KA Core: 1 hr) . . . 173
3.7.2. FDS/Optimización para el Aprendizaje Automático (CS Core: 1 hr, KA Core: 1 hr) . . . 174
3.7.3. FDS/Teoría de la Información (CS Core: 1 hr, KA Core: 1 hr) . . . 174
3.7.4. FDS/Reducción de Dimensionalidad (CS Core: 1 hr, KA Core: 1 hr) . . . 175
3.7.5. FDS/Fundamentos Matemáticos del Aprendizaje Profundo (CS Core: 1 hr, KA Core: 1 hr) . . . 175
3.8. Información Cuántica y Computación (QIC) . . . 175
3.8.1. QIC/Fundamentos de la Mecánica Cuántica para la Computación (CS Core: 1 hr, KA Core: 1 hr) . . . 176
3.8.2. QIC/Circuitos y Puertas Cuánticas (CS Core: 1 hr, KA Core: 1 hr) . . . 176
3.8.3. QIC/Algoritmos Cuánticos (CS Core: 1 hr, KA Core: 1 hr) . . . 176
3.8.4. QIC/Entrelazamiento y Desigualdades de Bell (CS Core: 1 hr, KA Core: 1 hr) 177
3.9. Electromagnetismo (ELM) . . . 177
3.9.1. ELM/Electrostática: Campo Eléctrico y Ley de Gauss (CS Core: 1 hr, KA Core: 1 hr) . . . 178
3.9.2. ELM/Corriente Eléctrica, Circuitos y Leyes de Kirchhoff (CS Core: 1 hr, KA Core: 1 hr) . . . 178

3.9.3. ELM/Ecuaciones de Maxwell y Transformaciones de Norma (CS Core: 1 hr, KA Core: 1 hr) . . . 178
3.10. Termodinámica y Mecánica Estadística (TSM) . . . 179
3.10.1. TSM/Leyes de la Termodinámica y Funciones de Estado (CS Core: 1 hr, KA Core: 1 hr) . . . 179
3.10.2. TSM/Teoría Cinética de los Gases (CS Core: 1 hr, KA Core: 1 hr) . . . 179
3.10.3. TSM/Máquinas de Calor, Entropía y la Segunda Ley (CS Core: 1 hr, KA Core: 1 hr) . . . 180
3.11. Física Computacional y Basada en Datos (CDP) . . . 180
3.11.1. CDP/Métodos Numéricos para Ecuaciones Diferenciales en Física (CS Core: 1 hr, KA Core: 1 hr) . . . 180
3.11.2. CDP/Métodos de Monte Carlo y Simulaciones (CS Core: 1 hr, KA Core: 1 hr) 181
3.11.3. CDP/Aprendizaje Automático para el Descubrimiento Físico (CS Core: 1 hr, KA Core: 1 hr) . . . 181
3.12. Mecánica Clásica (CME) . . . 181
3.12.1. CME/Movimiento Rotacional y Momento Angular (CS Core: 1 hr, KA Core: 1 hr) . . . 182
3.12.2. CME/Sistemas de Partículas y Momento Lineal (CS Core: 1 hr, KA Core: 1 hr) 183
3.12.3. CME/Cinemática y Dinámica de Partículas (CS Core: 1 hr, KA Core: 1 hr) . . 183
3.12.4. CME/Trabajo, Energía y Leyes de Conservación (CS Core: 1 hr, KA Core: 1 hr) 184
3.12.5. CME/Dinámica de Cuerpos Rígidos (CS Core: 1 hr, KA Core: 1 hr) . . . 184
3.12.6. CME/Oscilaciones y Ondas (CS Core: 1 hr, KA Core: 1 hr) . . . 185
3.12.7. CME/Mecánica Lagrangiana y Hamiltoniana (CS Core: 1 hr, KA Core: 1 hr) . 185
3.12.8. CME/Elasticidad (CS Core: 1 hr, KA Core: 1 hr) . . . 185
3.12.9. CME/Fluidos: Hidrostática, Hidrodinámica y Viscosidad (CS Core: 1 hr, KA Core: 1 hr) . . . 186
3.12.10. CME/Vibraciones y Ondas (CS Core: 1 hr, KA Core: 1 hr) . . . 186
3.12.11. CME/Fenómeno de Transporte (CS Core: 1 hr, KA Core: 1 hr) . . . 187
3.13. Química General (GCH) . . . 187
3.13.1. GCH/Estructura Atómica (CS Core: 1 hr, KA Core: 1 hr) . . . 188
3.13.2. GCH/Enlace Químico (CS Core: 1 hr, KA Core: 1 hr) . . . 188
3.13.3. GCH/Materia y Energía (CS Core: 1 hr, KA Core: 1 hr) . . . 189
3.13.4. GCH/Estados de la Materia (CS Core: 1 hr, KA Core: 1 hr) . . . 189
3.13.5. GCH/Estequiometría (CS Core: 1 hr, KA Core: 1 hr) . . . 189
3.13.6. GCH/Química Ambiental y del Agua (CS Core: 1 hr, KA Core: 1 hr) . . . 190
3.13.7. GCH/Polímeros y Geosintéticos (CS Core: 1 hr, KA Core: 1 hr) . . . 190
3.13.8. GCH/Equilibrio Químico (CS Core: 1 hr, KA Core: 1 hr) . . . 191
3.13.9. GCH/Equilibrio Iónico e Hidrólisis de Sales (CS Core: 1 hr, KA Core: 1 hr) . . 191
3.13.10. GCH/Sales, Solubilidad y Equilibrios de Precipitación (CS Core: 1 hr, KA Core: 1 hr) . . . 191
3.13.11. GCH/Electroquímica y Electrólisis (CS Core: 1 hr, KA Core: 1 hr) . . . 191
3.13.12. GCH/Química del Cemento (CS Core: 1 hr, KA Core: 1 hr) . . . 192
3.14. Química Computacional y Teórica (CTC) . . . 192
3.14.1. CTC/Modelado Molecular y Simulación (CS Core: 1 hr, KA Core: 1 hr) . . . 192
3.14.2. CTC/Aprendizaje Automático en el Descubrimiento Químico (CS Core: 1 hr, KA Core: 1 hr) . . . 193
3.14.3. CTC/Diseño Computacional de Fármacos (CS Core: 1 hr, KA Core: 1 hr) . . . 193
3.15. Bioquímica y Biología Molecular (BMB) . . . 194
3.15.1. BMB/Estructura y Función de las Biomoléculas (CS Core: 1 hr, KA Core: 1 hr) 194
3.15.2. BMB/Genética Molecular y Enzimología (CS Core: 1 hr, KA Core: 1 hr) . . . 195
3.15.3. BMB/Bioinformática y Biología Computacional (CS Core: 1 hr, KA Core: 1 hr) 195
3.16. Teoría de la Probabilidad (PRO) . . . 196
3.16.1. PRO/Espacios Muestrales y Eventos (CS Core: 1 hr, KA Core: 1 hr) . . . 196
3.16.2. PRO/Axiomas y Propiedades de la Probabilidad (CS Core: 1 hr, KA Core: 1 hr) 196
3.16.3. PRO/Probabilidad Condicional e Independencia (CS Core: 1 hr, KA Core: 1 hr) 197

3.16.4. PRO/Variables Aleatorias Discretas y Distribuciones (CS Core: 1 hr, KA Core: 1 hr) . . . 197
3.16.5. PRO/Variables Aleatorias Continuas y Distribuciones (CS Core: 1 hr, KA Core: 1 hr) . . . 197
3.16.6. PRO/Distribuciones Conjuntas, Marginales y Condicionales (CS Core: 1 hr, KA Core: 1 hr) . . . 198
3.16.7. PRO/Teoremas Límite: LGN y TCL (CS Core: 1 hr, KA Core: 1 hr) . . . 198
3.17. Estadística Descriptiva y Análisis de Datos (DSC) . . . 199
3.17.1. DSC/Resúmenes Numéricos: Posición, Dispersión y Forma (CS Core: 1 hr, KA Core: 1 hr) . . . 199
3.18. Inferencia Estadística (INF) . . . 199
3.18.1. INF/Estimación Puntual (CS Core: 1 hr, KA Core: 1 hr) . . . 199
3.18.2. INF/Estimación por Intervalos e Intervalos de Confianza (CS Core: 1 hr, KA Core: 1 hr) . . . 200
3.18.3. INF/Fundamentos de las Pruebas de Hipótesis (CS Core: 1 hr, KA Core: 1 hr) 200
3.18.4. INF/Inferencia Bayesiana (CS Core: 1 hr, KA Core: 1 hr) . . . 200
3.19. Regresión y Modelos Estadísticos (REG) . . . 201
3.19.1. REG/Regresión Lineal Simple (CS Core: 1 hr, KA Core: 1 hr) . . . 201

4. Plan de estudios 203
4.1. Codificación de los cursos . . . 203
4.2. Estructura Curricular . . . 204
4.3. Créditos por Actividades Diversas . . . 209
4.4. Condición de Egreso . . . 209
4.5. Tópicos distribuídos por curso . . . 211
4.5.1. Tópicos de Ciencia de la Computación–*Computer Science*/curso . . . 211
4.5.2. Tópicos de Ciencias Básicas para Computación/curso . . . 229
4.6. Resultados esperados distribuídos por curso . . . 233
4.7. Cursos por resultado (*Outcome*) . . . 240
4.7.1. Outcome: AG-C01) Impacto social de soluciones computacionales. . . . 240
4.7.2. Outcome: AG-C02) Ética profesional en computación. . . . 240
4.7.3. Outcome: AG-C03) Trabajo individual y en equipo. . . . 240
4.7.4. Outcome: AG-C04) Comunicación efectiva en computación. . . . 240
4.7.5. Outcome: AG-C05) Gestión de proyectos en computación. . . . 241
4.7.6. Outcome: AG-C06) Aprendizaje continuo en computación. . . . 241
4.7.7. Outcome: AG-C07) Conocimientos de computación. . . . 241
4.7.8. Outcome: AG-C08) Análisis de problemas computacionales. . . . 241
4.7.9. Outcome: AG-C09) Diseño y desarrollo de soluciones. . . . 242
4.7.10. Outcome: AG-C10) Indagación en computación. . . . 242
4.7.11. Outcome: AG-C11) Uso de herramientas computacionales. . . . 242
4.7.12. Outcome: AG-C12) Gestión de tecnologías seguras de computación. . . . 243
4.8. Distribución de cursos en la carrera . . . 243
4.9. Departamentos . . . 245
4.9.1. DAISI . . . 245
4.10. Compatibilidad de la carrera con relación a estandares internacionales . . . 246

5. Contenido detallado por curso 257
Primer Semestre . . . 257
BIC101 Introducción a la Computación (Obligatorio) . . . 257
BMA101 Algebra Lineal (Obligatorio) . . . 262
BMA102 Cálculo Diferencial (Obligatorio) . . . 264
BCH101 Química I (Obligatorio) . . . 267
BEI101 Inglés I (Obligatorio) . . . 272
Segundo Semestre . . . 276
CS100 Introducción a la Ciencia de la Computación (Obligatorio) . . . 276
CS112 Programación Orientada a Objetos I (Obligatorio) . . . 287

CS1D1 Estructuras Discretas (Obligatorio) . . . 292
BMA103 Cálculo Integral (Obligatorio) . . . 295
BFI101 Física I (Obligatorio) . . . 297
BEI102 Inglés II (Obligatorio) . . . 302
Tercer Semestre . . . 306
CS113 Programación Orientada a Objetos II (Obligatorio) . . . 306
CS2B1 Desarrollo Basado en Plataformas (Obligatorio) . . . 312
AI161 IA Aplicada (Obligatorio) . . . 316
BMA104 Cálculo Diferencial e Integral Avanzado (Obligatorio) . . . 320
STA251 Estadística y Probabilidades (Obligatorio) . . . 324
BEI201 Inglés III (Electivo) . . . 329
Cuarto Semestre . . . 333
CS210 Algoritmos y Estructuras de Datos (Obligatorio) . . . 333
CS211 Teoría de la Computación (Obligatorio) . . . 338
CS221 Arquitectura de Computadores (Obligatorio) . . . 343
CS271 Bases de Datos I (Obligatorio) . . . 348
CS401 Metodología de la Investigación (Obligatorio) . . . 353
MA202 Métodos Numéricos (Obligatorio) . . . 357
BEI202 Inglés IV (Electivo) . . . 362
Quinto Semestre . . . 366
CS212 Análisis y Diseño de Algoritmos (Obligatorio) . . . 366
CS261 Inteligencia Artificial (Obligatorio) . . . 371
CS272 Bases de Datos II (Obligatorio) . . . 378
CS291 Ingeniería de Software I (Obligatorio) . . . 382
CS2S1 Sistemas Operativos (Obligatorio) . . . 388
FG106 Teatro (Obligatorio) . . . 393
BEI203 Inglés V (Electivo) . . . 397
Sexto Semestre . . . 401
CS231 Redes y Comunicación (Obligatorio) . . . 401
CS311 Algoritmos para Problemas Complejos (Obligatorio) . . . 406
CS312 Estructuras de Datos Avanzadas (Obligatorio) . . . 409
CS342 Compiladores (Obligatorio) . . . 412
AI263 Introducción al Aprendizaje de Máquina (Obligatorio) . . . 415
FI201 Física Computacional (Obligatorio) . . . 418
Séptimo Semestre . . . 421
CS251 Computación Gráfica (Obligatorio) . . . 421
CS292 Ingeniería de Software II (Obligatorio) . . . 425
CS2H1 Experiencia de Usuario (UX) (Obligatorio) . . . 430
AI264 Aprendizaje Profundo (Obligatorio) . . . 435
FG211 Ética Profesional (Obligatorio) . . . 438
FG350 Liderazgo y Desempeño (Obligatorio) . . . 440
Octavo Semestre . . . 443
CS281 Computación en la Sociedad (Obligatorio) . . . 443
CS3I1 Seguridad en Computación (Obligatorio) . . . 449
CS3P1 Computación Paralela y Distribuída (Obligatorio) . . . 455
CS402 Proyecto de tesis 1 (Obligatorio) . . . 465
EX301 Actividades Extracurriculares (Obligatorio) . . . 469
ET201 Formación de Empresas de Base Tecnológica I (Obligatorio) . . . 471
AI268 Visión Computacional (Electivo) . . . 475
CS391 Ingeniería de Software III (Electivo) . . . 481
CS393 Sistemas de Información (Electivo) . . . 486
CS394 Arquitectura de Software y DevOps para Software en la Nube (Electivo) . . . 489
CS395 Calidad de Software (Electivo) . . . 495
Noveno Semestre . . . 503
CS370 Big Data (Obligatorio) . . . 503
CS400 Prácticas Pre-profesionales (Obligatorio) . . . 507

CS403 Proyecto de tesis 2 (Obligatorio) . . . 509
AI365 Modelos Generativos Avanzados en IA (Obligatorio) . . . 513
CB309 Bioinformática (Obligatorio) . . . 515
ET301 Formación de Empresas de Base Tecnológica II (Obligatorio) . . . 519
AI369 Robótica (Electivo) . . . 522
CS351 Tópicos en Computación Gráfica (Electivo) . . . 526
CS353 Computación Cuántica (Electivo) . . . 530
CS392 Tópicos en Ingeniería de Software (Electivo) . . . 534
CS3P3 Internet de las Cosas (Electivo) . . . 541
Décimo Semestre . . . 544
CS3P2 Cloud Computing (Obligatorio) . . . 544
CS404 Taller de Investigación (Obligatorio) . . . 548
AI367 Tópicos en Inteligencia Artificial (Obligatorio) . . . 552
AI368 Computación Evolutiva (Obligatorio) . . . 559

6. Profesores & Cursos **563**
Cursos asignados por profesor . . . 563
Profesor asignado por curso . . . 563

7. Equivalencias con otros planes curriculares **567**
Equivalencia del Plan 2027 al 2018 . . . 567
Equivalencia del Plan 2018 al 2027 . . . 571

8. Laboratorios **575**

Índice de figuras

1.1. Campo acción de la Ciencia de la Computación 2

4.1. Esquema de codificación para los cursos. 203
4.2. Créditos por área por semestre 244
4.3. Distribución de cursos por áreas considerando creditaje. 245
4.4. Distribución de créditos por niveles de cursos. 245
4.5. DAISI 246
4.6. Comparación de CS-UNI con CS de ACM/IEEE-CS (Gráfico de curvas). 249
4.7. Comparación de CS-UNI con CS de ACM/IEEE-CS (Gráfico de araña). 249
4.8. Comparación de CS-UNI con DS de ACM/IEEE-CS (Gráfico de curvas). 250
4.9. Comparación de CS-UNI con DS de ACM/IEEE-CS (Gráfico de araña). 250
4.10. Comparación de CS-UNI con SE de ACM/IEEE-CS (Gráfico de curvas). 251
4.11. Comparación de CS-UNI con SE de ACM/IEEE-CS (Gráfico de araña). 251
4.12. Comparación de CS-UNI con CY de ACM/IEEE-CS (Gráfico de curvas). 252
4.13. Comparación de CS-UNI con CY de ACM/IEEE-CS (Gráfico de araña). 252
4.14. Comparación de CS-UNI con CE de ACM/IEEE-CS (Gráfico de curvas). 253
4.15. Comparación de CS-UNI con CE de ACM/IEEE-CS (Gráfico de araña). 253
4.16. Comparación de CS-UNI con IT de ACM/IEEE-CS (Gráfico de curvas). 254
4.17. Comparación de CS-UNI con IT de ACM/IEEE-CS (Gráfico de araña). 254
4.18. Comparación de CS-UNI con IS de ACM/IEEE-CS (Gráfico de curvas). 255
4.19. Comparación de CS-UNI con IS de ACM/IEEE-CS (Gráfico de araña). 255

5.1. Mapa de Conexión. BIC101 Introducción a la Computación 258
5.2. Mapa de Conexión. BMA101 Algebra Lineal 262
5.3. Mapa de Conexión. BMA102 Cálculo Diferencial 264
5.4. Mapa de Conexión. BCH101 Química I 267
5.5. Mapa de Conexión. BEI101 Inglés I 272
5.6. Mapa de Conexión. CS100 Introducción a la Ciencia de la Computación 276
5.7. Mapa de Conexión. CS112 Programación Orientada a Objetos I 287
5.8. Mapa de Conexión. CS1D1 Estructuras Discretas 292
5.9. Mapa de Conexión. BMA103 Cálculo Integral 295
5.10. Mapa de Conexión. BFI101 Física I 297
5.11. Mapa de Conexión. BEI102 Inglés II 302
5.12. Mapa de Conexión. CS113 Programación Orientada a Objetos II 306
5.13. Mapa de Conexión. CS2B1 Desarrollo Basado en Plataformas 312
5.14. Mapa de Conexión. AI161 IA Aplicada 316
5.15. Mapa de Conexión. BMA104 Cálculo Diferencial e Integral Avanzado 320
5.16. Mapa de Conexión. STA251 Estadística y Probabilidades 324
5.17. Mapa de Conexión. BEI201 Inglés III 329
5.18. Mapa de Conexión. CS210 Algoritmos y Estructuras de Datos 333
5.19. Mapa de Conexión. CS211 Teoría de la Computación 338
5.20. Mapa de Conexión. CS221 Arquitectura de Computadores 343
5.21. Mapa de Conexión. CS271 Bases de Datos I 348
5.22. Mapa de Conexión. CS401 Metodología de la Investigación 353

5.23. Mapa de Conexión. MA202 Métodos Numéricos . . . 357
5.24. Mapa de Conexión. BEI202 Inglés IV . . . 362
5.25. Mapa de Conexión. CS212 Análisis y Diseño de Algoritmos . . . 366
5.26. Mapa de Conexión. CS261 Inteligencia Artificial . . . 371
5.27. Mapa de Conexión. CS272 Bases de Datos II . . . 378
5.28. Mapa de Conexión. CS291 Ingeniería de Software I . . . 382
5.29. Mapa de Conexión. CS2S1 Sistemas Operativos . . . 388
5.30. Mapa de Conexión. FG106 Teatro . . . 393
5.31. Mapa de Conexión. BEI203 Inglés V . . . 397
5.32. Mapa de Conexión. CS231 Redes y Comunicación . . . 401
5.33. Mapa de Conexión. CS311 Algoritmos para Problemas Complejos . . . 406
5.34. Mapa de Conexión. CS312 Estructuras de Datos Avanzadas . . . 409
5.35. Mapa de Conexión. CS342 Compiladores . . . 412
5.36. Mapa de Conexión. AI263 Introducción al Aprendizaje de Máquina . . . 415
5.37. Mapa de Conexión. FI201 Física Computacional . . . 418
5.38. Mapa de Conexión. CS251 Computación Gráfica . . . 421
5.39. Mapa de Conexión. CS292 Ingeniería de Software II . . . 425
5.40. Mapa de Conexión. CS2H1 Experiencia de Usuario (UX) . . . 430
5.41. Mapa de Conexión. AI264 Aprendizaje Profundo . . . 435
5.42. Mapa de Conexión. FG211 Ética Profesional . . . 438
5.43. Mapa de Conexión. FG350 Liderazgo y Desempeño . . . 440
5.44. Mapa de Conexión. CS281 Computación en la Sociedad . . . 443
5.45. Mapa de Conexión. CS3I1 Seguridad en Computación . . . 449
5.46. Mapa de Conexión. CS3P1 Computación Paralela y Distribuída . . . 455
5.47. Mapa de Conexión. CS402 Proyecto de tesis 1 . . . 465
5.48. Mapa de Conexión. EX301 Actividades Extracurriculares . . . 469
5.49. Mapa de Conexión. ET201 Formación de Empresas de Base Tecnológica I . . . 471
5.50. Mapa de Conexión. AI268 Visión Computacional . . . 475
5.51. Mapa de Conexión. CS391 Ingeniería de Software III . . . 481
5.52. Mapa de Conexión. CS393 Sistemas de Información . . . 486
5.53. Mapa de Conexión. CS394 Arquitectura de Software y DevOps para Software en la Nube 489
5.54. Mapa de Conexión. CS395 Calidad de Software . . . 495
5.55. Mapa de Conexión. CS370 Big Data . . . 503
5.56. Mapa de Conexión. CS400 Prácticas Pre-profesionales . . . 507
5.57. Mapa de Conexión. CS403 Proyecto de tesis 2 . . . 509
5.58. Mapa de Conexión. AI365 Modelos Generativos Avanzados en IA . . . 513
5.59. Mapa de Conexión. CB309 Bioinformática . . . 515
5.60. Mapa de Conexión. ET301 Formación de Empresas de Base Tecnológica II . . . 519
5.61. Mapa de Conexión. AI369 Robótica . . . 522
5.62. Mapa de Conexión. CS351 Tópicos en Computación Gráfica . . . 526
5.63. Mapa de Conexión. CS353 Computación Cuántica . . . 530
5.64. Mapa de Conexión. CS392 Tópicos en Ingeniería de Software . . . 534
5.65. Mapa de Conexión. CS3P3 Internet de las Cosas . . . 541
5.66. Mapa de Conexión. CS3P2 Cloud Computing . . . 544
5.67. Mapa de Conexión. CS404 Taller de Investigación . . . 548
5.68. Mapa de Conexión. AI367 Tópicos en Inteligencia Artificial . . . 552
5.69. Mapa de Conexión. AI368 Computación Evolutiva . . . 559

Índice de cuadros

2.1. Lista de KUs del área de Inteligencia Artificial. 14
2.2. Lista de KUs del área de Fundamentos Algorítmicos. 27
2.3. Lista de KUs del área de Arquitectura y Organización. 33
2.4. Lista de KUs del área de Gestión de Datos. 40
2.5. Lista de KUs del área de Fundamentos de los Lenguajes de Programación. 47
2.6. Lista de KUs del área de Gráficos y Técnicas Interactivas. 62
2.7. Lista de KUs del área de Interacción Humano-Computadora. 66
2.8. Lista de KUs del área de Fundamentos Matemáticos y Estadísticos. 69
2.9. Lista de KUs del área de Redes y Comunicaciones. 72
2.10. Lista de KUs del área de Sistemas Operativos. 76
2.11. Lista de KUs del área de Computación Paralela y Distribuida. 85
2.12. Lista de KUs del área de Fundamentos de Desarrollo de Software. 94
2.13. Lista de KUs del área de Ingeniería de Software. 98
2.14. Lista de KUs del área de Seguridad. 110
2.15. Lista de KUs del área de Sociedad, ética y la Profesión. 120
2.16. Lista de KUs del área de Fundamentos de Sistemas. 134
2.17. Lista de KUs del área de Desarrollo de Plataformas Especializadas. 141

3.1. Lista de KUs del área de Cálculo y Análisis. 156
3.2. Lista de KUs del área de Álgebra y Teoría de Números. 161
3.3. Lista de KUs del área de Matemática Discreta y Combinatoria. 163
3.4. Lista de KUs del área de Probabilidad y Estadística. 165
3.5. Lista de KUs del área de Análisis Numérico y Científico. 167
3.6. Lista de KUs del área de Modelado Matemático y Simulación. 171
3.7. Lista de KUs del área de Fundamentos Matemáticos de la Ciencia de Datos. 173
3.8. Lista de KUs del área de Información Cuántica y Computación. 176
3.9. Lista de KUs del área de Electromagnetismo. 177
3.10. Lista de KUs del área de Termodinámica y Mecánica Estadística. 179
3.11. Lista de KUs del área de Física Computacional y Basada en Datos. 180
3.12. Lista de KUs del área de Mecánica Clásica. 182
3.13. Lista de KUs del área de Química General. 188
3.14. Lista de KUs del área de Química Computacional y Teórica. 192
3.15. Lista de KUs del área de Bioquímica y Biología Molecular. 194
3.16. Lista de KUs del área de Teoría de la Probabilidad. 196
3.17. Lista de KUs del área de Estadística Descriptiva y Análisis de Datos. 199
3.18. Lista de KUs del área de Inferencia Estadística. 199
3.19. Lista de KUs del área de Regresión y Modelos Estadísticos. 201

4.1. Tópicos de Ciencia de la Computación–*Computer Science* por curso del 1er al 2do Semestre (Parte 1 de 2) 211
4.2. Tópicos de Ciencia de la Computación–*Computer Science* por curso del 1er al 2do Semestre (Parte 2 de 2) 212
4.3. Tópicos de Ciencia de la Computación–*Computer Science* por curso del 3er al 4to Semestre (Parte 1 de 2) 213

4.4. Tópicos de Ciencia de la Computación–*Computer Science* por curso del 3er al 4to Semestre (Parte 2 de 2) . . . 214
4.5. Tópicos de Ciencia de la Computación–*Computer Science* por curso del 5to al 6to Semestre (Parte 1 de 3) . . . 215
4.6. Tópicos de Ciencia de la Computación–*Computer Science* por curso del 5to al 6to Semestre (Parte 2 de 3) . . . 217
4.7. Tópicos de Ciencia de la Computación–*Computer Science* por curso del 5to al 6to Semestre (Parte 3 de 3) . . . 218
4.8. Tópicos de Ciencia de la Computación–*Computer Science* por curso del 7mo Semestre 219
4.9. Tópicos de Ciencia de la Computación–*Computer Science* por curso del 8vo Semestre (Parte 1 de 3) . . . 220
4.10. Tópicos de Ciencia de la Computación–*Computer Science* por curso del 8vo Semestre (Parte 2 de 3) . . . 222
4.11. Tópicos de Ciencia de la Computación–*Computer Science* por curso del 8vo Semestre (Parte 3 de 3) . . . 223
4.12. Tópicos de Ciencia de la Computación–*Computer Science* por curso del 9no al 10mo Semestre (Parte 1 de 2) . . . 224
4.13. Tópicos de Ciencia de la Computación–*Computer Science* por curso del 9no al 10mo Semestre (Parte 2 de 2) . . . 226
4.14. Tópicos de Ciencias Básicas para Computación por curso del 1er al 2do Semestre (Parte 1 de 2) . . . 229
4.15. Tópicos de Ciencias Básicas para Computación por curso del 1er al 2do Semestre (Parte 2 de 2) . . . 230
4.16. Tópicos de Ciencias Básicas para Computación por curso del 3er al 4to Semestre (Parte 1 de 2) . . . 231
4.17. Tópicos de Ciencias Básicas para Computación por curso del 3er al 4to Semestre (Parte 2 de 2) . . . 232
4.18. Tópicos de Ciencias Básicas para Computación por curso del 5to al 6to Semestre . . . 232
4.19. Tópicos de Ciencias Básicas para Computación por curso del 7mo Semestre . . . 233
4.20. Tópicos de Ciencias Básicas para Computación por curso del 8vo Semestre . . . 233
4.21. Tópicos de Ciencias Básicas para Computación por curso del 9no al 10mo Semestre . . . 233
4.22. Resultados esperados por curso 1er al 2do Semestre . . . 234
4.23. Resultados esperados por curso 3er al 4to Semestre . . . 235
4.24. Resultados esperados por curso 5to al 6to Semestre . . . 236
4.25. Resultados esperados por curso 7mo Semestre . . . 237
4.26. Resultados esperados por curso 8vo Semestre . . . 238
4.27. Resultados esperados por curso 9no al 10mo Semestre . . . 239
4.28. Distribución de créditos por área por semestre . . . 244

Agradecimientos

Además de los autores directos de este documento, también deseamos dejar manifiesto de nuestro agradecimiento a otros colegas de diversas universidades del país y del mundo que gentilmente han aportado parte de su tiempo a darnos sus sugerencias. Entre ellos debemos mencionar a:

- Todo de equipo de ACM/ IEEE-CS Computing Curricula (CC2020, CS2023 y CS2013).
- Cuerpo docente de: CS-UCSP, CS-UNSA, CS-UTEC, CS-UNI.

De forma muy especial deseamos expresar nuestro agradecimiento al Patronato de la Universidad Nacional de Ingeniería (UNI) por su apoyo y auspicio en el desarrollo de esta propuesta de Malla Curricular, así como por su compromiso constante con la mejora de la formación académica de sus estudiantes.

Por otro lado, también deseamos agradecer a los miembros dela comunidad que hecho diversas correcciones que han permitido mejorar este documento. Entre ellos debemos mencionar a: **Luís Antonio Selis Vasquez** (UNI, Perú), **Piero Castillo** (UNI, Perú), **Jairo Andre Calagua Mallqui** (UNI, Perú), **Piero Oviedo Reyes** (UNI, Perú), **Yan Cesar Huallpa Torres** (UNI, Perú), quienes han participado activamente en la corrección de este documento.

Todo este equipo de trabajo asumió como premisa que el centro de nuestro esfuerzo, es la formación académica y humana de los estudiantes para formar agentes de cambio positivo y disruptivo en la sociedad.

A todos ellos deseamos agradecerles por su aporte que ha permitido generar este documento, único en su género en nuestro país, que servirá para sentar las bases de una carrera más sólida en esta fantástica área que nos ha tocado estudiar y de la cual nos sentimos orgullosos de formar parte: **Computación**.

Abreviaturas

ACM *Association for Computing Machinery*

AIS *Association for Information Systems*

CC Ciencia de la Computación

CS Ciencia de la Computación – *Computer Science*

IEEE-CS IEEE *Computer Society*

.

Capítulo 1

Introducción

La computación ha pasado por un desarrollo impresionante en las últimas décadas, convirtiéndose en el motor del desarrollo científico, tecnológico, industrial, social, económico y cultural, transformando de manera significativa nuestro diario accionar.

El surgimiento del computador ha marcado una nueva era en la historia de la humanidad que era imposible de imaginar varias décadas atrás. La gran cantidad de aplicaciones que se han desarrollado en los últimos años están transformando el desarrollo de todas las disciplinas del saber, la comercialización en el ámbito globalizado en que vivimos, la manera en que nos comunicamos, los procesos de enseñanza-aprendizaje y hasta en la manera como nos entretenemos.

Para darnos una idea de la relevancia e importancia, que en nuestro país ha alcanzado esta disciplina, basta mencionar que actualmente se ofrecen aproximadamente más de 100 carreras de Computación a nivel nacional. Esto sin considerar los programas de nivel Técnico Superior No Universitario que se ofertan.

Todas estas carreras existentes tienen como centro de su estudio a la computación pero lo hacen con más de 50 nombres distintos como: Ingeniería de Sistemas, Ingeniería de Computación, Ingeniería de Computación y Sistemas, entre otros. A pesar de que todas ellas apuntan al mismo mercado de trabajo, resulta por lo menos sorprendente que no sea posible encontrar por lo menos dos que compartan la misma curricula aunque sea en el primer año.

Comúnmente, durante la década de los setenta, la Computación se desarrolló dentro de las Facultades de Ciencias en la mayoría de las universidades estadounidenses, británicas y de otros países. Durante la década de los ochenta, los grupos de computación en las universidades se esforzaron por lograr una legitimidad académica en su ámbito local. Frecuentemente, se transformaron en departamentos de Matemáticas y Computación, hasta finalmente dividirse en dos departamentos de Matemáticas y de Computación, en la década de los noventa. Es en esta década en que un número creciente de instituciones reconocieron la influencia penetrante de la Computación, creando unidades independientes como departamentos, escuelas o institutos dedicados a tal área de estudio, un cambio que ha demostrado tanto perspicacia como previsión.

En Perú, un número cada vez mayor de instituciones de educación superior han tratado de seguir el desarrollo de las universidades extranjeras (aunque no siempre en forma muy seria o exitosa), reconociendo a la Computación como un área de estudio en sí misma, así como su importancia estratégica en la educación, y creando departamentos, escuelas o institutos dedicados a su estudio. La Facultad de Ciencias no puede ser la excepción a este cambio, en el que ya se tiene un retraso relativo con muchas de las instituciones educativas dentro y fuera del país.

1.1. Definiciones básicas

La referencia más sólida a nivel mundial en cuanto a la propuesta de carreras de computación para nivel de pregrado es la que fue propuesta en conjunto por la *Association for Computing Machinery* (ACM), IEEE *Computer Society* (IEEE-CS) y la *Association for Information Systems* (AIS). Estas tres organizaciones propusieron la Computing Curricula en el documento denominado: *Joint Task Force for Computing Curricula 2005, Computing Curricula 2005. Overview Report*[Shakelford et al., 2005] cuya version mas reciente es la denominada *Computing Curricula 2020*[ACM/IEEE-CS, 2020].

La Ciencia de la Computación es un término de origen estadounidense Ciencia de la Computación – *Computer Science* (CS). Este término es conocido también como informática en el ámbito europeo[1].

Según el diccionario de la Real Academia de la Lengua Española (http://www.rae.es) ambos términos también son sinónimos.

A nivel internacional, la computación presenta siete perfiles claramente definidos:

- Ciencia de la Computación (*Computer Science*) [ACM/IEEE-CS/AAAI Joint Task Force, 2023],
- Ingeniería de Computación (*Computer Engineering*) [Soldan et al., 2016],
- Sistemas de Información (*Information Systems*) [The Joint ACM/AIS IS2020 Task Force, 2020],
- Ingeniería de Software (*Software Engineering*) [Díaz-Herrera and Hilburn, 2004],
- Tecnología de la Información (*Information Technology*) [ACM and IEEE-CS, 2017],
- Ciencia de Datos (*Data Science*) [ACM Data Science Task Force, 2021],
- Ciberseguridad (*Cybersecurity*) [ACM/IEEE-CS Joint Task Force on Cybersecurity Education, 2017].

La Figura 1.1 es tomada de la definición propuesta en la *Computing Curricula*[Shakelford et al., 2005] en el área de Ciencia de la Computación (CC) La CC cubre la mayor parte entre el extremo superior y el extremo inferior, porque el profesional en CC no trata "solamente con el hardware" que utiliza un software o de "solamente la organización" que hace uso de la información que la computación le puede proveer.

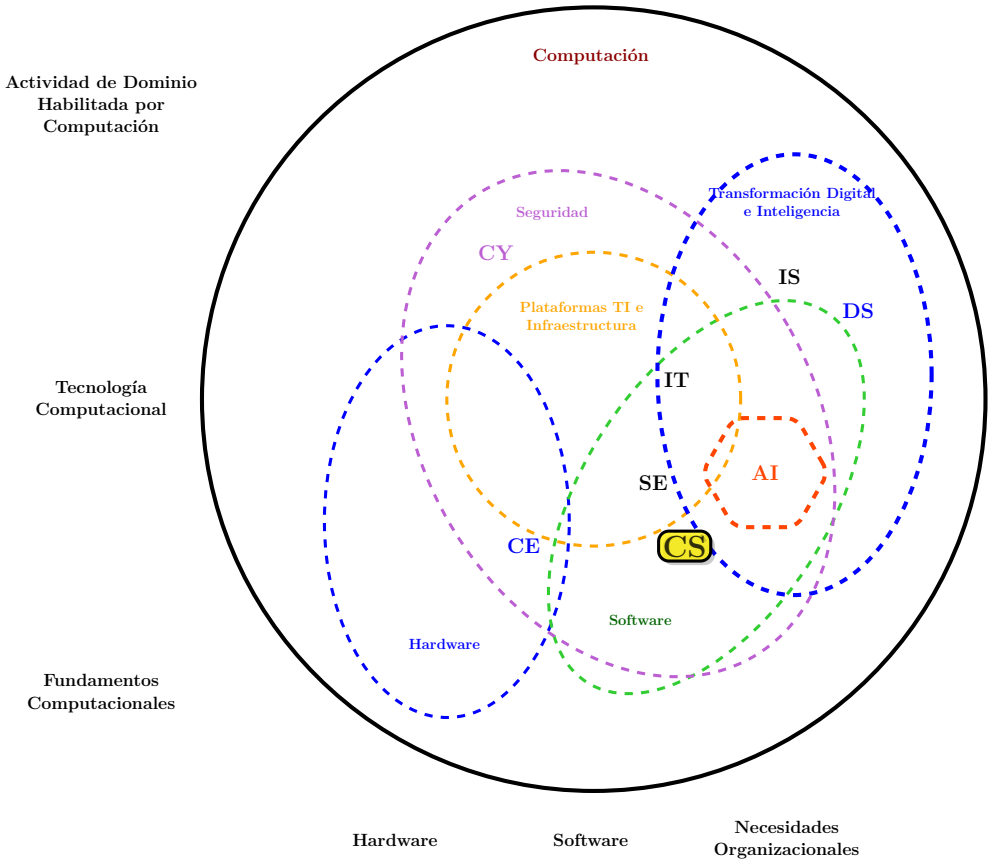

Figura 1.1: Campo acción de la Ciencia de la Computación

Ciencia de la Computación cubren un amplio rango, desde sus fundamentos teóricos y algorítmicos hasta los últimos desarrollos en robótica, visión por computadora, Inteligencia Artificial, bioinformática, y otras áreas emocionantes. Podemos pensar que el trabajo de un científico de la computación pertenece a las siguientes tres categorías:

- **Diseño e implementación de software**. Los científicos de computación se encargan de desafiantes labores de programación. También supervisan otros programadores, haciéndolos concientes de nuevas aproximaciones.

[1] El término europeo es derivado del vocablo francés *Informatique*.

- **Instrumentación de nuevas formas para usar computadoras**. El progreso en las áreas de ciencias de la computación como redes, bases de datos, e interfaces humano-computadora permitieron el desarrollo de la www y actualmente se trabaja en el desarrollo de metasistemas Grid. Además, los investigadores trabajan ahora en hacer que los robots sean ayudantes prácticos y demuestren inteligencia, utilizan las bases de datos para crear nuevos conocimientos, y están utilizando computadoras para decifrar los secretos de nuestro ADN.
- **Desarrollo de formas efectivas de resolver problemas de computación.** Por ejemplo, los científicos de la computación desarrollan las mejores formas posibles de almacenar información en bases de datos, enviar datos a través de la red, y desplegar imágenes complejas. Sus bases teóricas les permiten determinar el mejor desempeño posible, y su estudio de algoritmos les ayuda a desarrollar nuevas aproximaciones para proveer un mejor desempeño.

La Ciencia de la Computación cubren todo el rango desde la teoría hasta la programación. Mientras otras disciplinas pueden producir titulados mejor preparados para trabajos específicos, las ciencias de la computación ofrecen un amplio fundamento que permite a sus titulados adaptarse a nuevas tecnologías y nuevas ideas.

El profesional en CC se preocupa por casi todo en medio de estas áreas. En dirección hacia el hardware, este profesional llega a desarrollar software que permite el funcionamiento de dispositivos *devices*. En dirección a aspectos organizacionales, el profesional de CC ayuda a que los sistemas de información operen correctamente en las organizaciones. Él genera la tecnología que permite que otras áreas como los sistemas de información se desarrollen adecuadamente.

El profesional en CC diseña y desarrolla todo tipo de software, desde infraestructura de plataformas (sistemas operativos, programas de comunicación, etc.) hasta aplicación de tecnologías (navegadores de Internet, bases de datos, motores de búsqueda, etc.). Este profesional crea estas capacidades, pero no está orientado al uso de las mismas. Por lo tanto, el área sombreada (fig. 1.1) para CC se estrecha y finaliza en la medida que nos movamos hacia la aplicación y configuración de productos.

1.2. Campo y mercado ocupacional

Nuestro egresado podrá prestar sus servicios profesionales en empresas e instituciones públicas y privadas que requieran sus capacidades en función del desarrollo que oferta, entre ellas:

- Empresas dedicadas a la producción de software con calidad internacional.
- Empresas, instituciones y organizaciones que requieran software de calidad para mejorar sus actividades y/o servicios ofertados.

Nuestro egresado puede desempeñarse en el mercado laboral sin ningún problema ya que, en general, la exigencia del mercado y campo ocupacional está mucho más orientada al uso de herramientas. Sin embargo, es poco común que los propios profesionales de esta carrera se pregunten: ¿qué tipo de formación debería tener si yo quisiera crear esas herramientas además de saber usarlas?. Ambos perfiles (usuario y creador) son bastante diferentes pues no sería posible usar algo que todavía no fue creado. En otras palabras, los creadores de tecnología son los que dan origen a nuevos puestos de trabajo y abren la posibilidad de que otros puedan usar esa tecnología.

Debido a la formación basada en la investigación, nuestro profesional debe siempre ser un innovador donde trabaje. Esta misma formación permite que el egresado piense también en crear su propia empresa de desarrollo de software. Considerando que países en vias de desarrollo tienen un costo de vida mucho menor que Norte América ó Europa, una posibilidad que se muestra interesante es la exportación de software pero eso requiere que la calidad del producto sea al mismo nivel de lo ofrecido a nivel internacional.

Este perfil profesional también posibilita que los egresados se queden en el país; producir software en nuestro país y venderlo fuera es más rentable que salir al extranjero y comercializarlo allá.

El campo ocupacional de un egresado es amplio y está en continua expansión y cambio. Prácticamente toda empresa u organización hace uso de servicios de computación de algún tipo, y la buena formación básica de nuestros egresados hace que puedan responder a los requerimientos de las mismas exitosamente. Este egresado, no sólo podrá dar soluciones a los problemas existentes sino que deberá proponer innovaciones tecnológicas que impulsen la empresa hacia un progreso constante.

A medida que la informatización básica de las empresas del país avanza, la necesidad de personas capacitadas para resolver los problemas de mayor complejidad aumenta y el plan de estudios que

hemos desarrollado tiene como objetivo satisfacer esta demanda considerandola a mediano y largo plazo. El campo para las tareas de investigación y desarrollo de problemas complejos en computación es también muy amplio y está creciendo día a día a nivel mundial.

Debido a la capacidad innovadora de nuestro egresado, existe una mayor la probabilidad de registrar patentes con un alto nivel inventivo lo cual es especialmente importante en nuestros países.

1.3. Importancia de la carrera en la sociedad

Uno de los caminos que se espera que siga un profesional del área de Ciencia de la Computación es que el se dedique a producir software o que se integre a las empresas productoras de software. En el ámbito de la computación, es común observar que los países cuentas con Asociaciones de Productores de Software cuyas políticas están orientadas a la exportación. Siendo así, no tendría sentido preparar a nuestros alumnos sólo para el mercado local o nacional. Nuestros egresados deben estar preparados para desenvolverse en el mundo globalizado que nos ha tocado vivir.

Nuestros futuros profesionales deben estar orientados a crear nuevas empresas de base tecnológica que puedan incrementar las exportaciones de software peruano. Este nuevo perfil está orientado a generar industria innovadora. Si nosotros somos capaces de exportar software competitivo también estaremos en condiciones de atraer nuevas inversiones. Las nuevas inversiones generarían más puestos de empleo bien remunerados y con un costo bajo en relación a otros tipos de industria. Bajo esta perspectiva, podemos afirmar que esta carrera será un motor que impulsará al desarrollo del país de forma decisiva con una inversión muy baja en relación a otros campos.

Es necesario recordar que la mayor innovación de productos comerciales de versiones recientes utiliza tecnología que se conocía en el mundo académico hace varios años, décadas o más. Un ejemplo claro son las bases de datos que soportan datos y consultas espaciales desde hace muy pocos años. Sin embargo, utilizan estructuras de datos que ya existían hace algunas décadas. Es lógico pensar que la gente del área académica no se dedique a estudiar en profundidad la última versión de un determinado software cuando esa tecnología ya la conocían hace mucho tiempo. Por esa misma razón es raro en el mundo observar que una universidad tenga convenios con una transnacional de software para dictar solamente esa tecnología pues, nuestra función es generar esa tecnología **y no sólo saber usarla**.

Tampoco debemos olvidar que los alumnos que ingresan hoy saldrán al mercado dentro de 5 años aproximadamente y, en un mundo que cambia tan rápido, no podemos ni debemos enseñarles tomando en cuenta solamente el mercado actual. Nuestros profesionales deben estar preparados para resolver los problemas que habrá dentro de 10 o 15 años y eso sólo es posible a través de la investigación.

1.4. Información institutional

1.4.1. Misión

Por lo antes mencionado, nuestra misión es:

Contribuir al desarrollo científico, tecnológico y técnico del país formando profesionales competentes, orientados a la creación de nueva ciencia y tecnología computacional, como motor que impulse y consolide la industria del software en base a la investigación científica y tecnológica en áreas innovadoras formando, EN NUESTROS profesionales, un conjunto de habilidades y destrezas para la solución de problemas computacionales con un compromiso social.

1.4.2. Visión

- Queremos ser una carrera profesional acreditada con estándares internacionales que cuente con el reconocimiento en función de la calidad y competitividad de sus docentes y egresados.
- Queremos ser una carrera que trascienda por la relevancia y pertinencia de sus proyectos de investigación básica y aplicada.
- Buscamos ser una carrera que promueva el desarrollo de la industria del software relacionada a Ciencia de la Computación a nivel internacional, incorporando a sus egresados a la industria ya establecida o generando nuevas empresas en este mismo rubro.

- Queremos ser una carrera que comparta y difunda el conocimiento con todos los sectores de la población y contribuya a la solución de los problemas estratégicos de nuestra sociedad.

1.5. Información de acreditacion

1.5.1. Resultados de la carrera *Outcomes*

Al finalizar esta carrera, el(la) egresado(a), habrá logrado los siguientes resultados (*Outcomes*):

AG-C01) El Profesional y el Mundo: Analiza y evalúa el impacto de las soluciones a problemas complejos de computación en el desarrollo sostenible de la sociedad.

AG-C02) Ética: Aplica principios éticos y se compromete con la ética profesional y las normas de la práctica profesional de la computación.

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos.

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación.

AG-C05) Gestión de Proyectos: Aplica los principios de gestión en computación para gestionar proyectos.

AG-C06) Aprendizaje a lo largo de la vida: Reconoce la importancia del aprendizaje continuo y la adaptación a nuevas tecnologías.

AG-C07) Conocimientos de Computación: Aplica conocimientos de matemáticas, ciencias y computación.

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación.

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación.

AG-C10) Indagación: Estudia problemas complejos de computación usando métodos de ciencias de la información.

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas.

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora.

Las competencias y/o resultados propuestas en esta malla curricular corresponden a las competencias y/o resultados propuestas por Instituto de Acreditación de Carreras de Computación, Ingeniería y Tecnología (ICACIT).

1.5.2. Objetivos educacionales

Después de cinco años de egresado de la carrera profesional de Ciencia de la Computación, nuestros profesionales deben ser capaces de:

1. Cumplir y superar las expectativas de trabajo definidas por el entorno laboral.
2. Desempeñarse como miembro, o líder, de un equipo de trabajo tanto especializado como multidisciplinario.
3. Proponer soluciones al contexto laboral, donde se desenvuelve, basadas en la implementación o mejora del estado del arte en Ciencia de la Computación.
4. Comunicar de forma efectiva propuestas tecnológicas a personas de distintos niveles de conocimiento y de diferentes ámbitos sociales.

5. Actualizarse y adaptarse a nuevos conocimientos en Ciencia de la Computación y a diferentes ámbitos laborales, de forma autónoma o mediante estudios complementarios.
6. Demostrar un claro entendimiento de las consecuencias que surgen a partir de creaciones tecnológicas relacionadas a la Ciencia de la Computación en aspectos tales como lo social, ético, humano, moral, legal, ambiental, económico, entre otros.

1.5.3. Aprendizaje esperado (*Learning Outcomes*)

Cada KU dentro de un KA enumera tanto un conjunto de temas y los resultados de aprendizaje (*Learning Outcomes*) que los estudiantes deben alcanzar en lo que respecta a los temas especificados. Resultados de aprendizaje no son de igual tamaño y no tienen una asignación uniforme de horas curriculares; temas con el mismo número de horas pueden tener muy diferentes números de los resultados del aprendizaje asociados.

Cada resultado de aprendizaje tiene un nivel asociado de dominio. En la definición de los diferentes niveles que dibujamos de otros enfoques curriculares, especialmente la taxonomía de Bloom, que ha sido bien explorada dentro de la Ciencia de la Computación. En este documento no se aplicó directamente los niveles de Bloom en parte porque varios de ellos son impulsados por contexto pedagógico, que introduciría demasiada pluralidad en un documento de este tipo; en parte porque tenemos la intención de los niveles de dominio para ser indicativa y no imponer restricción teórica sobre los usuarios de este documento.

Nosotros usamos tres niveles de dominio esperados que son:

Nivel 1 Familiarizarse(*Familiarity*) : El estudiante **comprende** un concepto básicamente. Responde: **¿Qué sabe sobre esto?**

Nivel 2 Usar(*Usage*) : El estudiante **aplica** conceptos en situaciones prácticas (ej: programación). Responde: **¿Cómo lo haría?**

Nivel 3 Evaluar(*Assessment*) : El estudiante **evalúa y justifica** enfoques. Responde: **¿Por qué este método?**

Por ejemplo, para evaluar los niveles de dominio, consideremos la noción de iteración en el desarrollo de software (for, while e iteradores). En el plano de la "familiaridad", se espera que un estudiante tenga una definición del concepto de iteración en el desarrollo de software y saber por qué esta técnica es útil.

Con el fin de mostrar el dominio del nivel "Uso", el estudiante debe ser capaz de escribir un programa adecuadamente usando una forma de iteración.

En el nivel de "Evaluación", en la iteración se requeriría que un estudiante comprenda múltiples métodos de iteración y que sea capaz de seleccionar apropiadamente entre ellos para diferentes aplicaciones.

1.6. Perfiles

1.6.1. Perfil del estudiante

El aspirante a ingresar a la Escuela Profesional de Ciencia de la Computación de la Universidad Nacional de Ingeniería (UNI) debe tener:
Conocimientos de:

1. Conceptos fundamentales de álgebra, geometría, trigonometría y análisis matemático elemental.
2. Nociones básicas de lógica proposicional y razonamiento deductivo.
3. Principios elementales de algoritmos y secuencias lógicas para la resolución de problemas.
4. Nociones de probabilidad y estadística descriptiva para el análisis de datos.

Habilidades para:

1. Pensar de forma abstracta, identificando patrones, estructuras y relaciones más allá de lo concreto.
2. Modelar problemas del mundo real utilizando representaciones simbólicas y formales.
3. Descomponer problemas complejos en subproblemas más simples y manejables.

4. Aprender de manera autónoma y autorregulada, adquiriendo nuevas tecnologías y paradigmas sin necesidad de instrucción formal continua.
5. Adaptarse rápidamente a entornos tecnológicos cambiantes y a la evolución vertiginosa de herramientas y lenguajes.
6. Comunicar ideas técnicas y soluciones de forma clara, tanto oralmente como por escrito, utilizando terminología apropiada.
7. Mantener la concentración en tareas analíticas que demanden esfuerzo sostenido y atención al detalle.
8. Sintetizar información de múltiples fuentes y extraer conceptos esenciales para la toma de decisiones.
9. Evaluar críticamente soluciones existentes y proponer mejoras o alternativas innovadoras.

Actitudes de:

1. Interés genuino y curiosidad por la tecnología, la programación y el funcionamiento de los sistemas computacionales.
2. Disposición para el trabajo colaborativo en equipos de desarrollo y proyectos multidisciplinarios.
3. Iniciativa para explorar problemas más allá del currículo y perseverancia ante desafíos intelectuales como la depuración de código o la optimización.
4. Mentalidad de crecimiento y apertura para aprender continuamente en un campo que evoluciona permanentemente.
5. Compromiso con la calidad, la eficiencia y las buenas prácticas en el desarrollo de soluciones.

1.6.2. Perfil del graduado

El graduado del Programa Profesional de Ciencia de la Computación es capaz de:

1. Diseñar, desarrollar e implementar soluciones de software de calidad internacional que atiendan necesidades complejas de la industria, la sociedad y la investigación.
2. Aportar al desarrollo tecnológico del país mediante la creación de software innovador, seguro y escalable, considerando estándares globales y mejores prácticas de la industria (ISO, IEEE, CMMI).
3. Transformar procesos organizacionales y modelos de negocio a través de la aplicación estratégica de tecnologías computacionales emergentes (IA, computación en la nube, ciberseguridad, IoT).
4. Adaptarse con agilidad a la rápida evolución del campo, gracias a una formación fundamentada en principios científicos sólidos y capacidades de autoaprendizaje continuo.
5. Liderar equipos multidisciplinarios y distribuidos globalmente en proyectos de desarrollo tecnológico, aplicando metodologías ágiles (Scrum, DevOps) y comunicación efectiva en entornos bilingües.
6. Generar ventajas competitivas sostenibles para organizaciones mediante la arquitectura de sistemas robustos, el análisis de datos masivos y la automatización inteligente de procesos.
7. Fundar y hacer crecer empresas de base tecnológica (*startups*) con potencial para competir e innovar en mercados globales.
8. Acceder y sobresalir en programas de posgrado de primer nivel (maestrías y doctorados) en universidades de prestigio internacional, sustentado por una rigurosa formación en ciencias de la computación.
9. Contribuir al avance del conocimiento en el área mediante investigación aplicada, publicación en conferencias y revistas indexadas, y desarrollo de tecnologías con potencial de patente.

1.6.3. Perfil Profesional

El profesional en Ciencia de la Computación tiene las siguientes características:

- Desarrollar tecnología computacional buscando el bien común de los individuos, la sociedad y las organizaciones.
- Transformar, acelerar y ampliar los límites de cualquier área del conocimiento a través de soluciones innovadoras basadas en el uso eficiente de tecnología computacional.
- Adaptarse rápidamente a los cambios tecnológicos debido a su formación basada en la investigación constante.

- Trabajar y liderar equipos multidisciplinarios que llevan a cabo proyectos de innovación tecnológica.
- Incrementar las ventajas competitivas de cualquier organización a través del uso eficiente de tecnología computacional gracias a su alta capacidad de abstracción.
- Crear empresas de base tecnológica.
- Poder seguir estudios de postgrado con exigencia internacional en áreas relacionadas.

1.6.4. Perfil del docente

El docente de la carrera de Ciencia de la Computación es una persona que:

- Promueve la formación integral del estudiante mediante la enseñanza, investigación y extensión, transmitiendo conocimientos sólidos con claridad y simplicidad.
- Fomenta la investigación y el aprendizaje continuo para que los estudiantes se adapten a los avances del área y busquen el bien común de individuos, sociedad y organizaciones.
- Mantiene formación académica avanzada de calidad, con estudios a tiempo completo, buscando la excelencia académica a nivel internacional.
- Publica investigaciones en eventos internacionales y revistas indexadas de alto impacto, contribuyendo al avance del conocimiento y manteniéndose actualizado.
- Establece redes de cooperación con instituciones de nivel internacional, enriqueciendo la experiencia educativa.
- Participa activamente en la formación e integración del estudiante a la comunidad académica , promoviendo la exigencia continua y rigor académico.

1.7. Grados y Títulos

Cumplida la aprobación de los créditos académicos propuestos en la malla curricular y luego de haber cumplido con los requisitos estipulados en el reglamento de grados y títulos de la carrera, el egresado recibirá:

Grado Académico: Bachiller en Ciencia de la Computación y

Titulo Profesional: Licenciado en Ciencia de la Computación

1.8. Recursos para dictado de clases

Un profesional innovador debe estar al tanto de los últimos avances de su área siempre. Los últimos avances de esta área no son presentados en los libros necesariamente. Debemos utilizar publicaciones de revistas indexadas de circulación mundial.

Por esa razón, tomamos como base la suscripción institucional a la ACM y a la IEEE-CS. Es recomendado que el docente use este material para discutir en clase las tendencias en todas las áreas.

Los laboratorios de cómputo también deben ser renovados de acuerdo a la rapidez propia de esta área y esto implica renovaciones constantes para poder garantizar que los estudiantes estén actualizados.

El recurso físico en cuanto a laboratorios para cada curso está detallado en el Capítulo 8.

1.9. Organización del documento

El resto de este documento está organizado de la siguiente forma: el Capítulo 1, define y explica el campo de acción de la Ciencia de la Computación, además se hace una muy breve explicación de las distintas carreras relacionadas a esta disciplina. En el caso particular del area de computación, las propuestas de ACM/IEEE-CS *Computing Curricula* [ACM/IEEE-CS, 2020].

En el Capítulo 2, presentamos las áreas de Conocimiento de Ciencia de la Computación–*Computer Science*, indicando los tópicos y objetivos de aprendizaje de cada temas,

En el Capítulo 3, presentamos las áreas de Conocimiento de Ciencias Básicas para Computación, indicando los tópicos y objetivos de aprendizaje de cada temas,

El Capítulo 4 contiene la distribución por semestres, por áreas, por niveles, visión gráfica de la malla curricular, comparación con las diversas propuestas internacionales, distribución de tópicos por curso así como la distribución de habilidades por materia.

El Capítulo 5 contiene información detallada para cada uno de los cursos incluyendo las habilidades con las cuales contribuye, bibliografía por cada unidad así como el número de horas mínimas por cada unidad.

En el Capítulo 6 podemos observar un listado de cursos por profesor y de profesor por curso.

En el Capítulo 7 se presentan las tablas de equivalencias con otros planes curriculares.

En el Capítulo 8 se presenta una sugerencia de los laboratorios requeridos para el dictado de clases las mismas que podrían variar de acuerdo al volumen de alumnos que se tenga.

Capítulo 2

Cuerpo del conocimiento de Ciencia de la Computación–*Computer Science*

Las 17 áreas de conocimiento en Ciencia de la Computación–*Computer Science* son:

2.1 Inteligencia Artificial (AI) (Pág. 13)
- 2.1.1 Introducción a la IA: Conceptos y Comportamiento Inteligente (Pág. 15)
- 2.1.2 Características del Problema y Naturaleza de los Agentes (Pág. 15)
- 2.1.3 Estado y Búsqueda No Informada (Pág. 15)
- 2.1.4 Búsqueda Heurística y CSP (Pág. 16)
- 2.1.5 Búsqueda Adversarial y Optimización Estocástica (Pág. 16)
- 2.1.6 Representación del Conocimiento y Razonamiento Probabilístico Básico (Pág. 17)
- 2.1.7 Independencia, Modelos de Markov y Toma de Decisiones (Pág. 17)
- 2.1.8 Fundamentos de Aprendizaje Automático (Pág. 17)
- 2.1.9 Aprendizaje Automático: Datos, Representaciones y Evaluación (Pág. 18)
- 2.1.10 Redes Neuronales y Ética (Pág. 19)
- 2.1.11 Formulación del Aprendizaje Automático y Evaluación (Pág. 19)
- 2.1.12 Aprendizaje Profundo, RL y Ética (Pág. 19)
- 2.1.13 Entrenamiento y Optimización de Redes Neuronales (Pág. 20)
- 2.1.14 Técnicas de Regularización para Redes Profundas (Pág. 20)
- 2.1.15 Arquitecturas de Redes Convolucionales y Residuales (Pág. 20)
- 2.1.16 Arquitecturas Transformer y Modelos Generativos (Pág. 20)
- 2.1.17 Aplicaciones e Impacto Social de la IA (Pág. 21)
- 2.1.18 Aplicaciones Adicionales e Impacto Social de la IA (Pág. 21)
- 2.1.19 Representación Lógica y Razonamiento (Pág. 22)
- 2.1.20 Representación Probabilística y Razonamiento (Pág. 22)
- 2.1.21 Planificación (Pág. 23)
- 2.1.22 Agentes y Sistemas Cognitivos (Pág. 23)
- 2.1.23 Procesamiento del Lenguaje Natural (Pág. 24)
- 2.1.24 Robótica (Pág. 25)
- 2.1.25 Percepción y Visión por Computadora (Pág. 25)

2.2 Fundamentos Algorítmicos (AL) (Pág. 26)
- 2.2.1 Estructuras de Datos Fundamentales (Pág. 27)
- 2.2.2 Algoritmos Fundamentales (Pág. 27)
- 2.2.3 Algoritmos Avanzados (Pág. 28)
- 2.2.4 Estrategias Algorítmicas (Pág. 28)
- 2.2.5 Marco de Análisis de Complejidad (Pág. 29)
- 2.2.6 Notación Asintótica y Clases de Complejidad (Pág. 30)
- 2.2.7 Análisis de Complejidad: Recursión, Amortización y Cotas Ajustadas (Pág. 30)
- 2.2.8 Teoría de la Complejidad Computacional (Pág. 30)
- 2.2.9 Lenguajes Formales y Autómatas (Pág. 31)
- 2.2.10 Computabilidad y Decidibilidad (Pág. 31)
- 2.2.11 Sociedad, ética y la Profesión (Pág. 32)

2.3 Arquitectura y Organización (AR) (Pág. 33)
- 2.3.1 Lógica Digital y Sistemas Digitales (Pág. 33)
- 2.3.2 Representación de Datos a Nivel de Máquina (Pág. 34)
- 2.3.3 Organización de Máquina a Nivel de Ensamblador (Pág. 34)
- 2.3.4 Jerarquía de Memoria (Pág. 35)
- 2.3.5 Interconexión y Comunicación (Pág. 35)
- 2.3.6 Organización Funcional (Pág. 36)
- 2.3.7 Rendimiento y Eficiencia Energética (Pág. 36)
- 2.3.8 Arquitecturas Heterogéneas (Pág. 37)
- 2.3.9 Arquitecturas de Procesador Seguras (Pág. 37)
- 2.3.10 Arquitecturas Cuánticas (Pág. 38)
- 2.3.11 Problemas de Sostenibilidad (Pág. 38)

2.4 Gestión de Datos (DM) (Pág. 39)
- 2.4.1 El Rol de los Datos y el Ciclo de Vida de los Datos (Pág. 40)
- 2.4.2 Conceptos Fundamentales de Sistemas de Bases de Datos (Pág. 40)
- 2.4.3 Modelado de Datos (Pág. 41)
- 2.4.4 Bases de Datos Relacionales (Pág. 42)
- 2.4.5 Construcción de Consultas (Pág. 42)
- 2.4.6 Procesamiento de Consultas (Pág. 43)
- 2.4.7 Internos de DBMS (Pág. 43)
- 2.4.8 Sistemas NoSQL (Pág. 44)
- 2.4.9 Seguridad y Privacidad de Datos (Pág. 44)
- 2.4.10 Análisis de Datos (Pág. 45)
- 2.4.11 Bases de Datos Distribuidas/Computación en la Nube (Pág. 45)
- 2.4.12 Bases de Datos Semiestructuradas y No Estructuradas (Pág. 45)
- 2.4.13 Sociedad, ética y Profesión (Pág. 46)

2.5 Fundamentos de los Lenguajes de Programación (FPL) (Pág. 46)
- 2.5.1 Fundamentos de Programación Orientada a Objetos (Pág. 47)
- 2.5.2 Programación Orientada a Objetos: Encapsulación, Subtipado y Reflexión (Pág. 48)
- 2.5.3 Programación Funcional (Pág. 48)
- 2.5.4 Programación Lógica (Pág. 49)
- 2.5.5 Scripting de Shell (Pág. 50)
- 2.5.6 Programación Dirigida por Eventos y Reactiva (Pág. 50)
- 2.5.7 Computación Paralela y Distribuida (Pág. 51)
- 2.5.8 Programación Orientada a Aspectos (Pág. 52)
- 2.5.9 Fundamentos de Sistemas de Tipos (Pág. 52)
- 2.5.10 Sistemas de Tipos: Polimorfismo y Tipado Complementario (Pág. 53)
- 2.5.11 Teoría de Tipos Avanzada (Pág. 53)
- 2.5.12 Modelo de Ejecución y Memoria de Sistemas (Pág. 54)
- 2.5.13 Gestión de Memoria (Pág. 54)
- 2.5.14 Traducción y Ejecución de Lenguajes (Pág. 54)
- 2.5.15 Abstracción y Representación de Programas

(Pág. 55)
- 2.5.16 Análisis Sintáctico (Pág. 56)
- 2.5.17 Análisis Semántico del Compilador (Pág. 56)
- 2.5.18 Análisis de Programas y Analizadores (Pág. 57)
- 2.5.19 Generación de Código (Pág. 57)
- 2.5.20 Comportamiento en Tiempo de Ejecución y Sistemas (Pág. 58)
- 2.5.21 Construcciones de Programación Avanzadas (Pág. 59)
- 2.5.22 Pragmática del Lenguaje (Pág. 59)
- 2.5.23 Semántica Formal (Pág. 60)
- 2.5.24 Metodologías de Desarrollo Formal (Pág. 60)
- 2.5.25 Principios de Diseño de Lenguajes de Programación (Pág. 61)
- 2.5.26 Sociedad, ética y Profesión (Pág. 61)

2.6 Gráficos y Técnicas Interactivas (GIT) (Pág. 61)
- 2.6.1 Conceptos Fundamentales de Gráficos y Técnicas Interactivas (Pág. 62)
- 2.6.2 Visualización (Pág. 62)
- 2.6.3 Renderizado Aplicado y Técnicas (Pág. 63)
- 2.6.4 Modelado Geométrico (Pág. 63)
- 2.6.5 Sombreado y Renderizado Avanzado (Pág. 63)
- 2.6.6 Animación por Computadora (Pág. 64)
- 2.6.7 Simulación (Pág. 64)
- 2.6.8 Inmersión (Pág. 64)
- 2.6.9 Interacción (Pág. 64)
- 2.6.10 Procesamiento de Imágenes (Pág. 65)
- 2.6.11 Computación Tangible/Física (Pág. 65)
- 2.6.12 Sociedad, ética y Profesión (Pág. 65)

2.7 Interacción Humano-Computadora (HCI) (Pág. 66)
- 2.7.1 Comprender al Usuario: Objetivos individuales e interacciones con otros (Pág. 66)
- 2.7.2 Responsabilidad y Rendición de Cuentas en el Diseño (Pág. 67)
- 2.7.3 Accesibilidad y Diseño Inclusivo (Pág. 67)
- 2.7.4 Evaluación del Diseño (Pág. 67)
- 2.7.5 Diseño de Sistemas (Pág. 68)
- 2.7.6 Sociedad, ética y Profesión (Pág. 68)

2.8 Fundamentos Matemáticos y Estadísticos (MSF) (Pág. 68)
- 2.8.1 Matemáticas Discretas (Pág. 69)
- 2.8.2 Matemáticas Discretas Avanzadas (Pág. 69)
- 2.8.3 Probabilidad (Pág. 70)
- 2.8.4 Estadística (Pág. 70)
- 2.8.5 álgebra Lineal (Pág. 71)
- 2.8.6 Cálculo (Pág. 71)

2.9 Redes y Comunicaciones (NC) (Pág. 72)
- 2.9.1 Fundamentos de Redes y Comunicaciones (Pág. 72)
- 2.9.2 Aplicaciones en Red (Pág. 73)
- 2.9.3 Soporte de Fiabilidad (Pág. 73)
- 2.9.4 Enrutamiento y Reenvío (Pág. 74)
- 2.9.5 Comunicación de un solo salto (Pág. 74)
- 2.9.6 Seguridad de Red (Pág. 74)
- 2.9.7 Soporte de Movilidad (Pág. 75)
- 2.9.8 Temas Emergentes (Pág. 75)

2.10 Sistemas Operativos (OS) (Pág. 75)
- 2.10.1 Rol y Propósito de los Sistemas Operativos (Pág. 76)
- 2.10.2 Principios del Sistema Operativo (Pág. 77)
- 2.10.3 Concurrencia (Pág. 77)
- 2.10.4 Protección y Seguridad (Pág. 78)
- 2.10.5 Planificación (Pág. 79)
- 2.10.6 Modelo de Proceso (Pág. 79)
- 2.10.7 Gestión de Memoria (Pág. 80)
- 2.10.8 Gestión de Dispositivos (Pág. 80)
- 2.10.9 API de Sistemas de Archivos e Implementación (Pág. 81)
- 2.10.10 Sistemas de Archivos Avanzados (Pág. 81)
- 2.10.11 Virtualización (Pág. 82)
- 2.10.12 Sistemas en Tiempo Real y Embebidos (Pág. 83)
- 2.10.13 Tolerancia a fallos (Pág. 83)
- 2.10.14 Sociedad, ética y Profesión (Pág. 84)

2.11 Computación Paralela y Distribuida (PDC) (Pág. 84)
- 2.11.1 Programas: Paralelismo Declarativo (Pág. 85)
- 2.11.2 Programas: Inicio de Actividades (Pág. 85)
- 2.11.3 Programas: Propiedades de Ejecución (Pág. 86)
- 2.11.4 Programas: Distribución (Pág. 86)
- 2.11.5 Programas: Mapeos de Implementación (Pág. 86)
- 2.11.6 Programación GPU (Pág. 87)
- 2.11.7 Comunicación (Pág. 87)
- 2.11.8 Comunicación: Propiedades y Extensiones (Pág. 88)
- 2.11.9 Memoria y Consistencia (Pág. 88)
- 2.11.10 Coordinación (Pág. 89)
- 2.11.11 Coordinación: Sincronización y Atomicidad (Pág. 89)
- 2.11.12 Coordinación: Propiedades Avanzadas (Pág. 90)
- 2.11.13 Evaluación (Pág. 90)
- 2.11.14 Evaluación: Métodos Formales (Pág. 91)
- 2.11.15 Evaluación: Pruebas y Medición (Pág. 91)
- 2.11.16 Algoritmos (Pág. 92)
- 2.11.17 Algoritmos: Dominios de Aplicación (Pág. 92)
- 2.11.18 Algoritmos: Dominios Algorítmicos (Pág. 93)

2.12 Fundamentos de Desarrollo de Software (SDF) (Pág. 93)
- 2.12.1 Conceptos y Construcciones Básicas de Programación (Pág. 94)
- 2.12.2 Prácticas de Programación: Pruebas, Depuración y Documentación (Pág. 95)
- 2.12.3 Estructuras de Datos Fundamentales (Pág. 95)
- 2.12.4 Algoritmos (Pág. 95)
- 2.12.5 Prácticas de Desarrollo de Software (Pág. 96)
- 2.12.6 Sociedad, ética y la Profesión (Pág. 96)

2.13 Ingeniería de Software (SE) (Pág. 96)
- 2.13.1 Colaboración y Dinámica de Equipo (Pág. 98)
- 2.13.2 Diversidad, Stakeholders y Equipos Distribuidos (Pág. 99)
- 2.13.3 Control de Versiones y CI/CD (Pág. 99)
- 2.13.4 Entorno de Desarrollo e IDE (Pág. 100)
- 2.13.5 Ingeniería de Requisitos (Pág. 100)
- 2.13.6 Evolución y Estimación de Requisitos (Pág. 101)
- 2.13.7 Principios y Arquitectura de Software (Pág. 101)
- 2.13.8 Diseño de API y Modelado (Pág. 102)
- 2.13.9 Calidad y Evaluación del Diseño (Pág. 102)
- 2.13.10 Prácticas de Codificación (Pág. 103)
- 2.13.11 Construcción a Escala y Proceso (Pág. 103)
- 2.13.12 Conceptos de verificación y validación (Pág. 104)
- 2.13.13 Planificación y Tipos de Pruebas (Pág. 104)
- 2.13.14 Prácticas Avanzadas de Pruebas (Pág. 105)
- 2.13.15 Herramientas y Análisis de Pruebas (Pág. 105)
- 2.13.16 Pruebas de Rendimiento y Evaluación Comparativa (Pág. 106)
- 2.13.17 Compatibilidad y Versionado de API (Pág. 106)
- 2.13.18 Técnicas de Refactorización (Pág. 107)
- 2.13.19 Conceptos de Confiabilidad (Pág. 107)
- 2.13.20 Ingeniería de Confiabilidad (Pág. 108)
- 2.13.21 Métodos Formales (Pág. 108)

2.14 Seguridad (SEC) (Pág. 109)
- 2.14.1 Principios y Mentalidad de Seguridad (Pág. 110)
- 2.14.2 Control de Acceso y Aplicaciones (Pág. 111)
- 2.14.3 Seguridad en IA y Hardware (Pág. 111)
- 2.14.4 Ética, Ley y Sociedad en Seguridad (Pág. 112)
- 2.14.5 Privacidad y Seguridad por Diseño (Pág. 112)
- 2.14.6 Inyección y Validación de Entrada (Pág. 113)
- 2.14.7 Seguridad de Memoria y Tipos (Pág. 113)
- 2.14.8 Flujo de Información y No Interferencia (Pág. 114)
- 2.14.9 Análisis de Malware y Seguridad Avanzada (Pág. 114)
- 2.14.10 Fundamentos de Criptografía (Pág. 114)
- 2.14.11 Algoritmos Criptográficos (Pág. 115)
- 2.14.12 Protocolos, Aplicaciones y Criptografía Avanzada (Pág. 115)
- 2.14.13 Análisis de Amenazas e Ingeniería de Seguridad (Pág. 116)
- 2.14.14 Diseño e Ingeniería de Controles de Seguridad (Pág. 116)
- 2.14.15 Computación Confiable e Ingeniería de Privacidad (Pág. 117)
- 2.14.16 Técnicas de Informática Forense (Pág. 117)
- 2.14.17 Aspectos Legales y Manejo de Incidentes (Pág. 118)
- 2.14.18 Gobierno, Gestión y Política de Seguridad (Pág. 118)
- 2.14.19 Gestión de Riesgos y Datos (Pág. 119)

2.15 Sociedad, ética y la Profesión (SEP) (Pág. 119)
- 2.15.1 Impacto Social e IA (Pág. 120)
- 2.15.2 Equidad Digital y Accesibilidad (Pág. 121)
- 2.15.3 Fundamentos y Marcos del Análisis Ético (Pág. 121)
- 2.15.4 Métodos y Herramientas de Análisis Ético (Pág. 122)
- 2.15.5 Valores, Naturaleza y Actualización Profesional (Pág. 122)

- 2.15.6 Códigos de Ética y Responsabilidad Legal (Pág. 123)
- 2.15.7 Responsabilidad Profesional y Política Pública (Pág. 123)
- 2.15.8 Disidencia Ética y Contexto Cultural (Pág. 124)
- 2.15.9 Ética en el Lugar de Trabajo (Pág. 124)
- 2.15.10 Fundamentos de la Propiedad Intelectual (Pág. 124)
- 2.15.11 Licencias, Plagio y Autoría (Pág. 125)
- 2.15.12 Derechos de PI Digital y Derechos de Autor (Pág. 125)
- 2.15.13 Open Source, Piratería y DRM (Pág. 125)
- 2.15.14 Datos, Vigilancia y Tecnología de Privacidad (Pág. 126)
- 2.15.15 Fundamentos Filosóficos y Legislación de Privacidad (Pág. 126)
- 2.15.16 Libertades Civiles y Expresión (Pág. 127)
- 2.15.17 Comunicación Técnica y Profesional (Pág. 127)
- 2.15.18 Comunicación en Equipo y Cultura (Pág. 127)
- 2.15.19 Huella Ambiental de los Sistemas Informáticos (Pág. 128)
- 2.15.20 Efectos Sistémicos y Contexto Social (Pág. 128)
- 2.15.21 Diseño Sostenible y Computación Ubicua (Pág. 129)
- 2.15.22 Fundamentos e Historia Temprana de la Informática (Pág. 129)
- 2.15.23 Historia Moderna: PC, Internet e IA (Pág. 130)
- 2.15.24 Modelos y Estrategias Económicas (Pág. 130)
- 2.15.25 Automatización, IA y Ética Económica (Pág. 131)
- 2.15.26 Delitos y Ataques Informáticos (Pág. 131)
- 2.15.27 Leyes, Políticas y Responsabilidad en Seguridad (Pág. 131)
- 2.15.28 Identidad y Diversidad en la Informática (Pág. 132)
- 2.15.29 Contexto Histórico e Intercultural de DEIA (Pág. 132)
- 2.15.30 Inclusión y Prácticas en Informática (Pág. 133)
- 2.15.31 Diseño Accesible (Pág. 133)

2.16 Fundamentos de Sistemas (SF) (Pág. 133)

- 2.16.1 Componentes Hardware y Lógica (Pág. 134)
- 2.16.2 Capas, Abstracciones y Escalabilidad (Pág. 134)
- 2.16.3 Sistemas Digitales y Lógica (Pág. 135)
- 2.16.4 Máquinas de Estado (Pág. 135)
- 2.16.5 Procesamiento Secuencial y Paralelo (Pág. 136)
- 2.16.6 Gestión de Recursos (Pág. 136)
- 2.16.7 Latencia, Caché y Jerarquía de Memoria (Pág. 137)
- 2.16.8 Virtualización y Aislamiento (Pág. 137)
- 2.16.9 Métricas y Benchmarks de Rendimiento (Pág. 138)
- 2.16.10 Modelos Analíticos de Rendimiento (Pág. 138)
- 2.16.11 Eventos, Herramientas y Experimentación (Pág. 138)
- 2.16.12 Confiabilidad del Sistema (Pág. 139)
- 2.16.13 Seguridad del Sistema (Pág. 139)
- 2.16.14 Diseño del Sistema (Pág. 140)
- 2.16.15 Sociedad, ética y la Profesión (Pág. 140)

2.17 Desarrollo de Plataformas Especializadas (SPD) (Pág. 140)

- 2.17.1 Aspectos Comunes: Plataformas, APIs y Restricciones (Pág. 141)
- 2.17.2 Aspectos Comunes: Lenguajes de Plataforma y Patrones Web (Pág. 142)
- 2.17.3 Frontend, Frameworks y Estándares Web (Pág. 142)
- 2.17.4 Arquitectura Cloud y Gestión de Datos (Pág. 142)
- 2.17.5 Plataformas Móviles: Fundamentos del Desarrollo (Pág. 143)
- 2.17.6 Plataformas Móviles: Arquitectura y Restricciones (Pág. 143)
- 2.17.7 Plataformas Móviles: Capacidades y Computación Asíncrona (Pág. 144)
- 2.17.8 Plataformas Robóticas (Pág. 144)
- 2.17.9 Características y Arquitecturas Embebidas (Pág. 144)
- 2.17.10 Sistemas, Programación y Sensores (Pág. 145)
- 2.17.11 Tiempo Real, Memoria y Verificación (Pág. 145)
- 2.17.12 Historia, Tipos y Características de Plataformas (Pág. 146)
- 2.17.13 Sensores, Controladores y Ecosistemas eSports (Pág. 146)
- 2.17.14 Arquitecturas y Renderizado (Pág. 147)
- 2.17.15 Algoritmos, Medios y Mundos Virtuales (Pág. 147)
- 2.17.16 Lenguajes, APIs y Herramientas de Desarrollo (Pág. 147)
- 2.17.17 Motores de Juego (Pág. 148)
- 2.17.18 Fundamentos del Diseño de Juegos (Pág. 148)
- 2.17.19 Géneros, Audiencias y Alcance Social (Pág. 148)
- 2.17.20 Plataformas Interactivas de Datos y Análisis (Pág. 149)
- 2.17.21 Programación con IA y Plataformas Cuánticas (Pág. 149)
- 2.17.22 SPD-SEP/Móvil (Pág. 150)
- 2.17.23 SPD-SEP Web (Pág. 150)
- 2.17.24 Propiedad Intelectual en los Juegos (Pág. 151)
- 2.17.25 Accesibilidad e Interfaz del Juego (Pág. 151)
- 2.17.26 Salud, Seguridad y Sostenibilidad (Pág. 151)
- 2.17.27 Representación, Inclusión e Igualdad (Pág. 152)
- 2.17.28 SPD-SEP/Robótica (Pág. 152)
- 2.17.29 SPD-SEP/Interactiva (Pág. 153)

2.1. Inteligencia Artificial (AI)

La inteligencia artificial (IA) estudia problemas que son difíciles o impracticables de resolver con enfoques algorítmicos tradicionales. Estos problemas a menudo recuerdan a aquellos que se consideran que requieren inteligencia humana, y las estrategias de solución de IA resultantes típicamente se generalizan sobre clases de problemas. Las técnicas de IA ahora son omnipresentes en la informática, apoyando aplicaciones cotidianas como el correo electrónico, las redes sociales, la fotografía, los mercados financieros y los asistentes virtuales inteligentes (por ejemplo, Siri, Alexa). Estas técnicas también se utilizan en el diseño y análisis de agentes autónomos que perciben su entorno e interactúan racionalmente con él, como vehículos autónomos y otros robots.

Tradicionalmente, la IA ha incluido una mezcla de enfoques simbólicos y subsimbólicos. Las soluciones que proporciona se basan en un amplio conjunto de esquemas de representación del conocimiento generales y especializados, mecanismos de resolución de problemas y técnicas de optimización. Estos enfoques abordan la percepción (por ejemplo, reconocimiento de voz, comprensión del lenguaje natural, visión por computadora), la resolución de problemas (por ejemplo, búsqueda, planificación, optimización), la generación (por ejemplo, narrativa, conversación, imágenes, modelos, recomendaciones), la actuación (por ejemplo, robótica, automatización de tareas, control) y las arquitecturas necesarias para soportarlas (por ejemplo, agentes únicos, sistemas multiagente). El aprendizaje automático puede utilizarse dentro de cada uno de estos aspectos e incluso puede emplearse de extremo a extremo en

todos ellos. El estudio de la Inteligencia Artificial prepara a los estudiantes para determinar cuándo un enfoque de IA es apropiado para un problema dado, identificar representaciones y mecanismos de razonamiento apropiados, implementarlos y evaluarlos tanto con respecto al rendimiento como a su impacto social más amplio.

Durante la última década, el término ”inteligencia artificial” se ha vuelto común en empresas, artículos de noticias y conversaciones cotidianas, impulsado en gran medida por una serie de aplicaciones de aprendizaje automático de alto impacto. Estos avances fueron posibles gracias a la amplia disponibilidad de grandes conjuntos de datos, el aumento de la potencia computacional y las mejoras algorítmicas. En particular, ha habido un cambio desde representaciones diseñadas manualmente hacia representaciones aprendidas automáticamente a través de la optimización sobre grandes conjuntos de datos. Los avances resultantes han puesto términos como ”redes neuronales” y ”aprendizaje profundo” en el vocabulario cotidiano. Las empresas ahora anuncian soluciones basadas en IA como valor añadido a sus servicios, de modo que ”inteligencia artificial” es ahora tanto un término técnico como una palabra de moda de marketing. Otras disciplinas, como la biología, el arte, la arquitectura y las finanzas, utilizan cada vez más técnicas de IA para resolver problemas dentro de sus disciplinas.

Por primera vez en nuestra historia, la población en general tiene acceso a herramientas sofisticadas impulsadas por IA, incluyendo herramientas para generar ensayos o poemas a partir de un indicador, obras de arte a partir de una descripción, y fotografías o videos falsos que representan personas reales. La tecnología de IA ahora se utiliza ampliamente en el comercio de acciones, la curación de nuestras noticias y feeds de redes sociales, la evaluación automatizada de solicitantes de empleo, la detección de condiciones médicas y la influencia en las sentencias penales a través de la predicción de reincidencia. En consecuencia, la tecnología de IA puede tener impactos sociales significativos y consideraciones éticas que deben entenderse y considerarse al desarrollarla y aplicarla.

Área de Conocimiento (KA)	**CS Core**	**KA Core**
⭐ 2.1.1 Introducción a la IA: Conceptos y Comportamiento Inteligente (Pág. 15)	2	1
⭐ 2.1.2 Características del Problema y Naturaleza de los Agentes (Pág. 15)	1	1
⭐ 2.1.3 Estado y Búsqueda No Informada (Pág. 15)	1	2
⭐ 2.1.4 Búsqueda Heurística y CSP (Pág. 16)	1	2
⭐ 2.1.5 Búsqueda Adversarial y Optimización Estocástica (Pág. 16)	1	3
⭐ 2.1.6 Representación del Conocimiento y Razonamiento Probabilístico Básico (Pág. 17)	1	1
⭐ 2.1.7 Independencia, Modelos de Markov y Toma de Decisiones (Pág. 17)	1	1
⭐ 2.1.8 Fundamentos de Aprendizaje Automático (Pág. 17)	1	2
⭐ 2.1.9 Aprendizaje Automático: Datos, Representaciones y Evaluación (Pág. 18)	1	2
⭐ 2.1.10 Redes Neuronales y Ética (Pág. 19)	1	1
⭐ 2.1.11 Formulación del Aprendizaje Automático y Evaluación (Pág. 19)	1	2
⭐ 2.1.12 Aprendizaje Profundo, RL y Ética (Pág. 19)	1	2
⭐ 2.1.13 Entrenamiento y Optimización de Redes Neuronales (Pág. 20)	1	1
⭐ 2.1.14 Técnicas de Regularización para Redes Profundas (Pág. 20)	1	1
⭐ 2.1.15 Arquitecturas de Redes Convolucionales y Residuales (Pág. 20)	1	1
⭐ 2.1.16 Arquitecturas Transformer y Modelos Generativos (Pág. 20)	1	1
⭐ 2.1.17 Aplicaciones e Impacto Social de la IA (Pág. 21)	2	2
⭐ 2.1.18 Aplicaciones Adicionales e Impacto Social de la IA (Pág. 21)	2	2
2.1.19 Representación Lógica y Razonamiento (Pág. 22)	Electivo	
2.1.20 Representación Probabilística y Razonamiento (Pág. 22)	Electivo	
2.1.21 Planificación (Pág. 23)	Electivo	
2.1.22 Agentes y Sistemas Cognitivos (Pág. 23)	Electivo	
2.1.23 Procesamiento del Lenguaje Natural (Pág. 24)	Electivo	
2.1.24 Robótica (Pág. 25)	Electivo	
2.1.25 Percepción y Visión por Computadora (Pág. 25)	Electivo	

Cuadro 2.1: Lista de KUs del área de Inteligencia Artificial.

2.1.1. AI/Introducción a la IA: Conceptos y Comportamiento Inteligente (CS Core: 2 hrs, KA Core: 1 hr)

Temas:
Core

- Visión general de los problemas de IA, Ejemplos de aplicaciones recientes exitosas de IA
- Definiciones de agentes con ejemplos (por ejemplo, reactivos, deliberativos)
- ¿Qué es el comportamiento inteligente?: 1. La prueba de Turing y sus defectos 2. Entrada y salida multimodal 3. Simulación del comportamiento inteligente 4. Razonamiento racional versus no racional
- Visión general de las Aplicaciones de IA, crecimiento e impacto (económico, social, ético)

Non Core

- Problemas filosóficos
- Historia de la IA

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir la prueba de Turing y el experimento mental de la "Habitación China" [Explicar]
2. Diferenciar entre razonamiento/comportamiento óptimo y razonamiento/comportamiento similar al humano [Evaluar]
3. Diferenciar los términos: IA, aprendizaje automático y aprendizaje profundo [Explicar]

2.1.2. AI/Características del Problema y Naturaleza de los Agentes (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Características del problema: 1. Totalmente observable versus parcialmente observable 2. Agente único versus multiagente 3. Determinista versus estocástico 4. Estático versus dinámico 5. Discreto versus continuo
- Naturaleza de los agentes: 1. Autonomía autónoma, semiautónoma, de iniciativa mixta 2. Reflexivos, basados en metas y basados en utilidad 3. Toma de decisiones bajo incertidumbre y con información incompleta 4. La importancia de la percepción y las interacciones ambientales 5. Agentes basados en aprendizaje 6. Agentes corporeizados: *a*) sensores, dinámicas, efectores
- Practicar la identificación de características del problema en entornos de ejemplo
- Profundidad adicional sobre la naturaleza de los agentes con ejemplos
- Profundidad adicional sobre Aplicaciones de IA, Crecimiento e Impacto (económico, social, ético, seguridad)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enumerar las características de un problema específico [Aplicar]

2.1.3. AI/Estado y Búsqueda No Informada (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Representación del espacio de estados de un problema: 1. Especificación de estados, metas y operadores 2. Factorización de estados en representaciones (espacios de hipótesis) 3. Resolución de problemas mediante búsqueda en grafos: *a*) por ejemplo, Grafos como un espacio y recorridos de árboles como exploración de ese espacio *b*) Construcción dinámica del grafo (no dada de antemano)
- Búsqueda en grafo no informada para resolución de problemas: 1. Búsqueda en amplitud 2. Búsqueda en profundidad: *a*) Con profundización iterativa 3. Búsqueda de costo uniforme
- Complejidades de espacio y tiempo de los algoritmos de búsqueda en grafos
- Búsqueda bidireccional

Non Core

- Comprensión del espacio de búsqueda: 1. Construcción de árboles de búsqueda 2. Espacios de búsqueda dinámicos 3. Explosión combinatoria del espacio de búsqueda 4. Topología del espacio de búsqueda (por ejemplo, crestas, puntos de silla, mínimos locales)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Diseñar la representación del espacio de estados para un rompecabezas (por ejemplo, N-reinas o problema de los 3 jarrones) [Aplicar]
2. Seleccionar e implementar un algoritmo de búsqueda no informada apropiado para un problema (por ejemplo, tres en raya), y caracterizar sus complejidades de tiempo y espacio [Aplicar]

2.1.4. AI/Búsqueda Heurística y CSP (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Búsqueda en grafo heurística para resolución de problemas: 1. Construcción heurística y admisibilidad 2. Ascenso de colinas 3. Mínimos locales y el paisaje de búsqueda: *a*) Soluciones locales versus globales 4. Búsqueda primero el mejor voraz 5. Búsqueda A*
- Búsqueda en haz
- Implementación de la búsqueda A*
- Satisfacción de restricciones

Non Core

- Variaciones de A* (IDA*, SMA*, RBFS)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Seleccionar e implementar un algoritmo de búsqueda informada apropiado para un problema después de diseñar una función heurística útil (por ejemplo, un robot navegando en un mundo de cuadrícula 2D) [Aplicar]
2. Evaluar si una heurística para un problema dado es admisible/puede garantizar una solución óptima [Evaluar]
3. Diseñar e implementar búsqueda A*/en haz para resolver un problema, y compararla con otros algoritmos de búsqueda en términos del costo de la solución, número de nodos expandidos, etc [Aplicar]
4. Comparar y contrastar varias búsquedas heurísticas con respecto a la aplicabilidad a un problema dado [Evaluar]
5. Modelar un rompecabezas lógico o Sudoku como un problema de satisfacción de restricciones, resolverlo con búsqueda con retroceso y determinar cuánto la consistencia de arcos puede reducir el espacio de búsqueda [Aplicar]

2.1.5. AI/Búsqueda Adversarial y Optimización Estocástica (CS Core: 1 hr, KA Core: 3 hrs)

Temas:
Core

- Juegos adversariales de dos jugadores: 1. Búsqueda minimax 2. Poda alfa-beta: *a*) Corte por profundidad

Non Core

- Búsqueda local
- Búsqueda tabú
- Juegos adversariales de dos jugadores: 1. El efecto horizonte 2. Libros de apertura/soluciones de finales 3. Qué significa ”resolver” un juego (por ejemplo, damas)
- Implementación de búsqueda minimax, búsqueda en haz
- Búsqueda expectimax (resolución MDP) y nodos de azar
- Búsqueda estocástica: 1. Recocido simulado 2. Algoritmos genéticos 3. Búsqueda en árbol Monte Carlo

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Aplicar búsqueda minimax en un juego adversarial de dos jugadores (por ejemplo, conecta 4), usando evaluación heurística en una profundidad particular para calcular las puntuaciones a propagar [Aplicar]
2. Aplicar búsqueda minimax con poda alfa-beta para podar el espacio de búsqueda en un juego adversarial de dos jugadores (por ejemplo, conecta 4) [Aplicar]
3. Comparar y contrastar algoritmos genéticos con técnicas de búsqueda clásicas, explicando cuándo es más apropiado usar un algoritmo genético para aprender un modelo versus otras formas de optimización (por ejemplo, descenso de gradiente) [Evaluar]

NonCore:

4. Diseñar e implementar una solución de algoritmo genético para un problema [Aplicar]
5. Diseñar e implementar un programa de recocido simulado para evitar mínimos locales en un problema [Aplicar]

2.1.6. AI/Representación del Conocimiento y Razonamiento Probabilístico Básico (CS Core: 1 hr, KA Core: 1 hr)

Temas:

Core

- Tipos de representaciones: 1. Simbólicas, lógicas: *a*) Crear una representación a partir de un enunciado de problema en lenguaje natural 2. Representaciones subsimbólicas aprendidas 3. Modelos gráficos (por ejemplo, Bayes ingenuo, red bayesiana)
- Repaso del razonamiento probabilístico, teorema de Bayes
- Razonamiento bayesiano: 1. Inferencia bayesiana
- Variables aleatorias y distribuciones de probabilidad: 1. Axiomas de probabilidad 2. Inferencia probabilística 3. Regla de Bayes (derivación) 4. Inferencia bayesiana (ejemplos más complejos)

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Dado un enunciado de problema en lenguaje natural, codificarlo como una representación simbólica o lógica [Aplicar]
2. Calcular una inferencia probabilística en un problema del mundo real usando el teorema de Bayes para determinar la probabilidad de una hipótesis dada la evidencia [Aplicar]
3. Aplicar la regla de Bayes para determinar la probabilidad de una hipótesis dada la evidencia [Aplicar]
4. Calcular la probabilidad de resultados y probar si los resultados son independientes [Aplicar]

2.1.7. AI/Independencia, Modelos de Markov y Toma de Decisiones (CS Core: 1 hr, KA Core: 1 hr)

Temas:

Core

- Independencia
- Independencia condicional
- Cadenas de Markov y modelos de Markov
- Utilidad y toma de decisiones

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar cómo podemos tomar decisiones bajo incertidumbre, usando conceptos como el teorema de Bayes y la utilidad [Explicar]

2.1.8. AI/Fundamentos de Aprendizaje Automático (CS Core: 1 hr, KA Core: 2 hrs)

Temas:

Core

- Definición y ejemplos de una amplia variedad de tareas de aprendizaje automático: 1. Aprendizaje supervisado: *a*) Clasificación *b*) Regresión 2. Aprendizaje por refuerzo 3. Aprendizaje no supervisado: *a*) Agrupamiento
- Ideas fundamentales: 1. Teorema de no hay almuerzo gratis: ningún aprendiz puede resolver todos los problemas; las decisiones de diseño de representación tienen consecuencias. 2. Fuentes de error e indecidibilidad en el aprendizaje automático
- Un aprendizaje supervisado simple basado en estadísticas como regresión lineal o árboles de decisión: 1. Enfocarse en cómo funcionan sin entrar en detalles matemáticos o de optimización; suficiente para entender y usar implementaciones existentes correctamente
- El problema del sobreajuste/control de la complejidad de la solución (regularización, poda - solo intuición): 1. El compromiso entre sesgo (subajuste) - varianza (sobreajuste)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir las diferencias entre los tres estilos principales de aprendizaje (supervisado, por refuerzo y no supervisado) y determinar cuál es apropiado para un dominio de problema particular [Explicar]
2. Diferenciar los términos: IA, aprendizaje automático y aprendizaje profundo [Evaluar]
3. Formular una aplicación como un problema de clasificación, incluyendo las características de entrada disponibles y la salida a predecir (por ejemplo, identificar caracteres alfabéticos a partir de entrada de cuadrícula de píxeles) [Aplicar]
4. Identificar el sobreajuste en el contexto de un problema y curvas de aprendizaje y describir soluciones al sobreajuste [Evaluar]
5. Explicar cómo funciona el aprendizaje automático como un proceso de optimización/búsqueda [Explicar]

2.1.9. AI/Aprendizaje Automático: Datos, Representaciones y Evaluación (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Trabajar con Datos: 1. Preprocesamiento de datos: *a*) Importancia y peligros de las elecciones de preprocesamiento 2. Manejo de valores faltantes (imputación, marcar como faltante): *a*) Implicaciones de imputar versus marcar como faltante 3. Codificación de variables categóricas, codificación de datos de valor real 4. Normalización/estandarización 5. énfasis en datos reales, no ejemplos de libros de texto
- Representaciones: 1. Espacios de hipótesis y complejidad 2. Expansión de características de base simple, como elevar al cuadrado características univariadas 3. Representaciones de características aprendidas
- Evaluación del aprendizaje automático: 1. Separación de conjuntos de entrenamiento, validación y prueba 2. Métricas de rendimiento para clasificadores 3. Estimación del rendimiento en prueba en datos retenidos 4. Ajuste de los parámetros de un modelo de aprendizaje automático con un conjunto de validación 5. Importancia de entender qué está haciendo un modelo, dónde están sus peligros/limitaciones y las implicaciones de sus decisiones

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Aplicar dos o más algoritmos de aprendizaje estadístico simple a una tarea de clasificación y medir la precisión de los clasificadores [Aplicar]
2. Implementar un algoritmo de aprendizaje estadístico y el proceso de optimización correspondiente para entrenar el clasificador y obtener una predicción en nuevos datos [Aplicar]
3. Explicar los procedimientos adecuados de evaluación de AA, incluyendo las diferencias entre rendimiento en entrenamiento y prueba, y qué puede salir mal con el proceso de evaluación llevando a informes inexactos del rendimiento de AA [Explicar]
4. Comparar dos algoritmos de aprendizaje automático en un conjunto de datos, implementando la metodología de preprocesamiento de datos y evaluación (por ejemplo, métricas y manejo de divisiones entrenamiento/prueba) desde cero [Aplicar]

2.1.10. AI/Redes Neuronales y Ética (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Redes neuronales básicas: 1. Fundamentos de comprensión de cómo funcionan las redes neuronales y su proceso de entrenamiento, sin detalles de los cálculos 2. Introducción básica a redes neuronales generativas (por ejemplo, modelos de lenguaje grandes)
- ética para el Aprendizaje Automático: 1. Enfocarse en datos reales, escenarios reales y estudios de caso 2. Sesgo de conjunto de datos/algorítmico/de evaluación y consecuencias no deseadas

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir el proceso de entrenamiento de redes neuronales y las representaciones aprendidas resultantes [Explicar]

2.1.11. AI/Formulación del Aprendizaje Automático y Evaluación (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Formulación del aprendizaje automático simple como un problema de optimización, como regresión lineal por mínimos cuadrados o regresión logística: 1. Función objetivo 2. Descenso de gradiente 3. Regularización para evitar sobreajuste (formulación matemática)
- Conjuntos de modelos: 1. Combinación de mayoría ponderada simple
- Evaluación de rendimiento: 1. Otras métricas para clasificación (por ejemplo, error, precisión, exhaustividad) 2. Métricas de rendimiento para regresores 3. Matriz de confusión 4. Validación cruzada: *a*) Ajuste de parámetros (búsqueda en cuadrícula/aleatoria, mediante validación cruzada)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comparar y contrastar varias técnicas de aprendizaje (por ejemplo, árboles de decisión, regresión logística, Bayes ingenuo, redes neuronales y redes de creencia), proporcionando ejemplos de cuándo cada estrategia es superior [Comparar]
2. Evaluar el rendimiento de un sistema de aprendizaje simple en un conjunto de datos del mundo real [Evaluar]

NonCore:

3. Caracterizar el estado del arte en teoría del aprendizaje, incluyendo sus logros y limitaciones [Explicar]
4. Explicar el problema del sobreajuste, junto con técnicas para detectar y manejar el problema [Explicar]
5. Explicar el triple compromiso entre el tamaño de un espacio de hipótesis, el tamaño del conjunto de entrenamiento y la precisión del rendimiento [Explicar]

2.1.12. AI/Aprendizaje Profundo, RL y Ética (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Aprendizaje profundo: 1. Redes feed-forward profundas (solo intuición, sin matemáticas) 2. Redes neuronales convolucionales (solo intuición, sin matemáticas) 3. Visualización de representaciones de características aprendidas de redes profundas 4. Otras arquitecturas (NN generativas, NN recurrentes, transformers, etc.)
- Visión general de los métodos de aprendizaje por refuerzo
- Dos o más aplicaciones de algoritmos de aprendizaje automático: 1. Por ejemplo, medicina y salud, economía, visión, lenguaje natural, robótica, juego
- ética para el Aprendizaje Automático: 1. Continuar enfocándose en datos reales, escenarios reales y estudios de caso 2. Privacidad 3. Equidad 4. Propiedad intelectual 5. Explicabilidad

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Visualizar el progreso del entrenamiento de una red neuronal a través de curvas de aprendizaje en un kit de herramientas establecido (por ejemplo, TensorBoard) y visualizar las características aprendidas de la red [Aplicar]

NonCore:

2. Dada una aplicación real de aprendizaje automático, describir problemas éticos respecto a las elecciones de datos, pasos de preprocesamiento, selección de algoritmo y visualización/presentación de resultados [Aplicar]

2.1.13. AI/Entrenamiento y Optimización de Redes Neuronales (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Backpropagation moderno: grafos computacionales y diferenciación automática
- Optimizadores avanzados: momentum, RMSProp, Adam, y programación de la tasa de aprendizaje

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Implementar una red neuronal profunda desde cero, incluyendo su paso de backpropagation [Aplicar]
2. Seleccionar y ajustar un optimizador y su programación de tasa de aprendizaje para la estabilidad del entrenamiento [Aplicar]

2.1.14. AI/Técnicas de Regularización para Redes Profundas (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Regularización por Dropout
- Normalización por lotes (Batch Normalization)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Diagnosticar el sobreajuste en un entrenamiento usando diagnósticos de regularización [Analizar]
2. Aplicar Dropout y Batch Normalization para mejorar la generalización de un modelo [Aplicar]

2.1.15. AI/Arquitecturas de Redes Convolucionales y Residuales (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Arquitecturas de redes neuronales convolucionales profundas
- Conexiones residuales y ResNets

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Diseñar una arquitectura CNN profunda para una tarea de visión dada [Aplicar]
2. Adaptar una arquitectura residual a un dominio de visión específico [Aplicar]

2.1.16. AI/Arquitecturas Transformer y Modelos Generativos (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Transformers y el mecanismo de atención

- Modelos generativos: autoencoders variacionales (VAEs) y redes generativas adversariales (GANs)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Entrenar un modelo transformer en un dataset real [Aplicar]
2. Analizar y contrastar arquitecturas de modelos generativos VAE y GAN [Analizar]

2.1.17. AI/Aplicaciones e Impacto Social de la IA (CS Core: 2 hrs, KA Core: 2 hrs)

Temas:
Core

- Al menos una aplicación de IA a un problema y campo específico, como medicina, salud, sostenibilidad, redes sociales, economía, educación, robótica, etc. (elegir al menos uno para el Núcleo CS): 1. Formular y evaluar una aplicación específica como un problema de IA: *a*) Cómo lidiar con problemas mal especificados o mal planteados 2. Disponibilidad/escasez y limpieza de datos: *a*) Limpieza y preprocesamiento básico de datos *b*) Sesgo del conjunto de datos 3. Sesgo algorítmico 4. Sesgo de evaluación 5. Evaluación de las implicaciones sociales de la aplicación
- Modelos generativos profundos desplegados: 1. Visión general de alto nivel de modelos generativos de imágenes profundas (por ejemplo, a 2023, DALL-E, Midjourney, Stable Diffusion, etc.), sus usos y sus limitaciones/peligros. 2. Visión general de alto nivel de modelos de lenguaje grandes (por ejemplo, a 2023, ChatGPT, Bard, etc.), sus usos y sus limitaciones/peligros.
- Visión general del impacto social de la IA: 1. ética 2. Equidad 3. Confianza/explicabilidad 4. Privacidad y uso de datos de entrenamiento 5. Autonomía humana y supervisión/regulaciones/requisitos legales 6. Sostenibilidad

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Dado un dominio de aplicación y problema del mundo real, formular una solución de IA para ello, identificando datos/entrada apropiados, preprocesamiento, representaciones, técnicas de IA y métricas/metodología de evaluación [Aplicar]

2.1.18. AI/Aplicaciones Adicionales e Impacto Social de la IA (CS Core: 2 hrs, KA Core: 2 hrs)

Temas:
Core

- Una o más aplicaciones adicionales de IA a un amplio conjunto de problemas y diversos campos, como medicina, salud, sostenibilidad, redes sociales, economía, educación, robótica, etc. (elegir un área diferente de la elegida para el Núcleo CS): 1. Formular y evaluar una aplicación específica como un problema de IA: *a*) Cómo lidiar con problemas mal especificados o mal planteados 2. Disponibilidad/escasez y limpieza de datos: *a*) Limpieza y preprocesamiento básico de datos *b*) Sesgo del conjunto de datos 3. Sesgo algorítmico 4. Sesgo de evaluación 5. Evaluación de las implicaciones sociales de la aplicación
- Profundidad adicional sobre modelos generativos profundos desplegados: 1. Introducción a cómo funcionan los modelos generativos de imágenes profundas (por ejemplo, a 2023, DALL-E, Midjourney, Stable Diffusion) incluyendo discusión de atención 2. Introducción a cómo funcionan los modelos de lenguaje grandes (por ejemplo, a 2023, ChatGPT, Bard) incluyendo discusión de atención 3. Idea de modelos fundamentales, cómo usarlos, y los beneficios/problemas con entrenarlos a partir de grandes datos
- Análisis y discusión del impacto social de la IA: 1. ética 2. Equidad 3. Confianza/explicabilidad 4. Privacidad y uso de datos de entrenamiento 5. Autonomía humana y supervisión/regulaciones/requisitos legales 6. Sostenibilidad

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Analizar el impacto social de una o más aplicaciones específicas de IA del mundo real, identificando problemas respecto a ética, equidad, sesgo, confianza y explicabilidad [Evaluar]

2. Describir algunos de los modos de falla de los modelos generativos profundos actuales para lenguaje o imágenes, y cómo esto podría afectar su uso en una aplicación [Describir]

2.1.19. AI/Representación Lógica y Razonamiento

Temas:
Non Core

- Repaso de lógica proposicional y de predicados
- Resolución y demostración de teoremas (solo lógica proposicional): 1. Encadenamiento hacia adelante, encadenamiento hacia atrás
- Problemas de representación del conocimiento: 1. Lógicas de descripción 2. Ingeniería de ontologías
- Web semántica
- Razonamiento no monótono (por ejemplo, lógicas no clásicas, razonamiento por defecto)
- Argumentación
- Razonamiento sobre acción y cambio (por ejemplo, cálculo de situaciones y eventos)
- Razonamiento temporal y espacial
- Programación lógica: 1. Prolog, Programación de conjuntos de respuestas
- Sistemas expertos basados en reglas
- Redes semánticas
- Razonamiento basado en modelos y basado en casos

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Traducir una oración en lenguaje natural (por ejemplo, inglés) a una declaración de lógica de predicados [Aplicar]
2. Convertir una declaración lógica a forma causal [Aplicar]
3. Aplicar resolución a un conjunto de declaraciones lógicas para responder a una consulta [Aplicar]
4. Comparar y contrastar los modelos más comunes utilizados para la representación estructurada del conocimiento, destacando sus fortalezas y debilidades [Evaluar]
5. Identificar los componentes del razonamiento no monótono y su utilidad como mecanismo representacional para sistemas de creencias [Evaluar]
6. Comparar y contrastar las técnicas básicas para representar la incertidumbre [Evaluar]
7. Comparar y contrastar las técnicas básicas para la representación cualitativa [Evaluar]
8. Aplicar cálculo de situaciones y eventos a problemas de acción y cambio [Aplicar]
9. Explicar la distinción entre razonamiento temporal y espacial, y cómo se interrelacionan [Explicar]
10. Explicar la diferencia entre técnicas de razonamiento basadas en reglas, basadas en casos y basadas en modelos [Explicar]
11. Definir el concepto de un sistema de planificación y cómo difiere de las técnicas de búsqueda clásicas [Definir]
12. Describir las diferencias entre planificación como búsqueda, planificación basada en operadores y planificación proposicional, proporcionando ejemplos de dominios donde cada una es más aplicable [Describir]
13. Explicar la distinción entre inferencia monótona y no monótona [Explicar]

2.1.20. AI/Representación Probabilística y Razonamiento

Temas:
Non Core

- Repaso de independencia condicional
- Representaciones del conocimiento: 1. Redes bayesianas: *a*) Inferencia exacta y su complejidad *b*) Mantas de Markov y d-separación *c*) Métodos de muestreo aleatorizado (Monte Carlo) (por ejemplo, muestreo de Gibbs) 2. Redes de Markov 3. Modelos de probabilidad relacional 4. Modelos ocultos de Markov
- Teoría de la decisión: 1. Preferencias y funciones de utilidad 2. Maximización de la utilidad esperada 3. Teoría de juegos

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Calcular la probabilidad de una hipótesis dada la evidencia en una red bayesiana [Aplicar]
2. Explicar cómo las afirmaciones de independencia condicional permiten una mayor eficiencia de los sistemas probabilísticos [Explicar]
3. Identificar ejemplos de representaciones del conocimiento para razonar bajo incertidumbre [Listar/Enumerar]
4. Establecer la complejidad de la inferencia exacta. Identificar métodos para inferencia aproximada [Enunciar/Establecer]
5. Diseñar e implementar al menos una representación del conocimiento para razonar bajo incertidumbre [Aplicar]
6. Describir las complejidades del razonamiento probabilístico temporal [Describir]
7. Diseñar e implementar un HMM como un ejemplo de sistema probabilístico temporal [Aplicar]
8. Describir la relación entre preferencias y funciones de utilidad [Describir]
9. Explicar cómo las funciones de utilidad y el razonamiento probabilístico pueden combinarse para tomar decisiones racionales [Explicar]

2.1.21. AI/Planificación

Temas:
Non Core

- Repaso de lógica proposicional y de primer orden
- Operadores de planificación y representaciones de estado
- Planificación de orden total
- Planificación de orden parcial
- Grafos de plan y GraphPlan
- Planificación jerárquica
- Lenguajes y representaciones de planificación: 1. PDDL
- Planificación multiagente
- Planificación basada en MDP
- Interconexión de planificación, ejecución y replanificación dinámica: 1. Planificación condicional 2. Planificación continua 3. Planificación probabilística

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Construir la representación de estado, meta y operadores para un problema de planificación dado [Aplicar]
2. Codificar un problema de planificación en PDDL y usar un planificador para resolverlo [Aplicar]
3. Dado un conjunto de operadores, estado inicial y estado meta, dibujar el grafo de planificación de orden parcial e incluir restricciones de orden para resolver todos los conflictos [Aplicar]
4. Construir el grafo de planificación completo para que GraphPlan resuelva un problema dado [Aplicar]

2.1.22. AI/Agentes y Sistemas Cognitivos

Temas:
Non Core

- Arquitecturas de agentes (por ejemplo, reactivas, en capas, cognitivas)
- Teoría de agentes (incluyendo formalismos matemáticos)
- Racionalidad, Teoría de juegos: 1. Agentes decisorios-teóricos 2. Procesos de decisión de Markov (MDP) 3. Algoritmos de bandidos
- Agentes de software, asistentes personales y acceso a información: 1. Agentes colaborativos 2. Agentes de recolección de información 3. Agentes creíbles (personajes sintéticos, modelado de emociones en agentes)
- Agentes de aprendizaje
- Sistemas cognitivos: 1. Arquitecturas cognitivas (por ejemplo, ACT-R, SOAR, ICARUS, FORR) 2. Capacidades (por ejemplo, percepción, toma de decisiones, predicción, mantenimiento del co-

nocimiento) 3. Representación, organización, utilización, adquisición y refinamiento del conocimiento 4. Aplicaciones y evaluación de sistemas cognitivos
- Sistemas multiagente: 1. Agentes colaboradores 2. Equipos de agentes 3. Agentes competitivos (por ejemplo, subastas, votación) 4. Sistemas de enjambre y modelos biológicamente inspirados 5. Aprendizaje multiagente
- Interacción humano-agente: 1. Metodologías de comunicación (verbal y no verbal) 2. Problemas prácticos 3. Aplicaciones: *a*) Agentes comerciales, gestión de cadena de suministro *b*) Problemas éticos de las interacciones de IA con humanos *c*) Regulación y requisitos legales de los sistemas de IA para interactuar con humanos

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Caracterizar y contrastar las arquitecturas estándar de agentes [Evaluar]
2. Describir las aplicaciones de la teoría de agentes a dominios como agentes de software, asistentes personales y agentes creíbles, y discutir las implicaciones éticas asociadas [Describir]
3. Describir los paradigmas principales utilizados por los agentes de aprendizaje [Describir]
4. Demostrar usando ejemplos apropiados cómo los sistemas multiagente soportan la interacción de agentes [Demostrar]
5. Construir un agente inteligente usando una arquitectura cognitiva establecida (ACT-R, SOAR) para resolver un problema específico [Aplicar]

2.1.23. AI/Procesamiento del Lenguaje Natural

Temas:
Non Core

- Gramáticas deterministas y estocásticas
- Algoritmos de análisis sintáctico: 1. CFGs y analizadores de tabla (por ejemplo, CYK) 2. CFGs probabilísticos y CYK ponderado
- Representación del significado/Semántica: 1. Representaciones del conocimiento basadas en lógica 2. Roles semánticos 3. Representaciones temporales 4. Creencias, deseos e intenciones
- Métodos basados en corpus
- N-gramas y HMMs
- Suavizado y retroceso
- Ejemplos de uso: Etiquetado POS y morfología
- Recuperación de información: 1. Modelo de espacio vectorial: *a*) TF & IDF 2. Precisión y exhaustividad
- Extracción de información
- Traducción de lenguaje
- Clasificación de texto, categorización: 1. Modelo de bolsa de palabras
- Aprendizaje profundo para PLN: 1. RNNs 2. Transformers 3. Incrustaciones multimodales (por ejemplo, imágenes + texto) 4. Modelos de lenguaje generativos

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Definir y contrastar gramáticas deterministas y estocásticas, proporcionando ejemplos para mostrar la adecuación de cada una [Definir]
2. Simular, aplicar o implementar algoritmos clásicos y estocásticos para analizar lenguaje natural [Aplicar]
3. Identificar los desafíos de representar el significado [Analizar]
4. Listar las ventajas de usar corpus estándar. Identificar ejemplos de corpus actuales para una variedad de tareas de PLN [Listar/Enumerar]
5. Identificar técnicas para recuperación de información, traducción de lenguaje y clasificación de texto [Analizar]
6. Implementar una transformación TF/IDF, usarla para extraer características de un corpus, y entrenar un algoritmo de aprendizaje automático estándar usando esas características para hacer clasificación de texto [Aplicar]

2.1.24. AI/Robótica

Temas:
Non Core

- Visión general: problemas y progreso: 1. Sistemas robóticos de última generación, incluyendo sus sensores y una visión general de su procesamiento sensorial 2. Arquitecturas de control de robots, por ejemplo, control deliberativo versus reactivo y vehículos de Braitenberg 3. Modelado del mundo y modelos del mundo 4. Incertidumbre inherente en la percepción y en el control
- Sensores y efectores: 1. Sensores: por ejemplo, LIDAR, sonar, visión, profundidad, estereoscópico, cámaras de eventos, micrófonos, háptica 2. Efectores: por ejemplo, ruedas, brazos, pinzas
- Marcos de coordenadas, traslación y rotación (2D y 3D)
- Espacio de configuración y mapas ambientales
- Interpretación de datos sensoriales inciertos
- Localización y mapeo
- Navegación y control
- Cinemática directa e inversa
- Planificación de trayectorias de movimiento y optimización de trayectorias
- Manipulación y agarre
- Control y dinámica de articulaciones
- Control basado en visión
- Coordinación y colaboración múltiple de robots
- Interacción humano-robot: 1. Espacios de trabajo compartidos 2. Equipos humano-robot e HRI físico 3. Robots de asistencia social 4. Predicción de movimiento/tarea/meta 5. Colaboración y comunicación (explícita vs implícita, verbal o simbólica vs no verbal o visual) 6. Confianza
- Aplicaciones y Problemas Sociales, Económicos y éticos: 1. Implicaciones sociales, económicas, de derecho al trabajo 2. Implicaciones éticas y de privacidad de las aplicaciones robóticas 3. Responsabilidad en robótica autónoma 4. Armas autónomas y ética 5. Supervisión y control humano

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Listar capacidades y limitaciones de los sistemas robóticos de última generación actuales, incluyendo sus sensores y el procesamiento sensorial crucial que informa a esos sistemas [Listar/Enumerar]
2. Integrar sensores, actuadores y software en un robot diseñado para emprender una tarea específica [Aplicar]
3. Programar un robot para realizar tareas simples usando arquitecturas de control deliberativas, reactivas y/o híbridas [Aplicar]
4. Implementar algoritmos fundamentales de planificación de movimiento dentro de un espacio de configuración de robot [Aplicar]
5. Caracterizar las incertidumbres asociadas con sensores y actuadores robóticos comunes; articular estrategias para mitigar estas incertidumbres [Caracterizar]
6. Listar las diferencias entre las representaciones del entorno externo de los robots, incluyendo sus fortalezas y limitaciones [Listar/Enumerar]
7. Comparar y contrastar al menos tres estrategias para la navegación de robots dentro de entornos conocidos y/o desconocidos, incluyendo sus fortalezas y limitaciones [Comparar]
8. Describir al menos un enfoque para coordinar las acciones y percepción de varios robots para realizar una única tarea [Describir]
9. Comparar y contrastar un enfoque de coordinación multirobot y un enfoque de colaboración humano-robot y atribuir sus diferencias a diferencias entre los entornos del problema [Comparar]
10. Analizar los problemas sociales, económicos y éticos de una aplicación robótica del mundo real [Analizar]

2.1.25. AI/Percepción y Visión por Computadora

Temas:
Non Core

- Visión por computadora: 1. Adquisición, representación, procesamiento y propiedades de imágenes 2. Representación de formas, reconocimiento de objetos y segmentación 3. Análisis de movimiento 4. Modelos generativos
- Audio y reconocimiento de voz
- Tacto y propiocepción
- Otras modalidades (por ejemplo, olfato)
- Modularidad en el reconocimiento
- Enfoques para el reconocimiento de patrones: 1. Algoritmos de clasificación y medidas de calidad de clasificación 2. Técnicas estadísticas 3. Técnicas de aprendizaje profundo

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Resumir la importancia del reconocimiento de imágenes y objetos en IA e indicar varias aplicaciones significativas de esta tecnología [Resumir]
2. Listar al menos tres enfoques de segmentación de imágenes, como umbralización, algoritmos basados en bordes y basados en regiones, junto con sus características definitorias, fortalezas y debilidades [Listar/Enumerar]
3. Implementar reconocimiento de objetos 2D basado en representaciones de forma basadas en contorno y/o región [Aplicar]
4. Distinguir los objetivos del reconocimiento de sonido, reconocimiento de voz y reconocimiento de hablante, e identificar cómo se manejará diferente la señal de audio cruda en cada uno de estos casos [Distinguir]
5. Proporcionar al menos dos ejemplos de transformación de una fuente de datos de un dominio sensorial a otro, por ejemplo, datos táctiles interpretados como imágenes 2D de banda única [Generar]
6. Implementar un algoritmo de extracción de características en datos reales, por ejemplo, un detector de bordes o esquinas para imágenes o vectores de coeficientes de Fourier que describen un segmento corto de señal de audio [Aplicar]
7. Implementar un algoritmo que combine características en percepciones de nivel superior, por ejemplo, un contorno o polígono a partir de primitivas visuales o hipótesis de fonemas a partir de una señal de audio [Aplicar]
8. Implementar un algoritmo de clasificación que segmenta percepciones de entrada en categorías de salida y evalúa cuantitativamente la clasificación resultante [Aplicar]
9. Evaluar el rendimiento de la extracción de características subyacente, relativo a al menos un enfoque alternativo posible (ya sea implementado o no) en su contribución a la tarea de clasificación (8), anterior [Evaluar]
10. Describir al menos tres enfoques de clasificación, sus prerrequisitos para aplicabilidad, sus fortalezas y sus limitaciones [Describir]
11. Implementar y evaluar una solución de aprendizaje profundo para problemas en visión por computadora, como reconocimiento de objetos o escenas [Aplicar]

2.2. Fundamentos Algorítmicos (AL)

Los algoritmos y las estructuras de datos son fundamentales para la informática, ya que todo cálculo teórico y programa aplicado consiste en algoritmos que operan sobre elementos de datos que poseen alguna estructura subyacente. Seleccionar soluciones computacionales apropiadas para problemas del mundo real se beneficia de comprender las capacidades y limitaciones teóricas y prácticas de los algoritmos y paradigmas disponibles, incluyendo su impacto en el medio ambiente y la sociedad. Además, esta comprensión proporciona una visión de la naturaleza intrínseca de la computación, los problemas computacionales y la resolución de problemas computacionales, así como de las posibles técnicas de solución independientes del lenguaje de programación, el paradigma de programación, el hardware de la computadora u otros aspectos de implementación.

Esta área de conocimiento se centra en la naturaleza de la computación, incluyendo los conceptos y habilidades necesarios para diseñar y analizar algoritmos para resolver problemas computacionales del mundo real. Complementa la implementación de algoritmos y estructuras de datos que se encuentran en el área de conocimiento de Fundamentos de Desarrollo de Software (SDF). Como los algoritmos

y las estructuras de datos son esenciales en todas las áreas avanzadas de la informática, esta área proporciona los fundamentos algorítmicos que se espera que todo graduado en informática conozca. La exposición a la amplitud de estos temas fundamentales de AL está diseñada para proporcionar a los estudiantes la base para estudiar estos temas en mayor profundidad, para estudiar temas adicionales de computación y algoritmos, y para aprender algoritmos avanzados en una variedad de áreas de conocimiento de CS y disciplinas CS+X.

Área de Conocimiento (KA)	**CS Core**	**KA Core**
⭐ 2.2.1 Estructuras de Datos Fundamentales (Pág. 27)	5	3
⭐ 2.2.2 Algoritmos Fundamentales (Pág. 27)	4	2
⭐ 2.2.3 Algoritmos Avanzados (Pág. 28)	3	2
⭐ 2.2.4 Estrategias Algorítmicas (Pág. 28)	6	
⭐ 2.2.5 Marco de Análisis de Complejidad (Pág. 29)	2	1
⭐ 2.2.6 Notación Asintótica y Clases de Complejidad (Pág. 30)	1	1
⭐ 2.2.7 Análisis de Complejidad: Recursión, Amortización y Cotas Ajustadas (Pág. 30)	2	1
⭐ 2.2.8 Teoría de la Complejidad Computacional (Pág. 30)	2	1
⭐ 2.2.9 Lenguajes Formales y Autómatas (Pág. 31)	4	10
⭐ 2.2.10 Computabilidad y Decidibilidad (Pág. 31)	6	14
2.2.11 Sociedad, ética y la Profesión (Pág. 32)	Electivo	

Cuadro 2.2: Lista de KUs del área de Fundamentos Algorítmicos.

2.2.1. AL/Estructuras de Datos Fundamentales (CS Core: 5 hrs, KA Core: 3 hrs)

Temas:
Core

- Tipo de Dato Abstracto (ADT) y operaciones sobre un ADT: 1. Operaciones de diccionario (insertar, eliminar, encontrar)
- Arreglos: 1. Numéricos vs no numéricos, cadenas de caracteres 2. Unidimensionales (vector) vs multidimensionales (matriz)
- Registros/Estructuras/Tuplas y Objetos
- Listas enlazadas (por razones históricas): 1. Simples vs Dobles y Lineales vs Circulares
- Pilas
- Colas y deques: 1. Cola de prioridad basada en montículo
- Tablas/mapas hash: 1. Resolución de colisiones y complejidad (por ejemplo, sondeo, encadenamiento, rehash)
- árboles: 1. Binarios, n-arios y árboles de búsqueda 2. Balanceados (por ejemplo, AVL, Rojo-Negro, Montículo)
- Conjuntos

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Para cada ADT/Estructura de Datos en esta unidad: *a*) Explicar su definición, propiedades, representación(es) y operaciones de ADT asociadas. *b*) Explicar paso a paso cómo las operaciones de ADT asociadas con la estructura de datos la transforman. [Explicar]
2. Explicar cómo se maneja la evitación de colisiones y la resolución de colisiones en tablas hash [Explicar]
3. Explicar la propiedad de montículo y el uso de montículos como una implementación de una cola de prioridad [Explicar]

2.2.2. AL/Algoritmos Fundamentales (CS Core: 4 hrs, KA Core: 2 hrs)

Temas:
Core

- Grafos (por ejemplo, [no]dirigidos, [a]cíclicos, [no]conexos y [no]ponderados): 1. Representación de grafos: lista de adyacencia vs matriz

- Algoritmos de búsqueda: 1. Complejidad $O(n)$ (por ejemplo, búsqueda lineal/secuencial en arreglo/lista) 2. Complejidad $O(\log_2 n)$ (por ejemplo, búsqueda binaria) 3. Complejidad $O(\log_b n)$ (por ejemplo, búsqueda en árbol no informada en profundidad/amplitud)
- Algoritmos de ordenamiento (por ejemplo, estables, inestables): 1. Complejidad $O(n^2)$ (por ejemplo, inserción, selección) 2. Complejidad $O(n \log n)$ (por ejemplo, quicksort, merge, timesort)
- Algoritmos de grafos: 1. Camino más corto (por ejemplo, Dijkstra, Floyd) 2. árbol de expansión mínima (por ejemplo, Prim, Kruskal)
- Algoritmos de ordenamiento: 1. Complejidad $O(n \log n)$ heapsort 2. Pseudo $O(n)$ complejidad (por ejemplo, bucket, counting, radix)
- Algoritmos de grafos: 1. Clausura transitiva (por ejemplo, Warshall) 2. Ordenamiento topológico

- Emparejamiento: 1. Emparejamiento eficiente de cadenas (por ejemplo, Boyer-Moore, Knuth-Morris-Pratt) 2. Emparejamiento de subsecuencia común más larga 3. Emparejamiento de expresiones regulares

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Para cada algoritmo en esta unidad explicar paso a paso cómo opera el algoritmo [Explicar]
2. Para cada enfoque algorítmico (por ejemplo, ordenamiento) en esta unidad aplicar un ejemplo prototípico del enfoque (por ejemplo, ordenamiento por mezcla) [Aplicar]
3. Dados los requisitos para un problema, desarrollar múltiples soluciones usando varias estructuras de datos y algoritmos. Posteriormente, evaluar la idoneidad, fortalezas y debilidades seleccionando un enfoque que satisfaga mejor los requisitos [Crear]
4. Explicar factores más allá de la eficiencia computacional que influyen en la elección de algoritmos, como el tiempo de programación, la mantenibilidad y el uso de patrones específicos de la aplicación en los datos de entrada [Explicar]
5. Para cada uno de los algoritmos y enfoques algorítmicos en los temas del Núcleo KA: *a*) Explicar un ejemplo prototípico del algoritmo, y *b*) Explicar paso a paso cómo opera el algoritmo. [Explicar]

2.2.3. AL/Algoritmos Avanzados (CS Core: 3 hrs, KA Core: 2 hrs)

Temas:
Non Core

- Algoritmos de criptografía (por ejemplo, SHA-256)
- Algoritmos paralelos
- Algoritmos de consenso (por ejemplo, Blockchain): 1. Prueba de trabajo vs prueba de participación
- Algoritmos de computación cuántica: 1. Basados en oráculo (por ejemplo, Deutsch-Jozsa, Bernstein-Vazirani, Simon) 2. Aceleración superpolinomial mediante QFT (por ejemplo, Shor) 3. Aceleración polinomial mediante amplificación de amplitud (por ejemplo, Grover)
- Algoritmo de Transformada Rápida de Fourier (FFT)
- Algoritmo de evolución diferencial

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Una apreciación de la computación cuántica y su aplicación a ciertos problemas [Explicar]

2.2.4. AL/Estrategias Algorítmicas (CS Core: 6 hrs)

Temas:
Core

- Paradigmas: 1. Fuerza Bruta (por ejemplo, búsqueda lineal, ordenamiento por selección, viajante de comercio, mochila) 2. Disminuir y Vencer: *a*) Por una Constante (por ejemplo, ordenamiento por inserción, ordenamiento topológico) *b*) Por un Factor Constante (por ejemplo, búsqueda binaria) *c*) Por un Tamaño Variable (por ejemplo, Euclides) 3. Dividir y Vencer (por ejemplo, búsqueda binaria, quicksort, mergesort, Strassen) 4. Voraz (por ejemplo, Dijkstra, Kruskal,

Mochila) 5. Transformar y Vencer: *a*) Simplificación de instancia (por ejemplo, encontrar duplicados mediante preordenamiento de lista) *b*) Cambio de representación (por ejemplo, heapsort) *c*) Reducción de problema (por ejemplo, mínimo común múltiplo, programación lineal) *d*) Programación dinámica (por ejemplo, Floyd, Marshall, Bellman-Ford) 6. Compromisos espacio vs tiempo (por ejemplo, hash)
- Manejo del crecimiento exponencial (por ejemplo, heurística A*, ramificación y poda, retroceso)

- Iteración vs recursión (por ejemplo, factorial, búsqueda en árbol)
- Paradigmas: 1. Algoritmos de aproximación 2. Mejora iterativa (por ejemplo, Ford-Fulkerson, simplex) 3. Algoritmos aleatorizados/estocásticos (por ejemplo, corte máximo, bolas y cubos)

Non Core
- Computación cuántica

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Para cada uno de los paradigmas en esta unidad: *a*) Explicar sus características definitorias *b*) Explicar un ejemplo que demuestre el paradigma incluyendo cómo este ejemplo satisface las características del paradigma. [Explicar]
2. Para cada uno de los algoritmos en la unidad Estructuras de Datos Fundamentales, Algoritmos Fundamentales, Algoritmos Avanzados, explicar el paradigma utilizado por el algoritmo y cómo ejemplifica este paradigma [Explicar]
3. Dado un algoritmo, explicar el paradigma utilizado por el algoritmo y cómo ejemplifica este paradigma [Explicar]
4. Dar un problema del mundo real, evaluar paradigmas algorítmicos apropiados y algoritmos de estos paradigmas que aborden el problema incluyendo evaluar los compromisos entre los paradigmas y algoritmos seleccionados [Evaluar]
5. Dar ejemplos de algoritmos iterativos y recursivos que resuelvan el mismo problema, explicar los beneficios y desventajas de cada enfoque [Explicar]
6. Evaluar si un enfoque voraz conduce a una solución óptima [Evaluar]
7. Explicar varios enfoques para abordar problemas computacionales cuyas soluciones algorítmicas son exponenciales [Explicar]

2.2.5. AL/Marco de Análisis de Complejidad (CS Core: 2 hrs, KA Core: 1 hr)

Temas:
Core
- Marco de Análisis de Complejidad: 1. Rendimiento de un algoritmo en el mejor caso, caso promedio y peor caso 2. Mediciones empíricas y relativas (Orden de Crecimiento) 3. Tamaño de entrada y operaciones primitivas 4. Eficiencia de tiempo y espacio
- Mediciones empíricas de rendimiento
- Compromisos tiempo-espacio en algoritmos

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Preparar una presentación que explique a estudiantes de primer año los conceptos básicos de complejidad algorítmica incluyendo comportamiento de algoritmo en mejor caso, caso promedio y peor caso, notaciones Big-O, Omega y Theta, clases de complejidad, compromisos tiempo-espacio, medición empírica e impacto en problemas prácticos [Explicar]
2. Para cada algoritmo en la unidad Estructuras de Datos Fundamentales, Algoritmos Fundamentales, Algoritmos Avanzados, explicar su clase de complejidad de tiempo de ejecución y por qué pertenece a esta clase [Explicar]
3. Evaluar informalmente la clase de complejidad fundamental de algoritmos simples [Evaluar]
4. Desarrollar estudios empíricos para determinar y validar hipótesis sobre la complejidad de tiempo de ejecución de varios algoritmos ejecutando algoritmos con entradas de varios tamaños y comparando el rendimiento real con el análisis teórico [Crear]
5. Explicar ejemplos que ilustren los compromisos tiempo-espacio de algoritmos [Explicar]
6. Explicar cómo el balance del árbol afecta la eficiencia de las operaciones de árbol de búsqueda binaria [Explicar]

2.2.6. AL/Notación Asintótica y Clases de Complejidad (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Análisis de complejidad asintótica (cotas promedio y peor caso): 1. Notaciones formales Big-O, Big-Omega y Big-Theta 2. Clases de Complejidad Fundamentales y Ejemplos/Problemas Representativos: *a*) $O(1)$ Constante (por ejemplo, acceso a arreglo) *b*) $O(\log_2 n)$ Logarítmica (por ejemplo, búsqueda binaria) *c*) $O(n)$ Lineal (por ejemplo, búsqueda lineal) *d*) $O(n \log_2 n)$ Log Lineal (por ejemplo, mergesort) *e*) $O(n^2)$ Cuadrática (por ejemplo, ordenamiento por selección) *f*) $O(n^c)$ Polinomial (por ejemplo, $O(n^3)$ eliminación gaussiana) *g*) $O(2^n)$ Exponencial (por ejemplo, Mochila, Satisfactibilidad (SAT), Viajante de Comercio (TSP), todos los subconjuntos) *h*) $O(n!)$ Factorial (por ejemplo, circuito hamiltoniano, todas las permutaciones)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Usando ejemplos, explicar cada una de las clases de complejidad fundamentales en esta unidad [Explicar]
2. Para cada clase de complejidad fundamental en esta unidad, explicar un algoritmo que demuestre la complejidad de tiempo de ejecución asociada [Explicar]
3. Explicar a una audiencia no técnica la importancia de los algoritmos tratables versus intratables usando una explicación intuitiva de la complejidad Big-O [Explicar]
4. Aplicar la notación Big-O para dar cotas superiores en la complejidad tiempo/espacio de algoritmos [Aplicar]

2.2.7. AL/Análisis de Complejidad: Recursión, Amortización y Cotas Ajustadas (CS Core: 2 hrs, KA Core: 1 hr)

Temas:
Core

- Notaciones Little-o, Little-Omega y Little Theta
- Análisis recursivo formal
- Análisis amortizado

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Dado un problema para programar para el cual puede haber varios enfoques algorítmicos, evaluarlos y determinar cuáles son factibles, y seleccionar uno que sea óptimo en implementación y comportamiento de tiempo de ejecución [Evaluar]
2. Usar relaciones de recurrencia para evaluar la complejidad de tiempo de algoritmos definidos recursivamente [Aplicar]
3. Aplicar relaciones de recurrencia elementales usando una forma del Teorema Maestro [Aplicar]

2.2.8. AL/Teoría de la Complejidad Computacional (CS Core: 2 hrs, KA Core: 1 hr)

Temas:
Core

- Tractabilidad e intractabilidad: 1. Clases de Complejidad P, NP y NP-Completo 2. Problemas NP-Completos (por ejemplo, SAT, Mochila, TSP) 3. Reducciones
- Modelos de complejidad basados en Máquinas de Turing: 1. Complejidad de tiempo: *a*) Clases P, NP, NP-C y EXP *b*) Teorema de Cook-Levin 2. Complejidad de espacio: *a*) NSpace y PSpace *b*) Teorema de Savitch

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar la importancia de la NP-Completitud [Explicar]
2. Explicar cómo NP-Duro es una cota inferior y NP es una cota superior para la NP-Completitud [Explicar]

3. Explicar ejemplos de problemas NP-completos [Explicar]
4. Explicar el Teorema de Cook-Levin y la NP-Completitud de SAT [Explicar]
5. Explicar las clases P y NP [Explicar]
6. Demostrar que un problema es NP-Completo reduciendo un problema NP-C clásico conocido a él (por ejemplo, 3SAT y Clique) [Crear]
7. Explicar la clase P-espacio y su relación con la clase EXP [Explicar]

2.2.9. AL/Lenguajes Formales y Autómatas (CS Core: 4 hrs, KA Core: 10 hrs)

Temas:
Core

- Autómatas formales: 1. Estado Finito 2. Pila 3. Linealmente Acotado 4. Máquina de Turing
- Lenguajes formales, gramáticas y Jerarquía de Chomsky: 1. Regulares (Tipo-3): *a*) Expresiones Regulares 2. Libres de Contexto (Tipo-2) 3. Sensibles al Contexto (Tipo-1) 4. Recursivamente Enumerables (Tipo-0)
- Relaciones entre autómatas formales, lenguajes y gramáticas
- Autómatas deterministas y no deterministas
- Demostraciones del Lema de Bombeo: 1. Demostración de la limitación de Estado Finito/Lenguaje Regular 2. Limitación de Autómatas de Pila/Lenguaje Libre de Contexto
- Modelos equivalentes de computación algorítmica: 1. Máquinas de Turing y Variaciones (por ejemplo, multicinta, no deterministas) 2. Cálculo Lambda 3. Funciones Mu-Recursivas

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Para cada autómata formal en esta unidad: *a*) Explicar su definición comparando sus características con los otros autómatas de esta unidad *b*) Usando un ejemplo, explicar paso a paso cómo el autómata opera sobre la entrada incluyendo si acepta la entrada asociada *c*) Explicar un ejemplo de entradas que pueden y no pueden ser aceptadas por el autómata. [Explicar]
2. Dado un problema, desarrollar un autómata apropiado que aborde el problema [Crear]
3. Desarrollar una expresión regular para un lenguaje regular dado expresado en lenguaje natural [Crear]
4. Explicar la diferencia entre expresiones regulares (aceptadores Tipo-3) y las expresiones regulares (aceptadores Tipo-2) usadas en lenguajes de programación [Explicar]
5. Para cada modelo formal en esta unidad: *a*) Explicar su definición comparando sus características con las otras en esta unidad *b*) Explicar ejemplos de entradas que son y no pueden ser aceptadas por el lenguaje/gramática. [Explicar]
6. Para cada autómata formal en esta unidad explicar (comparar/contrastar) sus capacidades deterministas y no deterministas [Explicar]
7. Aplicar lemas de bombeo, o medios alternativos, para demostrar las limitaciones de los autómatas de Estado Finito y Pila [Aplicar]
8. Convertir entre notaciones equivalentemente poderosas para un lenguaje, incluyendo entre DFAs, NFAs y expresiones regulares, y entre PDAs y CFGs [Aplicar]

2.2.10. AL/Computabilidad y Decidibilidad (CS Core: 6 hrs, KA Core: 14 hrs)

Temas:
Core

- Decidabilidad, (in)computabilidad y parada
- La tesis de Church-Turing
- Corrección algorítmica: 1. Invariantes (por ejemplo, en iteración, recursión, búsqueda en árbol)
- Decidabilidad: 1. Aritmetización y diagonalización
- Reducibilidad y reducciones
- Complejidad de tiempo basada en Máquina de Turing
- Complejidad de espacio (por ejemplo, Pspace, Teorema de Savitch)

Non Core

- Computación cuántica: 1. Postulados de la mecánica cuántica: *a*) Espacio de estados *b*) Evolución del estado *c*) Composición de estados *d*) Medición del estado 2. Representaciones de vector columna de qubits 3. Representaciones matriciales de operaciones cuánticas 4. Compuertas cuánticas simples (por ejemplo, XNOT, CNOT)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar una Máquina de Turing universal y su operación [Explicar]
2. Presentar a una audiencia de compañeros de trabajo y gerentes la imposibilidad de proporcionarles un programa que verifique todos los otros programas, incluyendo algunos aparentemente simples, en busca de bucles infinitos incluyendo una explicación del problema de la parada, por qué no tiene solución algorítmica y su importancia para la computación algorítmica del mundo real [Explicar]
3. Explicar ejemplos de problemas clásicos incomputables [Explicar]
4. Explicar la Tesis de Church-Turing y su importancia para la computación algorítmica [Explicar]
5. Explicar cómo los invariantes (de bucle) pueden usarse para demostrar la corrección de un algoritmo [Explicar]
6. Aplicar aritmetización y diagonalización para demostrar que el Problema de la Parada para Máquinas de Turing es Indecidible [Aplicar]
7. Dado un lenguaje indecidible conocido, aplicar una reducción por mapeo o historia computacional para demostrar que otro lenguaje es indecidible [Aplicar]
8. Explicar el teorema de Rice y su importancia [Explicar]
9. Explicar un ejemplo de demostración de un problema que es incomputable reduciendo un problema incomputable clásico conocido a él [Explicar]
10. Explicar las funciones Recursivas Primitivas y Generales (cero, sucesor, selección, recursión primitiva, composición y Mu), su importancia e implementaciones en Máquina de Turing [Explicar]
11. Explicar cómo se realiza la computación en el Cálculo Lambda (por ejemplo, conversión alfa y reducción beta) [Explicar]

NonCore:

12. Para un sistema cuántico dar ejemplos que expliquen los siguientes postulados: *a*) Espacio de Estados - estado del sistema representado como un vector unitario en espacio de Hilbert *b*) Evolución del Estado - el uso de operadores unitarios para evolucionar el estado del sistema *c*) Composición de Estados - el uso de producto tensorial para componer estados del sistema *d*) Medición del Estado - la salida probabilística de medir un estado del sistema. [Explicar]
13. Explicar la operación de una compuerta cuántica XNOT o CNOT sobre un bit cuántico representado como una matriz y vector columna, respectivamente [Explicar]

2.2.11. AL/Sociedad, ética y la Profesión

Temas:
Core

- Algoritmos sociales, éticos y seguros
- Equidad algorítmica
- Anonimato (por ejemplo, Privacidad Diferencial)
- Responsabilidad/Transparencia
- Algoritmos responsables
- Impactos económicos y otros de algoritmos ineficientes
- Sostenibilidad
- Computación consciente del contexto

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Desarrollar soluciones algorítmicas a problemas sociales del mundo real, como enrutar una ambulancia a un hospital [Crear]
2. Explicar el impacto que un algoritmo puede tener en el medio ambiente y la sociedad cuando se usa para resolver un problema del mundo real considerando su sostenibilidad y que puede afectar a diferentes grupos sociales de diferentes maneras [Explicar]

3. Preparar una presentación que justifique la selección de estructuras de datos y/o algoritmos apropiados para resolver un problema del mundo real dado [Explicar]
4. Explicar un ejemplo que articule cómo la privacidad diferencial protege el conocimiento de los datos de un individuo [Explicar]
5. Explicar los impactos ambientales de las decisiones de diseño que se relacionan con el diseño de algoritmos [Explicar]
6. Explicar los compromisos involucrados en los algoritmos de prueba de trabajo y prueba de participación [Explicar]

2.3. Arquitectura y Organización (AR)

Los profesionales de la informática pasan considerable tiempo escribiendo código eficiente para resolver un problema particular en un dominio de aplicación. A medida que ocurre el cambio del procesamiento secuencial al paralelo, se requiere una comprensión más profunda de las arquitecturas de computadoras subyacentes. La arquitectura ya no puede verse como una caja negra donde los principios de una arquitectura pueden aplicarse a otra. En cambio, los programadores deben mirar dentro de la caja negra y usar componentes específicos para mejorar el rendimiento del sistema y la eficiencia energética.

El área de conocimiento de Arquitectura y Organización (AR) tiene como objetivo desarrollar una comprensión más profunda de los entornos de hardware sobre los cuales se basa casi toda la computación, y las interfaces relevantes proporcionadas a las capas de software superiores. El hardware objetivo comprende desde procesadores de sistemas embebidos de baja gama hasta multiprocesadores empresariales de alta gama.

Los temas en esta área de conocimiento beneficiarán a los estudiantes permitiéndoles apreciar los principios arquitectónicos fundamentales de los sistemas informáticos modernos, incluyendo el desafío de aprovechar el paralelismo para sostener las mejoras de rendimiento y energía en el futuro. Esta KA ayudará a los estudiantes de informática a alejarse del enfoque de caja negra y a ser más conscientes del sistema informático subyacente y las eficiencias que arquitecturas específicas pueden lograr.

Área de Conocimiento (KA)	**CS Core**	**KA Core**
⭐ 2.3.1 Lógica Digital y Sistemas Digitales (Pág. 33)		3
⭐ 2.3.2 Representación de Datos a Nivel de Máquina (Pág. 34)	1	
⭐ 2.3.3 Organización de Máquina a Nivel de Ensamblador (Pág. 34)	1	2
⭐ 2.3.4 Jerarquía de Memoria (Pág. 35)	6	
⭐ 2.3.5 Interconexión y Comunicación (Pág. 35)	1	
⭐ 2.3.6 Organización Funcional (Pág. 36)		2
⭐ 2.3.7 Rendimiento y Eficiencia Energética (Pág. 36)		3
⭐ 2.3.8 Arquitecturas Heterogéneas (Pág. 37)		2
⭐ 2.3.9 Arquitecturas de Procesador Seguras (Pág. 37)		2
⭐ 2.3.10 Arquitecturas Cuánticas (Pág. 38)		2
2.3.11 Problemas de Sostenibilidad (Pág. 38)	Electivo	

Cuadro 2.3: Lista de KUs del área de Arquitectura y Organización.

2.3.1. AR/Lógica Digital y Sistemas Digitales (KA Core: 3 hrs)

Temas:
Core

- Lógica combinacional vs secuencial/arreglos de puertas programables en campo (FPGAs): 1. Lógica combinacional fundamental 2. Bloque de construcción de lógica secuencial
- Herramientas de diseño asistido por computadora que procesan representaciones de hardware y arquitectura
- Síntesis de alto nivel: 1. Notación de transferencia de registros 2. Lenguaje de descripción de hardware (por ejemplo, Verilog/VHDL/Chisel)
- Flujo de diseño de sistema en chip (SoC)

- Restricciones físicas: 1. Retardos de puertas 2. Fan-in y fan-out 3. Energía/potencia 4. Velocidad de la luz

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Discutir la progresión de los componentes de tecnología informática desde tubos de vacío hasta VLSI, desde arquitecturas de computadoras mainframe hasta la organización de computadoras a escala de almacén [Debatir]
2. Describir el paralelismo y las dependencias de datos entre y dentro de componentes en una arquitectura de computadora heterogénea moderna [Describir]
3. Explicar la relación entre paralelismo y consumo de energía [Explicar]
4. Construir el diseño de bloques básicos para una computadora: unidad aritmético-lógica (nivel de puertas), registros (nivel de puertas), unidad central de procesamiento (nivel de transferencia de registros) y memoria (nivel de transferencia de registros) [Crear]
5. Evaluar bloques básicos simples (por ejemplo, unidad aritmético-lógica, registros, movimiento entre registros) de un diseño de computadora simple [Evaluar]
6. Analizar el comportamiento de temporización de un procesador segmentado, identificando problemas de dependencia de datos [Analizar]

2.3.2. AR/Representación de Datos a Nivel de Máquina (CS Core: 1 hr)

Temas:
Core

- Visión general e historia de la arquitectura de computadoras
- Bits, bytes y palabras
- Representaciones sin signo, con signo y complemento a dos
- Representación de datos numéricos y bases numéricas: 1. Punto fijo 2. Punto flotante
- Representación de datos no numéricos
- Representación de registros, arreglos y tipos de datos UTF

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Discutir por qué todo en las computadoras son datos, incluyendo instrucciones [Debatir]
2. Explicar cómo las representaciones numéricas de longitud fija pueden afectar la exactitud y precisión [Explicar]
3. Describir cómo se almacenan los enteros negativos en representaciones de magnitud con signo y complemento a dos [Describir]
4. Discutir cómo diferentes formatos pueden representar datos numéricos [Debatir]
5. Explicar la representación a nivel de bits de datos no numéricos, como caracteres, cadenas, registros y arreglos [Explicar]
6. Traducir datos numéricos de un formato a otro [Traducir]
7. Describir cómo un sumador único (sin detección de desbordamiento) puede manejar tanto entrada con signo (complemento a dos) como sin signo (binario) sin ”saber” qué formato está usando una entrada dada [Describir]

2.3.3. AR/Organización de Máquina a Nivel de Ensamblador (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Arquitectura de máquina von Neumann
- Unidad de control: captación, decodificación y ejecución de instrucciones
- Introducción a SIMD vs MIMD y la taxonomía de Flynn
- Organización de multiprocesadores/multinúcleo de memoria compartida
- Arquitectura del conjunto de instrucciones (ISA) (por ejemplo, x86, ARM y RISC-V): 1. Conjuntos de instrucciones de ancho fijo vs variable 2. Formatos de instrucción 3. Manipulación de datos, control, E/S 4. Modos de direccionamiento 5. Programación en lenguaje de máquina 6. Programación en lenguaje ensamblador

- Mecanismos de llamada y retorno de subrutinas
- E/S e interrupciones
- Segmentos de montículo, estáticos, pila y código

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Discutir cómo las unidades funcionales clásicas de von Neumann se implementan en sistemas embebidos, particularmente en memoria dentro y fuera del chip [Debatir]
2. Describir cómo se ejecutan las instrucciones en una máquina von Neumann clásica, con extensiones para hilos, sincronización de multiprocesadores y ejecución SIMD [Describir]
3. Evaluar un diagrama de ejemplo con paralelismo a nivel de instrucción y riesgos para describir cómo se manejan en las canalizaciones de procesadores típicos [Evaluar (valorar)]
4. Discutir cómo se representan las instrucciones a nivel de máquina y en el contexto de un ensamblador simbólico [Debatir]
5. Mapear un ejemplo de patrones de lenguaje de alto nivel a notaciones de lenguaje ensamblador/máquina [Mapear/Mapa conceptual]
6. Contrastar diferentes formatos de instrucción considerando aspectos como direcciones por instrucción y formatos de longitud variable vs longitud fija [Contrastar]
7. Analizar un diagrama de subrutina para comentar cómo se manejan las llamadas de subrutina a nivel de ensamblador [Analizar]
8. Describir conceptos básicos de interrupciones y operaciones de E/S [Describir]
9. Escribir un programa simple en lenguaje ensamblador para procesamiento y manipulación de cadenas/arreglos [Escribir]

2.3.4. AR/Jerarquía de Memoria (CS Core: 6 hrs)

Temas:
Core

- Jerarquía de memoria: la importancia de la localidad temporal y espacial
- Organización y operaciones de memoria principal
- Memoria persistente (por ejemplo, SSD, discos estándar)
- Latencia, tiempo de ciclo, ancho de banda e intercalado
- Memorias caché: 1. Mapeo de direcciones 2. Tamaño de bloque 3. Política de reemplazo y almacenamiento 4. Prebúsqueda
- Coherencia de caché multiprocesador
- Memoria virtual (soporte de hardware)
- Manejo de fallas y confiabilidad
- Confiabilidad: 1. Codificación de error 2. Compresión de datos 3. Integridad de datos
- Procesamiento en Memoria (PIM)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Usando un diagrama del sistema de memoria, identificar los principales tipos de tecnología de memoria (por ejemplo, SRAM, DRAM) y su costo y rendimiento relativos [Analizar]
2. Medir el efecto de la latencia de memoria en el tiempo de ejecución [Evaluar]
3. Enumerar las funciones de un sistema con gestión de memoria virtual [Enumerar]
4. Calcular el tiempo promedio de acceso a memoria bajo varias configuraciones de caché y memoria y mezclas de referencias de instrucciones y datos [Computar/Calcular]

2.3.5. AR/Interconexión y Comunicación (CS Core: 1 hr)

Temas:
Core

- Fundamentos de E/S: 1. Protocolo de enlace y almacenamiento en búfer 2. E/S programada 3. E/S basada en interrupciones
- Estructuras de interrupción: vectorizadas y priorizadas, acuse de recibo de interrupción
- Dispositivos de E/S (por ejemplo, ratón, teclado, pantalla, cámara, sensores, actuadores)
- Almacenamiento externo, organización física y unidades

- Fundamentos de buses: 1. Protocolos de bus 2. Arbitraje 3. Acceso directo a memoria (DMA)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Analizar un diagrama de control de interrupciones para comentar cómo se usan las interrupciones para implementar control de E/S y transferencias de datos [Analizar]
2. Enumerar varios tipos de buses en un sistema informático [Enumerar]
3. Enumerar las ventajas de los discos magnéticos y contrastarlas con las de los discos de estado sólido [Listar/Enumerar]

2.3.6. AR/Organización Funcional (KA Core: 2 hrs)

Temas:
Core

- Implementación de rutas de datos simples, incluyendo segmentación de instrucciones, detección de riesgos y resolución (por ejemplo, paradas, reenvío)
- Unidad de control: 1. Implementación cableada 2. Realización microprogramada
- Segmentación de instrucciones
- Introducción al paralelismo a nivel de instrucción (ILP)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comparar implementaciones alternativas de rutas de datos en arquitecturas de computadoras modernas [Comparar]
2. Producir un conjunto de señales de control para sumar dos enteros usando implementaciones cableadas y microprogramadas [Producir]
3. Discutir el paralelismo a nivel de instrucción usando segmentación y los riesgos significativos que pueden ocurrir [Debatir]
4. Diseñar un procesador completo, incluyendo ruta de datos y control [Diseñar]
5. Calcular los ciclos promedio por instrucción para una implementación dada de procesador y sistema de memoria [Computar/Calcular]

2.3.7. AR/Rendimiento y Eficiencia Energética (KA Core: 3 hrs)

Temas:
Core

- Evaluación de rendimiento-energía (introducción): rendimiento, consumo de energía, costos de memoria y comunicación
- Predicción de bifurcaciones, ejecución especulativa, ejecución fuera de orden, algoritmo de Tomasulo
- Mejoras para procesadores vectoriales y GPUs
- Soporte de hardware para multihilos: 1. Condiciones de carrera 2. Implementaciones de bloqueo 3. Sincronización punto a punto 4. Implementación de barreras
- Escalabilidad
- Arquitecturas alternativas incluyendo VLIW/EPIC, aceleradores y otros procesadores de propósito especial
- Escalado dinámico de voltaje y frecuencia (DVFS)
- Silicio Oscuro

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Discutir métricas de evaluación de rendimiento y eficiencia energética [Debatir]
2. Analizar un diagrama de ejecución especulativa y escribir sobre las decisiones que se pueden tomar [Analizar]
3. Crear un diagrama de evaluación comparativa de rendimiento-vatio de GPU [Crear]
4. Escribir un programa multihilo que sume (en paralelo) elementos de dos vectores enteros [Escribir]
5. Recomendar un conjunto de decisiones de diseño para arquitecturas de computadoras alternativas [Recomendar]

6. Enumerar conceptos clave asociados con el escalado dinámico de voltaje y frecuencia [Enumerar]
7. Medir la mejora en ahorro de energía para una cuantización de entero de 8 bits comparada con una cuantización de 32 bits [Evaluar]

2.3.8. AR/Arquitecturas Heterogéneas (KA Core: 2 hrs)

Temas:
Core

- Arquitecturas SIMD y MIMD (por ejemplo, GPUs de Propósito General, TPUs y NPUs)
- Sistemas de memoria heterogéneos: 1. Memoria compartida versus memoria distribuida 2. Memoria volátil vs no volátil 3. Protocolos de coherencia
- Arquitecturas Específicas de Dominio (DSAs): 1. Acelerador de Aprendizaje Automático 2. Computación en red 3. Sistemas embebidos para aplicaciones emergentes 4. Computación neuromórfica 5. Dispositivos de computación perimetral
- Soluciones de empaquetado e integración como 3DIC y chiplets
- Aprendizaje automático en diseño de arquitectura: 1. Algoritmos de IA para análisis de carga de trabajo 2. Optimización de configuraciones de arquitectura para rendimiento y eficiencia energética

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Analizar un diagrama de sistema con arquitecturas paralelas alternativas, por ejemplo, SIMD y MIMD, e identificar las diferencias clave [Analizar]
2. Discutir qué problemas de gestión de memoria se encuentran en multiprocesadores que no están presentes en uniprocesadores y cómo estos problemas podrían resolverse [Debatir]
3. Indicar las diferencias entre placa base de memoria, interconexión de memoria del procesador y memoria remota a través de redes, sus implicaciones para la latencia de acceso y su impacto en el rendimiento del programa [Diferenciar]
4. Discutir cómo determinarías cuándo usar un acelerador específico de dominio en lugar de una CPU de propósito general [Debatir]
5. Enumerar diferencias clave en los principios de diseño arquitectónico entre una unidad de procesamiento basada en vector y basada en escalar [Enumerar]
6. Enumerar las ventajas y desventajas de una arquitectura PIM [Listar/Enumerar]

2.3.9. AR/Arquitecturas de Procesador Seguras (KA Core: 2 hrs)

Temas:
Core

- Principios de Hardware Seguro: 1. Análisis de Riesgos de Seguridad, Protección de Activos y Modelo de Amenazas 2. Aceleración Criptográfica con Hardware 3. Soporte para virtualización (por ejemplo, aislamiento de SO)
- Raíces de confianza en hardware, Funciones Físicamente No Clonables (PUF)
- Generadores de Números Aleatorios por Hardware
- Extensiones de protección de memoria: 1. Verificación de límites de puntero en tiempo de ejecución (por ejemplo, desbordamiento de búfer) 2. Protección a nivel microarquitectónico 3. Protección a nivel de ISA
- Entorno de Ejecución Confiable (TEE): 1. Protecciones de Base de Computadora Confiable 2. Protección de máquinas virtuales 3. Protección de contenedores 4. Módulos de software confiables (Enclaves)
- Cifrado homomórfico para procesamiento de datos que preserva la privacidad

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Discutir principios de hardware seguro, explorando un marco para análisis de riesgos y protección de activos [Debatir]
2. Resumir cómo las Funciones Físicamente No Clonables (PUF) pueden ser un identificador único de dispositivo en aplicaciones de seguridad [Resumir]

3. Distinguir un generador de números aleatorios con soporte de hardware dedicado de generadores sin hardware dedicado a generar entropía [Distinguir]
4. Enumerar las ventajas y desventajas de la protección de memoria a nivel de ISA [Listar/Enumerar]
5. Describir problemas de diseño clave de un entorno de ejecución confiable (TEE) para soportar máquinas virtuales [Describir]

2.3.10. AR/Arquitecturas Cuánticas (KA Core: 2 hrs)

Temas:
Core

- Principios: 1. El principio de dualidad onda-partícula 2. El principio de incertidumbre en el experimento de doble rendija 3. ¿Qué es un Qubit? Superposición, interferencia y medición. Fotones como qubits 4. Sistemas de dos qubits, Entrelazamiento, estados de Bell, El teorema de No Señalización
- Axiomas de la MC: principio de superposición, axioma de medición, evolución unitaria
- Compuertas de un solo qubit para el modelo de circuito de computación cuántica: X, Z, H
- Compuertas de dos qubits y productos tensoriales, trabajando con matrices
- El Teorema de No Clonación. El protocolo de Teleportación Cuántica
- Algoritmos: 1. Algoritmos cuánticos simples: Bernstein-Vazirani, algoritmo de Simon 2. Implementar Deutsch-Josza con Interferómetros Mach-Zehnder 3. Factorización cuántica (Algoritmo de Shor) 4. Búsqueda cuántica (Algoritmo de Grover)
- Aspectos de implementación: 1. La implementación física de qubits 2. Control clásico de una Unidad de Procesamiento Cuántico (QPU) 3. Mitigación y control de errores, NISQ y más allá 4. Enfoques de medición
- Aplicaciones Emergentes: 1. Cifrado post-cuántico 2. La Internet Cuántica 3. Computación cuántica adiabática (AQC) y recocido cuántico

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Discutir cómo un objeto cuántico producido como partícula se propaga como una onda y se detecta como partícula con una distribución de probabilidad correspondiente a la onda [Debatir]
2. Discutir la naturaleza a nivel cuántico que es inherentemente probabilística [Debatir]
3. Expresar tu punto de vista sobre el entrelazamiento que puede usarse para crear correlaciones no clásicas, pero no hay forma de usar el entrelazamiento cuántico para enviar mensajes más rápido que la velocidad de la luz [Evaluar]
4. Describir el paralelismo cuántico y el papel de la interferencia constructiva vs destructiva en algoritmos cuánticos dada la naturaleza probabilística de las mediciones [Describir]
5. Analizar un fragmento de código proporcionando el papel de la transformada cuántica de Fourier (QFT) en el algoritmo de Shor [Analizar]
6. Escribir un programa para implementar el algoritmo de Shor en un simulador, destacando los componentes clásicos y aspectos del algoritmo de Shor [Escribir]
7. Enumerar los detalles de cada modalidad de qubit (por ejemplo, ion atrapado, superconductor, espín de silicio, fotónico, punto cuántico, átomo neutro, topológico, centro de color, electrón sobre helio) [Enumerar]
8. Contrastar AQC con el modelo de compuertas de computación cuántica y los problemas para los que cada uno es más adecuado resolver [Contrastar]

2.3.11. AR/Problemas de Sostenibilidad

Temas:
Non Core

- Impactos ambientales de decisiones de implementación: 1. Objetivos de sostenibilidad, consumo de recursos y viabilidad económica 2. Huella de carbono, desechos electrónicos de hardware 3. La huella energética de los centros de datos en varias cargas de trabajo (por ejemplo, entrenamiento y uso de modelos de IA) 4. Directrices para estándares de diseño sostenible

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Evaluar los impactos ambientales del despliegue de un proyecto dado (por ejemplo, el consumo de energía de CPUs y GPUs, contribución a desechos electrónicos, y efecto de la virtualización de hardware en centros de datos) [Evaluar (valorar)]

2.4. Gestión de Datos (DM)

Desde mediados de la década de 1970, el estudio de la Gestión de Datos (DM) ha significado un estudio casi exclusivo de los sistemas de bases de datos relacionales. Dependiendo del contexto institucional, los estudiantes han estudiado, en proporciones variables, lo siguiente.

- Modelado de datos y diseño de bases de datos: por ejemplo, modelo de datos E-R, modelo relacional, teoría de normalización
- Construcción de consultas: por ejemplo, álgebra relacional, SQL
- Procesamiento de consultas: por ejemplo, índices (árbol B+, hash), algoritmos (por ejemplo, ordenamiento externo, selección, proyección, unión), optimización de consultas (transformaciones, selección de índices)
- Internos de DBMS: por ejemplo, concurrencia/bloqueos, gestión de transacciones, gestión de búfer

Se espera que los graduados de hoy posean habilidades de usuario (en lugar de implementador) de DBMS. Estas incluyen principalmente modelado de datos y construcción de consultas; capacidad para tomar una colección no organizada de datos, organizarla usando un DBMS y acceder/actualizar la colección mediante consultas.

Además, los estudiantes necesitan estudiar lo siguiente:

- El rol que juegan los datos en una organización. Esto incluye el Ciclo de Vida de los Datos: Creación-Procesamiento-Revisión/Reporte-Retención/Recuperación-Destrucción.
- Los aspectos sociales/legales de la recolección de datos: por ejemplo, escala, privacidad de datos, privacidad de base de datos (cumplimiento) por diseño, desidentificación, propiedad, confiabilidad, seguridad de base de datos y aplicaciones intencionales y no intencionales.
- Tecnologías emergentes y avanzadas que están aumentando/reemplazando los sistemas relacionales tradicionales, particularmente aquellas usadas para soportar análisis de datos (grandes), incluyendo NoSQL (por ejemplo, JSON, XML, bases de datos de almacén clave-valor), bases de datos en la nube, MapReduce y dataframes.
- Los roles existentes y emergentes para aquellos involucrados con la gestión de datos, que incluyen los siguientes.
  - Profesionales de características de producto: aquellos que usan tanto bases de datos operativas SQL como NoSQL.
  - Profesionales analíticos/Profesionales en datos: aquellos que escriben código SQL, Python y Scala analítico para construir activos de datos para grupos de negocios.
  - Analistas de negocio: aquellos que construyen/gestionan datos más frecuentemente con hojas de cálculo Excel.
  - Profesionales de infraestructura de datos: aquellos que implementan un sistema de gestión de datos en una variedad de aplicaciones de datos (por ejemplo, OLTP).
  - **Todos** los que producen o consumen datos deben comprender los problemas sociales, éticos y profesionales asociados.

Un rol que trasciende todas las categorías anteriores es el de custodio de datos. Anteriormente, los datos eran vistos como un recurso para ser gestionado (Gestión de Sistemas de Información) al igual que otros recursos empresariales. Hoy, los datos son vistos en un contexto más amplio. Los datos sobre clientes ahora pueden verse como pertenecientes a (o en algunos contextos nacionales, como propiedad de) esos clientes. Ahora existe una comprensión aceptada de que el almacenamiento y uso seguro y ético de los datos institucionales es parte de ser un custodio de datos responsable.

Además, reconocemos la tensión entre un enfoque curricular en preparación profesional versus el estudio de un área de conocimiento como un esfuerzo científico. Esto es particularmente cierto con la Gestión de Datos. Por ejemplo, demostrar (o al menos conocer) la completitud de los Axiomas de Armstrong es fundamental en la teoría de dependencias funcionales. Sin embargo, la mayoría de los graduados en informática nunca utilizarán este concepto durante sus carreras profesionales. Lo mismo puede decirse de muchos otros temas en el canon de Gestión de Datos. Por el contrario, si nuestros

graduados solo pueden normalizar datos en forma normal de Boyce-Codd (usando una herramienta automatizada) y escribir consultas SQL, sin comprender el rol que juegan los índices en la ejecución eficiente de consultas, les hemos hecho un perjuicio a ellos y a la sociedad.

Con este fin, el número de horas de Núcleo CS es relativamente pequeño en relación con las horas de Núcleo KA. Este enfoque está diseñado para permitir que las instituciones con contextos diferentes personalicen sus planes de estudio apropiadamente. Una institución que se enfoque en implementación OLTP, por ejemplo, priorizaría el almacenamiento eficiente y el acceso a datos, mientras que una institución que se enfoque en características de producto priorizaría el acceso programático a bases de datos existentes.

Sin embargo, como una institución maneje esta tensión, deseamos dar voz a una de las ironías de los planes de estudio de informática. Los estudiantes típicamente pasan gran parte de su vida educativa leyendo (y escribiendo) datos de un archivo o interactivamente, mientras que fuera de la academia los datos predominantes provienen de bases de datos a las que se accede programáticamente. Quizás en un futuro no muy lejano los estudiantes aprenderán acceso programático a bases de datos desde temprano y luego continuarán esta práctica a medida que avanzan en su plan de estudios.

Finalmente, entendemos que aunque el KA de Gestión de Datos puede ser ortogonal a los KAs de SEC (Seguridad) y SEP (Sociedad, ética y la Profesión), también es punto cero para estas (y otras) áreas de conocimiento. Al diseñar almacenes de datos persistentes, la pregunta de qué debe almacenarse debe examinarse desde perspectivas legales y éticas. ¿Hay preocupaciones de privacidad? Y tan importante como eso, ¿qué tan bien protegidos están los datos?

Área de Conocimiento (KA)	**CS Core**	**KA Core**
⭐ 2.4.1 El Rol de los Datos y el Ciclo de Vida de los Datos (Pág. 40)	2	
⭐ 2.4.2 Conceptos Fundamentales de Sistemas de Bases de Datos (Pág. 40)	2	1
⭐ 2.4.3 Modelado de Datos (Pág. 41)	2	3
⭐ 2.4.4 Bases de Datos Relacionales (Pág. 42)	1	3
⭐ 2.4.5 Construcción de Consultas (Pág. 42)	2	4
⭐ 2.4.6 Procesamiento de Consultas (Pág. 43)		4
⭐ 2.4.7 Internos de DBMS (Pág. 43)		4
⭐ 2.4.8 Sistemas NoSQL (Pág. 44)		2
⭐ 2.4.9 Seguridad y Privacidad de Datos (Pág. 44)	1	2
⭐ 2.4.10 Análisis de Datos (Pág. 45)		3
2.4.11 Bases de Datos Distribuidas/Computación en la Nube (Pág. 45)	Electivo	
2.4.12 Bases de Datos Semiestructuradas y No Estructuradas (Pág. 45)	Electivo	
2.4.13 Sociedad, ética y Profesión (Pág. 46)	Electivo	

Cuadro 2.4: Lista de KUs del área de Gestión de Datos.

2.4.1. DM/El Rol de los Datos y el Ciclo de Vida de los Datos (CS Core: 2 hrs)

Temas:
Core

- Ciclo de Vida de los Datos: Creación-Procesamiento-Revisión/Reporte-Retención/Recuperación-Destrucción

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar las cinco etapas del Ciclo de Vida de los Datos [Analizar]

2.4.2. DM/Conceptos Fundamentales de Sistemas de Bases de Datos (CS Core: 2 hrs, KA Core: 1 hr)

Temas:
Core

- Propósito y ventajas de los sistemas de bases de datos
- Componentes de los sistemas de bases de datos

- Diseño de funciones fundamentales de DBMS (por ejemplo, mecanismos de consulta, gestión de transacciones, gestión de búfer, métodos de acceso)
- Arquitectura de base de datos, independencia de datos y abstracción de datos
- Gestión de transacciones
- Normalización
- Enfoques para gestionar grandes volúmenes de datos (por ejemplo, sistemas de bases de datos NoSQL, uso de MapReduce)
- Cómo soportar aplicaciones solo de CRUD
- Bases de datos distribuidas/sistemas basados en la nube
- Datos estructurados, semiestructurados y no estructurados
- Uso de un lenguaje de consulta declarativo
- Sistemas que soportan contenido estructurado y/o de flujo

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar al menos cuatro ventajas que proporciona el uso de un sistema de base de datos [Analizar]
2. Enumerar los componentes de un sistema de base de datos (relacional) [Enumerar]
3. Seguir una consulta a medida que es procesada por los componentes de un sistema de base de datos (relacional) [Analizar]
4. Defender el valor de la independencia de datos [Defender]
5. Componer una consulta simple de selección-proyección-unión en SQL [Componer]
6. Enumerar las cuatro propiedades de un gestor de transacciones correcto [Enumerar]
7. Describir las ventajas de eliminar datos duplicados repetidos [Describir]
8. Esbozar cómo MapReduce usa paralelismo para procesar datos eficientemente [Esquematizar/Esbozar]
9. Evaluar las diferencias entre bases de datos estructuradas y semiestructuradas/no estructuradas [Evaluar]

2.4.3. DM/Modelado de Datos (CS Core: 2 hrs, KA Core: 3 hrs)

Temas:
Core

- Modelado de datos
- Modelo de datos relacional
- Modelos conceptuales (por ejemplo, entidad-relación, diagramas UML)
- Modelos de datos semiestructurados (expresados usando DTD, XML o JSON Schema, por ejemplo)

Non Core

- Modelos de hojas de cálculo
- Modelos orientados a objetos: 1. GraphQL
- Nuevas características en SQL
- Temas de Modelado de Datos Especializados: 1. Datos de series temporales (agregación, unión) 2. Datos de grafos (recorrido de enlaces) 3. Técnicas para evitar acceso ineficiente a datos crudos (por ejemplo, "precio promedio diario"): vistas materializadas y estructuras de datos especiales (por ejemplo, Hyperloglog, mapa de bits) 4. Datos Geoespaciales (por ejemplo, bases de datos GIS)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir los componentes del modelo de datos relacional [Describir]
2. Modelar relaciones 1:1, 1:n y n:m usando el modelo de datos relacional [Usar]
3. Describir los componentes del modelo E-R (o algún otro modelo no relacional) [Describir]
4. Modelar un entorno dado usando un modelo de datos conceptual [Analizar]
5. Modelar un entorno dado usando el modelo de datos basado en documentos o basado en almacén clave-valor [Aplicar]

2.4.4. DM/Bases de Datos Relacionales (CS Core: 1 hr, KA Core: 3 hrs)

Temas:
Core

- Integridad de entidad y referencial: Clave candidata, superclaves
- Diseño de base de datos relacional
- Mapeo de esquema conceptual a un esquema relacional
- Diseño físico de base de datos: estructuras de archivos y almacenamiento
- Introducción a la teoría de dependencias funcionales
- Teoría de Normalización: 1. Descomposición de un esquema; propiedades de unión sin pérdida y preservación de dependencias de una descomposición 2. Formas normales (BCNF) 3. Desnormalización (para eficiencia)

Non Core

- Teoría de dependencias funcionales: 1. Clausura de un conjunto de atributos 2. Cubierta canónica
- Teoría de normalización: 1. Dependencia multivaluada (4NF) 2. Dependencia de unión (PJNF, 5NF) 3. Teoría de representación

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir las características definitorias detrás del modelo de datos relacional [Describir]
2. Comentar sobre la diferencia entre una clave foránea y una superclave [Explicar]
3. Enumerar los diferentes tipos de restricciones de integridad [Enumerar]
4. Componer un esquema relacional a partir de un esquema conceptual que contenga relaciones 1:1, 1:n y n:m [Componer]
5. Mapear estructura de archivo apropiada a relaciones e índices [Mapear/Mapa conceptual]
6. Describir cómo la teoría de dependencias funcionales generaliza la noción de clave [Describir]
7. Defender una descomposición dada como sin pérdida y/o que preserva dependencias [Defender]
8. Detectar qué forma normal produce una descomposición dada [Detectar]
9. Comentar sobre las razones para desnormalizar una relación [Interpretar]

2.4.5. DM/Construcción de Consultas (CS Core: 2 hrs, KA Core: 4 hrs)

Temas:
Core

- Formación de Consultas SQL: 1. Ejecución interactiva de SQL 2. Ejecución programática de una consulta SQL
- álgebra Relacional
- SQL: 1. Definición de datos incluyendo especificaciones de integridad y otras restricciones 2. Sublenguaje de actualización

Non Core

- Cálculo Relacional
- QBE y entornos de 4ta generación
- Diferentes formas de invocar consultas no procedimentales en lenguajes convencionales
- Introducción a otros lenguajes de consulta principales (por ejemplo, XPATH, SPARQL)
- Procedimientos almacenados

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Componer consultas SQL que incorporen selección, proyección, unión, intersección, diferencia de conjuntos y división de conjuntos [Componer]
2. Determinar cuándo una consulta SQL anidada está correlacionada o no [Determinar]
3. Iterar sobre datos recuperados programáticamente de una base de datos a través de una consulta SQL [Crear]
4. Definir, en SQL, un esquema de relación, incluyendo todas las restricciones de integridad y activadores de eliminación/actualización [Definir]
5. Componer una consulta SQL para actualizar una tupla en una relación [Componer]

2.4.6. DM/Procesamiento de Consultas (KA Core: 4 hrs)

Temas:
Core

- Estructuras de página
- Estructuras de índice: 1. árboles B+ 2. Índices hash: estáticos y dinámicos 3. Creación de índices en SQL
- Estructuras de archivos: 1. Archivos de montón (*heap*) 2. Archivos hash
- Algoritmos para operadores de consulta: 1. Ordenamiento externo 2. Selección 3. Proyección; con y sin eliminación de duplicados 4. Uniones naturales: Bucle anidado, Mezcla-ordenada, Unión hash 5. Análisis de eficiencia de algoritmos
- Transformaciones de consultas
- Optimización de consultas: 1. Rutas de acceso 2. Construcción de plan de consulta 3. Estimación de selectividad 4. Planes solo de índice
- Procesamiento de Consultas en Paralelo (por ejemplo, exploración paralela, unión paralela, agregación paralela)
- Afinamiento/rendimiento de base de datos: 1. Selección de índices 2. Impacto de índices en el rendimiento de consultas 3. Desnormalización

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir el propósito y organización de las estructuras de índice tanto de árbol B+ como hash [Describir]
2. Componer un comando SQL para crear un índice (de cualquier tipo) [Componer]
3. Especificar los pasos para los diversos algoritmos de operadores de consulta: ordenamiento externo, proyección con eliminación de duplicados, unión por mezcla-ordenada, unión hash, unión por bucle anidado por bloques [Analizar]
4. Derivar el tiempo de ejecución (en solicitudes de E/S) para cada uno de los algoritmos anteriores [Analizar]
5. Transformar una consulta en álgebra relacional a su equivalente apropiado para una ejecución canalizada de profundidad izquierda [Rediseñar]
6. Calcular estimaciones de selectividad para una operación de selección y/o unión dada [Computar/Calcular]
7. Describir cómo modificar una estructura de índice para facilitar una operación solo de índice para una relación dada [Describir]
8. Para un escenario dado decidir qué índices soportar para la ejecución eficiente de un conjunto de consultas [Evaluar (valorar)]
9. Describir cómo los DBMS aprovechan el paralelismo para acelerar el procesamiento de consultas dividiendo el trabajo entre múltiples procesadores o nodos [Describir]

2.4.7. DM/Internos de DBMS (KA Core: 4 hrs)

Temas:
Core

- Gestión de Búfer de DB
- Gestión de Transacciones: 1. Niveles de Aislamiento 2. ACID 3. Serializabilidad 4. Transacciones Distribuidas
- Control de Concurrencia: 1. Bloqueo en 2 Fases 2. Estrategias de manejo de bloqueos mutuos 3. Modelos de consistencia basados en quórum
- Gestor de Recuperación: 1. Relación con el Gestor de Búfer

Non Core

- Control de Concurrencia: 1. Control de concurrencia optimista 2. Control de concurrencia por marca de tiempo
- Gestor de Recuperación: 1. Registro por escritura anticipada 2. Sistema de recuperación ARIES (Análisis, REDO, UNDO)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir cómo un DBMS gestiona su Grupo de Búferes [Describir]

2. Describir las cuatro propiedades para un gestor de transacciones correcto [Describir]
3. Esbozar el principio de serializabilidad [Esquematizar/Esbozar]

2.4.8. DM/Sistemas NoSQL (KA Core: 2 hrs)

Temas:
Core

- ¿Por qué NoSQL? (por ejemplo, Desajuste de impedancia entre Aplicación [CRUD] y RDBMS)
- Modelo de datos Clave-Valor y Documento

Non Core

- Sistemas de almacenamiento (por ejemplo, sistemas Clave-Valor, Lagos de Datos (*Data Lakes*))
- Modelos de Distribución (Fragmentación y Replicación)
- Bases de Datos de Grafos
- Modelos de Consistencia (Actualización y Lectura, consistencia de quórum, teorema CAP)
- Modelo de procesamiento (por ejemplo, Map-Reduce, map-reduce multi-etapa, map-reduce incremental)
- Estudios de Caso: Sistema de almacenamiento en la nube (por ejemplo, S3); Bases de datos de grafos; "Cuándo no usar NoSQL"

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Desarrollar un caso de uso para el uso de NoSQL sobre RDBMS [Crear]
2. Describir las características definitorias detrás de los modelos de datos basados en Clave-Valor y Documentos [Describir]

2.4.9. DM/Seguridad y Privacidad de Datos (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Diferencias entre seguridad de datos y privacidad de datos
- Proteger datos y sistemas de bases de datos de ataques, incluyendo ataques de inyección como inyección SQL
- Información de identificación personal (PII) y su protección
- Consideraciones éticas para asegurar la seguridad y privacidad de los datos
- Necesidad de, y diferentes enfoques para asegurar datos en reposo, en tránsito y durante el procesamiento
- Auditoría de base de datos y su rol en forensía digital
- Inferencia de datos y prevención de ataques
- Leyes y regulaciones que gobiernan la seguridad de datos y la privacidad de datos

Non Core

- Factores de riesgo típicos y medidas de prevención para asegurar la integridad de datos
- Ransomware y prevención de pérdida y destrucción de datos

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir las diferencias en los objetivos para seguridad de datos y privacidad de datos [Describir]
2. Identificar y mitigar riesgos asociados con diferentes enfoques para proteger datos [Analizar]
3. Describir consideraciones legales y éticas de seguridad y privacidad de datos de extremo a extremo [Describir]
4. Desarrollar un sistema de auditoría de base de datos dadas consideraciones de riesgo [Crear]
5. Aplicar varios enfoques de exploración de datos para comprender conjuntos de datos no familiares [Aplicar]

2.4.10. DM/Análisis de Datos (KA Core: 3 hrs)

Temas:
Core

- Técnicas exploratorias de datos (motivación, representación, estadísticas descriptivas, visualizaciones)
- Ciclo de vida de ciencia de datos: comprensión del negocio, comprensión de datos, preparación de datos, modelado, evaluación, despliegue y aceptación del usuario
- Algoritmos de minería de datos y aprendizaje automático: por ejemplo, clasificación, agrupamiento, asociación, regresión
- Adquisición y gobernanza de datos
- Consideraciones de seguridad y privacidad de datos
- Equidad y sesgo de datos
- Técnicas de visualización de datos y su uso en análisis de datos
- Resolución de Entidades

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir varios enfoques de exploración de datos, incluyendo visualización, para comprender conjuntos de datos no familiares [Describir]
2. Aplicar varios enfoques de exploración de datos para comprender conjuntos de datos no familiares [Aplicar]
3. Describir algoritmos básicos de aprendizaje automático/minería de datos y cuándo son apropiados para su uso [Describir]
4. Aplicar varios algoritmos de aprendizaje automático/minería de datos [Aplicar]
5. Describir consideraciones legales y éticas en la adquisición, uso y modificación de conjuntos de datos [Describir]
6. Describir problemas de equidad y sesgo en la recolección y uso de datos [Describir]

2.4.11. DM/Bases de Datos Distribuidas/Computación en la Nube

Temas:
Non Core

- DBMS Distribuido: 1. Almacenamiento de datos distribuido 2. Procesamiento de consultas distribuido 3. Modelo de transacción distribuida 4. Soluciones homogéneas y heterogéneas 5. Bases de datos distribuidas cliente-servidor
- DBMS Paralelo: 1. Arquitecturas de DBMS paralelo: memoria compartida, disco compartido, nada compartido 2. Aceleración y escalamiento, por ejemplo, uso del modelo de procesamiento MapReduce 3. Replicación de datos y modelos de consistencia débil

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Describir los componentes clave de un DBMS distribuido, incluyendo almacenamiento de datos distribuido, procesamiento de consultas y gestión de transacciones [Describir]
2. Analizar las ventajas y desventajas entre arquitecturas de DBMS paralelo: memoria compartida, disco compartido y nada compartido [Analizar]
3. Describir estrategias de replicación de datos y modelos de consistencia débil en sistemas de bases de datos distribuidas [Describir]

2.4.12. DM/Bases de Datos Semiestructuradas y No Estructuradas

Temas:
Non Core

- Datos no estructurados vectorizados (texto, video, audio, etc.) y almacenamiento vectorial: 1. Vectorizador TF-IDF con n-grama 2. Word2Vec 3. Base de datos de arreglos o manejo de tipo de datos arreglo
- Bases de datos semiestructuradas (por ejemplo, JSON): 1. Almacenamiento: *a*) Codificación y compresión de tipos de datos anidados 2. Indexación: *a*) árbol B, índice de salto, filtro Bloom *b*) Índice invertido y compresión de mapa de bits *c*) Indexación por curva de llenado de espacio

para datos geo-semiestructurados 3. Procesamiento de consultas para casos de uso OLTP y OLAP: *a*) Compromisos de inserción, selección, actualización/eliminación *b*) Estudios de caso sobre Postgres/JSON, MongoDB y Snowflake/JSON

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Describir representaciones vectorizadas de datos no estructurados (por ejemplo, TF-IDF, Word2Vec) y sus modelos de almacenamiento [Describir]
2. Aplicar técnicas de indexación (por ejemplo, árbol B, índice invertido, filtro Bloom) a datos semiestructurados almacenados en formato JSON [Aplicar]
3. Analizar las ventajas y desventajas del procesamiento de consultas en bases de datos semiestructuradas para casos de uso OLTP y OLAP [Analizar]

2.4.13. DM/Sociedad, ética y Profesión

Temas:
Core

- Problemas relacionados con la escala
- Privacidad de datos en general: 1. Privacidad por diseño y por defecto
- Anonimato de datos
- Propiedad/custodia de datos
- Aplicaciones intencionales y no intencionales de los datos almacenados
- Fiabilidad de los datos
- Procedencia, linaje de datos y gestión de metadatos
- Seguridad de datos

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enumerar tres problemas sociales y tres legales relacionados con grandes colecciones de datos [Enumerar]
2. Describir el valor de la privacidad de datos [Describir]
3. Identificar las partes interesadas en competencia con respecto a la propiedad de los datos [Analizar]
4. Enumerar tres consecuencias negativas no intencionales de una aplicación centrada en datos (bien conocida) dada (por ejemplo, Facebook, LastPass, Ashley Madison) [Enumerar]
5. Describir el significado de procedencia y linaje de datos [Describir]
6. Identificar cómo una base de datos podría contribuir a la seguridad de los datos, así como cómo puede introducir inseguridades [Analizar]

2.5. Fundamentos de los Lenguajes de Programación (FPL)

Los fundamentos de los lenguajes de programación están enraizados en las matemáticas discretas, la lógica y los lenguajes formales, y proporcionan una base para la comprensión de los complejos lenguajes de programación modernos. Aunque los lenguajes de programación varían según el paradigma del lenguaje y el dominio del problema, y evolucionan en respuesta tanto a las necesidades sociales como al avance tecnológico, comparten un modelo abstracto subyacente de computación y desarrollo de programas. Esto sigue siendo cierto incluso a medida que el hardware del procesador y su interfaz con las herramientas de programación se entrelazan y complejizan progresivamente. Una comprensión de las abstracciones y paradigmas de programación comunes permite un aprendizaje más rápido de los lenguajes de programación.

El área de conocimiento de Fundamentos de los Lenguajes de Programación se ocupa de articular los conceptos y principios subyacentes de los lenguajes de programación, la especificación formal de un lenguaje de programación y el comportamiento de un programa, explicar cómo se implementan los lenguajes de programación, comparar las fortalezas y debilidades de varios paradigmas de programación, y describir cómo los lenguajes de programación interactúan con entidades como sistemas operativos y hardware. Los conceptos aquí cubiertos son aplicables a varios lenguajes y comprender estos principios ayuda al aprendiz a pasar fácilmente de un lenguaje a otro, así como a seleccionar un paradigma y lenguaje de programación que mejor se adapte al problema en cuestión.

Los lenguajes de programación son el medio a través del cual los programadores describen conceptos con precisión, formulan algoritmos y razonan sobre soluciones. A lo largo de su carrera, un científico de la computación aprenderá y trabajará con muchos lenguajes diferentes, por separado o juntos. Los desarrolladores de software deben entender diferentes modelos de programación, características y construcciones de programación, y conceptos subyacentes para tomar decisiones de diseño informadas entre lenguajes que admiten múltiples enfoques complementarios. Sería útil saber cómo se definen, componen e implementan las características de los lenguajes de programación para mejorar la eficiencia de ejecución y el mantenimiento a largo plazo del software desarrollado. También es útil un conocimiento básico de la traducción de lenguajes, el análisis de programas, el comportamiento en tiempo de ejecución, la gestión de memoria y la interacción de procesos concurrentes que se comunican entre sí mediante paso de mensajes, memoria compartida y sincronización. Finalmente, algunos desarrolladores e investigadores necesitarán diseñar nuevos lenguajes, un ejercicio que requiere una mayor familiaridad con los principios básicos.

Área de Conocimiento (KA)	**CS Core**	**KA Core**
⭐ 2.5.1 Fundamentos de Programación Orientada a Objetos (Pág. 47)	2	1
⭐ 2.5.2 Programación Orientada a Objetos: Encapsulación, Subtipado y Reflexión (Pág. 48)	2	1
⭐ 2.5.3 Programación Funcional (Pág. 48)	4	3
⭐ 2.5.4 Programación Lógica (Pág. 49)		2
⭐ 2.5.5 Scripting de Shell (Pág. 50)	2	
⭐ 2.5.6 Programación Dirigida por Eventos y Reactiva (Pág. 50)	2	2
⭐ 2.5.7 Computación Paralela y Distribuida (Pág. 51)	2	2
2.5.8 Programación Orientada a Aspectos (Pág. 52)	Electivo	
⭐ 2.5.9 Fundamentos de Sistemas de Tipos (Pág. 52)	2	2
⭐ 2.5.10 Sistemas de Tipos: Polimorfismo y Tipado Complementario (Pág. 53)	1	1
⭐ 2.5.11 Teoría de Tipos Avanzada (Pág. 53)	2	2
⭐ 2.5.12 Modelo de Ejecución y Memoria de Sistemas (Pág. 54)	2	
⭐ 2.5.13 Gestión de Memoria (Pág. 54)	1	
⭐ 2.5.14 Traducción y Ejecución de Lenguajes (Pág. 54)	2	3
⭐ 2.5.15 Abstracción y Representación de Programas (Pág. 55)		3
2.5.16 Análisis Sintáctico (Pág. 56)	Electivo	
2.5.17 Análisis Semántico del Compilador (Pág. 56)	Electivo	
2.5.18 Análisis de Programas y Analizadores (Pág. 57)	Electivo	
2.5.19 Generación de Código (Pág. 57)	Electivo	
2.5.20 Comportamiento en Tiempo de Ejecución y Sistemas (Pág. 58)	Electivo	
2.5.21 Construcciones de Programación Avanzadas (Pág. 59)	Electivo	
2.5.22 Pragmática del Lenguaje (Pág. 59)	Electivo	
2.5.23 Semántica Formal (Pág. 60)	Electivo	
2.5.24 Metodologías de Desarrollo Formal (Pág. 60)	Electivo	
2.5.25 Principios de Diseño de Lenguajes de Programación (Pág. 61)	Electivo	
2.5.26 Sociedad, ética y Profesión (Pág. 61)	Electivo	

Cuadro 2.5: Lista de KUs del área de Fundamentos de los Lenguajes de Programación.

2.5.1. FPL/Fundamentos de Programación Orientada a Objetos (CS Core: 2 hrs, KA Core: 1 hr)

Temas:
Core

- Programación imperativa como subconjunto de la programación orientada a objetos.
- Diseño orientado a objetos: 1. Descomposición en objetos que poseen estado y tienen comportamiento. 2. Diseño de jerarquía de clases para modelado.
- Definición de clases: campos, métodos y constructores.
- Subclases, herencia (incluyendo herencia múltiple) y sobrescritura de métodos.

- Despacho dinámico: definición de llamada a método.
- Manejo de excepciones.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enumerar las diferencias entre los paradigmas de programación imperativa y orientada a objetos [Enumerar]
2. Componer una clase a través del diseño, implementación y prueba para cumplir con los requisitos de comportamiento [Componer]
3. Construir una jerarquía de clases simple utilizando subclases que permita reutilizar código para subclases distintas [Crear]
4. Predecir y validar el flujo de control en un programa usando despacho dinámico [Evaluar]
5. Comparar y contrastar cómo difieren las soluciones computacionales a un problema en los enfoques procedural, funcional y orientado a objetos [Comparar]
6. Comparar y contrastar los mecanismos para definir y proteger elementos de datos dentro de los enfoques procedural, funcional y orientado a objetos [Comparar]

2.5.2. FPL/Programación Orientada a Objetos: Encapsulación, Subtipado y Reflexión (CS Core: 2 hrs, KA Core: 1 hr)

Temas:
Core

- Idiomas orientados a objetos para encapsulación: 1. Privacidad, ocultación de datos y visibilidad de miembros de clase. 2. Interfaces que revelan solo firmas de métodos. 3. Clases base abstractas, rasgos (*traits*) y mixins.
- Propiedades dinámicas vs estáticas.
- Composición vs herencia.
- Subtipado: 1. Polimorfismo de subtipos; conversiones ascendentes (*upcasts*) implícitas en lenguajes tipados. 2. Noción de reemplazo conductual: los subtipos actúan como el supertipo. 3. Relación entre subtipado y herencia.
- Clases de colecciones, iteradores y otros componentes comunes de biblioteca.
- Metaprogramación y reflexión.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comparar y contrastar los beneficios y costos/impacto de usar herencia (subclases) y composición (específicamente, cómo basar la composición en funciones de orden superior) [Comparar]
2. Explicar la relación entre la herencia orientada a objetos (compartición de código y sobrescritura) y el subtipado (la idea de que un subtipo sea usable en un contexto que espera el supertipo) [Explicar]
3. Usar mecanismos de encapsulación orientados a objetos como interfaces y miembros privados [Usar]
4. Definir y usar iteradores y otras operaciones en agregados, incluyendo operaciones que toman funciones como argumentos, en múltiples lenguajes de programación, seleccionando los idiomas más naturales para cada lenguaje [Definir]
5. Usar clases de colecciones e iteradores efectivamente para resolver un problema [Usar]
6. Comparar y contrastar cómo difieren las soluciones computacionales a un problema en los enfoques procedural, funcional y orientado a objetos [Comparar]
7. Comparar y contrastar los mecanismos para definir y proteger elementos de datos dentro de los enfoques procedural, funcional y orientado a objetos [Comparar]

2.5.3. FPL/Programación Funcional (CS Core: 4 hrs, KA Core: 3 hrs)

Temas:
Core

- Expresiones lambda y evaluación: 1. Enlace (binding) de variables y reglas de ámbito (scope). 2. Paso de parámetros. 3. Expresiones lambda anidadas y orden de reducción.

- Programación libre de efectos secundarios: 1. Las llamadas a funciones no tienen efectos secundarios, facilitando el razonamiento composicional. 2. Variables inmutables y copia de datos vs reducción. 3. Uso de recursión vs bucles vs tuberías (map/reduce).
- Procesamiento de datos estructurados (por ejemplo, árboles) mediante funciones con casos para cada variante de datos: 1. Funciones definidas sobre datos compuestos en términos de funciones aplicadas a las piezas constituyentes. 2. Estructuras de datos persistentes.
- Usar funciones de orden superior (tomar, devolver y almacenar funciones).
- Metaprogramación y reflexión.
- Clausuras (*closures*) de funciones (funciones que usan variables en el entorno léxico circundante): 1. Significado y definición básica: crear clausuras en tiempo de ejecución capturando el entorno. 2. Idiomas canónicos: devoluciones de llamada (*callbacks*), argumentos para iteradores, código reutilizable mediante argumentos de funciones. 3. Usar una clausura para encapsular datos en su entorno. 4. Evaluación diferida (*lazy*) vs ansiosa (*eager*).

Non Core

- Máquina de reducción de grafos (graph reduction) y llamada por necesidad (*call-by-need*).
- Implementar evaluación diferida.
- Integración con el paradigma de programación lógica usando conceptos como lógica ecuacional, estrechamiento (narrowing), residuación (residuation) y unificación semántica.
- Integración con otros paradigmas de programación como imperativo y orientado a objetos.

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Desarrollar algoritmos básicos que eviten asignar a estados mutables o considerar igualdad referencial [Crear]
2. Desarrollar funciones útiles que tomen y devuelvan otras funciones [Crear]
3. Explicar un ejemplo simple de una expresión lambda implementada usando una máquina virtual, como una máquina SECD, mostrando el almacenamiento y recuperación del entorno [Explicar]
4. Interpretar correctamente las variables y el ámbito léxico en un programa que usa clausuras de funciones [Interpretar]
5. Usar mecanismos de encapsulación funcional como clausuras e interfaces modulares [Usar]
6. Comparar y contrastar ejecución con estado (stateful) vs sin estado (stateless) [Comparar]
7. Definir y usar iteradores y otras operaciones en agregados, incluyendo operaciones que toman funciones como argumentos, en múltiples lenguajes de programación, seleccionando los idiomas más naturales para cada lenguaje [Definir]

NonCore:

8. Ilustrar la reducción de grafos usando una expresión λ con una subexpresión compartida [Aplicar]
9. Ilustrar la ejecución de una expresión λ anidada simple usando una máquina abstracta, como una máquina ABC [Aplicar]
10. Ilustrar estrechamiento (narrowing), residuación y unificación semántica usando ejemplos ilustrativos simples [Aplicar]
11. Ilustrar las construcciones de concurrencia usando ejemplos de programación simples de conceptos conocidos, como un búfer siendo leído y escrito concurrentemente o secuencialmente [Aplicar]

2.5.4. FPL/Programación Lógica (KA Core: 2 hrs)

Temas:

Core

- Cuantificadores universales vs existenciales.
- Lógica de predicados de primer orden vs lógica de orden superior.
- Expresar relaciones complejas usando conectivos lógicos y relaciones más simples.
- Definiciones de cláusula de Horn, hechos (facts), metas (goals) y submetas (subgoals).
- Unificación y algoritmo de unificación; unificación vs aserción vs evaluación de expresiones.
- Mezclar relaciones con funciones.
- Cortes (cuts), retroceso (backtracking) y no determinismo.
- Suposiciones de mundo cerrado (closed-world) vs mundo abierto (open-world).

Non Core

- Sobrecarga de memoria por copia de variables al manejar programas iterativos.
- Construcciones de programación para almacenar cómputo parcial y podar árboles de búsqueda.

- Mezclar programación funcional y programación lógica usando conceptos como lógica ecuacional, estrechamiento (narrowing), residuación y unificación semántica.
- Programación lógica de orden superior, con restricciones (constraint) e inductiva.
- Integración con otros paradigmas de programación como la programación orientada a objetos.
- Construcciones de programación avanzadas como listas de diferencia (difference-lists), creación de estructuras de datos definidas por el usuario, conjunto de (set of), etc.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Usar un lenguaje lógico para implementar un algoritmo convencional [Usar]
2. Usar un lenguaje lógico para implementar un algoritmo que emplee búsqueda implícita usando cláusulas, relaciones y cortes (cuts) [Usar]
3. Usar un ejemplo ilustrativo simple para mostrar la correspondencia entre la Lógica de Predicados de Primer Orden (FOPL) y los programas lógicos que usan cláusulas de Horn [Usar]
4. Usar ejemplos para ilustrar el algoritmo de unificación y su papel en el paso de parámetros en la reducción de consultas [Usar]
5. Usar programas lógicos simples que entrelazan relaciones, funciones y programación recursiva, como números factoriales y de Fibonacci, y relaciones complejas simples entre entidades, e ilustrar la ejecución y paso de parámetros usando unificación y retroceso [Usar]

NonCore:

6. Ilustrar el cómputo de programas simples como Fibonacci y mostrar la sobrecarga por recomputación, y luego mostrar cómo mejorar la sobrecarga de ejecución [Aplicar]

2.5.5. FPL/Scripting de Shell (CS Core: 2 hrs)

Temas:
Core

- Manejo de errores/excepciones
- Tuberías (piping)
- Comandos del sistema: 1. Interfaz con sistemas operativos
- Variables de entorno
- Abstracción de archivo y operadores
- Estructuras de datos, como arreglos y listas.
- Expresiones regulares
- Programas y procesos
- Flujo de trabajo

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Crear y ejecutar scripts automatizados para gestionar varias tareas del sistema [Crear]
2. Resolver varios problemas de procesamiento de texto mediante scripting [Resolver]

2.5.6. FPL/Programación Dirigida por Eventos y Reactiva (CS Core: 2 hrs, KA Core: 2 hrs)

Temas:
Core

- Programación procedural vs programación reactiva: ventajas de la programación reactiva para capturar eventos.
- Componentes de la programación reactiva: fuente de eventos, señales de eventos, escuchas (listeners) y despachadores (dispatchers), objetos de evento, adaptadores, manejadores de eventos (event-handlers).
- Modelos sin estado (stateless) y de transición de estado (state-transition) de programación basada en eventos.

- Usos canónicos como interfaces gráficas de usuario (GUIs), dispositivos móviles, robots, servidores.
- Usar un framework reactivo: 1. Definir manejadores/escuchas de eventos 2. Parametrización de emisores de eventos y argumentos de eventos 3. Eventos generados externamente y eventos generados por el programa
- Separación de modelo, vista y controlador
- Programas dirigidos por eventos y reactivos como sistemas de transición de estado

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Implementar manejadores de eventos para usar en sistemas reactivos, como GUIs [Implementar]
2. Examinar por qué un estilo de programación dirigida por eventos es natural en dominios donde los programas reaccionan a eventos externos [Examinar]
3. Definir y usar un framework reactivo [Definir]
4. Describir un sistema interactivo en términos de un modelo, una vista y un controlador [Describir]

2.5.7. FPL/Computación Paralela y Distribuida (CS Core: 2 hrs, KA Core: 2 hrs)

Temas:
Core

- Seguridad (safety) y vivacidad (liveness): 1. Condiciones de carrera (race conditions) 2. Dependencias/precondiciones 3. Modelos de fallos (fault models) 4. Terminación
- Modelos de programación: 1. Modelo de actores (*actor models*) 2. Modelos procedurales y reactivos 3. Modelos de programación síncrona/asíncrona 4. Paralelismo de datos (*data parallelism*)
- Propiedades: 1. Propiedades basadas en orden: *a*) Conmutatividad *b*) Independencia 2. Propiedades basadas en consistencia: *a*) Atomicidad *b*) Consenso
- Control de ejecución: 1. Espera asíncrona *Async await* 2. Promesas (*promises*) 3. Hilos (*threads*)
- Comunicación y coordinación: 1. Mutexes 2. Paso de mensajes (*message-passing*) 3. Memoria compartida (*shared memory*) 4. Cobegin-coend 5. Monitores 6. Canales (*channels*) 7. Hilos (*threads*) 8. Guardas (*guards*)
- Futuros (futures)
- Soporte del lenguaje para paralelismo de datos como forall, desenrollado de bucles (loop unrolling), map/reduce
- Efecto de los modelos de consistencia de memoria en la semántica del lenguaje y la generación correcta de código
- Interfaces de Programación de Aplicaciones de Transferencia de Estado Representacional (REST APIs)
- Tecnologías y enfoques: computación en la nube (*cloud computing*), computación de alto rendimiento, computación cuántica, computación ubicua
- Sobrecarga del paso de mensajes
- Granularidad del programa para una explotación eficiente de la concurrencia
- Concurrencia y otros paradigmas de programación (por ejemplo, funcional)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar por qué los lenguajes de programación no garantizan la consistencia secuencial en presencia de condiciones de carrera (data races) y qué deben hacer los programadores como resultado [Explicar]
2. Implementar programas concurrentes correctos usando múltiples modelos de programación, como memoria compartida, actores, futuros, construcciones de sincronización y primitivas de paralelismo de datos [Implementar]
3. Usar un modelo de paso de mensajes para analizar un protocolo de comunicación [Usar]
4. Usar construcciones de sincronización como métodos monitorizados/sincronizados en un programa simple [Usar]
5. Modelar dependencia de datos usando construcciones de programación simples que involucran variables, lectura y escritura [Implementar]

6. Modelar dependencia de control usando construcciones simples como selección e iteración [Implementar]
7. Explicar cómo las APIs REST integran aplicaciones y automatizan procesos [Explicar]
8. Explicar beneficios, limitaciones y desafíos relacionados con la computación distribuida y paralela [Explicar]

2.5.8. FPL/Programación Orientada a Aspectos

Temas:
Non Core

- Aspectos
- Puntos de unión (join points)
- Consejos (advice): 1. Antes (*before*) 2. Después (*after*) (como *finally*, *returning* o *throwing*) 3. Alrededor (*around*)
- Cortes de punto (point cuts): 1. Designadores
- Tejido (weaving) - estático y dinámico
- Alternativas incluyendo anotaciones y IDEs

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Describir los conceptos fundamentales de la programación orientada a aspectos, incluyendo aspectos, puntos de unión, consejos y cortes de punto [Describir]
2. Aplicar estrategias de tejido estático y dinámico para modularizar los conceptos transversales en un programa [Aplicar]
3. Analizar alternativas a la programación orientada a aspectos, como anotaciones y enfoques basados en IDEs, para el manejo de conceptos transversales [Analizar]

2.5.9. FPL/Fundamentos de Sistemas de Tipos (CS Core: 2 hrs, KA Core: 2 hrs)

Temas:
Core

- Un tipo como un conjunto de valores junto con un conjunto de operaciones: 1. Tipos primitivos (por ejemplo, números, booleanos) 2. Tipos compuestos construidos a partir de otros tipos (por ejemplo, registros/estructuras, uniones, arreglos, listas, funciones, referencias usando operaciones de conjuntos)
- Asociación de tipos a variables, argumentos, resultados y campos
- Seguridad de tipos (type safety) como un aspecto de la corrección del programa
- Seguridad de tipos y errores causados por usar valores inconsistentemente dados sus tipos pretendidos
- Objetivos y limitaciones del tipado estático y dinámico: detectar y eliminar errores lo antes posible.
- Equivalencia de tipos: equivalencia estructural vs nominal

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir, tanto para un tipo primitivo como para uno compuesto, los valores que tienen ese tipo [Describir]
2. Describir, para un lenguaje con un sistema de tipos estático, las operaciones que están prohibidas estáticamente, como pasar el tipo incorrecto de valor a una función o método [Describir]
3. Describir ejemplos de errores de programa detectados por un sistema de tipos [Describir]
4. Identificar propiedades del programa, para múltiples lenguajes de programación, que se verifican estáticamente y propiedades del programa que se verifican dinámicamente [Analizar]
5. Describir un programa de ejemplo que no verifica tipos en un lenguaje particular y que sin embargo no tendría error si se ejecutara [Describir]
6. Usar tipos y mensajes de error de tipos para escribir y depurar programas [Usar]
7. Explicar cómo las reglas de tipado definen el conjunto de operaciones que son legales para un tipo [Explicar]
8. Enumerar las reglas de tipo que gobiernan el uso de un tipo compuesto particular [Listar/Enumerar]

9. Explicar por qué la indecidibilidad requiere que los sistemas de tipos aproximen conservadoramente el comportamiento del programa [Explicar]
10. Explicar múltiples beneficios y limitaciones del tipado estático al escribir, mantener y depurar software [Explicar]

2.5.10. FPL/Sistemas de Tipos: Polimorfismo y Tipado Complementario (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Tipos genéricos (polimorfismo paramétrico): 1. Definición y ventajas del polimorfismo: paramétrico, de subtipado (subtyping), de sobrecarga (overloading) y de coerción 2. Comparación de tipos monomórficos y polimórficos 3. Comparación con polimorfismo ad-hoc (sobrecarga) y polimorfismo de subtipos 4. Parámetros genéricos y tipado 5. Uso de bibliotecas genéricas como colecciones 6. Comparación con polimorfismo ad hoc (sobrecarga) y polimorfismo de subtipos 7. Polimorfismo prescriptivo vs descriptivo 8. Modelos de implementación de tipos polimórficos 9. Subtipado
- Beneficios complementarios del tipado estático y dinámico: 1. Errores tempranos vs errores tardíos/evitados 2. Hacer cumplir invariantes durante el desarrollo y mantenimiento del código vs posponer decisiones de tipado mientras se prototipa y permitir convenientemente patrones de codificación flexibles como colecciones heterogéneas. 3. Reglas de tipado para tipos de función, producto y suma 4. Evitar mal uso del código vs permitir más reutilización de código 5. Detectar programas incompletos vs permitir que programas incompletos se ejecuten 6. Relación con el análisis estático 7. Decidibilidad

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Definir y usar piezas de programa (como funciones, clases, métodos) que usen tipos genéricos, incluyendo para colecciones [Definir]
2. Discutir las diferencias entre genéricos, subtipado y sobrecarga [Debatir]

2.5.11. FPL/Teoría de Tipos Avanzada (CS Core: 2 hrs, KA Core: 2 hrs)

Temas:
Non Core

- Constructores de tipos composicionales, como tipos producto (para agregados), tipos suma (para uniones), tipos función, tipos cuantificados y tipos recursivos
- Verificación de tipos (type checking)
- Subtipado: 1. Polimorfismo de subtipos; conversiones ascendentes implícitas en lenguajes tipados 2. Noción de reemplazo conductual: los subtipos actúan como el supertipo 3. Relación entre subtipado y herencia
- Seguridad de tipos como preservación más progreso (preservation plus progress)
- Inferencia de tipos
- Sobrecarga estática
- Proposiciones como tipos (implicación como función, conjunción como producto, disyunción como suma)
- Tipos dependientes (cuantificación universal como función dependiente, cuantificación existencial como producto dependiente).

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Definir un sistema de tipos de manera precisa y composicional [Definir]
2. Para varios constructores de tipos fundamentales, identificar los valores que describen y los invariantes que hacen cumplir [Analizar]
3. Describir precisamente los invariantes preservados por un sistema de tipos sano (sound) [Describir]
4. Demostrar seguridad de tipos para un lenguaje simple en términos de teoremas de preservación y progreso [Probar/Demostrar]

5. Implementar un algoritmo de inferencia de tipos basado en unificación para un lenguaje simple [Implementar]
6. Explicar cómo la sobrecarga estática y los algoritmos de resolución asociados influyen en el comportamiento dinámico de los programas [Explicar]

2.5.12. FPL/Modelo de Ejecución y Memoria de Sistemas (CS Core: 2 hrs)

Temas:
Core

- Estructuras de datos para traducción, ejecución, traducción y movilidad de código como pila, montón (heap), alias (compartición usando punteros), secuencia indexada y cadena
- Acceso directo, indirecto e indexado a ubicación de memoria
- Representación en tiempo de ejecución de abstracciones de datos como variables, arreglos, vectores, registros, elementos de datos basados en punteros como listas enlazadas y árboles, y objetos
- Máquina de bajo nivel abstracta con instrucción simple, pila y montón para explicar traducción y ejecución
- Distribución en memoria en tiempo de ejecución: registro de activación (con varios punteros), datos estáticos, pila de llamadas, montón: 1. Traducción de construcciones de selección e iterativas a diagramas de flujo de control 2. Traducción de diagramas de flujo de control a código abstracto de bajo nivel 3. Implementación de bucles, recursión y llamadas de cola (tail calls) 4. Traducción de llamadas a funciones/procedimientos y retorno de llamadas, incluyendo diferentes mecanismos de paso de parámetros usando una máquina abstracta
- Computación verde (green computing).

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar cómo se ejecuta una construcción central del lenguaje, como abstracciones de datos y abstracciones de control [Explicar]

2.5.13. FPL/Gestión de Memoria (CS Core: 1 hr)

Temas:
Core

- Asignación y acceso de bajo nivel a estructuras de datos de alto nivel como tipos de datos básicos, arreglos n-dimensionales, vectores, registros y objetos
- Retorno de procedimiento como mecanismo de desasignación automática para elementos de datos locales en la pila
- Gestión manual de memoria: asignar, desasignar y reutilizar memoria del montón
- Gestión automática de memoria: recolección de basura (garbage collection) como técnica automatizada usando la noción de alcanzabilidad (reachability)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar cómo las implementaciones de lenguajes de programación típicamente organizan la memoria en secciones de datos globales, texto, montón y pila, y cómo características como recursión y gestión de memoria se mapean a este modelo de memoria [Explicar]
2. Explicar por qué ocurren las fugas de memoria (memory leaks) y los problemas de punteros colgantes (dangling pointers), y qué puede hacer un programador para evitarlos/corregirlos [Explicar]

2.5.14. FPL/Traducción y Ejecución de Lenguajes (CS Core: 2 hrs, KA Core: 3 hrs)

Temas:
Core

- Modelos de ejecución para JIT (Just-In-Time), compilador, intérprete
- Uso de código intermedio, por ejemplo, bytecode
- Limitaciones y beneficios de JIT, compilador e intérprete

- Compiladores/transpiladores cruzados (cross compilers/transpilers)
- Representación BNF y BNF extendida de gramática libre de contexto
- árbol de análisis (parse tree) usando una oración simple como expresión aritmética o sentencia if-then-else
- Ejecución como código nativo o dentro de una máquina virtual
- Canalización de traducción de lenguaje: análisis sintáctico, análisis (parsing), verificación de tipos opcional, generación de código/traducción y optimización, enlace (linking), carga (loading), ejecución
- Representación en tiempo de ejecución de construcciones centrales del lenguaje como objetos (tablas de métodos) y funciones que pueden pasarse como parámetros a y devolverse desde funciones (clausuras)
- Desarrollo seguro de compiladores

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar y comprender las diferencias entre implementaciones de lenguaje compilado, JIT e interpretado, incluyendo los beneficios y limitaciones de cada una [Explicar]
2. Diferenciar sintaxis y análisis (parsing) de semántica y evaluación [Diferenciar]
3. Usar BNF y BNF extendida para especificar la sintaxis de construcciones simples como if-then-else, declaración de tipo y construcciones iterativas para lenguajes conocidos como C++ o Python [Usar]
4. Ilustrar el árbol de análisis usando una oración/expresión aritmética simple [Aplicar]
5. Ilustrar traducción de diagramas de sintaxis a BNF/BNF extendida para construcciones simples como if-then-else, declaración de tipo, construcciones iterativas, etc [Aplicar]
6. Ilustrar ambigüedad en el análisis usando if-then-else anidado/expresión aritmética y mostrar resolución usando orden de precedencia [Aplicar]
7. Discutir los beneficios y limitaciones de la recolección de basura, incluyendo la noción de alcanzabilidad [Debatir]

2.5.15. FPL/Abstracción y Representación de Programas (KA Core: 3 hrs)

Temas:
Core

- BNF y expresiones regulares
- Programas que toman (otros) programas como entrada, como intérpretes, compiladores, verificadores de tipos, generadores de documentación
- Componentes de un lenguaje: 1. Definiciones de alfabetos, delimitadores, oraciones, sintaxis y semántica 2. Sintaxis vs semántica
- Programa como un conjunto de oraciones significativas no ambiguas
- Abstracciones básicas de programación: constantes, variables, declaraciones (incluyendo declaraciones anidadas), comando, expresión, asignación, selección, iteración definida e indefinida, iteradores, función, procedimiento, módulos, manejo de excepciones
- Variables mutables vs inmutables: ventajas y desventajas de reutilizar ubicación de memoria existente vs ventajas de copiar y mantener valores antiguos; almacenar cómputo parcial vs recomputación
- Tipos de variables: estáticas, locales, no locales (*nonlocal*), globales; necesidad y problemas con variables no locales y globales.
- Reglas de ámbito (*scope*): estático vs dinámico; visibilidad de variables; efectos secundarios (*side-effects*).
- Efectos secundarios inducidos por variables no locales, variables globales y variables con alias (*aliased*).

Non Core

- Valores-L (*L-values*) y Valores-R (*R-values*): mapear nombre de variable mutable a Valores-L; mapear nombres de variables inmutables a Valores-R
- Entorno (*environment*) vs almacén (*store*) y sus propiedades
- Abstracción de datos y de control
- Mecanismos para intercambio de información entre unidades de programa como procedimientos,

funciones y módulos: variables no locales, variables globales, paso de parámetros, importación-exportación entre módulos
- Estructuras de datos para representar código para ejecución, traducción o transmisión.
- Representación de instrucciones de bajo nivel como instrucciones de máquina virtual, lenguaje ensamblador y representación binaria
- Cálculo lambda, enlace de variables (variable binding) y renombrado de variables.
- Tipos de semántica: operacional, axiomática, denotacional, conductual; definir y usar árboles de sintaxis abstracta; contrastar con sintaxis concreta.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Ilustrar el ámbito de variables y visibilidad usando programas simples [Aplicar]
2. Ilustrar diferentes tipos de paso de parámetros usando un lenguaje de programación pseudo simple [Aplicar]
3. Explicar el efecto secundario usando variables globales y no locales y cómo corregir tales programas [Explicar]
4. Explicar cómo los programas que procesan otros programas tratan a los otros programas como sus datos de entrada [Explicar]
5. Describir una gramática y un árbol de sintaxis abstracta para un lenguaje pequeño [Describir]
6. Describir los beneficios de tener representaciones de programas distintas de cadenas de código fuente [Describir]
7. Implementar un programa para procesar alguna representación de código para algún propósito, como un intérprete, un optimizador de expresiones o un generador de documentación [Implementar]

2.5.16. FPL/Análisis Sintáctico

Temas:
Non Core

- Gramáticas regulares vs gramáticas libres de contexto
- Escaneo (*scanning*) y análisis (*parsing*) basados en especificaciones de lenguaje
- Análisis léxico usando expresiones regulares
- Tokens y su uso
- Estrategias de análisis (*parsing*) incluyendo técnicas descendentes (*top-down*) (por ejemplo, descenso recursivo o LL) y ascendentes (*bottom-up*) (por ejemplo, LR o GLR): 1. Tablas de anticipación (*lookahead*) y su aplicación al análisis
- Teoría de lenguajes: 1. Jerarquía de Chomsky 2. Derivación más a la izquierda/más a la derecha y ambigüedad 3. Transformación de gramáticas
- Mecanismos de recuperación de errores del analizador (parser)
- Generar escáneres y analizadores a partir de especificaciones declarativas

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Usar gramáticas formales para especificar la sintaxis de lenguajes [Usar]
2. Ilustrar el papel de las tablas de anticipación en el análisis [Aplicar]
3. Usar herramientas declarativas para generar analizadores y escáneres [Usar]
4. Reconocer problemas clave en definiciones de sintaxis: ambigüedad, asociatividad, precedencia [Reconocer]

2.5.17. FPL/Análisis Semántico del Compilador

Temas:
Non Core

- árboles de sintaxis abstracta; contraste con sintaxis concreta
- Definir, recorrer y modificar representaciones de programa de alto nivel
- ámbito (scope) y resolución de enlaces (binding)
- Semántica estática: 1. Verificación de tipos. 2. Definir antes de usar 3. Anotaciones y entornos (*frameworks*) de verificación estática extendida (*extended static checking*).

- Valores-L/Valores-R
- Semántica de llamada
- Tipos de paso de parámetros con ilustraciones simples y comparación: llamada por valor (call by value), llamada por referencia (call by reference), llamada por valor-resultado (call by value-result), llamada por nombre (call by name), llamada por necesidad (call by need) y sus variaciones

- Especificaciones declarativas como gramáticas atribuidas (attribute grammars) y sus aplicaciones en manejo de gramáticas sensibles al contexto limitadas

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Dibujar el árbol de sintaxis abstracta para un lenguaje pequeño [Diagramar]
2. Implementar análisis estáticos sensibles al contexto, a nivel de fuente, como verificadores de tipos o resolver identificadores para identificar sus ocurrencias de enlace (binding) [Implementar]
3. Describir análisis semánticos usando una gramática atribuida [Describir]

2.5.18. FPL/Análisis de Programas y Analizadores

Temas:
Non Core

- Representaciones de programa relevantes, como bloques básicos (basic blocks), grafos de flujo de control (control-flow graphs), cadenas definición-uso (def-use chains) y asignación única estática (static single assignment)
- Indecidibilidad y consecuencias para el análisis de programas
- Análisis insensible al flujo (flow-insensitive), como verificación de tipos y análisis de punteros y alias escalables
- Análisis sensible al flujo (flow-sensitive), como análisis de flujo de datos (dataflow) hacia adelante y hacia atrás
- Análisis sensible a la ruta (path-sensitive), como verificación de modelos de software (software model checking) y verificación de software
- Herramientas y frameworks para implementar analizadores
- Papel del análisis estático en la optimización de programas y el análisis de dependencia de datos durante la explotación de concurrencia
- Papel del análisis de programas en la verificación (parcial) y la búsqueda de errores (bug-finding)

- Paralelización: 1. Análisis para auto-paralelización 2. Análisis para detectar errores de concurrencia

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Explicar la diferencia entre grafo de flujo de datos (dataflow graph) y grafo de flujo de control [Explicar]
2. Explicar por qué los análisis de programa no triviales y sólidos (sound) deben ser aproximados [Explicar]
3. Argumentar por qué un análisis es correcto (sound y terminante) [Argumentar]
4. Explicar por qué el alias potencial limita el análisis de programa sólido y cómo el análisis de alias puede ayudar [Explicar]
5. Usar los resultados de un análisis de programa para optimización de programas y/o corrección parcial del programa [Usar]

2.5.19. FPL/Generación de Código

Temas:
Non Core

- Conjuntos de instrucciones
- Flujo de control
- Gestión de memoria
- Llamadas a procedimientos y despacho de métodos (method dispatching)

- Compilación separada; enlace (linking)
- Selección de instrucciones
- Planificación de instrucciones (por ejemplo, encadenamiento - pipelining)
- Asignación de registros
- Optimización de código como una forma de análisis de programas
- Generación de programas mediante IA generativa

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Identificar todos los pasos esenciales para convertir automáticamente código fuente en lenguaje ensamblador u otros lenguajes de bajo nivel [Analizar]
2. Explicar el código de bajo nivel necesario para llamar a funciones/métodos en lenguajes modernos [Explicar]
3. Discutir por qué la compilación separada requiere convenciones de llamada uniformes [Debatir]
4. Discutir por qué la compilación separada limita la optimización debido a efectos desconocidos de las llamadas [Debatir]
5. Discutir oportunidades para la optimización introducidas por la traducción ingenua y enfoques para lograr la optimización, como selección de instrucciones, planificación de instrucciones, asignación de registros y optimización de mirilla (peephole optimization) [Debatir]

2.5.20. FPL/Comportamiento en Tiempo de Ejecución y Sistemas

Temas:
Non Core

- Modelos de proceso usando pilas y montones para asignar y desasignar registros de activación y recuperar entornos usando punteros de marco (frame pointers) y direcciones de retorno durante una llamada a procedimiento, incluyendo ejemplos de paso de parámetros
- Esquemas de búsqueda de código usando tablas hash para métodos en implementaciones de programas orientados a objetos
- Distribución de datos para objetos y registros de activación
- Asignación de objetos en el montón (heap)
- Implementar entidades virtuales y métodos virtuales; tablas de métodos virtuales (virtual method tables) y su aplicación
- Comportamiento en tiempo de ejecución de programas orientados a objetos
- Comparar y contrastar la asignación de memoria durante el intercambio de información usando paso de parámetros y variables no locales (usando cadena de enlaces estáticos).
- Enfoques y técnicas de gestión dinámica de memoria: malloc/free, recolección de basura (mark-sweep, copying, reference counting), regiones (también conocidas como arenas o zonas)
- Compilación justo a tiempo (just-in-time) y recompilación dinámica
- Interfaz con el sistema operativo (por ejemplo, para inicialización del programa)
- Interoperabilidad entre lenguajes de programación incluyendo mecanismos de paso de parámetros y representación de datos: 1. *Big endian*, *little endian* 2. Distribución de datos de tipos de datos compuestos como arreglos
- Otras características comunes de las máquinas virtuales, como carga de clases (class loading), hilos (threads) y verificación de seguridad
- Sandboxing

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Discutir beneficios y limitaciones de la gestión automática de memoria [Debatir]
2. Explicar el uso de metadatos en representaciones en tiempo de ejecución de objetos y registros de activación, como punteros de clase, longitudes de arreglo, direcciones de retorno y punteros de marco (frame pointers) [Explicar]
3. Comparar y contrastar asignación estática vs asignación basada en pila vs asignación basada en montón de elementos de datos [Comparar]
4. Explicar por qué algunos elementos de datos no pueden desasignarse automáticamente al final de una llamada a procedimiento/método (necesidad de recolección de basura) [Explicar]

5. Discutir ventajas, desventajas y dificultades de la compilación justo a tiempo y la recompilación dinámica [Debatir]
6. Discutir el uso del sandboxing en código móvil [Debatir]
7. Identificar los servicios proporcionados por los sistemas en tiempo de ejecución de lenguajes modernos [Analizar]

2.5.21. FPL/Construcciones de Programación Avanzadas

Temas:
Non Core

- Mecanismos de encapsulación
- Evaluación diferida (delayed evaluation) y flujos infinitos (infinite streams)
- Comparar y contrastar evaluación diferida vs evaluación ansiosa (eager)
- Unificación vs aserción vs evaluación de expresiones
- Abstracciones de control: manejo de excepciones, continuaciones (continuations), mónadas (monads).
- Abstracciones orientadas a objetos: herencia múltiple, mixins, rasgos (traits), multimétodos (multimethods)
- Metaprogramación: macros, programación generativa, desarrollo basado en modelos (model-based development)
- Manipulación de cadenas mediante coincidencia de patrones (pattern-matching) (expresiones regulares)
- Evaluación dinámica de código ("eval")
- Soporte del lenguaje para verificar aserciones, invariantes y pre/post-condiciones
- Lenguajes específicos del dominio, como lenguajes de bases de datos, lenguajes de ciencia de datos, lenguajes de computación embebida, lenguajes síncronos, lenguajes de interfaz de hardware
- Modelos y lenguajes de computación de alto rendimiento masivamente paralelos

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Usar varias construcciones e idiomas de programación avanzados correctamente [Usar]
2. Discutir cómo varias construcciones de programación avanzadas tienen como objetivo mejorar la estructura del programa, la calidad del software y la productividad del programador [Debatir]
3. Discutir cómo varias construcciones de programación avanzadas interactúan con la definición e implementación de otras características del lenguaje [Debatir]

2.5.22. FPL/Pragmática del Lenguaje

Temas:
Non Core

- Efecto de las necesidades tecnológicas y los requisitos de software en el desarrollo y evolución de los lenguajes de programación
- Dominios de problemas y paradigma de programación
- Criterios para un buen diseño de lenguaje de programación: 1. Principios de diseño de lenguajes como la ortogonalidad 2. Definir construcciones de control e iteración 3. Modularización de software grande
- Orden de evaluación, precedencia y asociatividad
- Evaluación ansiosa vs diferida
- Definir construcciones de control e iteración
- Llamadas externas y bibliotecas del sistema

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Discutir el papel de conceptos como la ortogonalidad y valores por defecto bien elegidos en el diseño de lenguajes [Debatir]
2. Evaluar objetivamente y justificar decisiones de diseño de lenguaje [Evaluar]

3. Implementar un programa de ejemplo cuyo resultado pueda diferir bajo diferentes reglas para el orden de evaluación, precedencia o asociatividad [Implementar]
4. Ilustrar usos de la evaluación diferida, como abstracciones de control definidas por el usuario [Aplicar]
5. Discutir la necesidad de permitir llamadas a llamadas externas y bibliotecas del sistema y las consecuencias para la implementación del lenguaje [Debatir]

2.5.23. FPL/Semántica Formal

Temas:
Non Core

- Sintaxis vs semántica
- Enfoques de semántica: axiomática, operacional, denotacional, basada en tipos
- Semántica axiomática de construcciones abstractas como asignación, selección, iteración usando precondición, postcondiciones e invariante de bucle
- Análisis de semántica operacional de construcciones abstractas y secuencia de tales como asignación, evaluación de expresión, selección, iteración usando entorno y almacén: 1. Ejecución simbólica (symbolic execution) 2. Verificadores de restricciones (constraint checkers)
- Semántica denotacional: 1. Cálculo Lambda.
- Demostraciones por inducción sobre la semántica del lenguaje
- Definiciones y demostraciones formales para sistemas de tipos: 1. Proposiciones como tipos (implicación como función, conjunción como producto, disyunción como suma) 2. Tipos dependientes (cuantificación universal como función dependiente, cuantificación existencial como producto dependiente) 3. Parametricidad

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Construir semántica formal para un lenguaje pequeño [Crear]
2. Escribir un programa de cálculo lambda y mostrar su evaluación a una forma normal [Escribir]
3. Discutir los diferentes enfoques de la semántica operacional, denotacional y axiomática [Debatir]
4. Usar inducción para demostrar propiedades de todos los programas en un lenguaje [Usar]
5. Usar inducción para demostrar propiedades de todos los programas en un lenguaje que está bien tipado según un sistema de tipos definido formalmente [Usar]
6. Usar parametricidad para establecer el comportamiento del código dado solo su tipo [Usar]

2.5.24. FPL/Metodologías de Desarrollo Formal

Temas:
Non Core

- Lenguajes y metodologías de especificación formal
- Demostradores de teoremas (theorem provers), asistentes de prueba (proof assistants) y lógicas
- Verificadores de restricciones (constraint checkers)
- Tipos dependientes (cuantificación universal como función dependiente, cuantificación existencial como producto dependiente)
- Especificación y descarga de pruebas (proof discharge) para sistemas de software completamente verificados usando pre/post condiciones, tipos de refinamiento (refinement types), etc.
- Modelado formal e implementación/refinamiento manual de sistemas de software.
- Uso de pruebas simbólicas (symbolic testing) y fuzzing en el desarrollo de software.
- Verificación de modelos (model checking).
- Comprensión de situaciones donde los métodos formales pueden aplicarse efectivamente y cómo estructurar el desarrollo para maximizar su valor.

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Usar técnicas de modelado formal para desarrollar y validar arquitecturas [Usar]
2. Usar lenguajes de programación asistidos por pruebas (proof assisted) para desarrollar artefactos de software completamente especificados y verificados [Usar]

3. Usar soporte de verificador y especificación en lenguajes de programación para validar formalmente propiedades del sistema [Usar]
4. Integrar herramientas de validación simbólica en un flujo de trabajo de programación [Integrar]
5. Discutir cuándo y cómo los métodos formales pueden usarse efectivamente en el proceso de desarrollo [Debatir]

2.5.25. FPL/Principios de Diseño de Lenguajes de Programación

Temas:
Non Core

- Principios de diseño de lenguajes: 1. Simplicidad 2. Seguridad 3. Traducción rápida 4. Código objeto eficiente 5. Ortogonalidad 6. Legibilidad 7. Completitud 8. Estrategias de implementación
- Diseñar un lenguaje para ajustarse a un dominio o problema específico
- Interoperabilidad entre lenguajes de programación
- Portabilidad del lenguaje
- Descripción formal de un lenguaje de programación
- Principios de computación verde (green computing)

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Entender qué constituye un buen diseño de lenguaje y aplicar ese conocimiento para evaluar un lenguaje de programación real [Evaluar]

2.5.26. FPL/Sociedad, ética y Profesión

Temas:
Non Core

- Impacto de los lenguajes de programación centrados en inglés
- Mejorar la accesibilidad e inclusión para personas con discapacidades - Compatibilidad con tecnologías de asistencia
- Factores humanos relacionados con los lenguajes de programación y la usabilidad: 1. Impacto de la sintaxis en la accesibilidad 2. Compatibilidad con diferencias culturales (por ejemplo, moneda, decimales, fechas) 3. Neurodiversidad
- Etimología de términos como ”clase” (class), ”maestro” (master) y ”esclavo” (slave) en lenguajes de programación
- Aumentar la accesibilidad mediante el soporte de múltiples idiomas dentro de las aplicaciones (UTF)

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Diseñar conscientemente lenguajes de programación para que sean inclusivos y no ofensivos [Diseñar]

2.6. Gráficos y Técnicas Interactivas (GIT)

Los Gráficos y Técnicas Interactivas (GIT) engloban los principios, algoritmos y tecnologías involucrados en el renderizado, modelado, animación e interacción con contenido visual e interactivo. Esta área de conocimiento integra conceptos de gráficos por computadora, interacción humano-computadora, visualización y procesamiento de imágenes para crear experiencias digitales inmersivas y efectivas. Con la proliferación de aplicaciones interactivas en entretenimiento, educación, visualización científica e interfaces de usuario, GIT juega un papel crucial en la computación moderna. Los estudiantes que estudian GIT aprenden a diseñar, implementar y evaluar sistemas que generan contenido visual, facilitan la interacción del usuario y aprovechan tecnologías emergentes como la realidad virtual, la realidad aumentada y la computación tangible. El área enfatiza tanto los fundamentos teóricos como las habilidades prácticas, preparando a los graduados para contribuir en diversos campos, incluyendo el desarrollo de videojuegos, simulación, visualización de datos y medios interactivos.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 2.6.1 Conceptos Fundamentales de Gráficos y Técnicas Interactivas (Pág. 62)	4	3
⭐ 2.6.2 Visualización (Pág. 62)		6
⭐ 2.6.3 Renderizado Aplicado y Técnicas (Pág. 63)		15
⭐ 2.6.4 Modelado Geométrico (Pág. 63)		6
⭐ 2.6.5 Sombreado y Renderizado Avanzado (Pág. 63)		6
⭐ 2.6.6 Animación por Computadora (Pág. 64)		6
⭐ 2.6.7 Simulación (Pág. 64)		6
⭐ 2.6.8 Inmersión (Pág. 64)		6
⭐ 2.6.9 Interacción (Pág. 64)		4
⭐ 2.6.10 Procesamiento de Imágenes (Pág. 65)		6
⭐ 2.6.11 Computación Tangible/Física (Pág. 65)		6
2.6.12 Sociedad, ética y Profesión (Pág. 65)	Electivo	

Cuadro 2.6: Lista de KUs del área de Gráficos y Técnicas Interactivas.

2.6.1. GIT/Conceptos Fundamentales de Gráficos y Técnicas Interactivas (CS Core: 4 hrs, KA Core: 3 hrs)

Temas:
Core

- Descripción general del pipeline de gráficos por computadora y sistemas interactivos
- Conceptos básicos de renderizado: rasterización, trazado de rayos (*ray tracing*), modelos de sombreado
- Teoría del color y espacios de color
- Sistemas de coordenadas y transformaciones
- Técnicas básicas de interacción: dispositivos de entrada, manejo de eventos
- Conceptos avanzados de renderizado: iluminación global (*global illumination*), renderizado basado en física
- Hardware gráfico y técnicas de aceleración
- Percepción humana y cognición visual

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar las etapas básicas del pipeline de gráficos por computadora [Explicar]
2. Describir técnicas básicas de renderizado como rasterización y trazado de rayos [Describir]
3. Aplicar teoría del color para diseñar gráficos visualmente efectivos [Aplicar]
4. Implementar transformaciones de coordenadas para gráficos 2D y 3D [Implementar]
5. Diseñar aplicaciones interactivas básicas utilizando dispositivos de entrada comunes y manejo de eventos [Diseñar]
6. Comparar técnicas avanzadas de renderizado como iluminación global y renderizado basado en física [Comparar]
7. Analizar el rol del hardware gráfico en acelerar tareas de renderizado [Analizar]
8. Explicar cómo la percepción humana influye en el diseño de gráficos e interacción [Explicar]

2.6.2. GIT/Visualización (KA Core: 6 hrs)

Temas:
Core

- Principios de visualización de datos: percepción, cognición y codificación efectiva
- Tipos de visualizaciones: científicas, de información, geográficas
- Técnicas de visualización para datos multivariados, temporales y jerárquicos
- Visualización interactiva y exploración
- Evaluación de la efectividad de la visualización

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Aplicar principios de percepción y cognición para diseñar visualizaciones efectivas [Aplicar]
2. Seleccionar técnicas de visualización apropiadas para diferentes tipos de datos y tareas [Evaluar]
3. Implementar visualizaciones interactivas para exploración de datos [Implementar]
4. Evaluar la efectividad de una visualización usando métodos empíricos [Evaluar]

2.6.3. GIT/Renderizado Aplicado y Técnicas (KA Core: 15 hrs)

Temas:
Core

- Algoritmos de rasterización: conversión por barrido (*scan conversion*), antialiasing
- Trazado de rayos (*ray tracing*): estructuras de aceleración, muestreo (*sampling*), sombreado
- Mapeo de texturas (*texture mapping*) y filtrado
- Modelos de sombreado: Lambertiano, Phong, BRDF
- Técnicas avanzadas: mapeo de sombras (*shadow mapping*), mapeo ambiental (*environment mapping*), renderizado diferido (*deferred rendering*)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Implementar algoritmos de rasterización para renderizado de primitivas [Implementar]
2. Implementar trazado de rayos básico con estructuras de aceleración [Implementar]
3. Aplicar técnicas de mapeo de texturas y filtrado para mejorar el realismo visual [Aplicar]
4. Comparar diferentes modelos de sombreado y sus aplicaciones [Comparar]
5. Implementar técnicas avanzadas de renderizado como mapeo de sombras o renderizado diferido [Implementar]

2.6.4. GIT/Modelado Geométrico (KA Core: 6 hrs)

Temas:
Core

- Representaciones geométricas: polígonos, curvas, superficies, formas implícitas
- Técnicas de modelado: curvas spline, superficies de subdivisión, geometría sólida constructiva
- Modelado procedural y generación
- Algoritmos geométricos: intersección, distancia, descomposición

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir diferentes representaciones geométricas y sus compensaciones (trade-offs) [Describir]
2. Implementar técnicas de modelado como curvas spline o superficies de subdivisión [Implementar]
3. Crear modelos procedurales usando algoritmos o reglas [Crear]
4. Aplicar algoritmos geométricos para resolver problemas como pruebas de intersección o procesamiento de mallas (mesh) [Aplicar]

2.6.5. GIT/Sombreado y Renderizado Avanzado (KA Core: 6 hrs)

Temas:
Core

- Sombreado avanzado: sombreado basado en física, dispersión subsuperficial (subsurface scattering)
- Iluminación global (global illumination): radiosidad, mapeo de fotones (photon mapping), trazado de caminos (path tracing)
- Técnicas de renderizado en tiempo real: nivel de detalle (level of detail), culling, oclusión
- Programación GPU y sombreadores (shaders)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Implementar modelos de sombreado avanzados para materiales realistas [Implementar]
2. Comparar algoritmos de iluminación global y sus compensaciones [Comparar]
3. Optimizar el renderizado en tiempo real usando técnicas como nivel de detalle y *culling* [Crear]
4. Desarrollar sombreadores GPU para efectos de renderizado personalizados [Crear]

2.6.6. GIT/Animación por Computadora (KA Core: 6 hrs)

Temas:
Core

- Principios de animación: tiempo (timing), espaciado (spacing), anticipación, seguimiento (follow-through)
- Animación por fotogramas clave (keyframe) e interpolación
- Animación procedural: sistemas de partículas, comportamiento de bandadas (flocking), basada en física
- Animación de personajes: animación esquelética, skinning, cinemática inversa
- Captura de movimiento (motion capture) y retargeting

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Aplicar principios de animación para crear movimiento creíble [Aplicar]
2. Implementar animación por fotogramas clave con técnicas de interpolación [Implementar]
3. Crear animaciones procedurales como sistemas de partículas o simulaciones basadas en física [Crear]
4. Implementar animación de personajes usando sistemas esqueléticos y skinning [Implementar]
5. Procesar datos de captura de movimiento para retargeting de animación [Implementar]

2.6.7. GIT/Simulación (KA Core: 6 hrs)

Temas:
Core

- Simulación basada en física: cuerpos rígidos, fluidos, tela
- Métodos numéricos para simulación: integración, detección de colisiones
- Restricciones y optimizaciones de simulación en tiempo real
- Aplicaciones en videojuegos, efectos visuales y visualización científica

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Implementar simulaciones basadas en física para cuerpos rígidos o fluidos [Implementar]
2. Aplicar métodos de integración numérica para simulaciones dinámicas [Aplicar]
3. Optimizar simulaciones para rendimiento en tiempo real [Crear]
4. Desarrollar simulaciones para aplicaciones específicas como videojuegos o visualización científica [Crear]

2.6.8. GIT/Inmersión (KA Core: 6 hrs)

Temas:
Core

- Conceptos de realidad virtual (VR) y realidad aumentada (AR)
- Tecnologías de visualización inmersiva: pantallas montadas en la cabeza (head-mounted displays), CAVEs
- Seguimiento (tracking) y entrada para entornos inmersivos
- Presencia y comodidad en sistemas inmersivos

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar los conceptos centrales y diferencias entre VR y AR [Explicar]
2. Evaluar tecnologías de visualización inmersiva y sus aplicaciones [Evaluar]
3. Implementar soluciones de seguimiento y entrada para entornos inmersivos [Implementar]
4. Diseñar experiencias inmersivas que maximicen la presencia y minimicen la incomodidad [Diseñar]

2.6.9. GIT/Interacción (KA Core: 4 hrs)

Temas:
Core

- Paradigmas de interacción: manipulación directa, basada en gestos, tangible
- Diseño de interfaz de usuario para aplicaciones gráficas
- Interacción multimodal: combinando retroalimentación visual, auditiva, háptica
- Evaluación de técnicas de interacción: usabilidad, experiencia de usuario

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comparar diferentes paradigmas de interacción y su idoneidad para varias tareas [Comparar]
2. Diseñar interfaces de usuario para aplicaciones intensivas en gráficos [Diseñar]
3. Implementar sistemas de interacción multimodal incorporando múltiples canales de retroalimentación [Implementar]
4. Realizar evaluaciones de usabilidad de técnicas de interacción [Evaluar]

2.6.10. GIT/Procesamiento de Imágenes (KA Core: 6 hrs)

Temas:
Core

- Representación y formatos de imagen
- Filtrado y mejora: convolución, dominio de frecuencia
- Segmentación de imágenes y detección de características
- Operaciones morfológicas
- Aplicaciones en gráficos, visión y multimedia

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Manipular imágenes usando diferentes representaciones y formatos [Procesar]
2. Aplicar técnicas de filtrado para mejora y análisis de imágenes [Aplicar]
3. Implementar algoritmos de segmentación de imágenes para extracción de características [Implementar]
4. Usar operaciones morfológicas para tareas de procesamiento de imágenes [Usar]
5. Desarrollar aplicaciones que integren procesamiento de imágenes con sistemas de gráficos o visión [Crear]

2.6.11. GIT/Computación Tangible/Física (KA Core: 6 hrs)

Temas:
Core

- Interfaces tangibles y computación física
- Sensores y actuadores para sistemas interactivos
- Sistemas embebidos para arte y diseño interactivo
- Instalaciones y entornos interactivos

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Diseñar interfaces tangibles que unan interacciones físicas y digitales [Diseñar]
2. Integrar sensores y actuadores para crear sistemas responsivos [Integrar]
3. Desarrollar sistemas embebidos para proyectos de arte o diseño interactivo [Crear]
4. Crear instalaciones interactivas que involucren a los usuarios en espacios físicos [Crear]

2.6.12. GIT/Sociedad, ética y Profesión

Temas:
Core

- Consideraciones éticas en gráficos e interacción: representación, accesibilidad, privacidad
- Impacto social de las tecnologías inmersivas y la visualización
- Práctica profesional en industrias de gráficos e interactivas

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Analizar problemas éticos relacionados con representación, accesibilidad y privacidad en gráficos e interacción [Analizar]
2. Discutir las implicaciones sociales de las tecnologías inmersivas y la visualización de datos [Debatir]
3. Describir prácticas y estándares profesionales en industrias de gráficos e interactivas [Describir]

2.7. Interacción Humano-Computadora (HCI)

La Interacción Humano-Computadora (HCI) es un campo multidisciplinario que se centra en el diseño, evaluación e implementación de sistemas informáticos interactivos para uso humano, y el estudio de los principales fenómenos que los rodean. HCI integra conocimientos y métodos de la informática, psicología, diseño, sociología y otros campos para crear tecnologías que sean útiles, usables y satisfactorias para las personas. El objetivo central de HCI es mejorar las interacciones entre usuarios y computadoras haciendo los sistemas más usables, accesibles y eficientes, considerando al mismo tiempo las capacidades y limitaciones humanas.

En el mundo actual, donde la tecnología informática impregna todos los aspectos de la vida diaria-desde teléfonos inteligentes y sitios web hasta dispositivos médicos y sistemas de hogares inteligentes-los principios de HCI son esenciales para crear sistemas que las personas puedan usar de manera efectiva y segura. HCI enfatiza procesos de diseño centrados en el usuario, prototipado iterativo y evaluación empírica para asegurar que los sistemas satisfagan las necesidades reales de los usuarios. El campo aborda diversos temas que incluyen usabilidad, experiencia de usuario (UX), accesibilidad, diseño de interacción, métodos de investigación de usuarios y las implicaciones sociales de la tecnología.

HCI prepara a los estudiantes para diseñar, implementar y evaluar sistemas interactivos que no solo sean funcionales sino que también consideren factores humanos, implicaciones éticas e impacto social. Los graduados con experiencia en HCI trabajan en roles como diseñadores de UX, diseñadores de interacción, ingenieros de usabilidad y gerentes de productos en diversas industrias, incluyendo desarrollo de software, atención médica, educación, entretenimiento y políticas públicas.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 2.7.1 Comprender al Usuario: Objetivos individuales e interacciones con otros (Pág. 66)	2	5
⭐ 2.7.2 Responsabilidad y Rendición de Cuentas en el Diseño (Pág. 67)	2	2
⭐ 2.7.3 Accesibilidad y Diseño Inclusivo (Pág. 67)	2	2
⭐ 2.7.4 Evaluación del Diseño (Pág. 67)	1	2
⭐ 2.7.5 Diseño de Sistemas (Pág. 68)	1	5
2.7.6 Sociedad, ética y Profesión (Pág. 68)	Electivo	

Cuadro 2.7: Lista de KUs del área de Interacción Humano-Computadora.

2.7.1. HCI/Comprender al Usuario: Objetivos individuales e interacciones con otros (CS Core: 2 hrs, KA Core: 5 hrs)

Temas:
Core

- Capacidades y limitaciones humanas: percepción, cognición, habilidades motoras
- Características del usuario: demografía, experiencia, contexto cultural
- Objetivos, tareas y contextos de uso del usuario
- Aspectos sociales y organizacionales de la interacción
- Modelos y teorías cognitivas relevantes para HCI
- Aspectos emocionales y afectivos de la interacción
- Métodos de investigación de usuarios: entrevistas, encuestas, observaciones

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir las capacidades y limitaciones humanas que afectan el diseño de interacción [Describir]

2. Analizar cómo las características del usuario influyen en la interacción con sistemas informáticos [Analizar]
3. Identificar objetivos y tareas del usuario en contextos de uso específicos [Analizar]
4. Explicar factores sociales y organizacionales que impactan la adopción y uso de la tecnología [Explicar]
5. Aplicar modelos cognitivos para predecir y explicar el comportamiento del usuario [Aplicar]
6. Considerar factores emocionales y afectivos en el diseño de interacción [Analizar]
7. Realizar investigación de usuarios usando métodos apropiados como entrevistas u observaciones [Evaluar]

2.7.2. HCI/Responsabilidad y Rendición de Cuentas en el Diseño (CS Core: 2 hrs, KA Core: 2 hrs)

Temas:
Core

- Consideraciones éticas en HCI: equidad, transparencia, rendición de cuentas
- Diseño para valores y diseño sensible a valores
- Innovación responsable y ética del diseño
- Rendición de cuentas algorítmica y sesgo en sistemas interactivos
- Responsabilidad profesional en la práctica de HCI
- Consideraciones legales y regulatorias para sistemas interactivos

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar problemas éticos en el diseño e implementación de HCI [Analizar]
2. Aplicar enfoques de diseño sensible a valores al desarrollo tecnológico [Aplicar]
3. Discutir prácticas de innovación responsable en HCI [Debatir]
4. Analizar la rendición de cuentas algorítmica y el sesgo en sistemas interactivos [Analizar]
5. Describir responsabilidades profesionales en la práctica de HCI [Describir]
6. Explicar consideraciones legales y regulatorias relevantes para sistemas interactivos [Explicar]

2.7.3. HCI/Accesibilidad y Diseño Inclusivo (CS Core: 2 hrs, KA Core: 2 hrs)

Temas:
Core

- Principios y pautas de accesibilidad
- Discapacidades y tecnologías de asistencia
- Diseño inclusivo y principios de diseño universal
- Métodos de evaluación para accesibilidad
- Diseñar para diversas capacidades y contextos
- Estándares de accesibilidad y cumplimiento

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Aplicar principios de accesibilidad al diseño de sistemas interactivos [Aplicar]
2. Describir cómo diferentes discapacidades afectan la interacción con la tecnología [Describir]
3. Diseñar sistemas inclusivos usando principios de diseño universal [Diseñar]
4. Realizar evaluaciones de accesibilidad de sistemas interactivos [Evaluar]
5. Crear diseños que acomoden diversas capacidades y contextos de uso [Crear]
6. Garantizar el cumplimiento con estándares y regulaciones de accesibilidad [Evaluar]

2.7.4. HCI/Evaluación del Diseño (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Principios y heurísticas de usabilidad
- Métodos de evaluación: evaluación heurística, recorrido cognitivo (cognitive walkthrough)
- Pruebas con usuarios: planificación, ejecución, análisis

- Métodos avanzados de evaluación: experimentos controlados, estudios de campo
- Métricas y medición de UX
- Métodos de evaluación analítica: modelado y simulación

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Aplicar principios y heurísticas de usabilidad para evaluar diseños [Aplicar]
2. Realizar evaluaciones heurísticas y recorridos cognitivos [Evaluar]
3. Planificar, conducir y analizar sesiones de pruebas con usuarios [Planear]
4. Diseñar y conducir experimentos controlados para investigación en HCI [Diseñar]
5. Medir la experiencia de usuario usando métricas apropiadas [Evaluar]
6. Usar métodos analíticos para modelar y predecir el desempeño del usuario [Usar]

2.7.5. HCI/Diseño de Sistemas (CS Core: 1 hr, KA Core: 5 hrs)

Temas:
Core

- Proceso de diseño: diseño centrado en el usuario, diseño iterativo
- Diseño de interacción: conceptos, principios, patrones
- Técnicas de prototipado: baja fidelidad, alta fidelidad, interactivo
- Arquitectura de información y diseño de navegación
- Técnicas avanzadas de interacción: gestos, voz, multimodal
- Diseño para dominios específicos: móvil, web, computación ubicua
- Sistemas de computación colaborativa y social

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Aplicar procesos de diseño centrado en el usuario para desarrollar sistemas interactivos [Aplicar]
2. Diseñar interacciones usando principios y patrones establecidos [Diseñar]
3. Crear prototipos en diferentes niveles de fidelidad para exploración de diseño [Crear]
4. Estructurar información y navegación para una interacción efectiva del usuario [Estructurar]
5. Implementar técnicas avanzadas de interacción como interfaces de gestos o voz [Implementar]
6. Diseñar sistemas interactivos para dominios y plataformas específicos [Diseñar]
7. Diseñar sistemas colaborativos que apoyen la interacción grupal [Diseñar]

2.7.6. HCI/Sociedad, ética y Profesión

Temas:
Core

- Impacto social de las tecnologías interactivas
- Problemas éticos en la investigación y práctica de HCI
- Práctica profesional en HCI
- Consideraciones culturales e interculturales en HCI
- Sostenibilidad y HCI
- Tendencias futuras y desafíos en HCI

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Analizar el impacto social de las tecnologías interactivas en individuos y sociedad [Analizar]
2. Abordar problemas éticos en proyectos de investigación y diseño de HCI [Evaluar]
3. Describir prácticas profesionales y trayectorias profesionales en HCI [Describir]
4. Diseñar sistemas que tomen en cuenta diferencias y contextos culturales [Diseñar]
5. Incorporar consideraciones de sostenibilidad en el diseño de HCI [Integrar]
6. Discutir tendencias emergentes y desafíos futuros en HCI [Debatir]

2.8. Fundamentos Matemáticos y Estadísticos (MSF)

Los fundamentos matemáticos y estadísticos de la informática proporcionan las bases teóricas para la disciplina. Incluyen matemáticas discretas, probabilidad, estadística, álgebra lineal y cálculo.

Estas áreas son esenciales para comprender algoritmos, estructuras de datos, aprendizaje automático, gráficos, criptografía y muchos otros temas en informática.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 2.8.1 Matemáticas Discretas (Pág. 69)	15	6
⭐ 2.8.2 Matemáticas Discretas Avanzadas (Pág. 69)	15	6
⭐ 2.8.3 Probabilidad (Pág. 70)	11	29
⭐ 2.8.4 Estadística (Pág. 70)	10	30
⭐ 2.8.5 álgebra Lineal (Pág. 71)	5	35
⭐ 2.8.6 Cálculo (Pág. 71)		40

Cuadro 2.8: Lista de KUs del área de Fundamentos Matemáticos y Estadísticos.

2.8.1. MSF/Matemáticas Discretas (CS Core: 15 hrs, KA Core: 6 hrs)

Temas:
Core

- Conjuntos: operaciones, cardinalidad, conjuntos potencia, productos cartesianos
- Relaciones: propiedades, relaciones de equivalencia, órdenes parciales
- Funciones: inyectivas, sobreyectivas, biyectivas, composición, inversa
- Lógica: lógica proposicional y de predicados, cuantificadores, demostraciones
- Grafos: definiciones, caminos, ciclos, árboles, conectividad
- Conteo: permutaciones, combinaciones, teorema del binomio
- Relaciones de recurrencia: resolver recurrencias simples

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Definir operaciones y relaciones básicas de conjuntos [Explicar]
2. Aplicar operaciones de conjuntos para resolver problemas [Aplicar]
3. Definir propiedades de las relaciones [Explicar]
4. Determinar si una relación es una relación de equivalencia [Aplicar]
5. Definir tipos de funciones [Explicar]
6. Componer funciones y encontrar inversas [Aplicar]
7. Explicar lógica proposicional y de predicados [Explicar]
8. Construir demostraciones lógicas simples [Aplicar]
9. Definir conceptos básicos de grafos [Explicar]
10. Resolver problemas básicos de grafos [Aplicar]
11. Explicar permutaciones y combinaciones [Explicar]
12. Aplicar técnicas de conteo para resolver problemas [Aplicar]
13. Definir relaciones de recurrencia [Explicar]
14. Resolver relaciones de recurrencia simples [Aplicar]

2.8.2. MSF/Matemáticas Discretas Avanzadas (CS Core: 15 hrs, KA Core: 6 hrs)

Temas:
Core

- Teoría avanzada de conjuntos: conjuntos infinitos, argumento diagonal de Cantor
- Relaciones avanzadas: clausuras, retículos
- Funciones avanzadas: funciones recursivas, computabilidad
- Lógica avanzada: sistemas de demostración, completitud, decidibilidad
- Teoría avanzada de grafos: planaridad, coloración, redes
- Conteo avanzado: funciones generadoras, principio de inclusión-exclusión
- Recurrencia avanzada: funciones generadoras, comportamiento asintótico

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar conceptos de conjuntos infinitos y cardinalidad [Explicar]
2. Aplicar operaciones de clausura a relaciones [Aplicar]

3. Analizar propiedades avanzadas de grafos [Evaluar]
4. Desarrollar funciones generadoras para problemas de conteo [Crear]

2.8.3. MSF/Probabilidad (CS Core: 11 hrs, KA Core: 29 hrs)

Temas:
Core

- Probabilidad básica: espacios muestrales, eventos, axiomas
- Probabilidad condicional, independencia, teorema de Bayes
- Variables aleatorias: discretas y continuas, distribuciones
- Esperanza, varianza, desviación estándar
- Distribuciones comunes: uniforme, binomial, normal
- Distribuciones conjuntas, covarianza, correlación
- Teoremas límite: ley de los grandes números, teorema del límite central
- Procesos estocásticos: cadenas de Markov
- Teoría de la información: entropía, información mutua

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Definir conceptos básicos de probabilidad [Explicar]
2. Calcular probabilidades de eventos [Aplicar]
3. Definir probabilidad condicional e independencia [Explicar]
4. Aplicar el teorema de Bayes para actualizar probabilidades [Aplicar]
5. Definir variables aleatorias y distribuciones [Explicar]
6. Calcular esperanza y varianza [Aplicar]
7. Describir distribuciones de probabilidad comunes [Explicar]
8. Aplicar distribuciones comunes para modelar fenómenos del mundo real [Aplicar]
9. Explicar distribuciones conjuntas y dependencia [Explicar]
10. Calcular covarianza y correlación [Aplicar]
11. Analizar las implicaciones de los teoremas límite [Evaluar]
12. Modelar sistemas simples usando cadenas de Markov [Crear]

2.8.4. MSF/Estadística (CS Core: 10 hrs, KA Core: 30 hrs)

Temas:
Core

- Estadística descriptiva: medidas de centro y dispersión
- Visualización de datos: histogramas, diagramas de caja, diagramas de dispersión
- Distribuciones muestrales: variabilidad muestral, teorema del límite central
- Estimación: estimaciones puntuales, intervalos de confianza
- Prueba de hipótesis: hipótesis nula y alternativa, valores p
- Regresión: regresión lineal simple y múltiple
- Análisis de varianza (ANOVA)
- Métodos no paramétricos
- Diseño experimental: aleatorización, control, enmascaramiento

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Definir medidas de estadística descriptiva [Explicar]
2. Calcular estadísticas descriptivas para un conjunto de datos [Aplicar]
3. Explicar técnicas comunes de visualización de datos [Explicar]
4. Crear visualizaciones apropiadas para datos dados [Aplicar]
5. Definir distribuciones muestrales y variabilidad [Explicar]
6. Aplicar el teorema del límite central para aproximar distribuciones [Aplicar]
7. Definir estimaciones puntuales e intervalos de confianza [Explicar]
8. Calcular intervalos de confianza para medias y proporciones [Aplicar]
9. Definir conceptos de prueba de hipótesis [Explicar]
10. Realizar una prueba de hipótesis simple e interpretar valores p [Aplicar]

11. Explicar regresión lineal simple y múltiple [Explicar]
12. Ajustar un modelo de regresión e interpretar coeficientes [Aplicar]
13. Analizar varianza usando ANOVA [Evaluar]
14. Diseñar un experimento simple con controles apropiados [Crear]

2.8.5. MSF/álgebra Lineal (CS Core: 5 hrs, KA Core: 35 hrs)

Temas:
Core

- Vectores: operaciones, producto punto, producto cruz
- Matrices: operaciones, inversas, determinantes
- Sistemas de ecuaciones lineales: eliminación gaussiana, forma matricial
- Espacios vectoriales: subespacios, base, dimensión
- Transformaciones lineales: representación matricial, núcleo, imagen
- Valores propios y vectores propios: diagonalización
- Ortogonalidad: bases ortonormales, Gram-Schmidt
- Descomposición en valores singulares (SVD)
- Aplicaciones: mínimos cuadrados, gráficos por computadora, aprendizaje automático

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Definir operaciones y propiedades de vectores [Explicar]
2. Calcular operaciones y productos vectoriales [Aplicar]
3. Definir operaciones y propiedades matriciales [Explicar]
4. Calcular operaciones matriciales, inversas y determinantes [Aplicar]
5. Explicar sistemas de ecuaciones lineales y métodos de solución [Explicar]
6. Resolver sistemas de ecuaciones lineales usando eliminación [Aplicar]
7. Definir espacios vectoriales, subespacios, base y dimensión [Explicar]
8. Determinar si un conjunto de vectores forma una base [Aplicar]
9. Definir transformaciones lineales y sus representaciones [Explicar]
10. Encontrar el núcleo y la imagen de una transformación lineal [Aplicar]
11. Explicar valores propios y vectores propios [Explicar]
12. Diagonalizar una matriz cuando sea posible [Aplicar]
13. Analizar ortogonalidad y bases ortonormales [Evaluar]
14. Aplicar descomposición en valores singulares a una matriz [Crear]

2.8.6. MSF/Cálculo (KA Core: 40 hrs)

Temas:
Core

- Límites y continuidad
- Derivadas: definición, reglas, aplicaciones
- Integrales: definidas e indefinidas, teorema fundamental
- Series: convergencia, series de Taylor
- Cálculo multivariable: derivadas parciales, gradientes
- Optimización: extremos locales y globales, multiplicadores de Lagrange
- Ecuaciones diferenciales: técnicas básicas de solución
- Métodos numéricos: integración, diferenciación

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Definir límites y continuidad [Explicar]
2. Calcular límites de funciones [Aplicar]
3. Definir derivadas y reglas de diferenciación [Explicar]
4. Calcular derivadas de funciones [Aplicar]
5. Definir integrales y el teorema fundamental del cálculo [Explicar]
6. Calcular integrales definidas e indefinidas [Aplicar]
7. Definir series y convergencia [Explicar]

8. Calcular expansiones en series de Taylor [Aplicar]
9. Explicar derivadas parciales y gradientes [Explicar]
10. Calcular derivadas parciales y gradientes [Aplicar]
11. Analizar problemas de optimización usando cálculo [Evaluar]
12. Resolver ecuaciones diferenciales simples [Crear]

2.9. Redes y Comunicaciones (NC)

Las redes y comunicaciones juegan un papel central en los sistemas informáticos interconectados que están transformando la vida diaria de miles de millones de personas. Internet público proporciona conectividad para aplicaciones en red que sirven a un número cada vez mayor de individuos y organizaciones en todo el mundo. Complementando al sector público, las principales redes propietarias aprovechan su alcance global para apoyar la computación, el almacenamiento y la entrega de contenido distribuidos rentables. Los avances en las redes satelitales amplían la conectividad a las zonas rurales. La comunicación dispositivo a dispositivo subyace al emergente Internet de las Cosas.

Esta área de conocimiento trata conceptos clave en redes y comunicaciones, así como sus instanciaciones representativas en Internet y otras redes informáticas. Además de los principios básicos de conmutación y estratificación, el área en su núcleo proporciona conocimiento sobre denominación, direccionamiento, fiabilidad, control de errores, control de flujo, control de congestión, jerarquía de dominios, enrutamiento, reenvío, modulación, codificación, entramado y control de acceso. El área también cubre unidades de conocimiento en seguridad de red y movilidad, como amenazas de seguridad, contramedidas, comunicación dispositivo a dispositivo y redes inalámbricas multiescalón. Además de los principios fundamentales, el área incluye su realización específica en Internet, así como habilidades prácticas en la implementación de conceptos de redes y comunicaciones. Finalmente, el área comprende temas emergentes como la virtualización de red y las redes cuánticas.

Como resultado de aprendizaje principal, los estudiantes desarrollan una comprensión exhaustiva del papel y funcionamiento de las redes y comunicaciones en sistemas informáticos en red. Aprenden cómo la estructura de red y los protocolos de comunicación afectan el comportamiento de las aplicaciones distribuidas. El área puede usarse para educar no solo los principios clave sino también sus instanciaciones específicas en Internet y equipar al estudiante con habilidades prácticas de implementación. Si bien las tecnologías de sistemas informáticos, redes y comunicaciones avanzan a un ritmo acelerado, el conocimiento fundamental adquirido permite al estudiante aplicar fácilmente los conceptos en nuevos entornos tecnológicos.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 2.9.1 Fundamentos de Redes y Comunicaciones (Pág. 72)	3	
⭐ 2.9.2 Aplicaciones en Red (Pág. 73)	4	
⭐ 2.9.3 Soporte de Fiabilidad (Pág. 73)		6
⭐ 2.9.4 Enrutamiento y Reenvío (Pág. 74)		4
⭐ 2.9.5 Comunicación de un solo salto (Pág. 74)		3
⭐ 2.9.6 Seguridad de Red (Pág. 74)		3
⭐ 2.9.7 Soporte de Movilidad (Pág. 75)		4
⭐ 2.9.8 Temas Emergentes (Pág. 75)		4

Cuadro 2.9: Lista de KUs del área de Redes y Comunicaciones.

2.9.1. NC/Fundamentos de Redes y Comunicaciones (CS Core: 3 hrs)

Temas:
Core

- Importancia de las redes en la informática contemporánea, y desafíos asociados. Ver también: Equidad Digital y Accesibilidad, Impacto Social e IA, Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad
- Organización de Internet (por ejemplo, usuarios, Proveedores de Servicios de Internet, sistemas autónomos, proveedores de contenido, redes de entrega de contenido)

- Técnicas de conmutación (por ejemplo, circuito y paquete)
- Capas y sus roles (aplicación, transporte, red, enlace de datos y física)
- Principios de estratificación (por ejemplo, encapsulación y modelo de reloj de arena). Ver también: Sistemas Digitales y Lógica, Procesamiento Secuencial y Paralelo, Máquinas de Estado

- Elementos de red (por ejemplo, routers, switches, hubs, puntos de acceso y hosts)
- Conceptos básicos de colas (por ejemplo, relación con latencia, congestión, niveles de servicio, etc.)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Articular la organización de Internet [Articular]
2. Listar y definir la terminología de red apropiada [Listar/Enumerar]
3. Describir la estructura en capas de una arquitectura de red típica [Describir]
4. Identificar los diferentes tipos de complejidad en una red (bordes, núcleo, etc.) [Analizar]

2.9.2. NC/Aplicaciones en Red (CS Core: 4 hrs)

Temas:
Core

- Esquemas de denominación y direccionamiento (por ejemplo, DNS e Identificadores Uniformes de Recursos)
- Paradigmas de aplicaciones distribuidas (por ejemplo, cliente/servidor, peer-to-peer, nube, edge y fog). Ver también: Comunicación, Coordinación
- Diversidad de demandas de aplicaciones en red (por ejemplo, latencia, ancho de banda y tolerancia a pérdidas). Ver también: Comunicación, Diseño Sostenible y Computación Ubicua, Huella Ambiental de los Sistemas Informáticos, Efectos Sistémicos y Contexto Social, Equidad Digital y Accesibilidad, Impacto Social e IA
- Cobertura de protocolos de capa de aplicación (por ejemplo, HTTP)
- Interacciones con APIs de TCP, UDP y Socket. Ver también: Programas: Distribución, Programas: Propiedades de Ejecución, Programación GPU, Programas: Mapeos de Implementación, Programas: Paralelismo Declarativo, Programas: Inicio de Actividades

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Definir los principios de denominación, direccionamiento y localización de recursos [Definir]
2. Analizar las necesidades de demandas específicas de aplicaciones en red [Analizar]
3. Describir los detalles de un protocolo de capa de aplicación [Describir]
4. Implementar una aplicación simple cliente-servidor basada en sockets [Implementar]

2.9.3. NC/Soporte de Fiabilidad (KA Core: 6 hrs)

Temas:
Core

- Entrega no fiable (por ejemplo, UDP)
- Principios de fiabilidad (por ejemplo, entrega sin pérdidas, duplicación o fuera de orden). Ver también: Confiabilidad del Sistema
- Control de errores (por ejemplo, retransmisión, corrección de errores)
- Control de flujo (por ejemplo, parar y esperar, basado en ventana)
- Control de congestión (por ejemplo, notificación implícita y explícita de congestión)
- TCP y problemas de rendimiento (por ejemplo, Tahoe, Reno, Vegas, Cubic)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir el funcionamiento de protocolos de entrega fiable [Describir]
2. Listar los factores que afectan el rendimiento de los protocolos de entrega fiable [Listar/Enumerar]
3. Describir algunos problemas de diseño de fiabilidad de TCP [Describir]
4. Diseñar e implementar un protocolo fiable simple [Diseñar]

2.9.4. NC/Enrutamiento y Reenvío (KA Core: 4 hrs)

Temas:
Core

- Paradigmas y jerarquía de enrutamiento (por ejemplo, intra/inter dominio, centralizado y descentralizado, enrutamiento de origen, circuitos virtuales, QoS)
- Métodos de reenvío (por ejemplo, tablas de reenvío y algoritmos de coincidencia)
- IP y problemas de escalabilidad (por ejemplo, NAT, CIDR, BGP, diferentes versiones de IP)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir varios paradigmas y jerarquías de enrutamiento [Describir]
2. Describir cómo se reenvían los paquetes en una red IP [Describir]
3. Describir cómo Internet aborda los desafíos de escalabilidad [Describir]

2.9.5. NC/Comunicación de un solo salto (KA Core: 3 hrs)

Temas:
Core

- Introducción a modulación, ancho de banda y medios de comunicación
- Codificación y Entramado
- Control de Acceso al Medio (MAC) (por ejemplo, acceso aleatorio y acceso programado)
- Ethernet y WiFi
- Conmutación (por ejemplo, árboles de expansión, VLANs).
- Topologías de Red de área Local (por ejemplo, centro de datos, redes de campus).

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir algunos aspectos básicos de modulación, ancho de banda y medios de comunicación [Describir]
2. Describir en detalle un protocolo MAC [Describir]
3. Demostrar comprensión de las compensaciones (trade-offs) de soluciones de codificación y entramado [Demostrar]
4. Describir detalles de la implementación de Ethernet [Describir]
5. Describir cómo funciona la conmutación [Describir]
6. Describir un tipo de topología de LAN [Describir]

2.9.6. NC/Seguridad de Red (KA Core: 3 hrs)

Temas:
Core

- Introducción general sobre seguridad (Amenazas, vulnerabilidades y contramedidas). Ver también: Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad, Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad, Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad
- Amenazas específicas de red y tipos de ataque (por ejemplo, denegación de servicio, suplantación, rastreo y redireccionamiento de tráfico, atacante en el medio, ataques de integridad de mensajes, ataques de enrutamiento, ransomware y análisis de tráfico). Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad, Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad
- Contramedidas: 1. Criptografía (por ejemplo, SSL, TLS, simétrica/asimétrica) 2. Arquitecturas para redes seguras (por ejemplo, canales seguros, protocolos de enrutamiento seguro, DNS seguro, VPNs, DMZ, Zero Trust Network Access, hiper seguridad de red, protocolos de comunicación anónima, aislamiento) 3. Monitoreo de red, detección de intrusiones, firewalls, protección contra suplantación y DoS, honeypots, trazabilidad, BGP Sec, RPKI

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir algunos de los modelos de amenaza de seguridad de red [Describir]
2. Describir contramedidas específicas basadas en red [Describir]
3. Analizar varios aspectos de seguridad de red a partir de un estudio de caso [Analizar]

2.9.7. NC/Soporte de Movilidad (KA Core: 4 hrs)

Temas:
Core

- Principios de comunicación celular (por ejemplo, 4G, 5G)
- Principios de LANs inalámbricas (principalmente 802.11)
- Comunicación dispositivo a dispositivo (por ejemplo, comunicación IoT)
- Redes inalámbricas multiescalón (por ejemplo, redes ad hoc, oportunistas, tolerantes a retrasos)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir algunos aspectos de comunicación celular como el registro [Describir]
2. Describir cómo 802.11 soporta usuarios móviles [Describir]
3. Describir usos prácticos de comunicación dispositivo a dispositivo, así como multiescalón [Describir]
4. Describir un tipo de red móvil como ad hoc [Describir]

2.9.8. NC/Temas Emergentes (KA Core: 4 hrs)

Temas:
Core

- Middleboxes (por ejemplo, avances en el uso de IA, redes basadas en intención, filtrado, inspección profunda de paquetes, balanceo de carga, NAT, CDN)
- Virtualización de Red (por ejemplo, SDN, Redes de Centros de Datos)
- Redes Cuánticas (por ejemplo, Introducción al dominio, teleportación, seguridad, Internet Cuántica)
- Satélite, mmWave, Luz Visible

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir el valor de los avances en middleboxes en redes [Describir]
2. Describir la importancia de las Redes Definidas por Software [Describir]
3. Describir algunos de los valores añadidos logrados al usar Redes Cuánticas [Describir]

2.10. Sistemas Operativos (OS)

El sistema operativo es una colección de servicios necesarios para interconectar de manera segura el hardware con las aplicaciones. Los temas centrales se centran en los mecanismos y políticas necesarios para virtualizar cómputo, memoria y entrada/salida (I/O). Los temas generales que se reutilizan en muchos niveles de los sistemas informáticos están bien ilustrados en los sistemas operativos (por ejemplo, sondeo vs interrupciones, caché, flexibilidad vs costos, enfoques de planificación de procesos, reemplazo de páginas, etc.). El área de conocimiento de Sistemas Operativos contiene los conceptos subyacentes clave para otras áreas de conocimiento: límites de confianza, concurrencia, persistencia y extensibilidad segura.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 2.10.1 Rol y Propósito de los Sistemas Operativos (Pág. 76)	2	
⭐ 2.10.2 Principios del Sistema Operativo (Pág. 77)	2	
⭐ 2.10.3 Concurrencia (Pág. 77)	2	1
⭐ 2.10.4 Protección y Seguridad (Pág. 78)	2	1
⭐ 2.10.5 Planificación (Pág. 79)		2
⭐ 2.10.6 Modelo de Proceso (Pág. 79)		2
⭐ 2.10.7 Gestión de Memoria (Pág. 80)		1
⭐ 2.10.8 Gestión de Dispositivos (Pág. 80)		1
⭐ 2.10.9 API de Sistemas de Archivos e Implementación (Pág. 81)		2
⭐ 2.10.10 Sistemas de Archivos Avanzados (Pág. 81)		1
⭐ 2.10.11 Virtualización (Pág. 82)		1
⭐ 2.10.12 Sistemas en Tiempo Real y Embebidos (Pág. 83)		1
⭐ 2.10.13 Tolerancia a fallos (Pág. 83)		1
2.10.14 Sociedad, ética y Profesión (Pág. 84)	Electivo	

Cuadro 2.10: Lista de KUs del área de Sistemas Operativos.

2.10.1. OS/Rol y Propósito de los Sistemas Operativos (CS Core: 2 hrs)

Temas:
Core

- Los sistemas operativos median entre el hardware de propósito general y el software específico de la aplicación.
- Funciones universales del sistema operativo (por ejemplo, proceso, interfaces de usuario y dispositivos, persistencia de datos)
- Funciones extendidas y/o especializadas del sistema operativo (por ejemplo, sistemas embebidos, tipos de servidor como archivos, web, multimedia, gestores de arranque y seguridad de arranque)
- Problemas de diseño (por ejemplo, eficiencia, robustez, flexibilidad, portabilidad, seguridad, compatibilidad, energía, seguridad, compensaciones entre verificación de errores y rendimiento, flexibilidad y rendimiento, y seguridad y rendimiento). Ver también: Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad
- Influencias de seguridad, redes, multimedia, computación paralela y distribuida
- Preocupación general de seguridad/protección: Omitir considerar la seguridad en cada capa crea una oportunidad para acceder de manera inapropiada a los recursos. 1. Se puede lograr acceso no autorizado a archivos en una unidad no cifrada moviendo el medio a otra computadora. 2. La seguridad impuesta por los sistemas operativos puede ser derrotada infiltrando la capa de arranque antes de que se cargue el sistema operativo. 3. El aislamiento de procesos puede ser subvertido por una verificación de autorización inadecuada en los límites de la API. 4. Las vulnerabilidades en el firmware del sistema pueden proporcionar vectores de ataque que eviten por completo el sistema operativo. 5. El aislamiento inadecuado de la memoria, cómputo y hardware de la máquina virtual puede exponer el sistema anfitrión a ataques desde sistemas invitados. 6. El sistema operativo puede necesitar mitigar la explotación de vulnerabilidades de hardware y firmware, lo que lleva a posibles reducciones de rendimiento (por ejemplo, mitigaciones de Spectre y Meltdown).
- Exposición de funciones del sistema operativo en shells y programación de sistemas. Ver también: Scripting de Shell

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comprender los objetivos y funciones de los sistemas operativos modernos [Explicar]
2. Evaluar los problemas de diseño en diferentes escenarios de uso (por ejemplo, sistema operativo en tiempo real, móvil, servidor) [Evaluar]
3. Comprender las funciones de un sistema operativo contemporáneo con respecto a conveniencia,

eficiencia y capacidad de evolucionar [Explicar]
4. Comprender cómo la evolución y la estabilidad son deseables y mutuamente antagónicas en las funciones del sistema operativo [Explicar]

2.10.2. OS/Principios del Sistema Operativo (CS Core: 2 hrs)

Temas:
Core

- Diseño y enfoques de software del sistema operativo (por ejemplo, monolítico, en capas, modular, micro-kernel, unikernel)
- Abstracciones, procesos y recursos
- Concepto de llamadas al sistema y enlaces a interfaces de programas de aplicación (por ejemplo, Win32, Java, Posix). Ver también: Organización de Máquina a Nivel de Ensamblador
- La evolución del vínculo entre la arquitectura del hardware y las funciones del sistema operativo
- Protección de recursos significa proteger algunas instrucciones/funciones de la máquina. Ver también: Organización de Máquina a Nivel de Ensamblador 1. Las aplicaciones no pueden acceder arbitrariamente a ubicaciones de memoria o direcciones de dispositivos de almacenamiento de archivos. 2. Protección de coprocesadores y dispositivos de red
- Aprovechamiento de interrupciones desde el nivel de hardware: rutinas de servicio e implementaciones. Ver también: Organización de Máquina a Nivel de Ensamblador 1. Interrupciones de temporizador para implementar segmentos de tiempo 2. Interrupciones de E/S para poner hilos bloqueados en espera sin sondeo
- Concepto de estado usuario/sistema y protección, transición al modo kernel usando llamadas al sistema. Ver también: Organización de Máquina a Nivel de Ensamblador
- Mecanismo para invocar llamadas al sistema, el cambio de modo y contexto correspondiente y retorno desde interrupción. Ver también: Organización de Máquina a Nivel de Ensamblador
- Costos de rendimiento de cambios de contexto y vaciados de caché asociados al realizar cambios de proceso en entornos mitigados para Spectre.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comprender cómo la aplicación de enfoques de diseño de software al diseño/implementación de sistemas operativos (por ejemplo, en capas, modular, etc.) afecta la robustez y mantenibilidad de un sistema operativo [Explicar]
2. Categorizar llamadas al sistema por propósito [Categorizar]
3. Comprender la dinámica de invocar una llamada al sistema (por ejemplo, pasar parámetros, cambio de modo) [Explicar]
4. Evaluar si una función puede implementarse en la capa de aplicación o solo puede realizarse mediante llamadas al sistema [Evaluar]
5. Aplicar técnicas de SO para aislamiento, protección y rendimiento a través de las funciones del SO (por ejemplo, similitudes de inanición en planificación de procesos, planificación de solicitudes de disco, semáforos, etc.) y más allá [Aplicar]
6. Comprender cómo la separación en modo kernel y usuario afecta la seguridad y el rendimiento [Explicar]
7. Comprender las ventajas y desventajas de usar procesamiento de interrupciones para habilitar la multiprogramación [Explicar]
8. Analizar vectores potenciales de ataque a través de los sistemas operativos y las características de seguridad diseñadas para proteger contra ellos [Analizar]

2.10.3. OS/Concurrencia (CS Core: 2 hrs, KA Core: 1 hr)

Temas:
Core

- Abstracción de hilos relativa a la concurrencia
- Condiciones de carrera, regiones críticas (rol de interrupciones, si es necesario). Ver también: Programas: Distribución, Programas: Propiedades de Ejecución, Programación GPU, Progra-

mas: Mapeos de Implementación, Programas: Paralelismo Declarativo, Programas: Inicio de Actividades
- Interbloqueos (deadlocks) e inanición. Ver también: Coordinación
- Problemas de multiprocesador (bloqueos de giro (spin-locks), reentrada).
- Concurrencia multiproceso vs multihilo
- Creación de hilos, estados, estructuras. Ver también: Sistemas Digitales y Lógica, Procesamiento Secuencial y Paralelo, Máquinas de Estado
- APIs de hilos
- Interbloqueos e inanición (condiciones necesarias/mitigaciones). Ver también: Coordinación
- Implementar código seguro para hilos (semáforos, bloqueos mutex, variables de condición). Ver también: Rendimiento y Eficiencia Energética, Eventos, Herramientas y Experimentación, Métricas y Benchmarks de Rendimiento, Modelos Analíticos de Rendimiento, Evaluación
- Condiciones de carrera en memoria compartida. Ver también: Coordinación

Non Core
- Gestionar acceso atómico a objetos del SO (por ejemplo, gran bloqueo del kernel vs muchos bloqueos pequeños vs estructuras de datos sin bloqueo como listas)

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Comprender las ventajas y desventajas de la concurrencia como funciones inseparables dentro del marco del sistema operativo [Explicar]
2. Comprender cómo la implementación a nivel de arquitectura resulta en problemas de concurrencia incluyendo condiciones de carrera [Explicar]
3. Comprender problemas de concurrencia en sistemas multiprocesador [Explicar]
4. Comprender el rango de mecanismos que pueden emplearse a nivel del sistema operativo para realizar sistemas concurrentes y describir los beneficios de cada uno [Explicar]
5. Comprender técnicas para lograr sincronización en un sistema operativo (por ejemplo, describir cómo puede implementarse un semáforo usando primitivas del SO) incluyendo control de intra-concurrencia y uso de atómicos de hardware [Explicar]
6. Analizar con precisión código para identificar condiciones de carrera y soluciones apropiadas para abordarlas [Analizar]

2.10.4. OS/Protección y Seguridad (CS Core: 2 hrs, KA Core: 1 hr)

Temas:
Core
- Descripción general de mecanismos de seguridad del sistema operativo. Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad
- Ataques y antagonismo (planificación, etc.). Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad
- Revisión de vulnerabilidades principales en sistemas operativos reales. Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad
- Estrategias de mitigación del sistema operativo como copias de seguridad. Ver también: Confiabilidad del Sistema
- Separación política/mecanismo. Ver también: Gestión de Riesgos y Datos, Gobierno, Gestión y Política de Seguridad
- Métodos y dispositivos de seguridad. Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad 1. Anillos de protección (historia desde Multics hasta x86 virtualizado) 2. Anillos -1 y -2 de x86_64 (hipervisor y ME/PSP)
- Protección, control de acceso y autenticación. Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad, Protocolos, Aplicaciones y Criptografía Avanzada, Algoritmos Criptográficos, Fundamentos de Criptografía

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Comprender el requisito de mecanismos de protección y seguridad en sistemas operativos [Explicar]
2. Listar y describir los vectores de ataque que aprovechan vulnerabilidades del SO [Listar/Enumerar]

3. Comprender los mecanismos disponibles en un SO para controlar el acceso a recursos [Explicar]
4. Resumir las características y limitaciones de un sistema operativo que impactan la protección y seguridad [Resumir]

2.10.5. OS/Planificación (KA Core: 2 hrs)

Temas:
Core

- Planificación con expropiación y sin expropiación
- Planificadores y políticas (por ejemplo, primero en llegar primero en ser servido, trabajo más corto primero, prioridad, round robin, multinivel). Ver también: Gestión de Recursos
- Conceptos de planificación de multiprocesador simétrico (SMP) y coherencia de caché Ver también: Jerarquía de Memoria
- Temporizadores (por ejemplo, construir muchos temporizadores a partir de temporizadores de hardware finitos). Ver también: Organización de Máquina a Nivel de Ensamblador
- Justicia e inanición

Non Core

- Subtemas de sistemas operativos como planificación consciente de energía y planificación en tiempo real. Ver también: Rendimiento y Eficiencia Energética, Características y Arquitecturas Embebidas, Tiempo Real, Memoria y Verificación, Sistemas, Programación y Sensores, Plataformas Móviles: Capacidades y Computación Asíncrona, Plataformas Móviles: Arquitectura y Restricciones, Plataformas Móviles: Fundamentos del Desarrollo
- Planificación cooperativa, como futexes de Linux y planificación en espacio de usuario.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comparar y contrastar los algoritmos comunes utilizados para planificación con y sin expropiación de tareas en sistemas operativos, como prioridad, comparación de rendimiento y esquemas de participación justa [Comparar]
2. Explicar las relaciones entre algoritmos de planificación y dominios de aplicación [Explicar]
3. Explicar las distinciones entre tipos de planificadores de procesador como corto plazo, mediano plazo, largo plazo y E/S [Explicar]
4. Evaluar un problema o solución para determinar la idoneidad para multiprocesamiento asimétrico y/o simétrico [Evaluar]
5. Evaluar un problema o solución para determinar la idoneidad como proceso vs hilos [Evaluar]
6. Listar algunos contextos que se benefician de la planificación con expropiación y con plazo límite [Listar/Enumerar]

NonCore:

7. Explicar las formas en que la lógica incorporada en los algoritmos de planificación es aplicable a otros mecanismos de sistemas operativos, como primero en llegar primero en ser servido o prioridad para E/S de disco, planificación de red, planificación de proyectos y problemas más allá de la computación [Explicar]

2.10.6. OS/Modelo de Proceso (KA Core: 2 hrs)

Temas:
Core

- Procesos e hilos relativos a virtualización de memoria protegida, estado del proceso, aislamiento de memoria, etc.
- Huella de memoria/segmentación (por ejemplo, pila, montón, etc.). Ver también: Organización de Máquina a Nivel de Ensamblador
- Creación y carga de ejecutables, bibliotecas compartidas y enlace dinámico. Ver también: Traducción y Ejecución de Lenguajes
- Despacho y cambio de contexto. Ver también: Organización de Máquina a Nivel de Ensamblador
- Comunicación entre procesos (por ejemplo, memoria compartida, paso de mensajes, señales, variables de entorno). Ver también: Comunicación

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comprender cómo los procesos e hilos usan características de concurrencia para virtualizar el control [Explicar]
2. Comprender las razones para usar interrupciones, despacho y cambio de contexto para apoyar concurrencia y virtualización en un sistema operativo [Explicar]
3. Comprender los diferentes estados por los que puede pasar una tarea, y las estructuras de datos necesarias para apoyar la gestión de muchas tareas [Explicar]
4. Comprender las diferentes formas de asignar memoria a tareas, citando los méritos relativos de cada una [Explicar]
5. Aplicar el mecanismo de comunicación entre procesos apropiado para un propósito específico en un artefacto de software programado [Aplicar]

2.10.7. OS/Gestión de Memoria (KA Core: 1 hr)

Temas:
Core

- Revisión de memoria física, traducción de direcciones y hardware de gestión de memoria. Ver también: Jerarquía de Memoria, Matemáticas Discretas
- Impacto de la jerarquía de memoria incluyendo concepto de caché, búsqueda en caché y almacenamiento en caché por CPU en mecanismos y políticas del sistema operativo Ver también: Jerarquía de Memoria, Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento
- Direccionamiento lógico y físico, virtualización de espacio de direcciones. Ver también: Jerarquía de Memoria, Matemáticas Discretas
- Conceptos de paginación, reemplazo de páginas, trashing y asignación de páginas y marcos
- Técnicas de asignación/desasignación/almacenamiento (algoritmos y estructura de datos) rendimiento y flexibilidad 1. Arenas, asignadores de losas (slab allocators), listas libres, clases de tamaño, páginas de tamaño heterogéneo (páginas enormes)
- Almacenamiento en caché de memoria y coherencia de caché y el efecto de vaciar la caché para evitar vulnerabilidades de ejecución especulativaVer también: Organización Funcional, Jerarquía de Memoria, Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento)
- Mecanismos y conceptos de seguridad en gestión de memoria incluyendo sandboxing, protección, aislamiento y vectores relevantes de ataque. Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad

Non Core

- Memoria virtual: aprovechando el hardware de memoria virtual para servicios del SO y eficiencia

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar jerarquía de memoria y compensaciones costo-rendimiento [Explicar]
2. Resumir los principios de memoria virtual aplicados a almacenamiento en caché y paginación [Resumir]
3. Evaluar las compensaciones en términos de tamaño de memoria (memoria principal, memoria caché, memoria auxiliar) y velocidad del procesador [Evaluar]
4. Describir la razón y el uso de la memoria caché (rendimiento y proximidad, cómo las cachés complican el aislamiento y la abstracción de máquina virtual) [Describir]
5. Codificar/Desarrollar programas eficientes que consideren los efectos del reemplazo de páginas y asignación de marcos en el rendimiento de un proceso y el sistema en el que se ejecuta [Crear]

NonCore:

6. Explicar cómo se utiliza el hardware para una virtualización eficiente [Explicar]

2.10.8. OS/Gestión de Dispositivos (KA Core: 1 hr)

Temas:
Core

- Estrategias de almacenamiento en búfer. Ver también: Interconexión y Comunicación

- Acceso Directo a Memoria (DMA) y E/S sondeada, E/S mapeada en memoria. Ver también: Interconexión y Comunicación 1. Protocolos de comunicación DMA (por ejemplo, búfers en anillo, etc.)
- Histórico y contextual - Gestión de dispositivos de almacenamiento persistente. Por ejemplo: Magnético, Dispositivo de Estado Sólido (SSD). Ver también: Fundamentos e Historia Temprana de la Informática, Historia Moderna: PC, Internet e IA

Non Core

- Abstracciones de interfaz de dispositivo, capa de abstracción de hardware
- Propósito del controlador de dispositivo, abstracción, implementación y desafíos de prueba
- Tolerancia a fallos de alto nivel en comunicación de dispositivos

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar la implementación de control de dispositivos a nivel de arquitectura y vincular mecanismos y políticas relevantes del sistema operativo (por ejemplo, estrategias de almacenamiento en búfer, acceso directo a memoria) [Explicar]
2. Explicar las capas de gestión de dispositivos del SO y la arquitectura (por ejemplo, controlador de dispositivo, controlador de dispositivo, abstracción de dispositivo) [Explicar]
3. Explicar la relación entre el hardware físico y los dispositivos virtuales mantenidos por el sistema operativo [Explicar]
4. Explicar el almacenamiento en búfer de datos de E/S y describir estrategias para implementarlo [Explicar]
5. Describir las ventajas y desventajas del acceso directo a memoria y discutir las circunstancias en las que su uso está justificado [Describir]

NonCore:

6. Describir la complejidad y mejores prácticas para la creación de controladores de dispositivos [Describir]

2.10.9. OS/API de Sistemas de Archivos e Implementación (KA Core: 2 hrs)

Temas:

Core

- Concepto de archivo incluyendo datos, metadatos, operaciones y modo de acceso
- Montaje de sistema de archivos
- Control de acceso a archivos
- Compartición de archivos
- Métodos básicos de asignación de archivos, incluida la tabla de asignación enlazada
- Estructuras del sistema de archivos que comprenden asignación de archivos incluyendo varias estructuras de directorio y métodos para identificar archivos de manera única (por ejemplo, nombre, identificador o ubicación de almacenamiento de metadatos)
- Técnicas de asignación/desasignación/almacenamiento (algoritmos y estructura de datos) impacto en rendimiento y flexibilidad (es decir, fragmentación interna y externa y compactación)
- Gestión de espacio libre como usar tablas de bits vs enlazado
- Implementación de directorios para segmentar y rastrear ubicación de archivos

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar las elecciones que deben tomarse en el diseño de sistemas de archivos [Explicar]
2. Evaluar diferentes enfoques para organización de archivos, reconociendo las fortalezas y debilidades de cada uno [Evaluar]
3. Aplicar constructores de software apropiadamente dado el conocimiento de la implementación del sistema de archivos [Aplicar]

2.10.10. OS/Sistemas de Archivos Avanzados (KA Core: 1 hr)

Temas:

Core

- Sistemas de archivos: particionamiento, montar/desmontar, sistemas de archivos virtuales
- Técnicas de implementación en profundidad
- Archivos mapeados en memoria. Ver también: Interconexión y Comunicación
- Sistemas de archivos de propósito especial
- Denominación, búsqueda, acceso, copias de seguridad
- Sistemas de archivos con registro (journaling) y estructurados en registro (log-structured). Ver también: Confiabilidad del Sistema

Non Core

- Sistemas de archivos distribuidos
- Sistemas de archivos cifrados
- Tolerancia a fallos

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar cómo los desarrollos de hardware han llevado a cambios en las prioridades para el diseño y la gestión de sistemas de archivos [Explicar]
2. Mapear abstracciones de archivo a una lista de dispositivos e interfaces relevantes [Mapear/Mapa conceptual]
3. Identificar y categorizar diferentes tipos de montaje [Analizar]
4. Explicar requisitos específicos de sistemas de archivos y las características de sistemas de archivos especializados que los satisfacen [Explicar]
5. Explicar el uso de registro y cómo los sistemas de archivos estructurados en registro mejoran la tolerancia a fallos [Explicar]

NonCore:

6. Explicar propósito y complejidad de los sistemas de archivos distribuidos [Explicar]
7. Listar ejemplos de protocolos de sistemas de archivos distribuidos [Listar/Enumerar]
8. Explicar mecanismos en sistemas de archivos para mejorar la tolerancia a fallos [Explicar]

2.10.11. OS/Virtualización (KA Core: 1 hr)

Temas:

Core

- Usar virtualización y aislamiento para lograr protección y rendimiento predecible. Ver también: Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento
- Paginación avanzada y memoria virtual. Ver también: Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento
- Sistemas de archivos virtuales y dispositivos virtuales.
- Contenedores y su comparación con máquinas virtuales.
- Trashing (por ejemplo, requisitos de Popek y Goldberg para sistemas recursivamente virtualizables).

Non Core

- Tipos de virtualización (incluyendo hardware/software, SO, servidor, servicio, red). Ver también: Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento
- Virtualización portátil; emulación vs aislamiento. Ver también: Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento
- Costo de virtualización. Ver también: Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento
- Máquinas virtuales y escapes de contenedores, peligros desde una perspectiva de seguridad. Ver también: Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad
- Hipervisores - extensiones de máquina virtual de hardware, anfitriones con soporte de kernel, QEMU KVM

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar cómo la arquitectura de hardware proporciona soporte y eficiencias para la virtualización [Explicar]
2. Explicar la diferencia entre emulación y aislamiento [Explicar]

3. Evaluar las compensaciones de la virtualización [Evaluar]

NonCore:

4. Explicar hipervisores y la necesidad de ellos junto con diferentes tipos de hipervisores [Explicar]

2.10.12. OS/Sistemas en Tiempo Real y Embebidos (KA Core: 1 hr)

Temas:

Core

- Planificación de procesos y tareas.
- Plazos límite y problemas de tiempo real. Ver también: Características y Arquitecturas Embebidas, Tiempo Real, Memoria y Verificación, Sistemas, Programación y Sensores
- Baja latencia vs "tiempo real suave" vs "tiempo real duro". Ver también: Características y Arquitecturas Embebidas, Tiempo Real, Memoria y Verificación, Sistemas, Programación y Sensores, Programación Dirigida por Eventos y Reactiva

Non Core

- Requisitos de gestión de memoria/disco en un entorno de tiempo real.
- Fallas, riesgos y recuperación.
- Preocupaciones especiales en sistemas de tiempo real (seguridad).

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar qué hace que un sistema sea un sistema de tiempo real [Explicar]
2. Explicar la latencia y sus fuentes en sistemas de software y sus características [Explicar]
3. Explicar las preocupaciones especiales que presentan los sistemas de tiempo real, incluido el riesgo, y cómo se abordan estas preocupaciones [Explicar]

NonCore:

4. Explicar características y mecanismos específicos de sistemas operativos de tiempo real [Explicar]

2.10.13. OS/Tolerancia a fallos (KA Core: 1 hr)

Temas:

Core

- Sistemas confiables y disponibles. Ver también: Confiabilidad del Sistema
- Enfoques de software y hardware para abordar la tolerancia (RAID). Ver también: Confiabilidad del Sistema

Non Core

- Redundancia espacial y temporal. Ver también: Confiabilidad del Sistema
- Métodos utilizados para implementar tolerancia a fallos. Ver también: Confiabilidad del Sistema
- Mecanismos de identificación y corrección de errores, sumas de comprobación (checksums) de memoria volátil en RAM. Ver también: Jerarquía de Memoria
- Verificación de consistencia y recuperación del sistema de archivos.
- Sistemas de archivos con registro y estructurados en registro. Ver también: Confiabilidad del Sistema
- Casos de uso para tolerancia a fallos (bases de datos, críticos para la seguridad). Ver también: Confiabilidad del Sistema
- Ejemplos de mecanismos del SO para detección, recuperación, reinicio para implementar tolerancia a fallos, uso de estas técnicas para los servicios propios del SO. Ver también: Confiabilidad del Sistema

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar cómo los sistemas operativos pueden facilitar la tolerancia a fallos, confiabilidad y disponibilidad [Explicar]
2. Explicar el rango de métodos para implementar tolerancia a fallos en un sistema operativo [Explicar]

3. Explicar cómo un sistema operativo puede continuar funcionando después de que ocurra una falla [Explicar]
4. Explicar las compensaciones de rendimiento y flexibilidad que impactan el uso de tolerancia a fallos [Explicar]

NonCore:

5. Describir en detalle los problemas y mecanismos de tolerancia a fallos del sistema operativo [Describir]

2.10.14. OS/Sociedad, ética y Profesión

Temas:

Core

- Código abierto en sistemas operativos. Ver también: Open Source, Piratería y DRM, Derechos de PI Digital y Derechos de Autor, Fundamentos de la Propiedad Intelectual, Licencias, Plagio y Autoría 1. Identificación de vulnerabilidades en kernels de código abierto, 2. Sistemas operativos invitados de código abierto, 3. Sistemas operativos anfitriones de código abierto, y 4. Cambios en monetización (actualizaciones pagas vs gratuitas).
- Problemas de fin de vida con la retirada de sistemas operativos. Concepto ejemplo: Implicaciones de privacidad del uso de sistemas operativos/entornos operativos propietarios, incluyendo telemetría, escaneo automatizado de datos personales, publicidad incorporada e integración automática en la nube.

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar ventajas y desventajas de encontrar y abordar errores en kernels de código abierto [Explicar]
2. Contextualizar la historia e impacto positivo y negativo de Linux como producto de código abierto [Contextualizar]
3. Listar complicaciones con la dependencia de sistemas operativos pasados de fin de vida [Listar/Enumerar]
4. Comprender diferencias en encontrar y abordar errores para varios modelos de pago de sistemas operativos [Explicar]

2.11. Computación Paralela y Distribuida (PDC)

La programación paralela y distribuida organiza, coordina y controla múltiples cálculos que ocurren al mismo tiempo en diferentes lugares. La ubicuidad del paralelismo y la distribución son consecuencias inevitables del creciente número de puertas en los procesadores, procesadores en las computadoras y computadoras en todas partes que pueden usarse para mejorar el rendimiento en comparación con los programas secuenciales, al mismo tiempo que se hace frente a la interconectividad intrínseca del mundo y la posibilidad de que algunos componentes o conexiones fallen o se comporten de manera maliciosa. La programación paralela y distribuida elimina las restricciones de la programación secuencial que requieren que los pasos computacionales ocurran en un orden serial en un solo lugar, revelando distinciones, técnicas y análisis adicionales que se aplican en cada capa de los sistemas informáticos.

En la mayoría de los usos convencionales, la programación "paralela" se centra en establecer y coordinar múltiples actividades que pueden ocurrir al mismo tiempo, la programación "distribuida" se centra en establecer y coordinar actividades que pueden ocurrir en diferentes lugares, y la programación "concurrente" se centra en las interacciones de las actividades en curso entre sí y con el entorno. Sin embargo, los tres términos pueden aplicarse en la mayoría de los contextos. El paralelismo generalmente implica alguna forma de distribución porque múltiples actividades que ocurren sin restricciones de orden secuencial suceden en múltiples lugares físicos (a menos que dependan de cambios de contexto o efectos cuánticos). Por el contrario, las acciones en diferentes lugares no necesitan tener un orden secuencial específico entre sí en ausencia de restricciones de comunicación.

Las técnicas de programación paralela, distribuida y concurrente forman el núcleo de la Computación de Alto Rendimiento (HPC), los sistemas distribuidos y, cada vez más, casi todas las aplicaciones informáticas. El área de conocimiento PDC ha evolucionado desde un conjunto diverso de temas

avanzados hasta un cuerpo central de conocimiento y práctica, que impregna casi todos los demás aspectos de la informática. El crecimiento del campo ha ocurrido de manera irregular en los diferentes subcampos de la informática, a veces con diferentes objetivos, terminología y prácticas, ocultando la superposición considerable de ideas y habilidades básicas que son el foco principal de esta área de conocimiento. Casi todos los problemas con una solución secuencial también admiten soluciones paralelas y/o distribuidas; surgen problemas y soluciones adicionales solo en el contexto de la concurrencia. Casi todos los dominios de aplicación de la computación paralela y distribuida son un área de estudio y/o ingeniería bien desarrollada, demasiado grande para enumerarla.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 2.11.1 Programas: Paralelismo Declarativo (Pág. 85)	1	1
⭐ 2.11.2 Programas: Inicio de Actividades (Pág. 85)	1	1
⭐ 2.11.3 Programas: Propiedades de Ejecución (Pág. 86)	1	1
⭐ 2.11.4 Programas: Distribución (Pág. 86)	1	1
⭐ 2.11.5 Programas: Mapeos de Implementación (Pág. 86)	1	1
2.11.6 Programación GPU (Pág. 87)	Electivo	
⭐ 2.11.7 Comunicación (Pág. 87)	2	4
⭐ 2.11.8 Comunicación: Propiedades y Extensiones (Pág. 88)	1	1
⭐ 2.11.9 Memoria y Consistencia (Pág. 88)	1	1
⭐ 2.11.10 Coordinación (Pág. 89)	1	3
⭐ 2.11.11 Coordinación: Sincronización y Atomicidad (Pág. 89)	1	2
⭐ 2.11.12 Coordinación: Propiedades Avanzadas (Pág. 90)	1	2
⭐ 2.11.13 Evaluación (Pág. 90)	1	3
⭐ 2.11.14 Evaluación: Métodos Formales (Pág. 91)	1	1
⭐ 2.11.15 Evaluación: Pruebas y Medición (Pág. 91)	1	1
⭐ 2.11.16 Algoritmos (Pág. 92)	1	3
⭐ 2.11.17 Algoritmos: Dominios de Aplicación (Pág. 92)	1	3
⭐ 2.11.18 Algoritmos: Dominios Algorítmicos (Pág. 93)	1	3

Cuadro 2.11: Lista de KUs del área de Computación Paralela y Distribuida.

2.11.1. PDC/Programas: Paralelismo Declarativo (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Paralelismo 1. Paralelismo declarativo: Determinar qué acciones pueden, o no deben, realizarse en paralelo, a nivel de instrucciones, funciones, clausuras, acciones compuestas, sesiones, tareas y servicios es la idea principal que subyace a los algoritmos PDC; no hacerlo es la principal fuente de errores. Ver también: Algoritmos 2. Definir orden: por ejemplo, usando relaciones de "sucede antes" o grafos acíclicos dirigidos serie/paralelo que representan programas. 3. Independencia: determinar cuándo el orden no importa, en términos de conmutatividad, dependencias, precondiciones. 4. Garantizar el orden entre acciones que de otro modo serían paralelas cuando sea necesario, incluyendo bloqueos, publicación segura; e imponer comunicación - enviar un mensaje sucede antes de recibirlo; y relajarlo cuando no sea necesario.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Mostrar gráficamente (como un Grafo Acíclico Dirigido - *Directed Acyclic Graph* (DAG)) cómo paralelizar una expresión numérica compuesta; por ejemplo, $a = (b + c) * (d + e)$ [Diseñar]
2. Explicar por qué los conceptos de consistencia y tolerancia a fallos no surgen en programas puramente secuenciales [Explicar]

2.11.2. PDC/Programas: Inicio de Actividades (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Iniciar actividades 1. Las opciones que permiten que las acciones se realicen (eventualmente) en lugares van desde el cableado directo hasta scripts de configuración; también establecer comunicación y gestión de recursos; estos se expresan de manera diferente en los lenguajes y contextos, generalmente confiando en el aprovisionamiento y gestión automatizados por las plataformas. Ver también: Gestión de Recursos 2. Procedimental: Permitir que múltiples acciones comiencen en un punto de programa dado; por ejemplo, iniciar nuevos hilos, posiblemente delimitando su alcance u organizándolos en grupos jerárquicos. 3. Reactiva: Habilitar al ocurrir un evento instalando un manejador de eventos, con menos control de cuándo comienzan o terminan las acciones, y puede aplicarse incluso en monoprocesadores. 4. Dependiente: Habilitar al completarse otras; por ejemplo, secuenciando conjuntos de acciones paralelas. Ver también: Coordinación. 5. Granularidad: El costo de ejecución de los cuerpos de las acciones debe superar la sobrecarga de organizarlos.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Escribir un servicio que cree un hilo (u otra forma de activación procedimental) para devolver una página web solicitada a cada nuevo cliente [Escribir]

2.11.3. PDC/Programas: Propiedades de Ejecución (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Propiedades de Ejecución 1. Ejecución no determinista de acciones sin orden. 2. Consistencia: Garantizar el acuerdo entre partes sobre valores y predicados cuando sea necesario para evitar carreras, mantener seguridad y atomicidad, o llegar a consenso. 3. Tolerancia a fallos: Manejar fallos en las partes o la comunicación, incluyendo (Bizantino) mal comportamiento debido a partes y protocolos no confiables, cuando sea necesario para mantener el progreso o la disponibilidad. Ver también: Confiabilidad del Sistema. 4. Los compromisos son un foco de evaluación. Ver también: Evaluación.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Escribir una función que cuente eventos, como recepciones de paquetes de red, de manera eficiente [Escribir]

2.11.4. PDC/Programas: Distribución (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Distribución 1. Definir lugares, como dispositivos que ejecutan acciones, incluyendo componentes de hardware, hosts remotos; también pueden incluir dispositivos, hosts y usuarios externos no controlados. Ver también: Interconexión y Comunicación 2. Un dispositivo puede dividir el tiempo o emular múltiples acciones paralelas mediante menos procesadores mediante planificación y virtualización. Ver también: Planificación 3. Nombrar o identificar lugares (por ejemplo, IDs de dispositivo) y acciones como partes (por ejemplo, IDs de hilo). 4. Las actividades a través de lugares pueden comunicarse a través de medios. Ver también: Comunicación

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Escribir un programa de filtro/mapeo/reducción en múltiples estilos [Escribir]

2.11.5. PDC/Programas: Mapeos de Implementación (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Una o más de las siguientes asignaciones y mecanismos a través de sistemas en capas: 1. Paralelismo a nivel de instrucción y datos en la CPU. Ver también: Organización Funcional. 2. SIMD

y paralelismo de datos heterogéneo. Ver también: Arquitecturas Heterogéneas. 3. Concurrencia planificada en multinúcleo, tareas, actores. Ver también: Planificación. 4. Clústeres, nubes; aprovisionamiento elástico. Ver también: Aspectos Comunes: Plataformas, APIs y Restricciones, Aspectos Comunes: Lenguajes de Plataforma y Patrones Web. 5. Sistemas distribuidos en red. Ver también: Aplicaciones en Red. 6. Tecnologías emergentes como la computación cuántica y la computación molecular.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar las compensaciones entre diferentes estrategias de mapeo en términos de rendimiento, costo y complejidad de implementación [Explicar]

2.11.6. PDC/Programación GPU

Temas:
Non Core

- Computación de Propósito General en GPU (GPGPU) 1. Arquitectura de GPU: multiprocesadores de flujo (SMs), núcleos CUDA y modelo de ejecución por warp. 2. Modelo de ejecución SIMT (Single Instruction, Multiple Threads) y sus diferencias respecto al SIMD de CPU. 3. Interacción host-dispositivo: transferencia de datos entre memoria de CPU (host) y GPU (dispositivo) vía PCIe. 4. Casos de uso: computación científica, aprendizaje automático, procesamiento de imágenes y simulación a gran escala.
- Modelo de Programación CUDA 1. Jerarquía de ejecución CUDA: grids, bloques de hilos e hilos individuales; mapeo al hardware GPU. 2. Definición y lanzamiento de kernels; parámetros de configuración (dimensiones de grid y bloque). 3. Indexación de hilos: threadIdx, blockIdx, blockDim, gridDim; cálculo de índices globales en 1D, 2D y 3D. 4. Divergencia de warp: implicaciones de las bifurcaciones dentro de un warp en el rendimiento. 5. Primitiva de sincronización intra-bloque: __syncthreads().
- Jerarquía de Memoria CUDA 1. Memoria global: grande, alta latencia, accesible por todos los hilos; patrones de acceso coalescente para el rendimiento. 2. Memoria compartida: memoria en chip de baja latencia compartida dentro de un bloque de hilos; conflictos de banco. 3. Registros y memoria local: almacenamiento privado por hilo. 4. Memoria constante y de textura: cachés de solo lectura optimizadas para patrones de acceso específicos. 5. Memoria unificada: modelo de programación simplificado con migración automática de datos entre host y dispositivo.

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Escribir un kernel CUDA que realice una operación data-paralela (p. ej., suma de vectores, multiplicación de matrices) y configurar correctamente las dimensiones de grid y bloque [Escribir]
2. Analizar el impacto de los patrones de acceso a memoria global (acceso coalescente vs. acceso entrelazado) y el uso de memoria compartida en el rendimiento de un kernel GPU [Analizar]
3. Diseñar un algoritmo paralelo usando el modelo de programación CUDA, incluyendo la jerarquía de hilos, estrategia de asignación de memoria y transferencias de datos entre host y dispositivo [Diseñar]

2.11.7. PDC/Comunicación (CS Core: 2 hrs, KA Core: 4 hrs)

Temas:
Core

- Medios 1. Variedades: canales (paso de mensajes o E/S), memoria compartida, heterogéneos, almacenes de datos. 2. Dependencia de la disponibilidad y naturaleza del hardware subyacente, conectividad y protocolos; soporte del lenguaje, emulación. Ver también: Interconexión y Comunicación.
- Canales 1. Medios de comunicación de parte a parte explícitos (generalmente nombrados). 2. APIs: Sockets, constructos arquitectónicos, basados en lenguaje y de kits de herramientas, como Message Passing Interface (MPI), y constructos en capas como Llamada a Procedimiento Remoto (RPC). Ver también: Fundamentos de Redes y Comunicaciones. 3. APIs de canales de E/S.

- Memoria 1. Arquitecturas de memoria compartida en las que las partes se comunican directamente solo con la memoria en direcciones dadas, con extensiones a memoria heterogénea que admite múltiples almacenes de memoria con transferencia de datos explícita entre ellos; por ejemplo, memoria local y compartida de GPU, Acceso Directo a Memoria (DMA). 2. Jerarquías de memoria: Múltiples capas de dominios, alcances y cachés compartidos; localidad: latencia, falso uso compartido. 3. Propiedades de consistencia: Límites de atomicidad a nivel de bits, coherencia, orden local.
- Almacenes de Datos 1. Estructuras de datos mantenidas cooperativamente que implementan mapas y TADs relacionados. 2. Variedades: Propiedad exclusiva, compartida, fragmentada, replicada, inmutable, versionada.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar las similitudes y diferencias entre: (1) La parte A envía un mensaje en el canal X con contenido 1 recibido por la parte B (2) A establece la variable compartida X a 1, leída por B (3) A establece "X=1" en un mapa compartido distribuido al que accede B [Explicar]
2. Escribir un programa que distribuya diferentes segmentos de un conjunto de datos a múltiples trabajadores y recoja los resultados (por ejemplo, sumar segmentos de un arreglo) [Escribir]
3. Escribir un programa paralelo que solicite datos de múltiples sitios y los resuma usando alguna forma de reducción [Escribir]
4. Comparar el rendimiento de versiones con y sin búfer de un programa productor-consumidor [Comparar]

2.11.8. PDC/Comunicación: Propiedades y Extensiones (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Una o más de las siguientes propiedades y extensiones: 1. Topologías: Unicast, Multicast, Buzones, Conmutadores; Enrutamiento a través de redes de interconexión de hardware y software. 2. Propiedades de concurrencia de medios: Orden, consistencia, idempotencia, superposición de comunicación con computación. 3. Rendimiento del medio: Latencia, ancho de banda (rendimiento), contención (congestión), capacidad de respuesta (vivacidad), confiabilidad (tasas de error y pérdida), progreso basado en protocolo (acks, tiempos de espera, mediación). 4. Propiedades de seguridad del medio: integridad, privacidad, autenticación, autorización. Ver también: Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada. 5. Formatos de datos: Serialización, validación, cifrado, compresión. 6. Políticas de canal: Puntos finales, sesiones, almacenamiento en búfer, respuesta a saturación (esperar vs. descartar), control de velocidad. 7. Multiplexación y demultiplexación de muchos dispositivos o partes de E/S relativamente lentos; técnicas basadas en finalización y basadas en planificador; APIs async-await, select y polling. 8. Formalización y análisis de comunicación por canales; por ejemplo, CSP. 9. Aplicaciones de la teoría de colas para modelar y predecir el rendimiento.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Determinar si un esquema de comunicación dado proporciona propiedades de seguridad suficientes para un uso dado [Determinar]
2. Dar un ejemplo de un escenario en el que los envíos de mensajes bloqueantes pueden causar un punto muerto (deadlock) [Crear]
3. Describir al menos una técnica de diseño para evitar fallos de vivacidad en programas que usan múltiples bloqueos [Describir]

2.11.9. PDC/Memoria y Consistencia (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Modelos de memoria, consistencia de datos y tolerancia a fallos: 1. Modelos de memoria: consistencia secuencial y de liberación/adquisición. 2. Gestión de memoria; incluyendo la reclamación de datos compartidos; conteo de referencias y alternativas. 3. Colocación y transferencia masiva de datos; reducción del tráfico de mensajes y mejora de la localidad; superposición de transferencia de datos y computación; impacto del diseño de datos como array de estructuras vs. estructura de arrays. 4. Emular memoria compartida: memoria compartida distribuida, Acceso Directo a Memoria Remota (RDMA). 5. Consistencia de almacenes de datos: Atomicidad, linealizabilidad, transaccionalidad, coherencia, orden causal, resolución de conflictos, consistencia eventual, cadenas de bloques. 6. Fallos, particionamiento y fallos parciales; votación; protocolos como Paxos y Raft. 7. Compromisos de diseño entre consistencia, disponibilidad, tolerancia a partición (fallos); imposibilidad de cumplir con todos a la vez. 8. Seguridad y confianza: Fallos bizantinos, prueba de trabajo y alternativas.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Dar un ejemplo de un orden de accesos entre actividades concurrentes (por ejemplo, un programa con una carrera de datos) que no sea secuencialmente consistente [Crear]
2. Escribir un programa que ilustre el reordenamiento de acceso a memoria o de mensajes [Escribir]
3. Describir los méritos relativos del control de concurrencia optimista versus conservador bajo diferentes tasas de contención entre actualizaciones [Describir]
4. Dar un ejemplo de un escenario en el que un intento de actualización optimista podría nunca completarse [Crear]
5. Modificar un sistema concurrente para usar un almacén de datos más escalable, confiable o disponible [Crear]
6. Usando una plataforma existente que admita almacenes de datos replicados, escribir un programa que mantenga un mapeo clave-valor incluso cuando uno o más hosts fallen [Escribir]

2.11.10. PDC/Coordinación (CS Core: 1 hr, KA Core: 3 hrs)

Temas:
Core

- Dependencias 1. La iniciación o progreso de una actividad puede depender de otras actividades, para evitar condiciones de carrera, garantizar la terminación o cumplir otros requisitos. 2. Garantizar el progreso evitando ciclos de dependencia, usando condiciones monótonas, eliminando dependencias no esenciales.
- Constructos de control y patrones de diseño: 1. Basados en finalización: Barreras, reuniones (joins), incluido el control de terminación. 2. Habilitados por datos: Colas, diseños productor-consumidor. 3. Basados en condición: Polling, reintentos, retrocesos, ayuda, suspensión, señalización, tiempos de espera. 4. Reactivos: Habilitar y activar continuaciones.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Mostrar cómo garantizar que un programa termine correctamente cuando todas las tareas concurrentes de un conjunto hayan finalizado [Diseñar]
2. Escribir una función que cuente eventos, como entradas de sensores o recepciones de paquetes de red, de manera eficiente [Escribir]
3. Escribir un programa de filtro/mapeo/reducción en múltiples estilos [Escribir]
4. Escribir un programa en el que la terminación de un conjunto de acciones paralelas sea seguida por otra [Escribir]
5. Escribir un servicio que cree un hilo (u otra forma de activación procedimental) para devolver una página web solicitada a cada nuevo cliente [Escribir]

2.11.11. PDC/Coordinación: Sincronización y Atomicidad (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Atomicidad 1. Instrucciones atómicas, ordenamientos de acceso local impuestos. 2. Bloqueos y exclusión mutua; granularidad de bloqueo. 3. Uso de bloqueos en un lenguaje específico; mantener la vivacidad sin introducir carreras. 4. Evitar punto muerto (*deadlock*): Ordenamiento, mayor granularidad, reintentos aleatorios; retrocesos, encapsulación mediante administradores de bloqueos. 5. Errores comunes: No bloquear o desbloquear cuando es necesario, mantener bloqueos mientras se invocan operaciones desconocidas. 6. Evitar bloqueos: replicación, solo lectura, propiedad, y construcciones no bloqueantes.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Mostrar cómo evitar o reparar un error de carrera en un programa dado [Analizar]

2.11.12. PDC/Coordinación: Propiedades Avanzadas (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Una o más de las siguientes propiedades y extensiones: 1. Propiedades de progreso incluyendo libre de bloqueo (*lock-free*), libre de espera (*wait-free*), equidad (*fairness*), planificación por prioridad, interacciones con consistencia, confiabilidad. 2. Rendimiento con respecto a contención, granularidad, convoy, escalabilidad. 3. Estructuras de datos y algoritmos no bloqueantes. 4. Propiedad y control de recursos. 5. Variantes y alternativas de bloqueos: bloqueos de secuencia, bloqueos de lectura-escritura; Actualizar-Copiando-Lectura (RCU), reentrada; tickets; control del ciclo activo (*spinning*) versus bloqueo. 6. Control basado en transacciones: Optimista y conservador. 7. Bloqueo distribuido: confiabilidad. 8. Alternativas a barreras: Relojes; contadores, relojes virtuales; flujo de datos y continuaciones; futuros y RPC; basadas en consenso, recolección de resultados con reductores y colectores. 9. Especulación, selección, cancelación; consecuencias en observabilidad y seguridad. 10. Control de recursos usando semáforos y variables de condición. 11. Flujo de control: Planificación de computaciones, bucles serie-paralelo con líderes (posiblemente elegidos), tuberías y flujos, paralelismo anidado. 12. Excepciones y fallos. Manejadores, detección, tiempos de espera, tolerancia a fallos, votación.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Escribir un programa que busque especulativamente una solución mediante múltiples actividades, terminando las demás cuando se encuentre una [Escribir]
2. Escribir un programa en el que una excepción numérica (como división por cero) en una actividad cause la terminación de las demás [Escribir]
3. Escribir un programa para que múltiples partes acuerden la hora actual del día; discutir sus limitaciones en comparación con protocolos como el protocolo de transferencia de red (NTP) [Escribir]

2.11.13. PDC/Evaluación (CS Core: 1 hr, KA Core: 3 hrs)

Temas:
Core

- Requisitos de seguridad (safety) y vivacidad (*liveness*) en términos de constructos de lógica temporal para expresar ”siempre” y ”eventualmente” Ver también: Computación Paralela y Distribuida.
- Identificar, probar y reparar violaciones, incluyendo formas comunes de errores como no garantizar el orden necesario (errores de carrera), atomicidad (incluyendo errores de ”verificar luego actuar”) y terminación (bloqueo activo).
- Métricas de requisitos de rendimiento para rendimiento (*throughput*), capacidad de respuesta, latencia, disponibilidad, consumo de energía, escalabilidad, uso de recursos, costos de comunicación, espera y control de velocidad, equidad; acuerdos de nivel de servicio. Ver también: Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento.
- Impacto en el rendimiento de las opciones de diseño e implementación, incluyendo granularidad, sobrecarga, costos de consenso y consumo de energía. Ver también: Diseño Sostenible y Compu-

tación Ubicua, Huella Ambiental de los Sistemas Informáticos, Efectos Sistémicos y Contexto Social.
- Estimar limitaciones de escalabilidad, por ejemplo usando la Ley de Amdahl o la Ley de Escalabilidad Universal. Ver también: Eventos, Herramientas y Experimentación, Métricas y Benchmarks de Rendimiento, Modelos Analíticos de Rendimiento.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Revisar una especificación para habilitar el paralelismo y la distribución sin violar otras propiedades o características esenciales [Rediseñar]
2. Explicar cómo las nociones concurrentes de seguridad (safety) y vivacidad (liveness) extienden sus contrapartes secuenciales [Explicar]
3. Especificar un conjunto de invariantes que deben mantenerse en cada paso de cómputo de paralelismo masivo [Analizar]
4. Escribir un programa de prueba que pueda revelar un error de carrera de datos; por ejemplo, perder una actualización cuando dos actividades intentan incrementar una variable [Escribir]
5. En un contexto dado, explicar hasta qué punto se esperaría que introducir paralelismo en un programa por lo demás secuencial mejoraría el rendimiento (*throughput*) y/o reduciría la latencia, y cómo podría afectar la eficiencia energética [Explicar]
6. Mostrar cómo cambian la escalabilidad y la eficiencia para problemas de muestra con y sin el supuesto de que el tamaño del problema cambia con el número de procesadores; además, explicar si y cómo cambiaría la escalabilidad bajo relajaciones de dependencias secuenciales [Diseñar]

2.11.14. PDC/Evaluación: Métodos Formales (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Métodos formales de verificación y análisis: 1. Extensiones a requisitos formales secuenciales como la linealizabilidad. 2. Especificaciones de protocolo, sesión y transacciones. 3. Uso de herramientas como Lenguaje Unificado de Modelado (UML), Lógica Temporal de Acciones (TLA), lógicas de programa. 4. Análisis de seguridad: seguridad y vivacidad en presencia de comportamientos hostiles o con errores de otras partes; propiedades requeridas de mecanismos de comunicación (por ejemplo, ausencia de fugas entre capas), filtrado de entrada, limitación de velocidad. Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad. 5. Análisis estático aplicado a corrección, rendimiento (*throughput*), latencia, recursos, energía. Ver también: Diseño Sostenible y Computación Ubicua, Huella Ambiental de los Sistemas Informáticos, Efectos Sistémicos y Contexto Social. 6. Análisis del modelo de Grafo Acíclico Dirigido (DAG) de eficiencia algorítmica (trabajo, span, caminos críticos).

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Especificar y medir el comportamiento cuando un servicio es solicitado por un número inesperadamente grande de clientes [Analizar]
2. Identificar y reparar un problema de rendimiento debido a cuellos de botella secuenciales [Analizar]
3. Comparar empíricamente el rendimiento (*throughput*) de dos implementaciones de un diseño común (quizás usando un marco de pruebas existente) [Comparar]

2.11.15. PDC/Evaluación: Pruebas y Medición (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Herramientas de prueba y técnicas de medición: 1. Pruebas y depuración; herramientas como detectores de carreras, fuzzers, verificadores de dependencia de bloqueos, pruebas de unidad/esfuerzo/tortura, visualizaciones, integración continua (CI), despliegue continuo (CD), y generadores de pruebas. 2. Medir y comparar rendimiento (*throughput*), sobrecarga, espera, contención, comunicación, movimiento de datos, localidad, uso de recursos, comportamiento en presencia de números excesivos de eventos, clientes o hilos. Ver también: Eventos, Herramientas y

Experimentación, Métricas y Benchmarks de Rendimiento, Modelos Analíticos de Rendimiento. 3. Análisis específicos del dominio de aplicación y técnicas de evaluación.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar y reparar un problema de rendimiento debido a latencia de comunicación o datos [Analizar]
2. Identificar y reparar un problema de rendimiento debido a sobrecarga en la gestión de recursos [Analizar]
3. Identificar y reparar un problema de confiabilidad o disponibilidad [Analizar]

2.11.16. PDC/Algoritmos (CS Core: 1 hr, KA Core: 3 hrs)

Temas:
Core

- Expresar e implementar algoritmos en lenguajes y marcos de trabajo dados, para iniciar actividades (por ejemplo hilos), usar constructos de memoria compartida, y APIs de canales, sockets y/o llamada a procedimiento remoto (RPC). Ver también: Computación Paralela y Distribuida. 1. Ejemplos de datos paralelos incluyendo mapeo/reducción. 2. Uso de APIs de canal, socket y/o RPC en un lenguaje dado, con control del programa para enviar (generalmente procedimental) vs recibir (generalmente reactivo o basado en RPC). 3. Uso de bloqueos, barreras y/o sincronizadores para mantener la vivacidad sin introducir carreras.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Implementar un componente paralelo/distribuido basado en un algoritmo conocido [Implementar]
2. Escribir un programa de datos paralelos que, por ejemplo, calcule el promedio de un arreglo de números [Escribir]
3. Escribir un programa productor-consumidor en el que un componente genere números y otro calcule su promedio. Medir las aceleraciones cuando los números son escalares pequeños versus valores de multiprecisión grandes [Escribir]

2.11.17. PDC/Algoritmos: Dominios de Aplicación (CS Core: 1 hr, KA Core: 3 hrs)

Temas:
Core

- Panorama de dominios de aplicación comunes en multinúcleo, reactivo, datos paralelos, clúster, nube, sistemas distribuidos abiertos y marcos de trabajo (con referencia a la siguiente tabla). 1. Multinúcleo: Agentes de ejecución típicos: Hilos. Mecanismos de comunicación típicos: Memoria compartida, Atómicos, bloqueos. Dominios algorítmicos típicos: Gestión de recursos, procesamiento de datos. Objetivos de ingeniería típicos: Rendimiento (*throughput*), latencia, energía. 2. Reactivo: Agentes de ejecución típicos: Manejadores, hilos. Mecanismos de comunicación típicos: Canales de E/S. Dominios algorítmicos típicos: Servicios, tiempo real. Objetivos de ingeniería típicos: Latencia. 3. Datos paralelos: Agentes de ejecución típicos: GPU, SIMD, aceleradores, híbridos. Mecanismos de comunicación típicos: Memoria heterogénea. Dominios algorítmicos típicos: álgebra lineal, gráficos, análisis de datos. Objetivos de ingeniería típicos: Rendimiento (*throughput*), energía. 4. Clúster: Agentes de ejecución típicos: Hosts gestionados. Mecanismos de comunicación típicos: Sockets, canales. Dominios algorítmicos típicos: Simulación, análisis de datos. Objetivos de ingeniería típicos: Rendimiento (*throughput*). 5. Nube: Agentes de ejecución típicos: Hosts aprovisionados. Mecanismos de comunicación típicos: APIs de servicio. Dominios algorítmicos típicos: Aplicaciones web. Objetivos de ingeniería típicos: Escalabilidad. 6. Distribuido abierto: Agentes de ejecución típicos: Hosts autónomos. Mecanismos de comunicación típicos: Sockets, Almacenes de datos. Dominios algorítmicos típicos: Almacenes de datos y servicios tolerantes a fallos. Objetivos de ingeniería típicos: Confiabilidad.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Extender un programa secuencial dirigido por eventos estableciendo una nueva actividad en un manejador de eventos (por ejemplo, un nuevo hilo en un manejador de acciones de GUI) [Diseñar]
2. Mejorar el rendimiento de un componente secuencial introduciendo paralelismo y/o distribución [Crear]
3. Elegir entre diferentes diseños paralelos/distribuidos para componentes de un sistema dado [Evaluar (valorar)]

2.11.18. PDC/Algoritmos: Dominios Algorítmicos (CS Core: 1 hr, KA Core: 3 hrs)

Temas:
Core

- Uno o más de los siguientes dominios algorítmicos. Ver también: Estrategias Algorítmicas: 1. álgebra lineal: Operaciones con vectores y matrices, precisión/estabilidad numérica, aplicaciones en análisis de datos y aprendizaje automático. 2. Procesamiento de datos: ordenación, búsqueda y recuperación, estructuras de datos concurrentes. 3. Grafos, búsqueda y combinatoria: Marcado, paralelización de aristas, acotamiento, especulación, análisis basado en redes. 4. Modelado y simulación: ecuaciones diferenciales; aleatorización, problemas de N-cuerpos, algoritmos genéticos. 5. Lógica computacional: satisfactibilidad (SAT), programación lógica concurrente. 6. Gráficos y geometría computacional: Transformaciones, renderizado, trazado de rayos. 7. Gestión de recursos: Asignar, colocar, reciclar y planificar procesadores, memoria, canales y hosts; recursos exclusivos vs compartidos; algoritmos estáticos, dinámicos y elásticos; Restricciones de tiempo real; Lotes, priorización, partición; descentralización mediante robo de trabajo y técnicas relacionadas. 8. Servicios: Implementar APIs web, moneda electrónica, sistemas de transacción, juegos multijugador.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Diseñar, implementar, analizar y evaluar un componente o aplicación para X que opere en un contexto dado, donde X esté en uno de los dominios listados, por ejemplo, un algoritmo genético para el diseño de una planta de fábrica [Diseñar]
2. Criticar el diseño e implementación de un componente o aplicación existente, o uno desarrollado por compañeros de clase [Críticar (análisis crítico)]
3. Comparar el rendimiento y la eficiencia energética de múltiples implementaciones de un diseño similar, por ejemplo, multinúcleo versus clúster versus GPU [Comparar]

2.12. Fundamentos de Desarrollo de Software (SDF)

La fluidez en el proceso de desarrollo de software es fundamental para el estudio de las ciencias de la computación. Para utilizar las computadoras de manera más efectiva para resolver problemas, los estudiantes deben ser competentes en la lectura y escritura de programas. Sin embargo, más allá de las habilidades de programación, deben poder seleccionar y utilizar estructuras de datos y algoritmos apropiados, así como herramientas modernas de desarrollo y prueba.

El área de conocimiento SDF reúne conceptos y habilidades fundamentales relacionados con el desarrollo de software, centrándose en aquellos que deben enseñarse al inicio de un programa de ciencias de la computación, típicamente en el primer año. Esto incluye conceptos de programación fundamentales y su uso efectivo para escribir programas, el uso de estructuras de datos fundamentales que pueden ser proporcionadas por el lenguaje de programación, conceptos básicos de buenas prácticas de programación para escribir programas de calidad, la lectura y comprensión de programas, y cierta comprensión del impacto de los algoritmos en el rendimiento de los programas. Las 43 horas de material en esta área de conocimiento pueden complementarse con material central de otras áreas a medida que los estudiantes avanzan a cursos de nivel intermedio y superior.

Esta área de conocimiento asume un lenguaje de programación contemporáneo con soporte incorporado para tipos de datos comunes, incluidos tipos de datos asociativos como diccionarios/mapas, como vehículo para introducir a los estudiantes a la programación (por ejemplo, Python, Java). Sin embargo, esto no pretende desalentar el uso de lenguajes más antiguos o de bajo nivel para SDF: las unidades de conocimiento a continuación pueden adaptarse adecuadamente al lenguaje real utilizado.

El surgimiento de la IA generativa y los Modelos de Lenguaje Grande (LLMs), que pueden generar programas para muchas tareas de programación, sin duda afectará a la profesión de programación y, en consecuencia, a la enseñanza de muchos temas de CS. Sin embargo, para poder utilizar efectivamente la IA generativa en tareas de programación, un programador debe tener una buena comprensión de los programas y, por lo tanto, aún debe aprender los fundamentos de la programación y desarrollar habilidades básicas de programación, que es el objetivo de SDF. En consecuencia, consideramos que los resultados deseados para SDF deberían permanecer iguales, aunque diferentes instructores ahora podrían dar más énfasis a la comprensión, documentación, especificaciones, análisis y pruebas de programas. (Esto es similar a enseñar a los estudiantes multiplicación, suma, etc., aunque se puedan usar calculadoras para hacerlo).

Área de Conocimiento (KA)	**CS Core**	**KA Core**
⭐ 2.12.1 Conceptos y Construcciones Básicas de Programación (Pág. 94)	10	
⭐ 2.12.2 Prácticas de Programación: Pruebas, Depuración y Documentación (Pág. 95)	10	
⭐ 2.12.3 Estructuras de Datos Fundamentales (Pág. 95)	12	
⭐ 2.12.4 Algoritmos (Pág. 95)	6	
⭐ 2.12.5 Prácticas de Desarrollo de Software (Pág. 96)	5	
2.12.6 Sociedad, ética y la Profesión (Pág. 96)	Electivo	

Cuadro 2.12: Lista de KUs del área de Fundamentos de Desarrollo de Software.

2.12.1. SDF/Conceptos y Construcciones Básicas de Programación (CS Core: 10 hrs)

Temas:
Core

- Conceptos básicos como variables, tipos de datos primitivos, expresiones y su evaluación
- Cómo funcionan los programas imperativos: estado y transiciones de estado en la ejecución de sentencias, flujo de control
- Construcciones básicas como sentencias de asignación, sentencias condicionales e iterativas, E/S básica
- Construcciones clave de modularidad como funciones (y métodos y clases, si son compatibles en el lenguaje) y conceptos relacionados como paso de parámetros, alcance, abstracción, encapsulamiento de datos Fundamentos de Programación Orientada a Objetos, Programación Orientada a Objetos: Encapsulación, Subtipado y Reflexión
- Entrada y salida usando archivos y APIs
- Tipos de datos estructurados disponibles en el lenguaje de programación elegido como secuencias (por ejemplo, arreglos, listas), contenedores asociativos (por ejemplo, diccionarios, mapas), otros (por ejemplo, conjuntos, tuplas) y cuándo y cómo usarlos Estructuras de Datos Fundamentales, Algoritmos Fundamentales, Algoritmos Avanzados

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Desarrollar programas que utilicen las construcciones fundamentales de programación: asignación y expresiones, E/S básica, sentencias condicionales e iterativas [Crear]
2. Desarrollar programas utilizando funciones con paso de parámetros [Crear]
3. Desarrollar programas que utilicen efectivamente los diferentes tipos de datos estructurados proporcionados en el lenguaje, como arreglos/listas, diccionarios y conjuntos [Crear]
4. Desarrollar programas que utilicen E/S de archivos para proporcionar persistencia de datos a través de múltiples ejecuciones [Crear]
5. Desarrollar programas que utilicen APIs para acceder o actualizar datos (por ejemplo, desde la web) [Crear]
6. Desarrollar programas que creen clases simples e instancien objetos de esas clases (si es compatible con el lenguaje) [Crear]
7. Rastrear el flujo de control durante la ejecución de un programa [Trazar/Rastrear]

8. Usar la terminología apropiada para identificar elementos de un programa (por ejemplo, identificador, operador, operando) [Usar]

2.12.2. SDF/Prácticas de Programación: Pruebas, Depuración y Documentación (CS Core: 10 hrs)

Temas:
Core

- Bibliotecas y frameworks proporcionados por el lenguaje (cuando/donde sea aplicable)
- Recursión
- Manejo de errores de tiempo de ejecución en programas (por ejemplo, manejo de excepciones).
- Conceptos básicos de errores de programación, pruebas y depuración Prácticas de Codificación, Construcción a Escala y Proceso, Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada
- Documentar/comentar código a nivel de programa y de módulo. Prácticas de Codificación, Construcción a Escala y Proceso
- Desarrollar una mentalidad de seguridad. Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Desarrollar programas que utilicen bibliotecas y frameworks proporcionados por el lenguaje (donde sea aplicable) [Crear]
2. Explicar el concepto de recursión e identificar cuándo y cómo usarlo efectivamente [Explicar]
3. Desarrollar funciones recursivas [Crear]
4. Desarrollar programas que puedan manejar errores de tiempo de ejecución [Crear]
5. Leer un programa dado y explicar lo que hace [Explicar]
6. Escribir comentarios para un programa o módulo especificando lo que hace [Escribir]

2.12.3. SDF/Estructuras de Datos Fundamentales (CS Core: 12 hrs)

Temas:
Core

- Tipos abstractos de datos estándar como listas, pilas, colas, conjuntos y mapas/diccionarios, incluyendo operaciones sobre ellos.
- Seleccionar y usar estructuras de datos apropiadas.
- Implicaciones de rendimiento de la elección de estructura(s) de datos.
- Cadenas y procesamiento de cadenas.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Escribir programas que utilicen cada uno de los tipos abstractos de datos clave proporcionados en el lenguaje (por ejemplo, arreglos, tuplas/registros/estructuras, listas, pilas, colas y tipos de datos asociativos como conjuntos, diccionarios/mapas) [Escribir]
2. Seleccionar la estructura de datos apropiada para un problema dado [Evaluar]
3. Explicar cómo puede cambiar el rendimiento de un programa al usar diferentes estructuras de datos u operaciones [Explicar]
4. Escribir programas que trabajen con texto utilizando las capacidades de procesamiento de cadenas proporcionadas por el lenguaje [Escribir]

2.12.4. SDF/Algoritmos (CS Core: 6 hrs)

Temas:
Core

- Concepto de algoritmo y noción de eficiencia algorítmica
- Algunos algoritmos comunes (por ejemplo, ordenación, búsqueda, recorrido de árboles, recorrido de grafos)
- Impacto de los algoritmos en la eficiencia tiempo-espacio de los programas

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar el rol de los algoritmos para escribir programas [Explicar]
2. Demostrar cómo un problema puede ser resuelto por diferentes algoritmos, cada uno con diferentes propiedades [Demostrar]
3. Explicar algunos algoritmos comunes (por ejemplo, ordenación, búsqueda, recorrido de árboles, recorrido de grafos) [Explicar]
4. Explicar el impacto en el rendimiento espacio/tiempo de algunos algoritmos [Explicar]

2.12.5. SDF/Prácticas de Desarrollo de Software (CS Core: 5 hrs)

Temas:
Core

- Pruebas básicas, incluido el diseño de casos de prueba
- Uso de un IDE de propósito general, incluido su depurador
- Estilo de programación que mejora la legibilidad
- Especificar la funcionalidad de un módulo en lenguaje natural.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Desarrollar pruebas para módulos y aplicar una variedad de estrategias para diseñar casos de prueba [Crear]
2. Explicar algunas limitaciones de las pruebas de programas [Explicar]
3. Compilar, ejecutar y depurar programas utilizando un IDE moderno y herramientas asociadas como depuradores visuales [Analizar]
4. Aplicar pautas básicas de estilo de programación para ayudar a la legibilidad de los programas, como comentarios, sangría, nombres adecuados de variables, etc [Aplicar]
5. Escribir especificaciones de un módulo como comentario de módulo que describa su funcionalidad [Escribir]

2.12.6. SDF/Sociedad, ética y la Profesión

Temas:
Core

- Derechos de propiedad intelectual de los programadores para los programas que desarrollan.
- Plagio e integridad académica.
- Responsabilidad y obligación de los programadores con respecto al código que desarrollan para soluciones. Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad
- ética profesional básica del trabajo de los programadores.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar/comprender algunos de los problemas de propiedad intelectual relacionados con programas [Explicar]
2. Explicar/comprender cuándo se puede utilizar código desarrollado por otros y las formas adecuadas de revelar su uso [Explicar]
3. Explicar/comprender la responsabilidad de los programadores al desarrollar código para una solución general (que puede ser desarrollada por un equipo) [Explicar]
4. Explicar/comprender uno o más códigos de conducta aplicables a los programadores [Explicar]

2.13. Ingeniería de Software (SE)

Ya a principios de la década de 1970, el informático británico Brian Randell supuestamente dijo: "La ingeniería de software es la construcción multi-persona de programas multi-versión". Esta es una visión esencial: mientras que la programación es la habilidad que gobierna nuestra capacidad para escribir un programa, la ingeniería de software es distinta en dos dimensiones: tiempo y personas.

Primero, un proyecto de ingeniería de software es un esfuerzo de equipo; ser un experto en programación solitario es insuficiente. Los ingenieros de software calificados deben demostrar experiencia en comunicación y colaboración. La programación puede ser una actividad individual, pero la ingeniería de software es colaborativa, profundamente ligada a cuestiones de profesionalismo, trabajo en equipo y comunicación.

Segundo, un proyecto de ingeniería de software suele ser "multi-versión". Tiene una vida útil esperada; necesita funcionar correctamente durante meses, años o décadas. Las funcionalidades pueden agregarse o eliminarse para cumplir con los requisitos del producto. El propio equipo de ingeniería probablemente cambiará. El contexto tecnológico cambiará, a medida que evolucionen nuestras plataformas informáticas, cambien los lenguajes de programación, se actualicen las dependencias, etc. Esta exposición a cuestiones de tiempo y cambio es novedosa en comparación con un proyecto de programación: no basta con construir algo que funcione; debe funcionar y seguir funcionando. Muchos de los temas más desafiantes en tecnología comparten "el tiempo conducirá al cambio" como causa raíz: compatibilidad con versiones anteriores, desfase de versiones, gestión de dependencias, cambios de esquema, evolución de protocolos.

La ingeniería de software presenta un desafío particularmente difícil para el aprendizaje en un entorno académico. Dado que las principales diferencias entre la programación y la ingeniería de software son el tiempo y el trabajo en equipo, es difícil generar lecciones que requieran un trabajo en equipo exitoso y que presenten fielmente los desafíos del tiempo. Además, algunos temas de ingeniería de software serán más auténticos y relevantes si nuestros estudiantes experimentan proyectos colaborativos y a largo plazo in vivo en lugar de en el aula. Independientemente de que esto suceda como una pasantía, la participación en un proyecto de código abierto o un rol de ingeniería a tiempo completo, un mes de experiencia práctica a tiempo completo tiene más horas disponibles que el curso promedio de ingeniería de software.

Por lo tanto, un plan de estudios de ingeniería de software debe centrarse en los conceptos necesarios para la mayoría de los nuevos contratados graduados, y que son novedosos para aquellos que se capacitan principalmente como programadores, o que son conceptos abstractos que pueden no declararse/compartirse explícitamente en el trabajo. Tales temas incluyen, entre otros:

- Pruebas
- Trabajo en equipo, colaboración
- Comunicación
- Diseño
- Mantenimiento y evolución
- Herramientas de ingeniería de software

Todas las pruebas sugieren que el papel del software en nuestra sociedad seguirá creciendo en el futuro previsible. Además, la era de "dos programadores en un garaje" parece haber llegado a su fin. La mayoría del software importante hoy en día es un esfuerzo de equipo, que se basa en código existente y aprovecha la funcionalidad existente. El estudio de las habilidades de ingeniería de software es un contrapunto profundamente importante a la experiencia cotidiana de los estudiantes de informática: debemos impresionarles con la realidad de que pocos proyectos de software se gestionan escribiendo desde cero como un esfuerzo en solitario. La comunicación, el trabajo en equipo, la planificación, las pruebas y las herramientas son mucho más importantes a medida que nuestros estudiantes pasan del aula y dejan su huella en el mundo.

Aunque la mayoría de los graduados en CS ocuparán un puesto en la industria que requiere este material, los temas del CS Core presentados aquí son valiosos independientemente de si los graduados van a la industria o a la academia.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 2.13.1 Colaboración y Dinámica de Equipo (Pág. 98)	2	2
⭐ 2.13.2 Diversidad, Stakeholders y Equipos Distribuidos (Pág. 99)	1	1
⭐ 2.13.3 Control de Versiones y CI/CD (Pág. 99)	1	2
⭐ 2.13.4 Entorno de Desarrollo e IDE (Pág. 100)	1	2
⭐ 2.13.5 Ingeniería de Requisitos (Pág. 100)		2
⭐ 2.13.6 Evolución y Estimación de Requisitos (Pág. 101)		1
⭐ 2.13.7 Principios y Arquitectura de Software (Pág. 101)	1	2
⭐ 2.13.8 Diseño de API y Modelado (Pág. 102)	1	1
⭐ 2.13.9 Calidad y Evaluación del Diseño (Pág. 102)	1	2
⭐ 2.13.10 Prácticas de Codificación (Pág. 103)	1	2
⭐ 2.13.11 Construcción a Escala y Proceso (Pág. 103)	1	2
⭐ 2.13.12 Conceptos de verificación y validación (Pág. 104)	1	1
⭐ 2.13.13 Planificación y Tipos de Pruebas (Pág. 104)	1	1
⭐ 2.13.14 Prácticas Avanzadas de Pruebas (Pág. 105)	1	1
⭐ 2.13.15 Herramientas y Análisis de Pruebas (Pág. 105)	1	1
⭐ 2.13.16 Pruebas de Rendimiento y Evaluación Comparativa (Pág. 106)	1	1
⭐ 2.13.17 Compatibilidad y Versionado de API (Pág. 106)		2
⭐ 2.13.18 Técnicas de Refactorización (Pág. 107)		1
⭐ 2.13.19 Conceptos de Confiabilidad (Pág. 107)		2
⭐ 2.13.20 Ingeniería de Confiabilidad (Pág. 108)		1
2.13.21 Métodos Formales (Pág. 108)	Electivo	

Cuadro 2.13: Lista de KUs del área de Ingeniería de Software.

2.13.1. SE/Colaboración y Dinámica de Equipo (CS Core: 2 hrs, KA Core: 2 hrs)

Temas:
Core

- Comunicación efectiva, incluida la oral y escrita, así como formal (correo electrónico, documentos, comentarios, presentaciones) e informal (chat de equipo, reuniones). Comunicación Técnica y Profesional, Comunicación en Equipo y Cultura
- Causas comunes de conflicto en el equipo y enfoques para la resolución de conflictos.
- Programación cooperativa: 1. Programación por parejas (pair programming) o Enjambre (swarming) 2. Revisión de código 3. Colaboración mediante control de versiones
- Roles y responsabilidades en un equipo de software: Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo 1. Ventajas del trabajo en equipo 2. Riesgos y complejidad de dicha colaboración
- Procesos del equipo: responsabilidades para las tareas, estimación de esfuerzo, estructura de reuniones, cronograma de trabajo

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Seguir prácticas efectivas de comunicación en equipo [Analizar]
2. Articular las fuentes, peligros y beneficios potenciales del conflicto en el equipo, especialmente enfocándose en el valor de estar en desacuerdo sobre ideas o propuestas sin insultar a las personas [Articular]
3. Facilitar una estrategia de resolución de conflictos y resolución de problemas en un entorno de equipo [Analizar]
4. Colaborar efectivamente en desarrollo/programación cooperativa [Analizar]
5. Proponer y delegar los roles y responsabilidades necesarios en un equipo de desarrollo de software [Proponer]
6. Redactar y seguir una agenda para una reunión de equipo [Componer]
7. Facilitar, a través de la participación en un proyecto de equipo, los elementos centrales de la formación de equipos, el establecimiento de una cultura de equipo saludable y la gestión de

equipos, incluida la creación y ejecución de un plan de trabajo del equipo [Crear]

2.13.2. SE/Diversidad, Stakeholders y Equipos Distribuidos (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Importancia de la diversidad e inclusión en el equipo. Comunicación Técnica y Profesional, Comunicación en Equipo y Cultura
- Interfaz con las partes interesadas, como equipo: 1. Gestión y otros equipos no técnicos 2. Clientes 3. Usuarios
- Riesgos asociados con equipos físicos, distribuidos, híbridos y virtuales, incluyendo comunicación, percepción, estructura, puntos de falla, mitigación y recuperación, etc.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Promover la importancia y los beneficios que la diversidad y la inclusión aportan a un equipo de desarrollo de software [Defender]
2. Referenciar, como equipo, la importancia de, y las estrategias para interactuar con partes interesadas fuera del equipo en niveles tanto técnicos como no técnicos [Referenciar]
3. Enumerar los riesgos asociados con equipos físicos, distribuidos, híbridos y virtuales y los posibles puntos de falla y cómo mitigarlos y recuperarse/aprender de los fallos [Enumerar]

2.13.3. SE/Control de Versiones y CI/CD (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Gestión de configuración de software y control de versiones: Prácticas de Desarrollo de Software 1. Configuración en control de versiones, builds/configuración reproducibles. 2. Estrategias de ramificación en control de versiones. Ramas de desarrollo vs ramas de lanzamiento. Desarrollo basado en tronco (*trunk-based development*). 3. Estrategias de fusión/rebase, cuando sean relevantes.
- Gestión de lanzamientos (*release management*).
- Herramientas de prueba, incluyendo herramientas de análisis estático y dinámico. Prácticas de Desarrollo de Software, Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada
- Automatización de procesos de software: 1. Sistemas de construcción (*build systems*): el valor de builds rápidos, herméticos y reproducibles, comparar/contrastar enfoques para construir un proyecto. 2. Integración Continua (CI): el uso de automatización y pruebas automatizadas para hacer una validación preliminar de que la revisión actual del tronco se construye y pasa las pruebas (básicas). 3. Despliegue Continuo (CD): el uso de automatización para liberar de forma automática cada cambio que supera las pruebas hacia el entorno de producción, garantizando entregas frecuentes y confiables. 4. Gestión de dependencias: actualización de dependencias externas/aguas arriba, gestión de paquetes, SemVer.
- Flujo de revisión colaborativa como puerta de calidad 1. Revisión obligatoria de *pull requests* antes de fusionar 2. Propiedad del código mediante reglas de tipo CODEOWNERS 3. Convenciones de mensajes de *commit* (*conventional commits*) 4. Reglas de protección de ramas que exigen verificaciones antes de fusionar
- Escaneo de vulnerabilidades de dependencias y de la cadena de suministro integrado en los pipelines de CI

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir la diferencia entre la gestión de configuración de software centralizada y distribuida [Describir]
2. Describir cómo el control de versiones puede usarse para ayudar en la gestión de lanzamientos de software [Describir]

3. Identificar elementos de configuración y usar una herramienta de control de código fuente en un proyecto pequeño basado en equipo [Analizar]
4. Describir cómo las herramientas de prueba estáticas y dinámicas disponibles pueden integrarse en el entorno de desarrollo de software [Describir]
5. Comprender el uso de sistemas de CI/CD como una fuente de verdad para el estado del código compartido del equipo (éxito en la construcción y pruebas) [Explicar]
6. Aplicar un flujo de revisión de *pull requests* con reglas de propiedad de código y protección de ramas para imponer puertas de calidad en un proyecto de equipo [Aplicar]
7. Usar un escáner de vulnerabilidades de dependencias/cadena de suministro como parte de un pipeline de CI para detectar dependencias vulnerables [Usar]

2.13.4. SE/Entorno de Desarrollo e IDE (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Herramientas de diseño y comunicación (documentos, diagramas, formas comunes de diagramas de diseño como UML).
- Conceptos y mecanismos de integración de herramientas. Prácticas de Desarrollo de Software
- Uso de las facilidades de un IDE moderno: depuración, refactorización, búsqueda/indexación, asistentes de código basados en ML, etc. Prácticas de Desarrollo de Software

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir los problemas importantes al seleccionar un conjunto de herramientas para el desarrollo de un sistema de software específico, incluyendo herramientas para seguimiento de requisitos, modelado de diseño, implementación, automatización de construcción y pruebas [Describir]
2. Demostrar la capacidad de usar herramientas de software para apoyar el desarrollo de un producto de software de tamaño mediano [Demostrar]

2.13.5. SE/Ingeniería de Requisitos (KA Core: 2 hrs)

Temas:
Core

- Describir requisitos funcionales usando, por ejemplo, casos de uso o historias de usuario. 1. Usar al menos un método para documentar y estructurar requisitos funcionales. 2. Comprender cómo el método apoya el diseño y la implementación. 3. Fortalezas y debilidades de usar un enfoque específico.
- Propiedades de los requisitos, incluyendo consistencia, validez, completitud y viabilidad.
- Obtención (*elicitation*) de requisitos. 1. Fuentes de requisitos, por ejemplo, usuarios, administradores o personal de soporte. 2. Métodos de recopilación de requisitos, por ejemplo, encuestas, entrevistas o análisis de comportamiento.
- Requisitos no funcionales, por ejemplo, seguridad, usabilidad o rendimiento, también llamados Atributos de Calidad. Diseño Sostenible y Computación Ubicua, Huella Ambiental de los Sistemas Informáticos, Efectos Sistémicos y Contexto Social
- Identificación y gestión de riesgos, incluyendo consideraciones éticas en torno al producto propuesto. Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo
- Comunicar y/o formalizar especificaciones de requisitos.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comparar diferentes métodos de obtención de requisitos a lo largo de múltiples ejes [Comparar]
2. Identificar diferencias entre dos métodos de describir requisitos funcionales (por ejemplo, entrevistas con clientes, estudios de usuarios) y las situaciones donde se preferiría cada uno [Analizar]
3. Identificar qué comportamientos son requeridos, permitidos o prohibidos a partir de un conjunto dado de requisitos y una lista de comportamientos candidatos [Analizar]
4. Recopilar un conjunto de requisitos para un sistema de software simple [Analizar]

5. Identificar áreas de un sistema de software que deben cambiarse, dada una descripción del sistema y un conjunto de nuevos requisitos a implementar [Analizar]
6. Identificar los requisitos funcionales y no funcionales en un conjunto de requisitos [Analizar]

2.13.6. SE/Evolución y Estimación de Requisitos (KA Core: 1 hr)

Temas:
Non Core

- Prototipado de una herramienta tanto para obtener como para validar/confirmar requisitos.
- Evolución del producto: cuando cambian los requisitos, cómo entender qué efecto tiene eso y qué cambios deben realizarse.
- Estimación de esfuerzo: 1. Aprender técnicas para estimar mejor el esfuerzo requerido para completar una tarea; 2. Practicar la estimación y compararla con cuánto tiempo toman las tareas; 3. La estimación de esfuerzo es bastante difícil, por lo que es probable que los estudiantes se equivoquen en muchos casos, pero ver el proceso desarrollarse con su propio trabajo es valioso.

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Crear un prototipo de un sistema de software para validar un conjunto de requisitos: construir una maqueta, MVP, etc [Crear]
2. Estimar el tiempo para completar un conjunto de tareas, luego comparar las estimaciones con el tiempo real tomado [Estimar]
3. Determinar una secuencia de implementación para un conjunto de tareas, respetando las dependencias entre ellas, con el objetivo de retirar el riesgo lo antes posible [Determinar]
4. Escribir una especificación de requisitos para un sistema de software simple [Escribir]

2.13.7. SE/Principios y Arquitectura de Software (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Principios de diseño de sistemas. Confiabilidad del Sistema 1. Niveles de abstracción (por ejemplo, diseño arquitectónico y diseño detallado) 2. Separación de preocupaciones (separation of concerns) 3. Ocultación de información (information hiding) 4. Acoplamiento y cohesión (coupling and cohesion)
- Arquitectura de software. Confiabilidad del Sistema 1. Paradigmas de diseño *a*) Descomposición funcional de arriba hacia abajo/diseño por capas *b*) Arquitectura orientada a datos *c*) Análisis y diseño orientado a objetos *d*) Diseño dirigido por eventos 2. Arquitecturas estándar (por ejemplo, cliente-servidor y arquitecturas de microservicios incluyendo discusiones sobre REST, n-capas, tuberías y filtros, Modelo Vista Controlador) 3. Identificar límites y dependencias de componentes
- Programación a gran escala vs programación a pequeña escala (programming in the large vs programming in the small). Confiabilidad del Sistema
- Malos olores del código (code smells) y otras indicaciones de la calidad del código, distintas de la corrección. Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar la arquitectura de software estándar de un diseño de alto nivel dado [Analizar]
2. Seleccionar y usar un paradigma de diseño apropiado para diseñar un sistema de software simple y explicar cómo se han aplicado los principios de diseño de sistemas en este diseño [Evaluar]
3. Adaptar un diseño de sistema defectuoso para que siga mejor principios como la separación de preocupaciones o la ocultación de información [Crear]
4. Identificar las dependencias entre un conjunto de componentes de software en un diseño arquitectónico [Analizar]

2.13.8. SE/Diseño de API y Modelado (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Principios de diseño de API 1. Consistencia *a*) Las API consistentes son más fáciles de aprender y menos propensas a errores *b*) La consistencia es tanto interna (entre diferentes partes de la API) como externa (siguiendo patrones comunes de API) 2. Capacidad de composición (*composability*) 3. Documentación de contratos *a*) Las operaciones de la API deben describir su efecto en el sistema, pero generalmente no su implementación *b*) Precondiciones, postcondiciones e invariantes 4. Capacidad de expansión (*expandability*) 5. Reporte de errores *a*) Los errores deben ser claros, predecibles y accionables *b*) La entrada que no coincida con el contrato debe producir un error *c*) Los errores que pueden gestionarse de manera confiable sin reportarse deben gestionarse
- Identificar y codificar invariantes de datos e invariantes temporales
- Modelos estructurales y de comportamiento de diseños de software

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Diseñar una API para un solo componente de un sistema de software grande, incluyendo la identificación y documentación de las invariantes, el contrato y las condiciones de error de cada operación [Diseñar]
2. Evaluar una descripción de API en términos de consistencia, capacidad de composición y capacidad de expansión [Evaluar]
3. Expandir un diseño existente para incluir una nueva funcionalidad [Crear]

2.13.9. SE/Calidad y Evaluación del Diseño (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Diseño de datos Modelado de Datos 1. Estructuras de datos 2. Sistemas de almacenamiento
- Trazabilidad de requisitos 1. Comprender qué requisitos son satisfechos por un diseño

Non Core

- Modelado de diseño, por ejemplo con diagramas de clases, diagramas de entidad-relación o diagramas de secuencia
- Medición y análisis de la calidad del diseño
- Principios de diseño y codificación seguros Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad 1. Principio de privilegio mínimo 2. Principio de valores predeterminados seguros ante fallos (fail-safe defaults) 3. Principio de aceptabilidad psicológica
- Evaluar compensaciones de diseño (tradeoffs) (por ejemplo, eficiencia vs confiabilidad, seguridad vs usabilidad)
- Modelo de características de calidad ISO/IEC 25010 y costo de la mala calidad 1. Características de calidad: adecuación funcional, eficiencia de desempeño, compatibilidad, usabilidad, confiabilidad, seguridad, mantenibilidad, portabilidad 2. Costo de la mala calidad y la deuda técnica como preocupación de negocio/económica, no solo técnica

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Diseñar un conjunto de estructuras de datos para implementar una superficie de API proporcionada [Diseñar]
2. Identificar qué requisitos son satisfechos por un diseño de software proporcionado [Analizar]

NonCore:

3. Traducir un diseño de software en lenguaje natural a diagramas de clases [Traducir]
4. Adaptar un diseño de sistema defectuoso para que siga mejor los principios de privilegio mínimo y valores predeterminados seguros ante fallos [Crear]
5. Contrastar dos diseños de software a través de diferentes cualidades, como eficiencia o usabilidad [Contrastar]
6. Evaluar un sistema de software frente a las características de calidad ISO/IEC 25010 y estimar el costo de negocio de la mala calidad/deuda técnica [Evaluar]

2.13.10. SE/Prácticas de Codificación (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Pruebas prácticas a pequeña escala Prácticas de Desarrollo de Software 1. Pruebas unitarias 2. Desarrollo guiado por pruebas (Test-driven development): esto es particularmente valioso para los estudiantes psicológicamente, ya que es mucho más fácil involucrarse constructivamente con el desafío de identificar entradas desafiantes para una API dada (casos límite, casos extremos) a priori. Si implementan primero, el instinto es a menudo evitar intentar hacer fallar su nueva creación, mientras que un enfoque de pruebas primero les da la satisfacción intelectual de detectar los casos problemáticos y luego ver cómo pasan más pruebas durante el proceso de desarrollo.
- Documentación Prácticas de Desarrollo de Software 1. Documentación de la interfaz: describir los requisitos de la interfaz, potencialmente incluyendo contratos (formales o informales), pre y post condiciones, invariantes. 2. La documentación de la implementación debe centrarse en partes de código complicadas y no obvias, ya sea porque el código usa características avanzadas del lenguaje o porque el comportamiento del código es complejo. (No agregar comentarios que reafirmen operaciones comunes/obvias y características simples del lenguaje.) *a*) Aclarar el flujo de datos, cómputo, etc., centrándose en lo que el código es. *b*) Identificar partes sutiles/complicadas del código y refactorizar para que sean autoexplicativas si es posible o proporcionar comentarios apropiados para aclarar.
- Estilo de codificación Prácticas de Desarrollo de Software 1. Guías de estilo 2. Comentarios 3. Nomenclatura (*naming*)
- "Mejores Prácticas" para la codificación: técnicas, patrones/idiomas, mecanismos para construir programas de calidad Prácticas de Desarrollo de Software, Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada 1. Prácticas de codificación defensiva 2. Prácticas y principios de codificación segura 3. Usar mecanismos de manejo de excepciones para hacer los programas más robustos, tolerantes a fallos
- Depuración (debugging) Prácticas de Desarrollo de Software
- Registro (logging)
- Uso de bibliotecas y frameworks desarrollados por otros Prácticas de Desarrollo de Software

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Escribir pruebas unitarias apropiadas para un componente pequeño (varias funciones, un solo tipo, etc.) [Escribir]
2. Escribir comentarios apropiados de interfaz y (si es necesario) de implementación para un componente pequeño [Escribir]
3. Describir técnicas, patrones de codificación y mecanismos para implementar diseños para lograr propiedades deseadas como confiabilidad, eficiencia y robustez [Describir]
4. Escribir código robusto utilizando mecanismos de manejo de excepciones [Escribir]
5. Describir prácticas de codificación segura y defensiva [Describir]
6. Seleccionar y usar un estándar de codificación definido en un proyecto de software pequeño [Usar]
7. Comparar y contrastar estrategias de integración, incluyendo integración de arriba hacia abajo (top-down), de abajo hacia arriba (bottom-up) y sándwich [Comparar]
8. Describir el proceso de analizar e implementar cambios en la base de código desarrollada para un proyecto específico [Describir]
9. Describir el proceso de analizar e implementar cambios en una base de código existente grande [Describir]

2.13.11. SE/Construcción a Escala y Proceso (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Non Core

- Pruebas a mayor escala 1. Dobles de prueba (test doubles) (*stubs*, *mocks*, *fakes*) 2. Inyección de dependencias (*dependency injection*)

- Secuenciación del trabajo, incluyendo identificación de dependencias, hitos y retiro de riesgo 1. Identificación de dependencias: identificar las dependencias entre diferentes tareas 2. Hitos: una colección de tareas que sirve como marcador de progreso cuando se completa. Idealmente, el hito abarca una unidad útil de funcionalidad. 3. Retiro de riesgo: identificar qué elementos de un proyecto son riesgosos y priorizar la finalización de tareas que aborden esos riesgos.
- Problemas potenciales de seguridad en programas Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada 1. Desbordamientos de búfer y otros tipos de desbordamientos 2. Condiciones de carrera (race conditions) 3. Inicialización incorrecta, incluida la elección de privilegios 4. Validación de entrada (input validation)
- Documentación (autogenerada)
- Contexto de desarrollo: "campo verde" (*green field*) vs base de código existente 1. Análisis de impacto del cambio (*change impact analysis*) 2. Realización del cambio (*change actualization*)
- Gestión de lanzamientos (release management)
- Prácticas de DevOps 1. Definition of Done como umbral de calidad compartido para el trabajo terminado 2. Postmortems sin culpa (*blameless postmortems*) para aprender de incidentes/fallas 3. Cultura de calidad en equipo: propiedad compartida, retroalimentación y aprendizaje continuo

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Reescribir un programa simple para eliminar vulnerabilidades comunes, como desbordamientos de búfer, desbordamientos de enteros y condiciones de carrera [Aplicar]
2. Escribir un componente de software que realice alguna tarea no trivial y sea resistente a errores de entrada y de tiempo de ejecución [Escribir]

2.13.12. SE/Conceptos de verificación y validación (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Conceptos de verificación y validación 1. Verificación: ¿Estamos construyendo la cosa correctamente? 2. Validación: ¿Construimos la cosa correcta?
- Por qué importan las pruebas: ¿El componente sigue siendo funcional a medida que el código evoluciona?
- Objetivos de las pruebas 1. Usabilidad 2. Confiabilidad 3. Conformidad con la especificación 4. Rendimiento 5. Seguridad

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar por qué las pruebas son importantes [Explicar]
2. Distinguir entre validación y verificación de programas [Distinguir]
3. Describir diferentes objetivos de las pruebas [Describir]

2.13.13. SE/Planificación y Tipos de Pruebas (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Tipos de pruebas 1. Unitarias 2. Integración 3. Validación 4. Sistema
- Diferencias estilísticas entre el código de prueba y el código de producción: DAMP vs DRY - se justifica más duplicación en el código de prueba.
- Planificación y generación de pruebas 1. Generación de casos de prueba, a partir de modelos formales, especificaciones, etc. 2. Cobertura de pruebas *a*) Matrices de prueba *b*) Cobertura de código - ¿cuánto del código está probado? *c*) Cobertura de entorno - ¿cuántas arquitecturas de hardware, sistemas operativos, navegadores, etc. están probados? 3. Datos y entradas de prueba

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comparar y contrastar los diferentes tipos y niveles de pruebas (regresión, unitarias, integración, sistemas y aceptación) [Comparar]
2. Describir técnicas para crear un plan de pruebas y generar casos de prueba [Describir]
3. Crear un plan de pruebas para un segmento de código de tamaño mediano que incluya una matriz de pruebas y generación de datos y entradas de prueba [Crear]
4. Implementar un plan de pruebas para un segmento de código de tamaño mediano [Implementar]

2.13.14. SE/Prácticas Avanzadas de Pruebas (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Desarrollo de pruebas 1. Desarrollo guiado por pruebas 2. Pruebas orientadas a objetos, uso de mocks e inyección de dependencias 3. Técnicas de prueba de caja opaca (anteriormente, caja negra) y caja transparente (anteriormente, caja blanca) 4. Herramientas de prueba, incluyendo cobertura de código, análisis estático y fuzzing
- Verificación y validación en el ciclo de desarrollo 1. Revisiones de código 2. Automatización de pruebas, incluyendo automatización de herramientas 3. Pruebas pre-commit y post-commit 4. Compensaciones entre cobertura de pruebas y rendimiento/latencia de las pruebas 5. Seguimiento y priorización de defectos: reproducibilidad de los defectos reportados
- Desafíos de verificación y validación específicos del dominio 1. Pruebas de rendimiento y evaluación comparativa (benchmarking) 2. Asincronía, paralelismo y concurrencia 3. Crítico para la seguridad (*safety-critical*) 4. Numérico

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar los principios fundamentales de los métodos de desarrollo guiado por pruebas y explicar el papel de las pruebas automatizadas en estos métodos [Analizar]
2. Discutir problemas relacionados con las pruebas de software orientado a objetos [Debatir]
3. Describir el uso de mocks y la inyección de dependencias y su aplicación [Describir]
4. Realizar, como parte de una actividad en equipo, una revisión de código de un segmento de código de tamaño mediano [Evaluar]
5. Describir el papel que las herramientas pueden desempeñar en la validación del software [Describir]
6. Automatizar las pruebas en un proyecto de software pequeño [Diseñar]
7. Explicar los roles, pros y contras de las pruebas pre-commit y post-commit [Explicar]
8. Discutir las compensaciones entre la cobertura de pruebas y el rendimiento/latencia de las pruebas y cómo esto puede afectar la verificación [Debatir]
9. Usar una herramienta de seguimiento de defectos para gestionar defectos de software en un proyecto de software pequeño [Usar]
10. Discutir las limitaciones de las pruebas en ciertos dominios [Debatir]

2.13.15. SE/Herramientas y Análisis de Pruebas (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Non Core

- Herramientas y automatización de verificación y validación 1. Análisis estático 2. Cobertura de código 3. Fuzzing 4. Análisis dinámico y contención de fallos (sanitizadores, etc.) 5. Registro y seguimiento de fallos
- Planificación y generación de pruebas 1. Estimación de fallos y terminación de pruebas, incluyendo siembra de defectos 2. Uso de números aleatorios y pseudoaleatorios en las pruebas
- Pruebas de sistemas asíncronos, paralelos y concurrentes
- Verificación y validación de artefactos que no son código (documentación, materiales de capacitación)
- Desarrollo guiado por comportamiento (BDD) 1. Expresar criterios de aceptación como escenarios Gherkin Given/When/Then 2. Ejecutar escenarios BDD como pruebas automatizadas de extremo a extremo

- Pruebas de mutación y análisis estático de calidad de código 1. Pruebas de mutación para evaluar la efectividad de una suite de pruebas 2. Detección estática de *code smells* y complejidad ciclomática excesiva

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Describir y comparar diferentes herramientas para verificación y validación [Describir]
2. Automatizar el uso de diferentes herramientas en un proyecto de software pequeño [Diseñar]
3. Explicar cómo y cuándo se deben usar números aleatorios en las pruebas [Explicar]
4. Describir enfoques para la estimación de fallos [Describir]
5. Estimar el número de fallos en una aplicación de software pequeña basándose en la densidad de fallos y la siembra de fallos [Estimar]
6. Describir técnicas y problemas con las pruebas de software asíncrono, concurrente y paralelo [Describir]
7. Crear un plan de pruebas para un segmento de código de tamaño mediano que contenga código asíncrono, concurrente y/o paralelo, incluyendo una matriz de pruebas y generación de datos y entradas de prueba [Crear]
8. Describir técnicas para la verificación y validación de artefactos que no son código [Describir]
9. Escribir escenarios Gherkin Given/When/Then para los criterios de aceptación de una funcionalidad y ejecutarlos como pruebas automatizadas de extremo a extremo [Escribir]
10. Analizar el puntaje de mutación de una suite de pruebas y refactorizar las pruebas para eliminar los mutantes sobrevivientes [Analizar]

2.13.16. SE/Pruebas de Rendimiento y Evaluación Comparativa (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Non Core

- Pruebas de rendimiento y evaluación comparativa 1. Rendimiento (*throughput*) y latencia 2. Degradación bajo carga (pruebas de estrés, manejo FIFO vs LIFO de solicitudes) 3. Aceleración (*speedup*) y escalado (*scaling*) *a*) Ley de Amdahl *b*) Ley de Gustafson *c*) Escalado suave (*soft*) y débil (*weak*) 4. Identificar y medir figuras de mérito (*figures of merit*) 5. Cuellos de botella de rendimiento comunes *a*) Limitado por cómputo (*compute-bound*) *b*) Limitado por ancho de banda de memoria *c*) Limitado por latencia 6. Métodos estadísticos y mejores prácticas para la evaluación comparativa (benchmarking) *a*) Estimación de la incertidumbre *b*) Intervalos de confianza 7. Análisis y presentación (gráficos, etc.) 8. Técnicas de medición de tiempo (*timing*)

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Describir el rendimiento (*throughput*) y la latencia y proporcionar ejemplos de cada uno [Describir]
2. Explicar la aceleración (speedup) y las diferentes formas de escalado (scaling) y cómo se calculan [Explicar]
3. Describir cuellos de botella de rendimiento comunes [Describir]
4. Describir métodos estadísticos y mejores prácticas para la evaluación comparativa (benchmarking) de software [Describir]
5. Explicar técnicas y desafíos para medir el tiempo al construir una evaluación comparativa (benchmark) [Explicar]
6. Identificar las figuras de mérito, construir y ejecutar una evaluación comparativa (benchmark) y analizar estadísticamente y visualizar los resultados para un proyecto de software pequeño [Analizar]

2.13.17. SE/Compatibilidad y Versionado de API (KA Core: 2 hrs)

Temas:
Core

- Ley de Hyrum / Ley de las Interfaces Implícitas

- Compatibilidad con versiones anteriores 1. La compatibilidad no es una propiedad de una sola entidad, es una propiedad de una relación. 2. La compatibilidad con versiones anteriores debe evaluarse en términos de proveedor + consumidor(es) o con un modelo bien especificado de qué formas de compatibilidad aspira/promete un proveedor.
- Control de versiones (versioning) 1. Versionado Semántico (SemVer) 2. Desarrollo basado en tronco (*trunk-based development*)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar tanto el comportamiento explícito como el implícito de una interfaz e identificar riesgos potenciales de la Ley de Hyrum [Analizar]
2. Identificar cambios que pueden considerarse ampliamente "compatibles con versiones anteriores", potencialmente con declaraciones explícitas sobre qué uso está o no soportado [Analizar]
3. Evaluar si un cambio propuesto es lo suficientemente seguro dada la metodología de control de versiones en uso para un proyecto dado [Evaluar]

2.13.18. SE/Técnicas de Refactorización (KA Core: 1 hr)

Temas:
Core

- Refactorización 1. Patrones de refactorización estándar (renombrar, integrar, extraer, etc.) 2. Uso de herramientas de refactorización en el IDE 3. Aplicación de herramientas de análisis estático (para identificar código que necesita refactorización, generar cambios, etc.) 4. Valor de la refactorización como remedio para la deuda técnica

Non Core

- Refactorización "a Gran Escala": técnicas cuando un cambio de refactorización es demasiado grande para comprometerlo de forma segura (proyectos grandes), o cuando es imposible sincronizar el cambio entre proveedor + todos los consumidores (múltiples repositorios, consumidores con código privado). 1. Expresar tanto las API antiguas como las nuevas para que puedan coexistir. 2. Minimizar el tamaño de los cambios de comportamiento. 3. ¿Por qué se requieren estas técnicas?. Por ejemplo, "consumidores de API que puedo ver" vs "consumidores que no puedo ver").

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Considerar las entradas de herramientas de análisis estático y/o los principios de Diseño de Software para identificar código que necesita refactorización [Analizar]
2. Refactorizar la implementación de una interfaz para mejorar el diseño, claridad, etc., con impacto mínimo/cero en los usuarios existentes [Crear]

NonCore:

3. Planificar una refactorización compleja de múltiples pasos para cambiar el comportamiento predeterminado de una API de manera segura [Planear]

2.13.19. SE/Conceptos de Confiabilidad (KA Core: 2 hrs)

Temas:
Core

- Concepto de confiabilidad como probabilidad de fallo o tiempo medio entre fallos, y fallas (faults) como causa de fallos (failures)
- Identificar requisitos de confiabilidad para diferentes tipos de software
- Fallos de software causados por defectos/errores (bugs), por lo que para alta confiabilidad el objetivo es tener un mínimo de defectos - inyectando menos defectos (mejor capacitación, educación, planificación) y eliminando la mayoría de los defectos inyectados (pruebas, revisión de código, etc.)
- Confiabilidad del software, confiabilidad del sistema y comportamiento de fallos
- Ciclo de inyección y eliminación de defectos, y diferentes enfoques para la eliminación de defectos

- Comparar el enfoque de "presupuesto de error" (error budget) para la confiabilidad con el enfoque "libre de errores" (error-free) e identificar dominios donde cada uno es relevante.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir cómo determinar el nivel de confiabilidad requerido por un sistema de software [Describir]
2. Explicar los problemas que existen para lograr niveles muy altos de confiabilidad [Explicar]
3. Comprender enfoques para minimizar fallas que pueden aplicarse en cada etapa del ciclo de vida del software [Explicar]

2.13.20. SE/Ingeniería de Confiabilidad (KA Core: 1 hr)

Temas:
Non Core

- Modelos de confiabilidad de software
- Técnicas y modelos de tolerancia a fallos de software 1. Diferencias contextuales en la tolerancia a fallos (por ejemplo, hacer fallar un sistema crítico para la aviación se evita firmemente, hacer fallar un sistema de procesamiento de datos antes de que se escriban datos corruptos en el almacenamiento es muy valioso)
- Prácticas de ingeniería de confiabilidad de software - incluyendo revisiones, pruebas, verificación de modelos práctica (practical model checking)
- Identificación de dominios de fallos dependientes e independientes, y su impacto en la confiabilidad del sistema
- Análisis basado en mediciones de la confiabilidad del software - telemetría, monitorización y alertas, paneles de control, métricas de calificación de lanzamientos, etc.

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Demostrar la capacidad de aplicar múltiples métodos para desarrollar estimaciones de confiabilidad para un sistema de software [Demostrar]
2. Identificar métodos que conducirán a la realización de una arquitectura de software que logre un nivel especificado de confiabilidad [Analizar]
3. Identificar formas de aplicar redundancia para lograr tolerancia a fallos [Analizar]
4. Identificar dependencias de punto único de fallo (single-point-of-failure, SPF) en un diseño de sistema [Analizar]

2.13.21. SE/Métodos Formales

Temas:
Non Core

- Especificación formal de interfaces 1. Especificación de pre y post condiciones 2. Lenguajes formales para escribir y analizar pre y post condiciones.
- áreas de problemas bien atendidas por los métodos formales 1. Programación libre de bloqueos (*lock-free*), condiciones de carrera de datos 2. Sistemas asíncronos y distribuidos, bloqueos (*deadlock*), bloqueos vivos (*livelock*), etc.
- Comparación con otras herramientas y técnicas para la detección de defectos 1. Pruebas 2. *Fuzzing*
- Enfoques formales para el modelado y análisis de software 1. Verificadores de modelos (model checkers) 2. Buscadores de modelos (model finders)

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Describir el papel que las técnicas de especificación y análisis formal pueden desempeñar en el desarrollo de software complejo y comparar su uso como técnicas de validación y verificación con las pruebas [Describir]
2. Aplicar técnicas de especificación y análisis formal a diseños y programas de software de baja complejidad [Aplicar]

3. Explicar los beneficios y desventajas potenciales del uso de lenguajes de especificación formal [Explicar]

2.14. Seguridad (SEC)

La informática sustenta casi cada faceta de la infraestructura crítica moderna: transporte, comunicación, salud, educación, generación y distribución de energía, por nombrar algunas. Con los ataques desenfrenados y las violaciones de esta infraestructura, los graduados en ciencias de la computación tienen un papel importante en el diseño, implementación y operación de sistemas de software que sean robustos, seguros y que protejan la seguridad (security).

El área de conocimiento de Seguridad (SEC) se centra en desarrollar una mentalidad de seguridad en la ética general de los graduados en ciencias de la computación, de modo que la seguridad esté incorporada en todos sus productos de trabajo. Los estudiantes de informática necesitan aprender sobre las vulnerabilidades del sistema y comprender las amenazas contra los sistemas informáticos. La elección del título Seguridad fue intencional para servir como un término general de una sola palabra para esta área de conocimiento, que también incluye conceptos para apoyar la privacidad, la criptografía, los sistemas seguros, los datos seguros y el código seguro.

El área de conocimiento SEC se basa en conceptos compartidos omnipresentes en todas las demás áreas de CS2023. Identifica siete conceptos transversales de la ciberseguridad: confidencialidad, integridad, disponibilidad, evaluación de riesgos, pensamiento sistémico, pensamiento adversarial y pensamiento centrado en el humano. El séptimo concepto, pensamiento centrado en el humano, se añade a los seis conceptos transversales definidos originalmente en los Planes de Estudios de Ciberseguridad 2017 (CSEC2017). Esta adición refuerza a los estudiantes que los humanos también son un eslabón en la cadena general de seguridad. Los principios para proteger sistemas incluyen seguridad por diseño (security-by-design), privacidad por diseño (privacy-by-design), defensa en profundidad (defense-in-depth) y confianza cero (zero-trust).

Otro concepto es la noción de garantía (assurance), que es una atestación de que los mecanismos de seguridad deben cumplir con las políticas de seguridad que se han definido para los datos, procesos y sistemas. La garantía está ligada a los conceptos de verificación y validación. Las consideraciones de privacidad y seguridad de los datos se comparten con las áreas de conocimiento DM (aspectos técnicos) y SEP.

El área de conocimiento SEC se sitúa así sobre varias de las otras áreas de conocimiento de CS2023, al tiempo que incluye conceptos adicionales que no están presentes en esas áreas de conocimiento. CS2023 trata la seguridad como un componente crucial del conjunto de habilidades de cualquier graduado en CS, y las horas necesarias para la preparación en seguridad provienen de todas las otras 16 áreas de conocimiento de CS2023.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 2.14.1 Principios y Mentalidad de Seguridad (Pág. 110)	1	3
⭐ 2.14.2 Control de Acceso y Aplicaciones (Pág. 111)	1	3
⭐ 2.14.3 Seguridad en IA y Hardware (Pág. 111)	1	2
⭐ 2.14.4 Ética, Ley y Sociedad en Seguridad (Pág. 112)	1	2
⭐ 2.14.5 Privacidad y Seguridad por Diseño (Pág. 112)	1	1
⭐ 2.14.6 Inyección y Validación de Entrada (Pág. 113)	1	2
⭐ 2.14.7 Seguridad de Memoria y Tipos (Pág. 113)	1	1
⭐ 2.14.8 Flujo de Información y No Interferencia (Pág. 114)	1	1
⭐ 2.14.9 Análisis de Malware y Seguridad Avanzada (Pág. 114)	1	3
⭐ 2.14.10 Fundamentos de Criptografía (Pág. 114)	1	2
⭐ 2.14.11 Algoritmos Criptográficos (Pág. 115)	1	2
⭐ 2.14.12 Protocolos, Aplicaciones y Criptografía Avanzada (Pág. 115)	1	2
⭐ 2.14.13 Análisis de Amenazas e Ingeniería de Seguridad (Pág. 116)	1	4
⭐ 2.14.14 Diseño e Ingeniería de Controles de Seguridad (Pág. 116)	1	3
⭐ 2.14.15 Computación Confiable e Ingeniería de Privacidad (Pág. 117)	1	3
⭐ 2.14.16 Técnicas de Informática Forense (Pág. 117)		4
⭐ 2.14.17 Aspectos Legales y Manejo de Incidentes (Pág. 118)		3
⭐ 2.14.18 Gobierno, Gestión y Política de Seguridad (Pág. 118)		2
⭐ 2.14.19 Gestión de Riesgos y Datos (Pág. 119)		1

Cuadro 2.14: Lista de KUs del área de Seguridad.

2.14.1. SEC/Principios y Mentalidad de Seguridad (CS Core: 1 hr, KA Core: 3 hrs)

Esta unidad de conocimiento introduce conceptos fundamentales de seguridad, principios transversales y el desarrollo de una mentalidad de seguridad. Cubre confidencialidad, integridad, disponibilidad, evaluación de riesgos, pensamiento sistémico, pensamiento adversarial y pensamiento centrado en el humano. También incluye conceptos básicos de autenticación, autorización, vulnerabilidades, amenazas, superficies de ataque, denegación de servicio, principios de protección y el impacto de la IA en la seguridad y la privacidad.

Temas:

Core

- Desarrollar una mentalidad de seguridad que incorpore conceptos transversales: confidencialidad, integridad, disponibilidad, evaluación de riesgos, pensamiento sistémico, pensamiento adversarial, pensamiento centrado en el humano.
- Vulnerabilidades, amenazas, superficies de ataque y vectores de ataque Protección y Seguridad.
- Denegación de Servicio (DoS) y Denegación de Servicio Distribuida (DDoS) Protección y Seguridad.
- Principios y prácticas de protección, por ejemplo, privilegio mínimo, diseño abierto, valores predeterminados seguros ante fallos, defensa en profundidad y confianza cero; y cómo pueden implementarse Principios del Sistema Operativo, Protección y Seguridad, Prácticas de Codificación, Construcción a Escala y Proceso, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.
- Consideraciones de optimización entre seguridad, privacidad, rendimiento y otros objetivos de diseño Diseño de Sistemas, Prácticas de Desarrollo de Software, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.
- Conceptos de confianza (trust) y confiabilidad (trustworthiness).

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Evaluar un sistema para posibles ataques que pueda lanzar un adversario [Evaluar]
2. Diseñar y desarrollar enfoques para proteger un sistema de un conjunto de amenazas identificadas [Diseñar]
3. Describir cómo se puede evitar el daño a la privacidad del usuario [Describir]

4. Desarrollar un sistema que incorpore varios principios de seguridad y privacidad [Crear]

2.14.2. SEC/Control de Acceso y Aplicaciones (CS Core: 1 hr, KA Core: 3 hrs)

Temas:
Core

- Conceptos básicos de autenticación y autorización/control de acceso.
- Modelos de control de acceso (por ejemplo, discrecional, obligatorio, basado en roles y basado en atributos).
- Controles de seguridad.
- Aplicaciones de una mentalidad de seguridad: web, nube y dispositivos móviles Diseño del Sistema, Aspectos Comunes: Plataformas, APIs y Restricciones, Aspectos Comunes: Lenguajes de Plataforma y Patrones Web.
- Protección de sistemas embebidos y ciberfísicos Características y Arquitecturas Embebidas, Tiempo Real, Memoria y Verificación, Sistemas, Programación y Sensores.
- Principios de seguridad usable y computación centrada en el humano Diseño de Sistemas, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comparar los diferentes modelos de control de acceso en términos de funcionalidad y rendimiento [Comparar]

2.14.3. SEC/Seguridad en IA y Hardware (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Impacto de la IA en la seguridad y la privacidad: usar la IA para reforzar las defensas y abordar las capacidades adversariales aumentadas debido a la IA Aplicaciones e Impacto Social de la IA, Aplicaciones Adicionales e Impacto Social de la IA, Sociedad, ética y Profesión, Diseño de Sistemas.
- Seguridad y confianza en sistemas de IA/aprendizaje automático, por ejemplo, adecuados para el propósito, límites operativos éticos, fuentes de conocimiento autorizadas, datos de entrenamiento verificados, pruebas de evaluación del sistema repetibles, atestación del sistema, validación/certificación independiente; consecuencias no deseadas de: efecto adverso Aplicaciones e Impacto Social de la IA, Aplicaciones Adicionales e Impacto Social de la IA, Formulación del Aprendizaje Automático y Evaluación, Aprendizaje Profundo, RL y Ética, Introducción a la IA: Conceptos y Comportamiento Inteligente, Características del Problema y Naturaleza de los Agentes, Fundamentos de Aprendizaje Automático, Aprendizaje Automático: Datos, Representaciones y Evaluación, Redes Neuronales y Ética, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.
- Riesgos de seguridad en la construcción y operación de sistemas de IA/aprendizaje automático (por ejemplo, sesgo algorítmico, sesgo del corpus de conocimiento, sesgo del corpus de entrenamiento, violación de derechos de autor) Aplicaciones e Impacto Social de la IA, Aplicaciones Adicionales e Impacto Social de la IA, Formulación del Aprendizaje Automático y Evaluación, Aprendizaje Profundo, RL y Ética, Introducción a la IA: Conceptos y Comportamiento Inteligente, Características del Problema y Naturaleza de los Agentes, Fundamentos de Aprendizaje Automático, Aprendizaje Automático: Datos, Representaciones y Evaluación, Redes Neuronales y Ética.
- Consideraciones de hardware en seguridad, por ejemplo, principios de hardware seguro, arquitecturas de procesador seguro, aceleración criptográfica, compartimentación, interacción software-hardware Organización de Máquina a Nivel de Ensamblador, Representación de Datos a Nivel de Máquina, Rol y Propósito de los Sistemas Operativos.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Mostrar cómo un adversario podría usar algoritmos de aprendizaje automático para reducir la seguridad de un sistema [Analizar]

2. Mostrar cómo un desarrollador podría mejorar la seguridad de un sistema usando algoritmos de aprendizaje automático [Analizar]
3. Describir vulnerabilidades de hardware (especialmente de la CPU) que pueden impactar al software [Describir]

2.14.4. SEC/Ética, Ley y Sociedad en Seguridad (CS Core: 1 hr, KA Core: 2 hrs)

Esta unidad cubre los aspectos sociales, éticos, legales y profesionales de la seguridad y la privacidad, incluyendo seguridad por diseño, privacidad por diseño, hacking ético y las implicaciones de las tecnologías emergentes.

Temas:

Core

- Impactos sociales de las fallas en seguridad y privacidad Equidad Digital y Accesibilidad, Impacto Social e IA, Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.
- Aplicabilidad de leyes y regulaciones sobre seguridad y privacidad Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.
- Consideraciones éticas profesionales al diseñar sistemas seguros y mantener la privacidad; hacking ético Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad, Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.
- Implicaciones de seguridad y privacidad de actores maliciosos de IA/aprendizaje automático, por ejemplo, identificación de deep fakes Formulación del Aprendizaje Automático y Evaluación, Aprendizaje Profundo, RL y Ética, Introducción a la IA: Conceptos y Comportamiento Inteligente, Características del Problema y Naturaleza de los Agentes, Fundamentos de Aprendizaje Automático, Aprendizaje Automático: Datos, Representaciones y Evaluación, Redes Neuronales y Ética, Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.
- Impactos sociales de los dispositivos del Internet de las Cosas (IoT) y otras tecnologías emergentes en la seguridad y la privacidad Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Calcular el impacto de una falla de seguridad en un sistema dado [Calcular]
2. Construir un sistema que se ajuste a las leyes de seguridad [Crear]
3. Aplicar un conjunto de regulaciones de privacidad para diseñar un sistema que proteja la privacidad [Aplicar]
4. Evaluar las ramificaciones legales de un sistema que no se ajusta a las leyes y regulaciones aplicables [Evaluar]

2.14.5. SEC/Privacidad y Seguridad por Diseño (CS Core: 1 hr, KA Core: 1 hr)

Temas:

Core

- Principios y prácticas de privacidad Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.
- Seguridad por diseño (security by design) Diseño del Sistema, Seguridad del Sistema.
- Privacidad por diseño (privacy by design) e ingeniería de la privacidad Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Construir un sistema diseñado para evitar daños a la privacidad del usuario [Crear]

2.14.6. SEC/Inyección y Validación de Entrada (CS Core: 1 hr, KA Core: 2 hrs)

Esta unidad cubre las vulnerabilidades comunes de software, las prácticas de codificación segura, la programación defensiva y las herramientas para el análisis estático y dinámico para prevenir y mitigar fallos de seguridad.

Temas:
Core

- Vulnerabilidades y debilidades comunes.
- Inyección SQL y otros ataques de inyección.
- Técnicas de cross-site scripting y mitigaciones.
- Validación de entrada y sanitización de datos Protección y Seguridad, Conceptos y Construcciones Básicas de Programación, Prácticas de Programación: Pruebas, Depuración y Documentación, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar problemas subyacentes en ejemplos dados de una enumeración de debilidades comunes y explicar cómo pueden evitarse [Analizar]
2. Aplicar técnicas de validación de entrada y sanitización de datos para mejorar la seguridad de un programa [Aplicar]
3. Evaluar y prevenir ataques de inyección SQL en una aplicación de base de datos [Evaluar]
4. Evaluar y prevenir ataques de cross-site scripting contra un sitio web [Evaluar]
5. Construir un programa que pruebe todos los errores de manejo de entrada [Crear]

2.14.7. SEC/Seguridad de Memoria y Tipos (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Seguridad de tipos (type safety) y lenguajes seguros en tipos Teoría de Tipos Avanzada, Modelo de Ejecución y Memoria de Sistemas, Fundamentos de Sistemas de Tipos, Sistemas de Tipos: Polimorfismo y Tipado Complementario, Protección y Seguridad, Conceptos y Construcciones Básicas de Programación, Prácticas de Programación: Pruebas, Depuración y Documentación, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.
- Desbordamientos de búfer (buffer overflows), aplastamiento de pila (*stack smashing*) y desbordamientos de enteros Organización de Máquina a Nivel de Ensamblador, Modelo de Ejecución y Memoria de Sistemas, Protección y Seguridad.
- Problemas de seguridad debido a condiciones de carrera (race conditions) Computación Paralela y Distribuida, Evaluación.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir cómo la selección de un lenguaje de programación puede impactar la seguridad del sistema que se está construyendo [Describir]
2. Reescribir un programa en un lenguaje seguro en tipos (por ejemplo, Java o Rust) originalmente escrito en un lenguaje de programación no seguro (por ejemplo, C/C++) [Rediseñar]
3. Evaluar un programa en busca de posibles ataques de desbordamiento de búfer y reescribirlo para prevenir tales ataques [Evaluar]
4. Evaluar un conjunto de programas relacionados en busca de posibles condiciones de carrera y evitar que un adversario las explote [Evaluar]

2.14.8. SEC/Flujo de Información y No Interferencia (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Principios de no interferencia (noninterference) y no deducibilidad (nondeducibility).
- Prevención de ataques de flujo de información.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Construir un sistema que esté protegido de flujos de información no autorizados [Crear]

2.14.9. SEC/Análisis de Malware y Seguridad Avanzada (CS Core: 1 hr, KA Core: 3 hrs)

Temas:
Core

- Técnicas de seguridad ofensiva como defensa.
- Técnicas de detección de malware asistidas por IA.
- Ransomware: creación, prevención y mitigación.
- Uso seguro de componentes de terceros Prácticas de Codificación, Construcción a Escala y Proceso, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.
- Malware: variedades, creación, ingeniería inversa y defensa contra ellos Traducción y Ejecución de Lenguajes, Modelo de Ejecución y Memoria de Sistemas.
- Garantía (assurance): pruebas (incluyendo fuzzing y pruebas de penetración), verificación y validación Protección y Seguridad, Conceptos y Construcciones Básicas de Programación, Prácticas de Programación: Pruebas, Depuración y Documentación, Prácticas de Codificación, Construcción a Escala y Proceso, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.
- Análisis estático y dinámico Análisis de Programas y Analizadores, Protección y Seguridad, Evaluación, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.
- Compiladores seguros y generación de código seguro Traducción y Ejecución de Lenguajes, Comportamiento en Tiempo de Ejecución y Sistemas.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir diferentes tipos de software malicioso [Describir]
2. Explicar los riesgos del mal uso de interfaces con código de terceros y cómo usar correctamente el código de terceros [Explicar]
3. Discutir la necesidad de actualizar el software para corregir vulnerabilidades de seguridad y la gestión del ciclo de vida de la corrección [Debatir]
4. Aplicar herramientas estáticas y dinámicas para identificar fallos de programación [Aplicar]
5. Evaluar un sistema para detectar la existencia de malware y eliminarlo [Evaluar]
6. Implementar técnicas preventivas para reducir la ocurrencia de ransomware [Implementar]

2.14.10. SEC/Fundamentos de Criptografía (CS Core: 1 hr, KA Core: 2 hrs)

Esta unidad introduce principios, algoritmos, protocolos criptográficos y sus aplicaciones en la protección de datos y comunicaciones, incluyendo encriptación simétrica/asimétrica, funciones hash, firmas digitales y temas emergentes como la criptografía cuántica y poscuántica.

Temas:
Core

- Diferencias entre las vistas algorítmica, aplicada y matemática de la criptografía.

- Preliminares matemáticos: aritmética modular, algoritmo euclidiano, independencia probabilística, conceptos básicos de álgebra lineal, teoría de números, campos finitos, complejidad, análisis asintótico Matemáticas Discretas, álgebra Lineal.
- Criptografía básica: criptografía de clave simétrica y de clave pública Estructuras de Datos Fundamentales, Algoritmos Fundamentales, Algoritmos Avanzados, Matemáticas Discretas.
- Criptosistemas clásicos, como cifrados de desplazamiento, sustitución, transposición, libros de códigos y máquinas Matemáticas Discretas.
- Principio de Kerckhoff y uso de bibliotecas evaluadas (vetted) Prácticas de Codificación, Construcción a Escala y Proceso.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar el papel de la criptografía en el soporte de la seguridad y la privacidad [Explicar]
2. Discutir los riesgos de inventar los propios métodos criptográficos [Debatir]
3. Discutir la importancia de los números primos en criptografía y explicar su uso en algoritmos criptográficos [Debatir]
4. Implementar y criptoanalizar cifrados clásicos [Implementar]

2.14.11. SEC/Algoritmos Criptográficos (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Bloques de construcción criptográficos básicos, incluyendo encriptación simétrica, encriptación asimétrica, funciones hash y autenticación de mensajes Matemáticas Discretas.
- Matemáticas adicionales: primalidad, factorización y criptografía de curva elíptica Matemáticas Discretas.
- Criptosistemas de clave privada: redes de sustitución-permutación, criptoanálisis lineal, criptoanálisis diferencial, DES y AES Matemáticas Discretas, Seguridad de Red.
- Criptosistemas de clave pública: Diffie-Hellman y RSA Matemáticas Discretas.
- Integridad y autenticación de datos: funciones hash y firmas digitales Seguridad y Privacidad de Datos, Matemáticas Discretas.
- Capacidades del atacante: ataque de mensaje elegido (para firmas), ataques de cumpleaños (birthday attacks), ataques de canal lateral y ataques de inyección de fallos Seguridad de Red.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir cómo se pueden gestionar las claves criptográficas de forma segura [Describir]
2. Comparar el rendimiento en espacio y tiempo de un conjunto dado de métodos criptográficos [Comparar]
3. Discutir cómo funcionan los criptosistemas modernos de clave privada y formas de criptoanalizarlos [Debatir]
4. Discutir cómo funcionan los criptosistemas modernos de clave pública y formas de criptoanalizarlos [Debatir]
5. Comparar diferentes algoritmos criptográficos en términos de seguridad [Comparar]
6. Explicar protocolos de intercambio de claves y mostrar enfoques para reducir su fallo [Explicar]

2.14.12. SEC/Protocolos, Aplicaciones y Criptografía Avanzada (CS Core: 1 hr, KA Core: 2 hrs)

Temas:
Core

- Uso de la criptografía en aplicaciones del mundo real, por ejemplo, dinero electrónico, canales seguros entre clientes y servidores, correo electrónico seguro, autenticación de entidades, emparejamiento de dispositivos, esteganografía y sistemas de votación Procesamiento de Imágenes, Seguridad de Red.
- Protocolos criptográficos: autenticación de desafío-respuesta, protocolos de conocimiento cero, compromisos (commitment), transferencia inconsciente, cómputo seguro de dos o múltiples partes, funciones hash, compartición de secretos y aplicaciones Matemáticas Discretas.

- Criptografía cuántica; criptografía poscuántica/resistente a la cuántica Estructuras de Datos Fundamentales, Algoritmos Fundamentales, Algoritmos Avanzados, Matemáticas Discretas.
- Blockchain y criptomonedas Matemáticas Discretas, Comunicación.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir aplicaciones reales de primitivas y protocolos criptográficos [Describir]
2. Discutir cómo funciona la criptografía cuántica y el impacto de la computación cuántica en los algoritmos criptográficos [Debatir]

2.14.13. SEC/Análisis de Amenazas e Ingeniería de Seguridad (CS Core: 1 hr, KA Core: 4 hrs)

Esta unidad cubre enfoques sistemáticos para la ingeniería de seguridad, incluyendo modelado de amenazas, análisis de riesgos, diseño de seguridad, dominios de ataque y computación confiable (trustworthy computing).

Temas:
Core

- Objetivos de la ingeniería de seguridad: construir sistemas que permanezcan confiables a pesar de errores, accidentes o adversarios maliciosos Prácticas de Codificación, Construcción a Escala y Proceso, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.
- Análisis de problemas y análisis situacional para abordar la seguridad del sistema Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.
- Análisis de seguridad, que cubre análisis de requisitos de seguridad; análisis de controles de seguridad; análisis de amenazas; y análisis de vulnerabilidades Análisis de Programas y Analizadores, Evaluación.
- Dominios de ataque de seguridad y superficies de ataque, por ejemplo, comunicaciones y redes, hardware, físico, ingeniería social, software y cadena de suministro Seguridad de Red.
- Modos, técnicas y tácticas de ataque de seguridad, por ejemplo, abuso de autenticación; fuerza bruta; manipulación de búfer; inyección de código; inserción de contenido; denegación de servicio; escucha; omisión de función; suplantación; ataque de integridad; intercepción; phishing; análisis de protocolo; abuso de privilegio; suplantación (spoofing); e inyección de tráfico Seguridad de Red, Protección y Seguridad, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Crear un modelo de amenazas para un sistema o diseño de sistema [Crear]
2. Aplicar análisis situacional para desarrollar soluciones seguras bajo un escenario especificado [Aplicar]
3. Identificar y mitigar vulnerabilidades y debilidades de seguridad en un sistema [Analizar]

2.14.14. SEC/Diseño e Ingeniería de Controles de Seguridad (CS Core: 1 hr, KA Core: 3 hrs)

Temas:
Core

- Análisis de compensaciones (tradeoff) de ingeniería basado en tiempo, costo, tolerancia al riesgo, aceptación de riesgo, retorno de la inversión, etc. Evaluación, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.
- Diseño e ingeniería de seguridad, incluyendo requisitos funcionales, subsistemas de seguridad, protección de información, pruebas de seguridad, evaluación de seguridad y evaluación (*assess-*

ment) Evaluación, Pruebas de Rendimiento y Evaluación Comparativa, Ingeniería de Requisitos, Evolución y Estimación de Requisitos, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.
- Diseño y desarrollo de sistemas ciberfísicos.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Evaluar un escenario dado para análisis de compensaciones (tradeoff) para el rendimiento del sistema, evaluación de riesgos y costos [Evaluar]
2. Diseñar un conjunto de controles de seguridad técnicos, contramedidas y protecciones de información para cumplir con los requisitos de seguridad y los objetivos de seguridad de un sistema [Diseñar]
3. Evaluar la efectividad de las funciones de seguridad, controles técnicos y componentes para un sistema [Evaluar]

2.14.15. SEC/Computación Confiable e Ingeniería de Privacidad (CS Core: 1 hr, KA Core: 3 hrs)

Temas:
Core

- Objetivos de la ingeniería de la privacidad: construir sistemas que diseñen, implementen y desplieguen características y controles de privacidad Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad.
- Atestación (attestation) de productos de software con respecto a su especificación y adaptabilidad Pruebas de Rendimiento y Evaluación Comparativa, Ingeniería de Requisitos, Evolución y Estimación de Requisitos, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.
- Consideraciones para la computación confiable (trustworthy computing), por ejemplo, empaquetado resistente a la manipulación, arranque confiable (trusted boot), núcleo confiable (*trusted kernel*), raíz de confianza de hardware (hardware root of trust), firma y verificación de software, criptografía basada en hardware, virtualización y contenedores Prácticas de Codificación, Construcción a Escala y Proceso, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Evaluar y predecir comportamientos emergentes en áreas como Ciencia de Datos, IA y Aprendizaje Automático [Evaluar]

2.14.16. SEC/Técnicas de Informática Forense (KA Core: 4 hrs)

Esta unidad cubre principios, metodologías y herramientas para la informática forense (digital forensics), incluyendo recolección de evidencia, análisis, respuesta a incidentes y consideraciones legales.
Temas:
Core

- Principios y metodologías básicas para la informática forense.
- Diseño de sistemas para informática forense.
- Informática forense en diferentes situaciones: sistemas operativos, sistemas de archivos, informática forense de aplicaciones, informática forense web, informática forense de red, informática forense de dispositivos móviles, uso de auditoría de bases de datos Seguridad de Red.
- Ataques a la informática forense y prevención de tales ataques.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar qué es una investigación digital y cómo se puede implementar Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad [Explicar]
2. Diseñar e implementar software para apoyar la informática forense [Diseñar]

3. Extraer datos de un disco duro para cumplir con la ley Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad [Extraer]
4. Recuperar datos basados en un término de búsqueda dado desde un sistema del que se ha hecho una imagen (imaged) [Analizar]
5. Reconstruir datos y eventos a partir de un historial de aplicación, o un artefacto web, o una base de datos en la nube, o un dispositivo móvil. Plataformas Móviles: Capacidades y Computación Asíncrona, Plataformas Móviles: Arquitectura y Restricciones, Plataformas Móviles: Fundamentos del Desarrollo, Arquitectura Cloud y Gestión de Datos, Frontend, Frameworks y Estándares Web [Aplicar]
6. Capturar y analizar tráfico de red. Seguridad de Red [Analizar]
7. Desarrollar enfoques para abordar los desafíos asociados con la informática forense de dispositivos móviles [Crear]
8. Aplicar herramientas de informática forense para investigar violaciones de seguridad [Aplicar]
9. Identificar y mitigar métodos anti-forenses [Analizar]

2.14.17. SEC/Aspectos Legales y Manejo de Incidentes (KA Core: 3 hrs)

Temas:
Core

- Procesos de manejo de incidentes.
- Reglas de la evidencia - conceptos generales y diferencias entre jurisdicciones Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.
- Problemas legales: protección y gestión de evidencia digital, cadena de custodia, reporte, actuar como testigo experto Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir los requisitos legales para el uso de datos incautados y su uso. Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad [Describir]
2. Describir e implementar una cadena de custodia de extremo a extremo desde la incautación inicial de evidencia digital hasta la disposición de la evidencia. Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad [Describir]
3. Discutir las responsabilidades y obligaciones profesionales de una persona cuando testifica como experto en informática forense Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo [Debatir]

2.14.18. SEC/Gobierno, Gestión y Política de Seguridad (KA Core: 2 hrs)

Esta unidad cubre el gobierno organizacional de seguridad, la gestión de riesgos, el desarrollo de políticas, la gestión del ciclo de vida de los datos y la protección de activos críticos.

Temas:
Core

- Proteger activos críticos de amenazas.
- Gobierno de seguridad (security governance): objetivos organizacionales y evaluación general de riesgos.
- Gestión de seguridad (security management): lograr y mantener niveles apropiados de confidencialidad, integridad, disponibilidad, responsabilidad (accountability), autenticidad y confiabilidad Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.
- Política de seguridad (security policy): políticas organizacionales, políticas específicas de temas, políticas específicas de sistemas.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir activos críticos y cómo pueden protegerse [Describir]
2. Diferenciar entre gobierno de seguridad, gestión y controles, dando ejemplos de cada uno [Diferenciar]
3. Describir un control técnico e implementarlo para mitigar amenazas específicas [Describir]

2.14.19. SEC/Gestión de Riesgos y Datos (KA Core: 1 hr)

Temas:
Core

- Enfoques para identificar y mitigar riesgos para la infraestructura informática.
- Políticas de gestión del ciclo de vida de los datos: recolección de datos, copias de seguridad y retención; almacenamiento y servicios en la nube; divulgación de violaciones (breach disclosure) Seguridad y Privacidad de Datos.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar y evaluar el riesgo de que programas y aplicaciones de base de datos causen violaciones [Analizar]
2. Diseñar e implementar estrategias de copia de seguridad apropiadas que se ajusten a una política dada [Diseñar]
3. Discutir una política de divulgación de violaciones basada en requisitos legales e implementar la política [Debatir]
4. Identificar los riesgos y beneficios de la externalización a la nube [Analizar]

2.15. Sociedad, ética y la Profesión (SEP)

El Código de ética y Conducta Profesional de la ACM establece: ”Las acciones de los profesionales de la informática cambian el mundo. Para actuar de manera responsable, deben reflexionar sobre los impactos más amplios de su trabajo, apoyando consistentemente el bien público”. El Código de ética del IEEE comienza reconociendo ”la importancia de nuestras tecnologías en la afectación de la calidad de vida en todo el mundo”. El Código de ética Profesional y Conducta de la AAAI comienza con ”Las acciones de los profesionales de la informática, y en particular, los profesionales de la IA, cambian el mundo. Para actuar de manera responsable, deben reflexionar sobre los impactos más amplios de su trabajo, apoyando consistentemente el bien público”.

Si bien los problemas técnicos dominan el plan de estudios de informática, no constituyen un programa educativo completo en un contexto más amplio. Es más evidente hoy que nunca que los estudiantes también deben estar expuestos al contexto social más amplio de la informática para desarrollar una comprensión de los problemas y responsabilidades sociales, éticos, legales y profesionales críticos y relevantes. Esta necesidad de incorporar el estudio de estos problemas no técnicos en el plan de estudios de la ACM se reconoció formalmente en 1991.

Los educadores en ciencias de la computación pueden optar por impartir el material en esta área de conocimiento dentro de los contextos de cursos técnicos y teóricos tradicionales, en cursos dedicados y como parte de cursos de proyecto integrador, proyecto y práctica profesional. El material en esta área de conocimiento se cubre mejor mediante una combinación de todo lo anterior.

El Código de ética y Conducta Profesional de la ACM, el Código de ética del IEEE y el Código de ética Profesional y Conducta de la AAAI proporcionan orientación que sirve como base para la conducta de todos los profesionales de la informática en su trabajo. El Código de la ACM enfatiza que el razonamiento ético no es un algoritmo a seguir, y se espera que los profesionales de la informática consideren cómo su trabajo impacta el bien público como consideración principal.

Un estudiante de ciencias de la computación no debe graduarse sin comprender cómo la sociedad y la ética influyen en la profesión de la informática. Tampoco debería ser posible completar un título en ciencias de la computación sin aprender cómo los profesionales de la informática influyen en la sociedad, las consideraciones éticas involucradas en dar forma a ese impacto, y el papel del estudiante convertido en graduado en estas relaciones, tanto como profesionales de la informática como miembros de la sociedad.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 2.15.1 Impacto Social e IA (Pág. 120)	2	1
⭐ 2.15.2 Equidad Digital y Accesibilidad (Pág. 121)	2	1
⭐ 2.15.3 Fundamentos y Marcos del Análisis Ético (Pág. 121)	2	1
⭐ 2.15.4 Métodos y Herramientas de Análisis Ético (Pág. 122)	1	1
⭐ 2.15.5 Valores, Naturaleza y Actualización Profesional (Pág. 122)	1	1
⭐ 2.15.6 Códigos de Ética y Responsabilidad Legal (Pág. 123)	1	1
⭐ 2.15.7 Responsabilidad Profesional y Política Pública (Pág. 123)	1	1
⭐ 2.15.8 Disidencia Ética y Contexto Cultural (Pág. 124)	1	1
⭐ 2.15.9 Ética en el Lugar de Trabajo (Pág. 124)	1	1
⭐ 2.15.10 Fundamentos de la Propiedad Intelectual (Pág. 124)	1	1
⭐ 2.15.11 Licencias, Plagio y Autoría (Pág. 125)	1	1
⭐ 2.15.12 Derechos de PI Digital y Derechos de Autor (Pág. 125)	1	1
⭐ 2.15.13 Open Source, Piratería y DRM (Pág. 125)	1	1
⭐ 2.15.14 Datos, Vigilancia y Tecnología de Privacidad (Pág. 126)	1	1
⭐ 2.15.15 Fundamentos Filosóficos y Legislación de Privacidad (Pág. 126)	1	1
⭐ 2.15.16 Libertades Civiles y Expresión (Pág. 127)	1	1
⭐ 2.15.17 Comunicación Técnica y Profesional (Pág. 127)	1	1
⭐ 2.15.18 Comunicación en Equipo y Cultura (Pág. 127)	1	1
⭐ 2.15.19 Huella Ambiental de los Sistemas Informáticos (Pág. 128)	1	1
⭐ 2.15.20 Efectos Sistémicos y Contexto Social (Pág. 128)	1	1
⭐ 2.15.21 Diseño Sostenible y Computación Ubicua (Pág. 129)	1	1
⭐ 2.15.22 Fundamentos e Historia Temprana de la Informática (Pág. 129)	1	1
⭐ 2.15.23 Historia Moderna: PC, Internet e IA (Pág. 130)	1	1
⭐ 2.15.24 Modelos y Estrategias Económicas (Pág. 130)		1
⭐ 2.15.25 Automatización, IA y Ética Económica (Pág. 131)		1
⭐ 2.15.26 Delitos y Ataques Informáticos (Pág. 131)	1	1
⭐ 2.15.27 Leyes, Políticas y Responsabilidad en Seguridad (Pág. 131)	1	1
⭐ 2.15.28 Identidad y Diversidad en la Informática (Pág. 132)	1	1
⭐ 2.15.29 Contexto Histórico e Intercultural de DEIA (Pág. 132)	1	1
⭐ 2.15.30 Inclusión y Prácticas en Informática (Pág. 133)	1	1
⭐ 2.15.31 Diseño Accesible (Pág. 133)	1	1

Cuadro 2.15: Lista de KUs del área de Sociedad, ética y la Profesión.

2.15.1. SEP/Impacto Social e IA (CS Core: 2 hrs, KA Core: 1 hr)

El Contexto Social proporciona la base para todas las demás unidades de conocimiento en SEP, particularmente ética Profesional. Cubre las implicaciones sociales de la informática en un mundo hiperconectado, el impacto de las aplicaciones informáticas en el bienestar y la seguridad individual, las consecuencias de involucrar tecnologías informáticas en la vida cívica y cómo los déficits en diversidad y accesibilidad en la informática afectan a la sociedad.

Temas:

Core

- Implicaciones sociales (por ejemplo, ideologías políticas y culturales) en un mundo hiperconectado donde las capacidades y el impacto de las redes sociales, la inteligencia artificial y la informática en general están evolucionando rápidamente.
- Impacto de las aplicaciones informáticas (por ejemplo, redes sociales, aplicaciones de inteligencia artificial) en el bienestar individual y la seguridad de todo tipo (por ejemplo, física, emocional, económica).
- Consecuencias de involucrar tecnologías informáticas, particularmente inteligencia artificial, tecnologías biométricas y sistemas de toma de decisiones algorítmicos, en la vida cívica (por ejemplo, tecnología de reconocimiento facial, etiquetas biométricas, algoritmos de distribución de recursos, software policial) y cómo la agencia y supervisión humana son cruciales.

- Crecimiento y control de internet, datos, informática e inteligencia artificial.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir las diferentes formas en que la tecnología informática (redes, computación móvil, inteligencia artificial) media la interacción social a nivel personal y colectivo [Describir]
2. Interpretar el contexto social de un diseño dado y su implementación [Interpretar]
3. Analizar la eficacia de un diseño e implementación dados utilizando datos empíricos [Analizar]
4. Comprender las implicaciones del uso de la tecnología (por ejemplo, redes sociales) para diferentes identidades, culturas y comunidades [Explicar]
5. Describir el papel de internet en facilitar la comunicación entre ciudadanos, gobiernos y entre sí [Describir]
6. Analizar los efectos de la dependencia de la informática en la implementación de la democracia (por ejemplo, entrega de servicios sociales, votación electrónica) [Analizar]
7. Describir el impacto de la falta de representación apropiada de personas de poblaciones históricamente minorizadas en la profesión de la informática (por ejemplo, cultura de la industria, diversidad de productos) [Describir]

2.15.2. SEP/Equidad Digital y Accesibilidad (CS Core: 2 hrs, KA Core: 1 hr)

Temas:
Core

- Cómo los déficits en diversidad y accesibilidad en la informática afectan a la sociedad y qué pasos se pueden tomar para mejorar la equidad en la informática.
- A menudo referido como la brecha digital, diferencias en el acceso a recursos de tecnología digital y sus ramificaciones resultantes para género, clase, etnia, geografía y/o países en desarrollo, incluyendo la consideración de la responsabilidad hacia aquellos que podrían ser menos ricos, bajo amenaza o que lucharían para que sus voces sean escuchadas.
- Problemas de accesibilidad, incluyendo requisitos legales como las Pautas de Accesibilidad al Contenido Web (www.w3.org/TR/WCAG21).
- Computación consciente del contexto (context-aware computing).

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar los supuestos y valores de los desarrolladores incorporados en el diseño de hardware y software, especialmente en lo que respecta a la usabilidad para poblaciones diversas, incluidos los desatendidos y aquellos con discapacidades [Analizar]
2. Discutir las implicaciones de la conciencia del contexto en sistemas de computación ubicua [Debatir]
3. Expresar cómo el acceso a internet y a las tecnologías informáticas afecta a diferentes sociedades [Explicar]
4. Identificar por qué/cómo el acceso a internet puede ser visto como un derecho humano [Analizar]

2.15.3. SEP/Fundamentos y Marcos del Análisis Ético (CS Core: 2 hrs, KA Core: 1 hr)

Las teorías y principios éticos son los fundamentos del análisis ético porque son los puntos de vista que pueden proporcionar orientación a lo largo del camino hacia una decisión. Cada teoría enfatiza diferentes supuestos y métodos para determinar la ética de una acción dada. Es importante que los estudiantes reconozcan que las decisiones en diferentes contextos pueden requerir diferentes teorías éticas (incluyendo combinaciones) para llegar a resultados éticamente aceptables, y lo que constituye 'aceptable' depende de una variedad de factores como el contexto cultural. Aplicar métodos para el análisis ético requiere tanto una comprensión de los principios y supuestos subyacentes que guían una herramienta dada como una conciencia del contexto social para esa decisión. Los marcos éticos tradicionales proporcionados por la filosofía occidental pueden ser útiles, pero no son inclusivos. Se debe hacer un esfuerzo para incluir perspectivas éticas decoloniales, indígenas e históricamente marginadas siempre que sea posible.

Temas:
Core

- Evitar falacias y tergiversaciones en la argumentación.
- Teorías éticas y toma de decisiones (marcos filosóficos y sociales, por ejemplo [1]).
- Reconocimiento del papel que juega la cultura en nuestra comprensión, adopción, diseño y uso de la tecnología informática.
- Por qué la ética es importante en la informática y cómo la ética es similar y diferente de las leyes y normas sociales.
- Introducción a los marcos éticos (por ejemplo, consecuencialismo como el utilitarismo, no consecuencialismo como el deber, derechos o justicia, centrado en el agente como la virtud o el feminismo, contractualismo, ética del cuidado) y su uso para analizar un dilema ético.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir cómo un contexto cultural dado impacta la toma de decisiones [Describir]
2. Expresar el uso de ejemplos y analogías en la argumentación ética [Analizar]
3. Analizar (y evitar) falacias lógicas básicas en un argumento [Analizar]
4. Analizar un argumento para identificar premisas y conclusión [Analizar]
5. Evaluar cómo y por qué la ética es tan importante en la informática y cómo se relaciona con las normas culturales, valores y leyes [Evaluar]
6. Justificar una decisión tomada sobre bases éticas [Justificar]
7. Discutir las ventajas y desventajas de los marcos éticos tradicionales [Debatir]
8. Analizar dilemas éticos relacionados con la creación y uso de tecnología desde múltiples perspectivas utilizando marcos éticos [Analizar]

2.15.4. SEP/Métodos y Herramientas de Análisis Ético (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Listas de verificación (checklists) profesionales.
- Rúbricas de evaluación.
- Análisis de partes interesadas (stakeholder analysis).
- Teoría del punto de vista (standpoint theory).

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Distinguir todas las posiciones de las partes interesadas en relación con su contexto cultural en una situación dada [Distinguir]
2. Analizar el potencial de introducir o perpetuar deuda ética (consideración diferida de impactos o implicaciones éticas) en decisiones técnicas [Analizar]

2.15.5. SEP/Valores, Naturaleza y Actualización Profesional (CS Core: 1 hr, KA Core: 1 hr)

Los profesionales de la informática tienen responsabilidades que van más allá de sus habilidades técnicas. Esta unidad explora los valores comunitarios y legales que guían la profesión, la naturaleza del profesionalismo —incluyendo el cuidado, la disciplina y la tutoría— y la responsabilidad del profesional de mantenerse actualizado en un campo en constante evolución.

Temas:
Core

- Valores de la comunidad y las leyes por las que vivimos.
- La naturaleza de ser un profesional, incluyendo cuidado, atención, disciplina, responsabilidad fiduciaria y tutoría (mentoring).
- Mantenerse actualizado como profesional de la informática en términos de familiaridad, herramientas, habilidades, marcos legales y profesionales, así como la capacidad y responsabilidad de autoevaluarse y progresar en el campo de la informática.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar problemas éticos que surgen en el diseño de software, prácticas de desarrollo y despliegue de software [Analizar]
2. Discutir cómo abordar problemas éticos en situaciones específicas [Debatir]
3. Describir los mecanismos que típicamente existen para que un profesional se mantenga actualizado en asuntos éticos [Describir]

2.15.6. SEP/Códigos de Ética y Responsabilidad Legal (CS Core: 1 hr, KA Core: 1 hr)

Los códigos de ética profesional, como los de la ACM, IEEE y AAAI, definen estándares de conducta y responsabilidad en la informática. Esta unidad examina la certificación profesional, los códigos de ética relevantes, y las dimensiones legales y éticas de la responsabilidad del software, incluyendo la corrección, confiabilidad y seguridad.

Temas:
Core

- Certificación profesional, códigos de ética, conducta y práctica, como los de la ACM, IEEE, AAAI y otras sociedades internacionales.
- Responsabilidad (accountability), obligación (responsibility) y responsabilidad legal (liability) (por ejemplo, corrección del software, confiabilidad y seguridad, garantía, negligencia, responsabilidad estricta, enfoques éticos para la divulgación de vulnerabilidades de seguridad), incluyendo si un producto/servicio debe construirse, no solo hacerlo porque es técnicamente posible.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Expresar la responsabilidad ética de garantizar la corrección, confiabilidad y seguridad del software, incluyendo de dónde surge esta responsabilidad (por ejemplo, Códigos de ética de ACM/IEEE/AAAI, leyes y regulaciones, políticas organizacionales) [Analizar]
2. Describir las fortalezas y debilidades de los códigos profesionales relevantes como expresiones de ser un profesional y guías para la toma de decisiones [Describir]

2.15.7. SEP/Responsabilidad Profesional y Política Pública (CS Core: 1 hr, KA Core: 1 hr)

Los profesionales de la informática operan dentro de marcos sociales, legales y culturales complejos. Esta unidad examina las teorías que explican cómo los seres humanos crean y utilizan la tecnología —incluyendo enfoques decoloniales y de justicia social— y explora las responsabilidades del profesional para reconocer, reportar y mantener conciencia de las consecuencias éticas de su trabajo y su participación en la política pública.

Temas:
Core

- Introducción a teorías que describen la creación y uso humano de la tecnología, incluyendo instrumentalismo, sociología de los sistemas tecnológicos, justicia de la discapacidad (disability justice), tesis de la neutralidad, pragmatismo y modelos decoloniales, incluyendo desarrollar y usar tecnología para corregir errores y hacer el bien.
- Estrategias para reconocer y reportar diseños, sistemas, software y conducta profesional (o sus resultados) que puedan violar la ley o códigos profesionales de ética.
- El papel del profesional de la informática y las sociedades profesionales en la política pública.
- Mantener la conciencia de las consecuencias.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Analizar un problema global de informática, observando el papel de los profesionales y funcionarios gubernamentales en la gestión de este problema [Analizar]
2. Describir formas en que los profesionales y las organizaciones profesionales pueden contribuir a la política pública [Describir]
3. Describir las consecuencias del comportamiento profesional inapropiado [Describir]

2.15.8. SEP/Disidencia Ética y Contexto Cultural (CS Core: 1 hr, KA Core: 1 hr)

La disidencia ética es un componente crítico de la conducta profesional responsable. Esta unidad aborda el whistleblowing como mecanismo de rendición de cuentas, y explora cómo el contexto cultural regional moldea la comprensión y respuesta ante dilemas éticos en la profesión informática.

Temas:

Core

- Disidencia ética y denuncia de irregularidades (whistleblowing).
- La relación entre la cultura regional y los dilemas éticos.

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Estar familiarizado con la denuncia de irregularidades (*whistleblowing*) y tener acceso a conocimiento para guiarlo a través de un incidente [Evaluar (valorar)]
2. Identificar ejemplos de cómo la cultura regional interactúa con dilemas éticos [Analizar]

2.15.9. SEP/Ética en el Lugar de Trabajo (CS Core: 1 hr, KA Core: 1 hr)

Temas:

Core

- Tratar con acoso y discriminación.
- Formas de acreditación profesional.
- Ergonomía y entornos informáticos saludables.
- Tiempo de comercialización (time-to-market) y consideraciones de costo versus estándares profesionales de calidad.

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Describir formas de acoso y discriminación y vías de asistencia [Describir]
2. Evaluar varias formas de acreditación profesional [Evaluar (valorar)]
3. Establecer la relación entre la ergonomía en los entornos informáticos y la salud de las personas [Enunciar/Establecer]
4. Describir problemas asociados con el impulso de las industrias para centrarse en el tiempo de comercialización (time-to-market) versus la aplicación de estándares profesionales de calidad [Describir]

2.15.10. SEP/Fundamentos de la Propiedad Intelectual (CS Core: 1 hr, KA Core: 1 hr)

La propiedad intelectual se refiere a un rango de derechos intangibles de propiedad sobre cualquier producto del intelecto humano, como un programa de software. Las leyes, que varían según la localidad, proporcionan diferentes métodos para proteger estos derechos de propiedad basados en su tipo. Idealmente, las leyes de propiedad intelectual equilibran los intereses de los creadores y usuarios de la propiedad. Hay cuatro tipos de derechos de propiedad intelectual relevantes para el software: patentes, derechos de autor (copyrights), secretos comerciales y marcas registradas. Además, los derechos de propiedad a menudo están protegidos por licencias de usuario. Cada uno proporciona un tipo diferente de protección legal.

Temas:

Core

- Derechos de propiedad intelectual.
- Fundamentos legales para la protección de la propiedad intelectual.
- Fundamentos filosóficos de la propiedad intelectual.
- Formas de propiedad intelectual (por ejemplo, derechos de autor, patentes, secretos comerciales, marcas registradas) y los derechos que protegen.

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Defender los usos legales y éticos de materiales con derechos de autor [Defender]

2. Discutir las bases filosóficas de la propiedad intelectual en un contexto apropiado (por ejemplo, país) [Debatir]
3. Contrastar las protecciones y obligaciones de los derechos de autor, patentes, secretos comerciales y marcas registradas [Contrastar]
4. Describir la justificación para la protección legal de la propiedad intelectual en el contexto apropiado (por ejemplo, país) [Describir]

2.15.11. SEP/Licencias, Plagio y Autoría (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Licencias de software comunes (por ejemplo, MIT, GPL y sus variantes, Apache, Mozilla, Creative Commons).
- Plagio y autoría.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Seleccionar una licencia de software apropiada para un proyecto dado [Analizar]
2. Interpretar la intención y implementación de las licencias de software [Interpretar]
3. Analizar los problemas éticos inherentes en varios mecanismos de detección de plagio [Analizar]
4. Identificar múltiples formas de plagio más allá de la copia textual de texto o software (por ejemplo, paráfrasis intencional, tergiversación de autoría y atribución inadecuada) [Analizar]

2.15.12. SEP/Derechos de PI Digital y Derechos de Autor (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Propiedad intelectual digital intangible (IDIP).
- Limitaciones a las protecciones de derechos de autor, incluyendo uso justo (fair use) y la doctrina de la primera venta (first sale doctrine).
- Leyes y tratados de propiedad intelectual que impactan la aplicación de los derechos de autor.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir y criticar la legislación y los precedentes destinados a las infracciones de derechos de autor digitales [Describir]
2. Identificar ejemplos contemporáneos de propiedad intelectual digital intangible [Analizar]
3. Discutir si un uso de material con derechos de autor es probable que sea uso justo (fair use) [Debatir]
4. Analizar el uso de obras con derechos de autor bajo los conceptos de uso justo (fair use) y la doctrina de la primera venta (first sale doctrine) [Analizar]
5. Discutir la naturaleza de las leyes anti-elusión (anti-circumvention) en el contexto de la protección de derechos de autor [Debatir]

2.15.13. SEP/Open Source, Piratería y DRM (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Piratería de software y métodos técnicos para hacer cumplir los derechos de propiedad intelectual, como la gestión de derechos digitales (digital rights management) y el software de código cerrado como secreto comercial.
- Fundamentos morales y legales del movimiento de código abierto.
- Sistemas que usan datos de otros (por ejemplo, modelos de lenguaje grande).

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Distinguir los problemas conflictivos involucrados en la obtención de patentes de software [Distinguir]

2. Identificar los objetivos del movimiento de código abierto y su impacto en campos más allá de la informática, como el movimiento del derecho a reparar (right-to-repair) [Analizar]
3. Resumir la naturaleza global de la piratería de software [Resumir]
4. Criticar el uso de medidas técnicas de gestión de derechos digitales (por ejemplo, encriptación, marcas de agua, restricciones de copia y bloqueos regionales) desde las perspectivas de múltiples partes interesadas [Criticar]

2.15.14. SEP/Datos, Vigilancia y Tecnología de Privacidad (CS Core: 1 hr, KA Core: 1 hr)

La recopilación masiva de datos en sistemas transaccionales, plataformas de vigilancia, computación en la nube e inteligencia artificial plantea desafíos fundamentales para la privacidad. Esta unidad examina las implicaciones de estos sistemas, las concepciones de anonimato y seudonimidad, y las soluciones tecnológicas para proteger la privacidad, como la encriptación de extremo a extremo y la privacidad diferencial.

Temas:

Core

- Implicaciones de privacidad de la recopilación generalizada de datos, incluyendo pero no limitado a bases de datos transaccionales, almacenes de datos, sistemas de vigilancia, computación en la nube e inteligencia artificial.
- Concepciones de anonimato, seudonimidad e identidad.
- Soluciones basadas en tecnología para la protección de la privacidad (por ejemplo, encriptación de extremo a extremo y privacidad diferencial).

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Evaluar soluciones a las amenazas a la privacidad en bases de datos transaccionales y almacenes de datos [Evaluar]
2. Describir el papel de la recopilación de datos en la implementación de sistemas de vigilancia generalizada (por ejemplo, RFID, reconocimiento facial, cobro de peajes, computación móvil) [Describir]
3. Distinguir los conceptos y objetivos del anonimato y la seudonimidad [Distinguir]
4. Describir las ramificaciones de las protecciones de privacidad basadas en tecnología, incluyendo la privacidad diferencial y la encriptación de extremo a extremo [Describir]

2.15.15. SEP/Fundamentos Filosóficos y Legislación de Privacidad (CS Core: 1 hr, KA Core: 1 hr)

El derecho a la privacidad tiene raíces filosóficas y legales profundas que varían según el contexto cultural y jurídico. Esta unidad examina las concepciones filosóficas y legales de la privacidad, los fundamentos legales internacionales como el GDPR, y la legislación sectorial como HIPAA y la Ley de IA de la UE, evaluando críticamente su intención, valor e implementación.

Temas:

Core

- Concepciones filosóficas y legales de la naturaleza de la privacidad, incluyendo el derecho a la privacidad.
- Fundamentos legales de la protección de la privacidad en jurisdicciones relevantes (por ejemplo, GDPR en la UE).
- Legislación de privacidad en áreas de práctica (por ejemplo, HIPAA en EE.UU., Ley de IA en la UE).

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Discutir la base filosófica para la protección legal de la privacidad personal en un contexto apropiado (por ejemplo, país) [Debatir]
2. Criticar la intención, valor potencial e implementación de varias formas de legislación de privacidad y principios más allá de lo que la ley requiere [Críticar (análisis crítico)]

2.15.16. SEP/Libertades Civiles y Expresión (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Libertades civiles, derechos de privacidad y diferencias culturales.
- Principios básicos de la investigación con sujetos humanos y principios más allá de lo que la ley requiere (por ejemplo, Informe Belmont, Declaración Universal de Derechos Humanos de la ONU y cómo esto se relaciona con la tecnología).
- Libertad de expresión y sus limitaciones.
- Contenido generado por usuarios, moderación de contenido y responsabilidad.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar diferencias culturales con respecto a la naturaleza y necesidad de la privacidad y otras libertades civiles [Analizar]
2. Identificar estrategias para permitir una libertad de expresión apropiada [Analizar]

2.15.17. SEP/Comunicación Técnica y Profesional (CS Core: 1 hr, KA Core: 1 hr)

La informática es una disciplina inherentemente colaborativa y social, lo que hace de la comunicación un aspecto esencial de la profesión. Gran parte, pero no toda, de esta comunicación ocurre en un entorno profesional donde los estilos, expectativas y normas de comunicación difieren de otros contextos donde se podría usar tecnología similar. Tanto la comunicación profesional como la informal transmiten información a varias audiencias que pueden tener diferentes objetivos y necesidades para esa información. Una buena comunicación también es necesaria para la transparencia y la confiabilidad. También es importante señalar que los profesionales de la informática no solo son comunicadores sino también oyentes que deben poder escuchar y utilizar reflexivamente los comentarios recibidos de diversas partes interesadas. Las habilidades de comunicación efectiva no son algo que uno 'simplemente sabe': se desarrollan y pueden aprenderse. Las habilidades de comunicación se enseñan mejor en contexto a lo largo del plan de estudios de pregrado.

Temas:
Core

- Comunicación oral, escrita y electrónica en equipo y grupo.
- Materiales de comunicación técnica (por ejemplo, código fuente y documentación, tutoriales, materiales de referencia, documentación de API).
- Comunicarse con diferentes partes interesadas, como clientes, liderazgo o el público.
- Compensaciones (tradeoffs) en factores competitivos que afectan los canales y elecciones de comunicación.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comprender la importancia de escribir documentos técnicos concisos y precisos siguiendo estándares bien definidos para el formato y para incluir tablas, figuras y referencias apropiadas [Producir]
2. Analizar documentación técnica escrita para verificar precisión técnica, concisión, falta de ambigüedad y conciencia de la audiencia [Analizar]
3. Componer y entregar una presentación formal consciente de la audiencia, accesible y organizada [Componer]
4. Identificar y describir cualidades de la comunicación efectiva (por ejemplo, virtual, cara a cara, intragrupo, documentos compartidos) [Analizar]
5. Elegir una forma apropiada de comunicar preocupaciones éticas delicadas [Evaluar (valorar)]

2.15.18. SEP/Comunicación en Equipo y Cultura (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Colaboración en equipo (incluyendo herramientas) y resolución de conflictos.
- Requisitos de accesibilidad e inclusividad para dirigirse a audiencias profesionales.

- Competencia cultural en la comunicación, incluyendo considerar el impacto de la diferencia en el lenguaje natural.
- Comunicarse para resolver problemas o hacer recomendaciones en el lugar de trabajo, como plantear preocupaciones éticas o abordar problemas de accesibilidad.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Planificar interacciones (por ejemplo, virtuales, cara a cara, documentos compartidos) con otros de maneras que inviten a la participación inclusiva, modelen la consideración respetuosa de las contribuciones de otros y valoren explícitamente la diversidad de ideas [Planear]
2. Comprender cómo comunicarse efectiva y apropiadamente como miembro de un equipo, incluyendo técnicas de resolución de conflictos [Aplicar]
3. Discutir formas de influir en el desempeño y los resultados en equipos diversos y multiculturales [Debatir]
4. Evaluar fortalezas y debilidades personales para trabajar de forma remota como parte de un equipo que proviene de diversos antecedentes y experiencias [Evaluar (valorar)]

2.15.19. SEP/Huella Ambiental de los Sistemas Informáticos (CS Core: 1 hr, KA Core: 1 hr)

El ciclo de vida del hardware y software tiene costos ambientales significativos. Esta unidad examina los impactos ambientales y sociales de los sistemas informáticos —desde residuos electrónicos y huella de carbono hasta el consumo energético de centros de datos y modelos de IA— evaluando su efecto en las decisiones de diseño y en la sociedad global.

Temas:
Core

- Impactos ambientales, sociales y culturales de las decisiones de implementación (por ejemplo, objetivos de sostenibilidad, sesgo/resultados algorítmicos, viabilidad económica y consumo de recursos).
- Impactos sociales y ambientales locales/regionales/globales de los sistemas informáticos y su uso (por ejemplo, huella de carbono, uso de recursos, residuos electrónicos) debido al hardware (por ejemplo, residuos electrónicos, centros de datos, utilización de elementos y recursos raros, reciclaje) y software (por ejemplo, servicios basados en la nube, blockchain, entrenamiento y uso de modelos de IA). Esto incluye el uso diario de hardware (hardware barato reemplazado con frecuencia) y software (navegación web, correo electrónico y otros servicios con demandas computacionales ocultas/remotas).

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Evaluar los impactos ambientales del despliegue de un proyecto dado (por ejemplo, consumo de energía, contribución a los residuos electrónicos, impacto de la fabricación) [Evaluar (valorar)]
2. Describir los impactos sociales y ambientales globales del uso y disposición de computadoras [Describir]
3. Describir los impactos ambientales de las decisiones de diseño dentro del campo de la informática que se relacionan con el diseño de algoritmos, diseño de sistemas operativos, diseño de redes, diseño de bases de datos, etc [Describir]
4. Evaluar aplicaciones informáticas en relación con problemas ambientales (por ejemplo, energía, contaminación, uso de recursos, reciclaje y reutilización, gestión y producción de alimentos) [Evaluar (valorar)]

2.15.20. SEP/Efectos Sistémicos y Contexto Social (CS Core: 1 hr, KA Core: 1 hr)

Las tecnologías informáticas complejas como la IA generativa, las redes sociales y el trabajo remoto generan efectos sistémicos difíciles de predecir. Esta unidad analiza cómo la sostenibilidad del software depende de los sistemas sociales que lo rodean —usuarios, organizaciones, mercados y políticas— y cómo los diseños tecnológicos transforman a su vez esos sistemas.

Temas:
Core

- Efectos sistémicos de tecnologías y fenómenos informáticos complejos (por ejemplo, IA generativa, centros de datos, redes sociales, externalización (offshoring), trabajo remoto).
- Cómo la sostenibilidad de los sistemas de software es interdependiente con los sistemas sociales, incluyendo el conocimiento y las habilidades de sus usuarios, los procesos y políticas organizacionales y su contexto social (por ejemplo, fuerzas del mercado, políticas gubernamentales).

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enumerar los efectos sostenibles de las prácticas y actividades modernas (por ejemplo, trabajo remoto, comercio electrónico, criptomonedas, modelos de IA, centros de datos) [Listar/Enumerar]
2. Analizar los impactos sociales y ambientales de nuevos diseños de sistemas [Analizar]

2.15.21. SEP/Diseño Sostenible y Computación Ubicua (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Pautas para estándares de diseño sostenible.
- Computación ubicua (pervasive computing): procesamiento de información que se ha integrado en objetos y actividades cotidianas, como sistemas inteligentes de energía, redes sociales y sistemas de retroalimentación para promover comportamientos sostenibles, transporte, monitoreo ambiental, ciencia ciudadana y activismo.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar formas de ser un profesional sostenible en un área específica o con un proyecto específico [Analizar]
2. Diseñar pautas para el diseño o despliegue de TI sostenible en áreas como sistemas inteligentes de energía, redes sociales, transporte, agricultura, sistemas de cadena de suministro, monitoreo ambiental y activismo ciudadano [Diseñar]

2.15.22. SEP/Fundamentos e Historia Temprana de la Informática (CS Core: 1 hr, KA Core: 1 hr)

La historia es importante porque proporciona un mecanismo para comprender por qué nuestros sistemas informáticos operan como lo hacen, los contextos sociales en los que surgieron los enfoques actuales y cómo estos continúan resonando en la disciplina hoy. No solo la informática afecta a la sociedad, sino también viceversa, resultando en un contexto socio-técnico complejo que está en constante cambio, requiriendo la perspectiva de la historia para poner el presente, así como los futuros posibles, en la perspectiva apropiada. También informa decisiones basadas en éxitos y fracasos del pasado, incluyendo el daño causado y cómo no repetirlos. La historia de la informática a menudo se enseña en contexto con conceptos fundamentales, como fundamentos de sistemas y fundamentos de desarrollo de software. Se debe poner un enfoque en aquellos que, debido a la marginación, no han figurado históricamente tan prominentemente como deberían.

Temas:
Core

- La historia de la informática: hardware, software y humano/organizacional.
- El papel de la historia en el presente, incluyendo dentro de diferentes contextos sociales, y la relevancia de esta historia en el futuro.
- Edad I (Pre-digital): Computación analógica antigua (Stonehenge, mecanismo de Anticitera, reloj de la Catedral de Salisbury, etc.), tablas numéricas calculadas por humanos, Euclides, Lovelace, Babbage, Gödel, Church, Turing, hardware pre-electrónico (electromecánico y mecánico).
- Edad II (Computación moderna temprana): ENIAC, UNIVAC, Bombas (Bletchley Park y descifradores de códigos), compañías informáticas (por ejemplo, IBM), mainframes, etc.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comprender la relevancia e impacto de la historia de la informática en eventos recientes, contexto presente y posibles resultados futuros, desde más de una perspectiva cultural [Evaluar (valorar)]
2. Discutir cómo las perspectivas sostenidas hoy han sido moldeadas por la historia, y que existen perspectivas alternativas (por ejemplo, temores de que la IA reemplace a los trabajadores humanos vs IA que aumenta el trabajo humano, varias visiones sobre la Ley de Moore) [Debatir]
3. Identificar tendencias formativas y consecuentes en la historia del campo de la informática [Analizar]
4. Identificar las contribuciones de individuos u organizaciones pioneras (laboratorios de investigación, compañías informáticas, oficinas gubernamentales) en el campo de la informática [Analizar]

2.15.23. SEP/Historia Moderna: PC, Internet e IA (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Edad III (Era de las PC): PCs, hardware y software informático moderno, Ley de Moore.
- Edad IV (Internet): Redes, arquitectura de internet, navegadores y su evolución, estándares, compañías y servicios nacidos en internet (por ejemplo, Google, Amazon, Microsoft, etc.), computación distribuida.
- Edad V (Móvil y Nube): Computación móvil y teléfonos inteligentes, computación en la nube y sus modelos (por ejemplo, SaaS), servidores remotos, seguridad y privacidad, redes sociales.
- Edad VI (IA): Sistemas de toma de decisiones, sistemas de recomendación, IA generativa y otras herramientas y tecnologías impulsadas por aprendizaje automático.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Discutir el contexto histórico para momentos importantes en la historia de la informática, como el cambio de tubos de vacío a transistores (TRADIC), sistemas operativos seminales tempranos (por ejemplo, OS 360), Xerox PARC y la primera computadora Apple con una GUI, la creación de paradigmas de lenguaje de programación específicos, el primer virus informático, la creación de internet, la creación de la WWW, la burbuja puntocom, Y2K, la introducción de teléfonos inteligentes, etc [Debatir]
2. Comparar la vida diaria antes y después del advenimiento de desarrollos históricos (por ejemplo, computadoras personales o internet) [Comparar]

2.15.24. SEP/Modelos y Estrategias Económicas (KA Core: 1 hr)

Las economías de la informática son importantes para aquellos que desarrollan y proporcionan recursos y servicios informáticos a otros, así como para la sociedad en general. Impactan a los usuarios de estos recursos y servicios, tanto profesionales como no profesionales. Los profesionales de la informática tienen el deber de conocer el impacto de estos temas en sus propios roles y actividades y cómo las elecciones hechas impactarán a los usuarios y a la sociedad.

Temas:
Core

- Modelos económicos: regulados y no regulados, monopolios, efectos de red (network effects) y mercado abierto; economías del conocimiento y de la atención (knowledge and attention economies).
- Estrategias de precios y despliegue: obsolescencia programada (planned obsolescence), suscripciones, freemium, licencias de software, código abierto (open-source), software libre (free software), adware.
- Impactos de las diferencias en el acceso a recursos informáticos y el efecto de la oferta y demanda de mano de obra calificada en la calidad de los productos informáticos.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Resumir los efectos sociales de los modelos económicos (por ejemplo, las economías del conocimiento y de la atención) [Resumir]
2. Describir las diferencias y similitudes de estrategias competitivas (por ejemplo, suscripción vs freemium vs gratis) [Describir]

3. Discutir ejemplos de brechas digitales [Debatir]

2.15.25. SEP/Automatización, IA y Ética Económica (KA Core: 1 hr)

Temas:
Core

- Automatización, IA y sus efectos en los mercados laborales, desarrolladores y usuarios.
- Preocupaciones éticas en torno a la economía de la atención (attention economy) y otras economías de la informática (por ejemplo, consentimiento informado, recopilación de datos, uso de jerga legal extensa en acuerdos de usuario).

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comprender los efectos de la automatización y la IA en la sociedad [Evaluar (valorar)]
2. Comprender las implicaciones éticas de las economías informáticas que dependen de la atención y los datos [Evaluar (valorar)]

2.15.26. SEP/Delitos y Ataques Informáticos (CS Core: 1 hr, KA Core: 1 hr)

Si bien las políticas de seguridad, las leyes y los delitos informáticos son temas importantes, es esencial que se vean con la base de otras unidades de conocimiento social y profesional, como Propiedad Intelectual, Privacidad y Libertades Civiles, Contexto Social y ética Profesional. Las computadoras, internet y la inteligencia artificial, quizás más que cualquier otra tecnología, han transformado la sociedad en los últimos 75 años. Al mismo tiempo, estas tecnologías han contribuido a amenazas sin precedentes a la privacidad; nuevas categorías de delitos informáticos y comportamientos antisociales; grandes interrupciones a organizaciones; y la concentración a gran escala de riesgo en los sistemas de información.

Temas:
Core

- Delitos informáticos, recurso legal para criminales informáticos e impacto en víctimas y sociedad.
- Ingeniería social, fraude habilitado por la informática, robo de identidad y recuperación de estos.
- Terrorismo cibernético, hacking criminal y hacktivismo.
- Malware, virus, gusanos.
- Ataques a infraestructura crítica como redes eléctricas y oleoductos.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enumerar ejemplos clásicos de delitos informáticos e incidentes de ingeniería social con impacto social [Listar/Enumerar]
2. Identificar problemas con las leyes que se aplican a los delitos informáticos [Analizar]
3. Describir la motivación y ramificaciones del terrorismo cibernético, robo de datos, hacktivismo, ransomware y otros ataques [Describir]
4. Examinar los problemas éticos y legales que rodean el mal uso del acceso y varias violaciones de seguridad [Examinar]

2.15.27. SEP/Leyes, Políticas y Responsabilidad en Seguridad (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Fundamentos no técnicos de la seguridad (por ejemplo, ingeniería humana, política, confidencialidad).
- Beneficios y desafíos de las leyes de delitos informáticos existentes y propuestas.
- Políticas de seguridad y los desafíos del cambio y el cumplimiento.
- Responsabilidad por la seguridad a lo largo del ciclo de vida de la informática.
- Leyes internacionales y locales y cómo se cruzan.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Discutir el papel del profesional en la seguridad y las compensaciones y desafíos involucrados [Debatir]
2. Investigar medidas que pueden tomar tanto individuos como organizaciones, incluidos gobiernos, para prevenir o mitigar los efectos indeseables de los delitos informáticos [Investigar]
3. Diseñar una política de seguridad para toda la compañía, que incluya procedimientos para la gestión de contraseñas y monitoreo de empleados [Diseñar]
4. Comprender cómo la legislación de una región puede afectar las actividades en otra (por ejemplo, cómo el GDPR de la UE se aplica globalmente, cuando hay personas de la UE involucradas) [Evaluar (valorar)]

2.15.28. SEP/Identidad y Diversidad en la Informática (CS Core: 1 hr, KA Core: 1 hr)

La identidad —incluyendo raza, género, origen y capacidad— moldea tanto la experiencia de los profesionales de la informática como los productos que crean. Esta unidad examina cómo la identidad interactúa con las tecnologías y entornos informáticos, y los beneficios demostrables de los equipos de desarrollo diversos, así como los costos de la falta de diversidad.

Temas:
Core

- Cómo la identidad impacta y es impactada por las tecnologías y entornos informáticos (académicos y profesionales).
- Los beneficios de los equipos de desarrollo diversos y los impactos de los equipos que no son diversos.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Definir y distinguir equidad, igualdad, diversidad, inclusión y accesibilidad [Definir]
2. Identificar ejemplos de los beneficios que los equipos diversos pueden aportar a los productos de software y cómo la falta de diversidad tiene costos [Analizar]
3. Evaluar el impacto del poder y el privilegio en la profesión de la informática en lo que respecta a la cultura, industria, productos y sociedad [Evaluar (valorar)]

2.15.29. SEP/Contexto Histórico e Intercultural de DEIA (CS Core: 1 hr, KA Core: 1 hr)

Las desigualdades actuales en la informática tienen raíces históricas profundas en mecanismos sociales sistémicos. Esta unidad examina la marginación histórica en la informática, los desafíos de infraestructura global, y las diferencias interculturales en DEIA, explorando cómo los cambios sistémicos pueden mejorar la equidad tanto en contextos familiares como desconocidos.

Temas:
Core

- Marginación histórica debido a mecanismos sociales sistémicos, supremacía tecnológica y desafíos de infraestructura global para la diversidad, equidad, inclusión y accesibilidad.
- Diferencias y necesidades interculturales en diversidad, equidad, inclusión y accesibilidad.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comparar la demografía de los estudiantes de ciencias de la computación y STEM de su institución con la demografía institucional general. Si difieren, identificar factores que contribuyen al acceso, compromiso y logro inequitativo en ciencias de la computación entre grupos marginados. Si no, evaluar por qué no [Comparar]
2. Desarrollar ejemplos de cambios sistémicos que podrían abordar positivamente la diversidad, equidad, inclusión y accesibilidad en un contexto familiar (es decir, en un curso introductorio de informática) y en un contexto no familiar, y cuándo estos podrían ser diferentes o iguales [Crear]

3. Comparar la demografía de su institución con la demografía general de la comunidad. Si difieren, identificar factores que contribuyen al acceso, compromiso y logro inequitativo entre grupos marginados. Si no, evaluar por qué no [Comparar]

2.15.30. SEP/Inclusión y Prácticas en Informática (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Lenguaje inclusivo y terminología cargada, y por qué su uso importa.
- Comportamientos inclusivos y por qué importan.
- Cómo los profesionales de la informática pueden influir e impactar la diversidad, equidad, inclusión y accesibilidad, incluyendo pero no solo a través del software que crean.
- Expertos y sus prácticas que reflejan las identidades del aula y del mundo a través de principios prácticos de DEIA.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar lenguaje, prácticas y comportamientos que pueden hacer que alguien se sienta incluido en un lugar de trabajo y/o equipo, y por qué es relevante. Evitar terminología cargada (véase Words Matter, www.acm.org/diversity-inclusion/words-matter); esto incluye identificar y acomodar a usuarios que a menudo son excluidos sin pensarlo y no son considerados en absoluto [Analizar]
2. Demostrar colegialidad y respeto cuando se trabaja con miembros del equipo que no comparten su identidad. No es suficiente simplemente asignar proyectos en equipo. La facultad debe preparar a los estudiantes para el trabajo en equipo y monitorear, guiar y evaluar la efectividad de sus equipos estudiantiles a lo largo de un proyecto [Demostrar]
3. Identificar los supuestos y valores de los desarrolladores incorporados en el diseño de hardware y software, especialmente en lo que respecta a la usabilidad por parte de poblaciones diversas [Analizar]
4. Analizar el trabajo de expertos que reflejan las identidades del aula y del mundo [Analizar]

2.15.31. SEP/Diseño Accesible (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Diseñar y desarrollar tecnología con la accesibilidad en mente.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Evaluar la accesibilidad de su aula o laboratorio. Evaluar la accesibilidad de su página web. (Véase www.w3.org/WAI y www.w3.org/TR/WCAG21) [Evaluar]

2.16. Fundamentos de Sistemas (SF)

Un sistema informático es un conjunto de infraestructuras de hardware y software sobre las cuales se construyen aplicaciones. Los sistemas informáticos se han convertido en un pilar de la vida diaria de las personas. Como tal, es esencial que los estudiantes aprendan conocimientos sobre sistemas informáticos, adquieran las habilidades para usar y diseñar estos sistemas y comprendan la lógica y los principios fundamentales en los sistemas informáticos. Esto podría equipar a los estudiantes con la competencia necesaria para una carrera relacionada con las ciencias de la computación.

En el plan de estudios de Ciencias de la Computación, el estudio de los sistemas informáticos generalmente abarca múltiples áreas de conocimiento, incluyendo, entre otros, sistemas operativos, sistemas paralelos y distribuidos, redes de comunicaciones, arquitectura y organización de computadoras, e ingeniería de software. El área de conocimiento de Fundamentos de Sistemas, como sugiere su nombre, se centra en los conceptos fundamentales y principios de diseño en sistemas informáticos que son compartidos por estos cursos dentro de sus respectivos núcleos. El objetivo de esta área de conocimiento es presentar una visión integradora de estos conceptos fundamentales y principios de diseño de una manera unificada aunque simplificada, proporcionando una base común para los diferentes

mecanismos y políticas especializados apropiados para el área de dominio específica. Los conceptos fundamentales en esta área de conocimiento incluyen una visión general de los sistemas informáticos, conceptos básicos como estado y transición de estado, asignación y planificación de recursos, y así sucesivamente. Además, esta área de conocimiento introduce principios básicos de diseño para mejorar la confiabilidad, disponibilidad, eficiencia y seguridad de los sistemas informáticos.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 2.16.1 Componentes Hardware y Lógica (Pág. 134)	2	
⭐ 2.16.2 Capas, Abstracciones y Escalabilidad (Pág. 134)	2	
⭐ 2.16.3 Sistemas Digitales y Lógica (Pág. 135)	2	
⭐ 2.16.4 Máquinas de Estado (Pág. 135)	2	
⭐ 2.16.5 Procesamiento Secuencial y Paralelo (Pág. 136)	2	
⭐ 2.16.6 Gestión de Recursos (Pág. 136)	1	1
⭐ 2.16.7 Latencia, Caché y Jerarquía de Memoria (Pág. 137)	2	2
⭐ 2.16.8 Virtualización y Aislamiento (Pág. 137)	1	1
⭐ 2.16.9 Métricas y Benchmarks de Rendimiento (Pág. 138)	1	1
⭐ 2.16.10 Modelos Analíticos de Rendimiento (Pág. 138)	1	1
⭐ 2.16.11 Eventos, Herramientas y Experimentación (Pág. 138)	1	1
⭐ 2.16.12 Confiabilidad del Sistema (Pág. 139)	2	1
⭐ 2.16.13 Seguridad del Sistema (Pág. 139)	2	1
⭐ 2.16.14 Diseño del Sistema (Pág. 140)	2	1
2.16.15 Sociedad, ética y la Profesión (Pág. 140)	Electivo	

Cuadro 2.16: Lista de KUs del área de Fundamentos de Sistemas.

2.16.1. SF/Componentes Hardware y Lógica (CS Core: 2 hrs)

Introduce los bloques de construcción y componentes básicos de una computadora, el hardware como paradigma computacional y el concepto de segmentación (pipelining).

Temas:

Core

- Bloques de construcción y componentes básicos de una computadora (compuertas, flip-flops, registros, interconexiones; ruta de datos (datapath) + control + memoria).
- Hardware como paradigma computacional: Bloques de construcción lógicos fundamentales; Expresiones lógicas, minimización, formas de suma de productos Lógica Digital y Sistemas Digitales.
- Concepto básico de segmentación (pipelining), etapas de procesamiento superpuestas Organización Funcional.

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Describir los bloques de construcción básicos de las computadoras y su papel en el desarrollo histórico de la arquitectura de computadoras [Describir]
2. Diseñar un circuito lógico simple usando los bloques de construcción fundamentales del diseño lógico para resolver un problema simple (por ejemplo, un sumador) [Diseñar]

2.16.2. SF/Capas, Abstracciones y Escalabilidad (CS Core: 2 hrs)

Cubre las abstracciones de programación, la distinción entre servicios de aplicación y del sistema operativo, el concepto de escalado y la organización en capas de los sistemas informáticos.

Temas:

Core

- Abstracciones de programación, interfaces, uso de bibliotecas Programas: Distribución, Programas: Propiedades de Ejecución, Programación GPU, Programas: Mapeos de Implementación, Programas: Paralelismo Declarativo, Programas: Inicio de Actividades.
- Distinción e interacción entre servicios de aplicación y del sistema operativo, llamada a procedimiento remoto (*remote procedure call*) Rol y Propósito de los Sistemas Operativos.

- Concepto básico de escalado (scaling): rendimiento vs tamaño del problema.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir cómo los sistemas informáticos se construyen de capa sobre capa, basados en la separación de preocupaciones, con interfaces bien definidas, ocultando detalles de las capas bajas a las capas más altas [Describir]
2. Describir que el hardware, el sistema operativo, la máquina virtual y la aplicación son capas adicionales de interpretación/procesamiento [Describir]
3. Describir los mecanismos de cómo los errores se detectan, se señalan hacia atrás y se manejan a través de las capas [Describir]
4. Construir un programa simple (por ejemplo, un cliente/servidor TCP) usando métodos de capas, detección y recuperación de errores, y reflejo del estado de error a través de las capas [Crear]
5. Identificar errores en un programa por capas usando herramientas para seguimiento de programas, ejecución paso a paso (single stepping) y depuración [Analizar]
6. Comprender el concepto de escalado fuerte (strong) vs débil (weak), es decir, cómo el rendimiento se ve afectado por la escala del problema vs la escala de los recursos para resolver el problema. Esto puede motivarse con ejemplos simples del mundo real [Evaluar (valorar)]

2.16.3. SF/Sistemas Digitales y Lógica (CS Core: 2 hrs)

Cubre sistemas digitales vs analógicos, compuertas lógicas, expresiones booleanas, relojes y lógica combinacional y secuencial, como base para comprender el funcionamiento de los sistemas informáticos.

Temas:
Core

- Sistemas Digitales vs Analógicos/Discretos vs Continuos.
- Compuertas lógicas simples, expresiones lógicas, simplificación de lógica booleana.
- Relojes, Estado, Secuenciación.
- Lógica Combinacional, Lógica Secuencial, Registros, Memorias Lógica Digital y Sistemas Digitales.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir las diferencias entre sistemas digitales y analógicos, y entre sistemas discretos y continuos. Puede dar ejemplos del mundo real de estos sistemas [Describir]

2.16.4. SF/Máquinas de Estado (CS Core: 2 hrs)

Introduce el concepto de estado y transición de estado, máquinas de estados finitos (FSM), y su aplicación en computadoras y protocolos de red como ejemplos de máquinas de estado.

Temas:
Core

- Estado y transición de estado (por ejemplo, estado inicial, estado final, ciclo de vida de estados) Lenguajes Formales y Autómatas.
- Máquinas de estados finitos (por ejemplo, NFA, DFA) Lenguajes Formales y Autómatas.
- Computadoras y Protocolos de Red como ejemplos de Máquinas de Estado Fundamentos de Redes y Comunicaciones.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir las computaciones como un sistema caracterizado por un conjunto conocido de configuraciones con transiciones de una configuración única (estado) a otra (estado) [Describir]
2. Describir la distinción entre sistemas cuya salida es solo una función de su entrada (sin estado - stateless) y aquellos con memoria/historial (con estado - stateful) [Describir]
3. Desarrollar descripciones de máquinas de estado para soluciones de enunciados de problemas simples (por ejemplo, secuenciación de semáforos, reconocedores de patrones) [Crear]
4. Describir una computadora como una máquina de estado que interpreta instrucciones de máquina [Describir]

5. Explicar cómo un programa o protocolo de red también puede expresarse como una máquina de estado y que pueden existir representaciones alternativas para la misma computación [Explicar]
6. Derivar el comportamiento de serie temporal de una máquina de estado a partir de su representación de máquina de estado (por ejemplo, máquina de estado de gestión de conexión TCP) [Analizar]

2.16.5. SF/Procesamiento Secuencial y Paralelo (CS Core: 2 hrs)

Cubre el procesamiento secuencial y paralelo, incluyendo modelos de un solo hilo y múltiples hilos a nivel de aplicación, y la evaluación del rendimiento de versiones paralelas de programas.

Temas:

Core

- Procesamiento secuencial vs paralelo. Concurrencia, Programas: Distribución, Programas: Propiedades de Ejecución, Programación GPU, Programas: Mapeos de Implementación, Programas: Paralelismo Declarativo, Programas: Inicio de Actividades.
- Procesamiento secuencial a nivel de aplicación: hilo único (single thread) Concurrencia, Programas: Distribución, Programas: Propiedades de Ejecución, Programación GPU, Programas: Mapeos de Implementación, Programas: Paralelismo Declarativo, Programas: Inicio de Actividades.
- Procesamiento paralelo simple a nivel de aplicación: a nivel de solicitud (servicios web/cliente-servidor/distribuido), hilo único por servidor, múltiples hilos con múltiples servidores, segmentación (pipelining) Concurrencia, Programas: Distribución, Programas: Propiedades de Ejecución, Programación GPU, Programas: Mapeos de Implementación, Programas: Paralelismo Declarativo, Programas: Inicio de Actividades.

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Escribir un problema secuencial simple y una versión paralela simple del mismo programa [Escribir]
2. Evaluar el rendimiento de versiones secuenciales y paralelas simples de un programa con diferentes tamaños de problema y ser capaz de describir las aceleraciones (speed-ups) logradas [Evaluar]
3. Describir en una línea de tiempo de ejecución cómo los eventos y operaciones de paralelismo pueden tener lugar simultáneamente (es decir, al mismo tiempo). Explicar cómo el trabajo puede realizarse en menos tiempo transcurrido si esto puede explotarse [Describir]

2.16.6. SF/Gestión de Recursos (CS Core: 1 hr, KA Core: 1 hr)

Esta unidad de conocimiento cubre diferentes tipos de recursos en sistemas informáticos y algoritmos comunes de asignación y planificación de recursos.

Temas:

Core

- Diferentes tipos de recursos (por ejemplo, participación del procesador, memoria, disco, ancho de banda de red) Fundamentos de Redes y Comunicaciones, API de Sistemas de Archivos e Implementación, Gestión de Memoria, Planificación.
- Algoritmos comunes de asignación/planificación de recursos (por ejemplo, primero en llegar primero en ser servido, planificación basada en prioridad, planificación justa (fair scheduling) y planificación con desalojo (preemptive scheduling)) Planificación.
- Ventajas y desventajas de los algoritmos de planificación comunes Planificación.

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Definir cómo los recursos finitos de computadora (por ejemplo, participación del procesador, memoria, almacenamiento y ancho de banda de red) se gestionan mediante su cuidadosa asignación a las entidades existentes [Definir]
2. Describir cómo funcionan los algoritmos comunes de asignación/planificación de recursos [Describir]
3. Desarrollar algoritmos de planificación comunes y evaluar sus rendimientos [Crear]

4. Describir los pros y los contras de los algoritmos de planificación comunes [Describir]

2.16.7. SF/Latencia, Caché y Jerarquía de Memoria (CS Core: 2 hrs, KA Core: 2 hrs)

Cubre las latencias en los sistemas informáticos, la localidad espacial y temporal en las memorias caché, la coherencia de caché, y la jerarquía de memoria del procesador con su fórmula de tiempo promedio de acceso.

Temas:

Core

- Latencias en sistemas informáticos: 1. Velocidad de la luz y las computadoras (un pie por nanosegundo vs relojes de un GHz) Organización Funcional 2. Latencias de memoria vs disco vs memoria a través de la red Jerarquía de Memoria, Rendimiento y Eficiencia Energética
- Memorias caché y los efectos de la localidad espacial y temporal en el rendimiento en procesadores y sistemas Jerarquía de Memoria, Rendimiento y Eficiencia Energética, Gestión de Memoria.
- Memorias caché y coherencia de caché en bases de datos, sistemas operativos, sistemas distribuidos y arquitectura de computadoras Jerarquía de Memoria, Internos de DBMS, Gestión de Memoria.
- Introducción a la jerarquía de memoria del procesador Jerarquía de Memoria, Rendimiento y Eficiencia Energética.
- La fórmula del tiempo promedio de acceso a memoria Jerarquía de Memoria.

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Describir el desglose de la latencia de los sistemas informáticos en términos de memoria, disco y red [Describir]
2. Explicar la importancia de la localidad en la determinación del rendimiento del sistema [Explicar]
3. Calcular el tiempo promedio de acceso a memoria y describir las compensaciones (tradeoffs) en el rendimiento de la jerarquía de memoria en términos de capacidad, tasa de fallos/aciertos (miss/hit rate) y tiempo de acceso [Calcular]

2.16.8. SF/Virtualización y Aislamiento (CS Core: 1 hr, KA Core: 1 hr)

Cubre la justificación del aislamiento y la virtualización para la protección y el rendimiento predecible, los niveles de indirección mediante memoria virtual, y los métodos para implementar memoria virtual y máquinas virtuales.

Temas:

Core

- Justificación de la virtualización y el aislamiento: protección y rendimiento predecible Virtualización.
- Niveles de indirección, ilustrados por la memoria virtual para gestionar recursos de memoria física Virtualización.
- Métodos para implementar memoria virtual y máquinas virtuales Virtualización.

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar por qué es importante aislar y proteger la ejecución de programas individuales y entornos que comparten recursos subyacentes comunes [Explicar]
2. Describir cómo el concepto de indirección puede crear la ilusión de una máquina dedicada y sus recursos incluso cuando se comparten físicamente entre múltiples programas y entornos [Describir]
3. Evaluar el rendimiento de dos instancias de aplicación ejecutándose en máquinas virtuales separadas y determinar el efecto del aislamiento de rendimiento [Evaluar]

2.16.9. SF/Métricas y Benchmarks de Rendimiento (CS Core: 1 hr, KA Core: 1 hr)

Cubre las figuras de mérito de rendimiento, las cargas de trabajo y benchmarks representativos, y los métodos de recolección y análisis para la evaluación cuantitativa del rendimiento de los sistemas informáticos.

Temas:

Core

- Figuras de mérito de rendimiento Rendimiento y Eficiencia Energética, Evaluación.
- Cargas de trabajo y puntos de referencia representativos (benchmarks), y métodos de recolección y análisis de figuras de mérito de rendimiento Rendimiento y Eficiencia Energética, Evaluación.

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar cómo los componentes de la arquitectura del sistema contribuyen a mejorar su rendimiento [Explicar]
2. Explicar las circunstancias en las que una figura dada de una métrica de rendimiento del sistema es útil [Explicar]
3. Explicar el uso e insuficiencias de los puntos de referencia (benchmarks) como medida del rendimiento del sistema [Explicar]

2.16.10. SF/Modelos Analíticos de Rendimiento (CS Core: 1 hr, KA Core: 1 hr)

Cubre la ecuación CPI como herramienta de diseño, la Ley de Amdahl y sus limitaciones, el análisis de orden de magnitud (Big O), el análisis de caminos críticos del sistema y las herramientas analíticas para la evaluación cuantitativa.

Temas:

Core

- Ecuación CPI (Ciclos por Instrucción) como herramienta para comprender las compensaciones (tradeoffs) en el diseño de conjuntos de instrucciones, tuberías de procesador (pipelines) y organizaciones del sistema de memoria Rendimiento y Eficiencia Energética, Evaluación.
- Ley de Amdahl: la parte de la computación que no se puede acelerar limita el efecto de las partes que sí se pueden Rendimiento y Eficiencia Energética, Evaluación.
- Análisis de orden de magnitud (notación Big O) Notación Asintótica y Clases de Complejidad, Análisis de Complejidad: Recursión, Amortización y Cotas Ajustadas, Marco de Análisis de Complejidad, Teoría de la Complejidad Computacional.
- Análisis de caminos lentos y rápidos de un sistema Organización Funcional.
- Herramientas analíticas para guiar la evaluación cuantitativa.

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Describir la Ley de Amdahl y discutir sus limitaciones [Describir]
2. Aplicar estudios de límite o cálculos simples para producir estimaciones de orden de magnitud para una métrica de rendimiento dada en un contexto dado [Aplicar]

2.16.11. SF/Eventos, Herramientas y Experimentación (CS Core: 1 hr, KA Core: 1 hr)

Cubre los eventos y su efecto en el rendimiento (detenciones, fallos de caché, fallos de página), la comprensión de sistemas por capas como plataformas de evaluación, las trampas de los microbenchmarks, y las herramientas y experimentos para medir el rendimiento del sistema.

Temas:

Core

- Eventos y su efecto en el rendimiento (por ejemplo, detenciones de instrucciones (stalls), fallos de caché, fallos de página) Organización Funcional, Gestión de Memoria.
- Comprender sistemas por capas, cargas de trabajo y plataformas, sus implicaciones para el rendimiento y los desafíos que representan para la evaluación.

- Trampas de los micro puntos de referencia (microbenchmark pitfalls).

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Aplicar herramientas de software para perfilar y medir el rendimiento del programa [Aplicar]
2. Diseñar y conducir un experimento orientado al rendimiento de un sistema común (por ejemplo, un sistema operativo y Spark) [Diseñar]
3. Diseñar un experimento de rendimiento en un sistema por capas para determinar el efecto de un parámetro del sistema en el rendimiento del sistema [Diseñar]

2.16.12. SF/Confiabilidad del Sistema (CS Core: 2 hrs, KA Core: 1 hr)

Esta unidad de conocimiento cubre la distinción entre errores (bugs), fallas (faults) y fallos (failures); confiabilidad vs disponibilidad; y confiabilidad mediante redundancia, incluyendo verificación y reintento, codificación redundante y duplicación.

Temas:
Core

- Distinción entre errores (bugs), fallas (faults) y fallos (failures) Coordinación, Conceptos de Confiabilidad, Ingeniería de Confiabilidad.
- Confiabilidad vs disponibilidad.
- Confiabilidad mediante redundancia: 1. verificación y reintento (*check and retry*) Soporte de Fiabilidad, Tolerancia a fallos 2. codificación redundante (códigos de corrección de errores, CRC, FEC, RAID) Jerarquía de Memoria, Bases de Datos Distribuidas/Computación en la Nube, Soporte de Fiabilidad 3. duplicación/reflejo (*mirroring*)/réplicas Bases de Datos Distribuidas/Computación en la Nube
- Otros enfoques para la confiabilidad (por ejemplo, registro por diario - journaling) Soporte de Fiabilidad, Tolerancia a fallos, Conceptos de Confiabilidad, Ingeniería de Confiabilidad.

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar la distinción entre errores de programa, errores del sistema y fallas de hardware (por ejemplo, memoria corrupta) y excepciones (por ejemplo, intento de dividir por cero) [Explicar]
2. Articular la distinción entre detectar, manejar y recuperarse de fallas y los métodos para su implementación [Articular]
3. Describir el papel de los códigos de corrección de errores en proporcionar técnicas de verificación y corrección de errores en memorias, almacenamiento y redes [Describir]
4. Aplicar algoritmos simples para explotar información redundante con el propósito de corrección de datos [Aplicar]
5. Comparar diferentes métodos de detección y corrección de errores por su sobrecarga de datos, complejidad de implementación y tiempo de ejecución relativo para codificar, detectar y corregir errores [Comparar]

2.16.13. SF/Seguridad del Sistema (CS Core: 2 hrs, KA Core: 1 hr)

Esta unidad de conocimiento cubre problemas comunes de seguridad del sistema y contramedidas, incluyendo criptografía y arquitectura de seguridad, así como sistemas de contramedida representativos, como sistemas de detección de intrusiones y sistemas antivirus.

Temas:
Core

- Problemas comunes de seguridad del sistema (por ejemplo, virus, ataques de denegación de servicio y escuchas - eavesdropping) Seguridad de Red, Protección y Seguridad, Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad, Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad.
- Contramedidas Seguridad de Red, Principios del Sistema Operativo, Protección y Seguridad: 1. Criptografía Protocolos, Aplicaciones y Criptografía Avanzada, Algoritmos Criptográficos, Fundamentos de Criptografía 2. Arquitectura de seguridad Diseño e Ingeniería de Controles de

Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad
- Sistemas de contramedida representativos: 1. Sistemas de detección de intrusiones, cortafuegos (firewalls) Seguridad de Red 2. Sistemas antivirus

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Describir algunos problemas comunes de seguridad del sistema y dar ejemplos [Describir]
2. Describir algunas contramedidas contra problemas de seguridad del sistema [Describir]
3. Describir sistemas de contramedida representativos [Describir]

2.16.14. SF/Diseño del Sistema (CS Core: 2 hrs, KA Core: 1 hr)

Esta unidad de conocimiento cubre criterios comunes de diseño de sistemas (por ejemplo, vivacidad (liveness), seguridad (safety), robustez, escalabilidad y seguridad (security)) y los diseños de sistemas representativos.
Temas:
Core
- Criterios comunes de diseño de sistemas (por ejemplo, vivacidad (liveness), seguridad (safety), robustez, escalabilidad y seguridad (security)) Evaluación.
- Diseños de sistemas representativos (por ejemplo, servidor web Apache, Spark y Linux).

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Describir criterios comunes de diseño de sistemas [Describir]
2. Dada la funcionalidad requerida de un sistema y sus criterios de diseño clave, proporcionar un diseño de alto nivel de este sistema [Generar]
3. Describir el diseño de algunos sistemas representativos [Describir]

2.16.15. SF/Sociedad, ética y la Profesión

Esta unidad de conocimiento cubre los derechos de propiedad intelectual de los sistemas informáticos, licencias de software comunes y delitos informáticos.
Temas:
Core
- Derechos de propiedad intelectual de los sistemas informáticos Open Source, Piratería y DRM, Derechos de PI Digital y Derechos de Autor, Fundamentos de la Propiedad Intelectual, Licencias, Plagio y Autoría.
- Licencias de software comunes Open Source, Piratería y DRM, Derechos de PI Digital y Derechos de Autor, Fundamentos de la Propiedad Intelectual, Licencias, Plagio y Autoría.
- Delitos informáticos Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Describir los derechos de propiedad intelectual de los sistemas informáticos [Describir]
2. Enumerar licencias de software representativas y comparar sus diferencias [Listar/Enumerar]
3. Enumerar delitos informáticos representativos [Listar/Enumerar]

2.17. Desarrollo de Plataformas Especializadas (SPD)

El área de conocimiento de Desarrollo de Plataformas Especializadas (SPD) se refiere a los atributos que involucran la creación de software dirigido a plataformas de hardware no tradicionales. El desarrollo para cada plataforma especializada, por ejemplo, robots, sistemas móviles, sistemas basados en web y sistemas embebidos, típicamente involucra consideraciones únicas.

Las necesidades sociales e industriales han creado una alta demanda de desarrolladores en plataformas especializadas, como aplicaciones móviles, plataformas web, plataformas robóticas y sistemas embebidos. Algunas habilidades profesionales únicas relevantes para esta área de conocimiento incluyen las siguientes:

- Crear aplicaciones que proporcionen una experiencia de usuario consistente en varios dispositivos, tamaños de pantalla y sistemas operativos.
- Desarrollar interfaces de programación de aplicaciones (APIs) para soportar la funcionalidad de cada plataforma especializada.
- Gestionar desafíos relacionados con limitaciones de recursos como cómputo, memoria, almacenamiento, y redes y comunicaciones.
- Aplicar preocupaciones transversales como optimización, seguridad, mejores prácticas de desarrollo, etc.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 2.17.1 Aspectos Comunes: Plataformas, APIs y Restricciones (Pág. 141)	3	2
⭐ 2.17.2 Aspectos Comunes: Lenguajes de Plataforma y Patrones Web (Pág. 142)	2	1
⭐ 2.17.3 Frontend, Frameworks y Estándares Web (Pág. 142)		3
⭐ 2.17.4 Arquitectura Cloud y Gestión de Datos (Pág. 142)		3
⭐ 2.17.5 Plataformas Móviles: Fundamentos del Desarrollo (Pág. 143)		2
⭐ 2.17.6 Plataformas Móviles: Arquitectura y Restricciones (Pág. 143)		2
⭐ 2.17.7 Plataformas Móviles: Capacidades y Computación Asíncrona (Pág. 144)		1
⭐ 2.17.8 Plataformas Robóticas (Pág. 144)		4
⭐ 2.17.9 Características y Arquitecturas Embebidas (Pág. 144)		1
⭐ 2.17.10 Sistemas, Programación y Sensores (Pág. 145)		2
⭐ 2.17.11 Tiempo Real, Memoria y Verificación (Pág. 145)		3
⭐ 2.17.12 Historia, Tipos y Características de Plataformas (Pág. 146)		1
⭐ 2.17.13 Sensores, Controladores y Ecosistemas eSports (Pág. 146)		1
⭐ 2.17.14 Arquitecturas y Renderizado (Pág. 147)		1
⭐ 2.17.15 Algoritmos, Medios y Mundos Virtuales (Pág. 147)		1
⭐ 2.17.16 Lenguajes, APIs y Herramientas de Desarrollo (Pág. 147)		1
⭐ 2.17.17 Motores de Juego (Pág. 148)		1
⭐ 2.17.18 Fundamentos del Diseño de Juegos (Pág. 148)		1
⭐ 2.17.19 Géneros, Audiencias y Alcance Social (Pág. 148)		1
⭐ 2.17.20 Plataformas Interactivas de Datos y Análisis (Pág. 149)		2
⭐ 2.17.21 Programación con IA y Plataformas Cuánticas (Pág. 149)		1
2.17.22 SPD-SEP/Móvil (Pág. 150)	Electivo	
2.17.23 SPD-SEP Web (Pág. 150)	Electivo	
2.17.24 Propiedad Intelectual en los Juegos (Pág. 151)	Electivo	
2.17.25 Accesibilidad e Interfaz del Juego (Pág. 151)	Electivo	
2.17.26 Salud, Seguridad y Sostenibilidad (Pág. 151)	Electivo	
2.17.27 Representación, Inclusión e Igualdad (Pág. 152)	Electivo	
2.17.28 SPD-SEP/Robótica (Pág. 152)	Electivo	
2.17.29 SPD-SEP/Interactiva (Pág. 153)	Electivo	

Cuadro 2.17: Lista de KUs del área de Desarrollo de Plataformas Especializadas.

2.17.1. SPD/Aspectos Comunes: Plataformas, APIs y Restricciones (CS Core: 3 hrs, KA Core: 2 hrs)

Temas:
Core

- Visión general de las plataformas de desarrollo (es decir, web, móvil, juegos, robótica, embebida e interactiva). 1. Entrada/sensores/dispositivos de control/dispositivos hápticos 2. Limitaciones de recursos *a*) Computacional *b*) Almacenamiento de datos *c*) Memoria *d*) Comunicación 3. Requisitos - seguridad, disponibilidad de tiempo de actividad, tolerancia a fallos Conceptos de Confiabilidad, Ingeniería de Confiabilidad, Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad 4. Salida/actuadores/dispositivos hápticos

- Programación mediante Interfaces de Programación de Aplicaciones (API) específicas de la plataforma vs construcción de aplicaciones tradicional
- Programación bajo restricciones y requisitos de la plataforma (por ejemplo, herramientas de desarrollo disponibles, desarrollo, consideraciones de seguridad) Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enumerar las restricciones de la programación móvil [Listar/Enumerar]

2.17.2. SPD/Aspectos Comunes: Lenguajes de Plataforma y Patrones Web (CS Core: 2 hrs, KA Core: 1 hr)

Temas:
Core

- Visión general de los Lenguajes de plataforma (por ejemplo, Python, Swift, Lua, Kotlin)
- Técnicas para aprender y dominar un lenguaje de programación específico de la plataforma

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enumerar las características de los lenguajes de scripting [Listar/Enumerar]
2. Describir el modelo de tres capas (three-tier) de la programación web [Describir]
3. Describir cómo se mantiene el estado en la programación web [Describir]

2.17.3. SPD/Frontend, Frameworks y Estándares Web (KA Core: 3 hrs)

Cubre los lenguajes de programación web, los frameworks, el modelo SaaS, los estándares web (DOM, accesibilidad) y las consideraciones de seguridad y privacidad.
Temas:
Core

- Lenguajes de programación web (por ejemplo, HTML5, JavaScript, Typescript, CSS)
- Plataformas, frameworks o meta-frameworks web 1. Servicios en la nube 2. API, Componentes Web (Web Components)
- Software como Servicio (SaaS)
- Estándares web como el modelo de objetos del documento (DOM), accesibilidad Accesibilidad y Diseño Inclusivo
- Consideraciones de Seguridad y Privacidad Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comparar y contrastar la programación web con la programación de propósito general [Comparar]
2. Describir las diferencias entre Software-como-Servicio (SaaS) y productos de software tradicionales [Describir]
3. Discutir cómo los estándares web impactan el desarrollo de software [Debatir]
4. Evaluar una aplicación web existente contra los estándares web actuales [Evaluar]

2.17.4. SPD/Arquitectura Cloud y Gestión de Datos (KA Core: 3 hrs)

Cubre el análisis de requisitos para aplicaciones web, los servicios de computación en la nube, la gestión de datos, las arquitecturas (microservicios, serverless, event-driven) y las soluciones de almacenamiento.
Temas:
Non Core

- Análisis de requisitos para aplicaciones web
- Servicios de computación Sistemas NoSQL 1. Alojamiento en la Nube (Cloud Hosting) 2. Escalabilidad (por ejemplo, Autoescalado (*Autoscaling*), Clústeres) 3. Estimación de costos para servicios

- Gestión de datos Conceptos Fundamentales de Sistemas de Bases de Datos 1. Residencia de datos (data residency): dónde se ubican los datos y qué rutas se pueden tomar para acceder a ellos 2. Integridad de datos (data integrity): garantizar que los datos sean accesibles y que se eliminen cuando sea requerido
- Arquitectura 1. Monolitos vs Microservicios 2. Micro-frontends 3. Arquitecturas Dirigidas por Eventos (Event-Driven) vs RESTful: ventajas y desventajas 4. Sin servidor (Serverless), computación en la nube bajo demanda
- Soluciones de almacenamiento Sistemas NoSQL, Bases de Datos Relacionales 1. Bases de datos relacionales 2. Bases de datos NoSQL

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Diseñar e implementar una aplicación basada en web utilizando un diseño de arquitectura de microservicios [Diseñar]
2. Describir las restricciones, como alojamiento, servicios y escalabilidad, relacionadas con las plataformas web [Describir]

2.17.5. SPD/Plataformas Móviles: Fundamentos del Desarrollo (KA Core: 2 hrs)

Temas:
Core

- Desarrollo con 1. Lenguajes de programación móvil 2. Entornos de programación móvil
- Restricciones de la plataforma móvil 1. Diseño de interfaz de usuario Comprender al Usuario: Objetivos individuales e interacciones con otros 2. Seguridad
- Acceso 1. Acceso a datos a través de APIs Construcción de Consultas 2. Diseño de puntos finales de API (endpoints) para aplicaciones móviles: trampas y consideraciones de diseño 3. Interfaces de red y web Modelado de Datos, Fundamentos de Redes y Comunicaciones

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Comparar la programación móvil con la programación de propósito general [Comparar]
2. Evaluar los pros y los contras del desarrollo de aplicaciones móviles nativas y multiplataforma [Evaluar]

2.17.6. SPD/Plataformas Móviles: Arquitectura y Restricciones (KA Core: 2 hrs)

Temas:
Non Core

- Desarrollo 1. Desarrollo nativo (native) versus multiplataforma (cross-platform) 2. Patrones de diseño/arquitectura de software para aplicaciones móviles Diseño de API y Modelado, Principios y Arquitectura de Software, Calidad y Evaluación del Diseño
- Restricciones de la plataforma móvil 1. Diseño de interfaz de usuario responsivo (responsive) Accesibilidad y Diseño Inclusivo 2. Heterogeneidad y movilidad de los dispositivos 3. Diferencias en las experiencias de usuario (por ejemplo, entre aplicaciones móviles y basadas en web) 4. Compensación (tradeoff) entre energía y rendimiento
- Especificación y pruebas Prácticas de Desarrollo de Software, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Diseñar la arquitectura de una aplicación móvil aplicando patrones de diseño de software apropiados, justificando la elección entre desarrollo nativo y multiplataforma [Diseñar]
2. Evaluar las restricciones de una plataforma móvil dada (interfaz responsiva, heterogeneidad de dispositivos, compensación energía-rendimiento) al especificar y probar una aplicación [Evaluar]

2.17.7. SPD/Plataformas Móviles: Capacidades y Computación Asíncrona (KA Core: 1 hr)

Temas:
Non Core

- Capacidades (*affordances*) de la computación móvil 1. Aplicaciones conscientes de la ubicación (*location-aware*) 2. Computación impulsada por sensores (por ejemplo, giroscopio, acelerómetro, datos de salud de un reloj) 3. Telefonía y mensajería instantánea 4. Realidad aumentada Inmersión
- Computación asíncrona Algoritmos 1. Diferencia con la programación síncrona tradicional 2. Manejo del éxito mediante devoluciones de llamada (*callbacks*) 3. Manejo de errores de forma asíncrona 4. Pruebas de código asíncrono y problemas típicos en las pruebas

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Desarrollar una aplicación móvil consciente de la ubicación con integración de API de datos [Crear]
2. Construir una aplicación móvil impulsada por sensores capaz de registrar datos en un servidor remoto [Crear]
3. Crear una aplicación de comunicación que incorpore telefonía y mensajería instantánea [Crear]

2.17.8. SPD/Plataformas Robóticas (KA Core: 4 hrs)

Temas:
Core

- Tipos de plataformas y dispositivos robóticos Robótica
- Sensores, cómputo embebido y efectores (actuadores) Computación Tangible/Física
- Lenguajes y bibliotecas específicos de robótica Robótica
- Arquitectura de software robótico (por ejemplo, usando el Sistema Operativo de Robot (ROS - *Robot Operating System*))
- Restricciones y consideraciones de diseño de la plataforma robótica Robótica
- Interconexiones con sistemas físicos o simulados Simulación, Computación Tangible/Física
- Algoritmos Robóticos Robótica, Animación por Computadora 1. Cinemática directa (*forward kinematics*) 2. Cinemática inversa (*inverse kinematics*) 3. Dinámica 4. Navegación y planificación de rutas (*path planning*) 5. Agarre (*grasping*) y manipulación
- Consideraciones de seguridad e interacción Equidad Digital y Accesibilidad, Impacto Social e IA, Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Diseñar e implementar una aplicación en una plataforma robótica dada [Diseñar]
2. Integrar un kit de robot basado en Arduino y programarlo para navegar por un laberinto [Integrar]
3. Comparar lenguajes y técnicas específicos de robótica con aquellos utilizados para el desarrollo de software de propósito general [Comparar]
4. Explicar la lógica detrás del diseño de la plataforma robótica y sus interconexiones con sistemas físicos o simulados [Explicar]
5. Dada una aplicación de alto nivel, diseñar una arquitectura de software de robot usando ROS especificando todos los componentes e interconexiones (temas ROS - ROS topics) para lograr esa aplicación [Diseñar]
6. Discutir las restricciones que una plataforma robótica dada impone a los desarrolladores [Debatir]

2.17.9. SPD/Características y Arquitecturas Embebidas (KA Core: 1 hr)

Cubre las características únicas de los sistemas embebidos frente a otros sistemas, las APIs para arquitecturas personalizadas (GPU, FPGA) y la evaluación de arquitecturas según requisitos.

Temas:
Core

- Introducción a las características únicas de los sistemas embebidos 1. Sistemas de tiempo real (*real-time*) vs tiempo real suave (*soft real-time*) y no tiempo real 2. Restricciones de recursos, como perfiles de memoria y plazos (*deadlines*) Jerarquía de Memoria
- API para arquitecturas personalizadas 1. Tecnología GPU Arquitecturas Heterogéneas, Sombreado y Renderizado Avanzado 2. Arreglos de Compuertas Programables en Campo (FPGA - *Field Programmable Gate Arrays*) Lógica Digital y Sistemas Digitales 3. Sistemas multiplataforma (*cross-platform*)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir las características únicas de los sistemas embebidos versus otros sistemas [Describir]
2. Identificar diferentes tipos de arquitecturas embebidas [Analizar]
3. Evaluar qué arquitectura es mejor para un conjunto dado de requisitos [Evaluar]

2.17.10. SPD/Sistemas, Programación y Sensores (KA Core: 2 hrs)

Cubre los sistemas embebidos con microcontroladores, interrupciones y eficiencia energética, la programación embebida, y la interfaz con sensores y actuadores.
Temas:
Core

- Sistemas Embebidos 1. Microcontroladores 2. Interrupciones y retroalimentación (*feedback*) 3. Manejadores de interrupciones en lenguajes de alto nivel Componentes Hardware y Lógica, Capas, Abstracciones y Escalabilidad 4. Interrupciones duras (*hard*) y suaves (*soft*) y salidas de trampa (*trap-exits*) Principios del Sistema Operativo 5. Interacción con hardware, actuadores y sensores 6. Eficiencia energética 7. Codificación débilmente temporizada (*loosely timed coding*) y sincronización 8. Adaptadores de software (software adapters)
- Programación embebida
- Sensores y actuadores

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Interfaz con sensores/actuadores [Programar]
2. Depurar un problema con una plataforma embebida existente [Analizar]
3. Diseñar y desarrollar software para interactuar y controlar hardware [Diseñar]

2.17.11. SPD/Tiempo Real, Memoria y Verificación (KA Core: 3 hrs)

Cubre los sistemas de tiempo real duro y suave, la gestión de recursos y memoria en tiempo real, las consideraciones de seguridad (safety), el análisis y verificación, y el diseño de aplicaciones embebidas.
Temas:
Core

- Sistemas de tiempo real duro (*hard*) vs sistemas de tiempo real suave (*soft*) Sistemas en Tiempo Real y Embebidos 1. Oportunidad (*timeliness*) 2. Sincronización/planificación del tiempo 3. Priorización 4. Latencia 5. Jitter de cómputo (*compute jitter*)
- Gestión de recursos en tiempo real
- Gestión de memoria 1. Mapeo de construcciones de programación (variable) a una ubicación de memoria Jerarquía de Memoria 2. Memoria compartida Gestión de Memoria 3. Gestión manual de memoria. 4. Recolección de basura (*garbage collection*) Traducción y Ejecución de Lenguajes
- Consideraciones de seguridad (*safety*) y análisis de seguridad Equidad Digital y Accesibilidad, Impacto Social e IA, Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo
- Análisis y verificación
- Diseño de aplicación

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Diseñar e implementar un pequeño sistema embebido para una plataforma dada (por ejemplo, un despertador inteligente o un dron) [Diseñar]
2. Diseñar métodos para sistemas de tiempo real [Diseñar]
3. Evaluar la planificación en tiempo real y el análisis de planificabilidad (*schedulability*) [Evaluar]
4. Evaluar la especificación y verificación formal de restricciones y propiedades de tiempo [Evaluar]

2.17.12. SPD/Historia, Tipos y Características de Plataformas (KA Core: 1 hr)

Cubre la evolución histórica de las plataformas de juegos, los tipos de plataformas típicas y sus características y restricciones técnicas.
Temas:
Core

- Evolución de las Plataformas de Juegos (por ejemplo, de Brown Box a Metaverso y más allá; mejora en las arquitecturas de computación (CPU y GPU); convergencia de plataformas y movilidad) Lógica Digital y Sistemas Digitales
- Plataformas de Juego Típicas (por ejemplo, Computadora Personal; Consola Doméstica; Consola Portátil; Máquina de Arcade; Televisión Interactiva; Teléfono Móvil; Tableta; Pantalla Montada en la Cabeza Integrada; Instalaciones Inmersivas y Simuladores; Dispositivos Habilitados para Internet de las Cosas; Sistemas CAVE; Navegadores Web; Sistemas de Transmisión Basados en la Nube)
- Características y Restricciones de Diferentes Plataformas de Juegos (por ejemplo, características: almacenamiento local, interconexión de redes, periféricos; rendimiento en tiempo de ejecución: frecuencia GPU/CPU, número de núcleos; conjuntos de chips: unidades de procesamiento de física, co-procesadores vectoriales; ancho de banda de expansión (PCIe); rendimiento de red (Ethernet); tipos y capacidades de memoria (DDR/GDDR); profundidad máxima de pila; consumo de energía; diseño térmico; Endian)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir las restricciones clave que una plataforma de juego específica probablemente impondrá a los desarrolladores [Describir]
2. Comparar las interfaces de diferentes plataformas de juegos, resaltando sus respectivas implicaciones para la interacción humano-computadora [Comparar]

2.17.13. SPD/Sensores, Controladores y Ecosistemas eSports (KA Core: 1 hr)

Cubre los sensores, controladores y actuadores típicos en plataformas de juego, y los ecosistemas de eSports incluyendo eventos, medios transmitidos y juego profesional.
Temas:
Core

- Sensores, Controladores y Actuadores Típicos (por ejemplo, diseños distintivos de sistemas de control - periféricos (ratón, teclado, joystick), controladores de juego, wearables, superficies interactivas; electrónica y hardware a medida; visión por computadora, seguimiento interno (*inside-out tracking*) y externo (*outside-in tracking*); electrónica y E/S habilitadas para IoT) Interacción

- Ecosistemas de eSports (por ejemplo, evolución del gameplay a través de plataformas; juegos y eSports; eventos de juegos como torneos LAN/arcade y eventos internacionales como la Serie Olímpica de eSports; medios transmitidos (*streamed*) y espectadores; tecnologías multimedia y gestión de transmisión; juego profesional; datos y aprendizaje automático para entrenamiento y coaching)

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar cómo los eSports se transmiten (streamed) a grandes audiencias a través de internet [Explicar]

2.17.14. SPD/Arquitecturas y Renderizado (KA Core: 1 hr)

Cubre las arquitecturas de CPU y GPU para juegos y las tuberías (pipelines) para renderizado gráfico y simulación física.

Temas:

Core

- Arquitecturas de CPU y GPU (por ejemplo, taxonomía de Flynn; paralelización; conjuntos de instrucciones; componentes estándar - arreglo de cómputo gráfico (*graphics compute array*), controlador de memoria gráfica (*graphics memory controller*), sistema básico de E/S de matriz de gráficos de video (*video graphics array basic input/output system*); interfaz de bus; unidad de gestión de energía (*power management unit*); unidad de procesamiento de video (*video processing unit*); interfaz de pantalla (*display interface*)) Arquitecturas Heterogéneas
- Tuberías (*pipelines*) para Renderizado Gráfico y Simulación Física (por ejemplo, basado en mosaicos (*tile-based*), modo inmediato (*immediate-mode*)) Renderizado Aplicado y Técnicas

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Recordar las características de las arquitecturas comunes de procesamiento gráfico de propósito general [Recordar/Evocar]
2. Identificar las etapas clave de la tubería de renderizado de modo inmediato (immediate-mode rendering pipeline) [Analizar]

2.17.15. SPD/Algoritmos, Medios y Mundos Virtuales (KA Core: 1 hr)

Cubre los algoritmos, estructuras de datos y funciones matemáticas comunes en juegos, y las representaciones de medios en mundos virtuales.

Temas:

Core

- Contextos Comunes para Algoritmos, Estructuras de Datos y Funciones Matemáticas (por ejemplo, bucles de juego (*game loops*); partición espacial, eliminación de viewport (*culling*) y nivel de detalle (*level of detail*); detección y resolución de colisiones (*collision detection and resolution*); simulación física; comportamiento para agentes inteligentes; generación procedural de contenido (*procedural content generation*)) Estructuras de Datos Fundamentales, Algoritmos Fundamentales, Algoritmos Avanzados, Matemáticas Discretas
- Representaciones de Medios en Mundos Virtuales (por ejemplo, E/S y técnicas de cómputo para mundos virtuales: audio; música; sprites; modelos y texturas; texto; diálogo; multimedia (por ejemplo, olfato, táctil)) Conceptos Fundamentales de Gráficos y Técnicas Interactivas

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Traducir funciones matemáticas complejas a código fuente de buen rendimiento [Traducir]
2. Optimizar una función para la detección de colisiones en un entorno simulado [Crear]

2.17.16. SPD/Lenguajes, APIs y Herramientas de Desarrollo (KA Core: 1 hr)

Cubre los lenguajes de programación y sombreado, las APIs y bibliotecas gráficas, y las herramientas y entornos de desarrollo para juegos.

Temas:

Core

- Herramientas y Técnicas de Desarrollo de Juegos 1. Lenguajes de Programación (por ejemplo, C++; C#; Lua; Python; JavaScript) 2. Lenguajes de Sombreado (*Shader Languages*) (por ejemplo, HLSL, GLSL; Shader Graph) 3. Bibliotecas y APIs Gráficas (por ejemplo, DirectX; SDL; OpenGL; Metal; Vulkan; WebGL). Renderizado Aplicado y Técnicas, Diseño de Sistemas 4. Herramientas y Entornos de Desarrollo Comunes (por ejemplo, IDEs; Depuradores (*Debuggers*); Perfiladores (*Profilers*); Sistemas de Control de Versiones incluyendo aquellos que manejan activos binarios; Kits de Desarrollo y Kits de Producción/Consumidor; Emuladores) Prácticas de Desarrollo de Software, Entorno de Desarrollo e IDE, Control de Versiones y CI/CD

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Usar una API gráfica estándar de la industria para renderizar un modelo 3D en una escena virtual [Usar]
2. Modificar un sombreador (shader) para cambiar un efecto visual de acuerdo con los requisitos establecidos [Crear]
3. Recomendar un conjunto apropiado de herramientas y técnicas de desarrollo para implementar un juego de un género particular para una plataforma dada [Recomendar]

2.17.17. SPD/Motores de Juego (KA Core: 1 hr)

Cubre los motores de juego de código abierto y las técnicas de producción, garantía de calidad, optimización, portado e internacionalización.

Temas:

Core

- Motores de Juego (*Game Engines*) 1. Motores de Juego Abiertos (por ejemplo, Unreal; Unity; Godot; CryEngine; Phyre; Source 2; Pygame y Ren'Py; Phaser; Twine; Spring RTS) 2. Técnicas (por ejemplo, Ideación, Prototipado, Diseño e Implementación Iterativos, Compilación de Ejecutables (*Compiling Executable Builds*), Operaciones de Desarrollo y Garantía de Calidad - Pruebas de Juego (*Play Testing*) y Pruebas Técnicas (*Technical Testing*), Perfilado (*Profiling*); Optimización, Portado (*Porting*); Internacionalización y Localización, Redes) Rendimiento y Eficiencia Energética, Ingeniería de Requisitos, Evolución y Estimación de Requisitos

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Implementar un juego para una plataforma particular según la especificación [Implementar]
2. Evaluar el rendimiento en tiempo de ejecución y de memoria de un juego usando una herramienta y entorno de desarrollo estándar de la industria [Evaluar (valorar)]
3. Discutir los desafíos clave en hacer un juego digital que sea compatible multiplataforma (cross-platform) [Debatir]

2.17.18. SPD/Fundamentos del Diseño de Juegos (KA Core: 1 hr)

Cubre el vocabulario y modelos del diseño de juegos, el pensamiento de diseño y el diseño de experiencia centrado en el usuario.

Temas:

Core

- Vocabulario del Diseño de Juegos (por ejemplo, definiciones de juego; modelo mecánica-dinámica-estética (*mechanics-dynamics-aesthetics model*); terminología de la industria; diseño de experiencia (*experience design*); modelos de experiencia y emoción)
- Pensamiento de Diseño (*Design Thinking*) y Diseño de Experiencia Centrado en el Usuario (por ejemplo, métodos de diseño de juegos; iteración, incremento y el doble diamante (*double-diamond*); fases de pre- y post-producción; garantía de calidad, incluyendo pruebas alfa y beta; participación de partes interesadas y clientes; gestión de la comunidad) Diseño de API y Modelado, Principios y Arquitectura de Software, Calidad y Evaluación del Diseño

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Expresar cómo los desarrolladores de juegos pueden mejorar la accesibilidad de una interfaz de juego [Demostrar]

2.17.19. SPD/Géneros, Audiencias y Alcance Social (KA Core: 1 hr)

Cubre los géneros de juegos, las audiencias y taxonomías de jugadores, y la proliferación de tecnologías de juegos a dominios más allá del entretenimiento.

Temas:

Core

- Géneros de Juegos (por ejemplo, aventura; simulador de caminata (*walking simulator*); disparos en primera persona (*first-person shooter*); estrategia en tiempo real (*real-time strategy*); arena de batalla multijugador en línea (MOBA); juego de rol (rpg))

- Audiencias y Taxonomías de Jugadores (por ejemplo, personas que juegan videojuegos; diversidad y ampliación de la participación; placeres, tipos de jugadores y preferencias; Bartle, Yee) Comprender al Usuario: Objetivos individuales e interacciones con otros
- Proliferación de Tecnologías de Juegos a Dominios Más Allá del Entretenimiento (por ejemplo, Educación y Entrenamiento; Juegos Serios (*Serious Games*); Producción Virtual; eSports; Gamificación; Diseño de Experiencias Inmersivas; Práctica de la Industria Creativa; Práctica Artística; Retórica Procedural (*Procedural Rhetoric*)) Aplicaciones e Impacto Social de la IA, Aplicaciones Adicionales e Impacto Social de la IA

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Crear formas novedosas de gameplay utilizando plataformas de juego de frontera [Crear]

2.17.20. SPD/Plataformas Interactivas de Datos y Análisis (KA Core: 2 hrs)

Cubre las plataformas de análisis de datos interactivo como cuadernos Jupyter, Google Colab, R, plataformas SQL en la nube y herramientas de diseño con bucles de retroalimentación de baja latencia.
Temas:
Non Core

- Plataformas de Cuadernos Computacionales (por ejemplo, Cuadernos Jupyter; Google Colab; R; SPSS; Observable)
- Plataformas de Análisis de Datos y SQL en la Nube (por ejemplo, BigQuery) Construcción de Consultas 1. Apache Spark 2. Visualizaciones de Datos Visualización
- Presentaciones Interactivas Respaldadas por Datos
- Herramientas de Diseño con Bucles de Retroalimentación de Baja Latencia (*Low-Latency Feedback Loops*) 1. Herramientas de renderizado 2. Herramientas de diseño gráfico

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Analizar grandes conjuntos de datos de forma interactiva [Analizar]
2. Crear una pista de acompañamiento para una interpretación musical, como la codificación en vivo (*live coding*) [Crear]
3. Crear cuadernos computacionales convincentes que construyan una narrativa para un objetivo/historia periodística dada [Crear]
4. Implementar código interactivo que use un conjunto de datos y genere gráficos exploratorios [Implementar]
5. Contextualizar los atributos de diferentes estilos de análisis de datos, como interactivo versus tubería (*pipeline*) de ingeniería [Contextualizar]
6. Escribir un programa usando una plataforma de computación de cuadernos (por ejemplo, búsqueda, ordenación o manipulación de grafos) [Escribir]

2.17.21. SPD/Programación con IA y Plataformas Cuánticas (KA Core: 1 hr)

Cubre la programación mediante prompts con modelos de lenguaje grande (LLMs) e IA generativa, y las plataformas cuánticas para programar operadores de lógica cuántica y servicios cuánticos disponibles.
Temas:
Non Core

- Programación mediante prompts (Prompt programming) 1. IA generativa (por ejemplo, ChatGPT de OpenAI, Codex de OpenAI, Copilot de GitHub) y los Modelos de Lenguaje Grande (LLMs) son accedidos/interactuados
- Plataformas Cuánticas Arquitecturas Cuánticas 1. Programar operadores de lógica cuántica en máquinas cuánticas. 2. Usar API para servicios cuánticos disponibles 3. Análisis de señales/Análisis de Fourier/Procesamiento de señales (para composición musical, análisis de audio/RF) Procesamiento de Imágenes

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Crear un programa que realice una tarea usando sistemas de Modelos de Lenguaje Grande (LLM) [Crear]
2. Contrastar un programa desarrollado por una plataforma de IA y por un humano [Contrastar]
3. Implementar un sistema que interactúe con un humano sin usar una pantalla [Implementar]
4. Demostrar un resultado de puerta cuántica (*quantum gate*) usando una plataforma cuántica [Demostrar]

2.17.22. SPD/SPD-SEP/Móvil

Temas:
Non Core

- Privacidad y protección de datos
- Accesibilidad en el diseño móvil
- Seguridad y ciberseguridad
- Impactos sociales de la tecnología móvil
- Uso ético de la IA y los algoritmos

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Comprender y defender las responsabilidades éticas para salvaguardar la privacidad del usuario y la protección de datos en aplicaciones móviles [Evaluar (valorar)]
2. Diseñar aplicaciones móviles con la accesibilidad en mente, asegurando un uso efectivo por personas con discapacidades [Diseñar]
3. Demostrar competencia en prácticas de codificación segura para mitigar riesgos asociados con diversas amenazas de seguridad en el desarrollo móvil [Demostrar]
4. Analizar los impactos sociales más amplios de la tecnología móvil, incluyendo su influencia en los patrones de comunicación, relaciones y salud mental [Analizar]
5. Comprender las consideraciones éticas del uso de IA en aplicaciones móviles, asegurando algoritmos imparciales y justos [Explicar]

2.17.23. SPD/SPD-SEP Web

Temas:
Non Core

- Preocupaciones de privacidad con las aplicaciones móviles
- Diseñar para la inclusividad y la accesibilidad
- Uso ético de la IA en aplicaciones móviles
- Desarrollo sostenible de aplicaciones y alojamiento de servidores
- Evitar spam o notificaciones intrusivas
- Abordar el ciberacoso y el acoso (*cyberbullying and harassment*)
- Promover comunidades en línea positivas
- Monetización y publicidad
- Uso ético de la gamificación

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Comprender cómo la computación móvil impacta las comunicaciones y el flujo de información dentro de la sociedad [Analizar]
2. Diseñar aplicaciones móviles que hayan hecho las tareas diarias más fáciles/rápidas [Diseñar]
3. Reconocer cómo la ubicuidad de la computación móvil ha afectado el equilibrio entre el trabajo y la vida personal [Reconocer]
4. Comprender cómo la computación móvil impacta el monitoreo de la salud y los servicios de atención médica [Analizar]
5. Definir cómo las aplicaciones móviles se usan para educar sobre y ayudar a lograr los objetivos de sostenibilidad de la ONU [Definir]

2.17.24. SPD/Propiedad Intelectual en los Juegos

Cubre los derechos de propiedad intelectual en las industrias creativas, incluyendo copyright, marcas, licenciamiento y cumplimiento.

Temas:

Non Core

- Derechos de Propiedad Intelectual en las Industrias Creativas 1. Propiedad de la Propiedad Intelectual: derechos de autor (*copyright*), marca registrada (*trademark*); derecho de diseño (*design right*), patente (*patent*), secreto comercial (*trade secret*), ley civil versus penal; acuerdos internacionales; generación procedural de contenido y las implicaciones de la inteligencia artificial generativa 2. Licenciamiento (Licensing): Excepciones de uso y uso justo (*fair usage*); acuerdos de licencia de código abierto (*open-source*); licenciamiento propietario y a medida (*bespoke*); cumplimiento (*enforcement*)

Aprendizaje esperado (*Learning Outcomes*):

NonCore:

1. Discutir cómo los creadores pueden proteger su propiedad intelectual [Debatir]
2. Comparar opciones para el licenciamiento de código abierto de una herramienta de desarrollo de juegos [Comparar]

2.17.25. SPD/Accesibilidad e Interfaz del Juego

Cubre la usabilidad y accesibilidad de las interfaces de juego, incluyendo métodos de evaluación heurística y diseño para jugadores con discapacidades.

Temas:

Non Core

- Acceso Justo al Juego (*Fair Access to Play*) 1. Usabilidad de la Interfaz de Juego: requisitos del usuario, capacidades (*affordances*), diseño ergonómico, investigación de usuarios, medición de experiencia y métodos de evaluación heurística para juegos 2. Accesibilidad de la Interfaz de Juego: formas de discapacidad y deficiencia (impairment and disability); medios para facilitar el acceso al juego; diseño universal; requisitos legislados para plataformas de juegos; evaluación de cumplimiento; desafíos en mecánicas de juego y acceso

Aprendizaje esperado (*Learning Outcomes*):

NonCore:

1. Identificar trampas comunes en las interfaces de juego que excluyen a jugadores con visión deficiente o no funcional [Analizar]
2. Describir cómo la evaluación heurística puede usarse para identificar problemas de usabilidad en interfaces de juego [Describir]
3. Recomendar cambios a una interfaz de juego específica para mejorar el acceso a jugadores que son sordos o cuya audición está de otra manera deficiente [Recomendar]

2.17.26. SPD/Salud, Seguridad y Sostenibilidad

Cubre las lesiones relacionadas con el juego y su mitigación, la evaluación de riesgos para eventos y fabricación, la salud mental de los jugadores, y la cadena de suministro y sostenibilidad del hardware de plataforma.

Temas:

Non Core

- Salud y Seguridad Relacionadas con los Juegos 1. Lesiones en el Juego: formas de mitigar lesiones comunes en la parte superior del cuerpo, como lesiones por esfuerzo repetitivo (repetitive strain injury); psicología del ejercicio y fisioterapia en eSports 2. Evaluación de Riesgos para Eventos y Fabricación: control de sustancias peligrosas para la salud (COSHH); seguridad contra incendios; seguridad eléctrica y electrónica; evaluación de riesgos para juegos y eventos de juegos; evaluación de riesgos para la fabricación 3. Salud Mental: motivación para jugar; gamificación y diseño lúdico (*gameful design*); psicología del juego - trastorno por juego en internet (*internet gaming disorder*)
- Cadena de Suministro y Sostenibilidad del Hardware de Plataforma 1. Ciclo de Vida de la Plataforma: composición de la plataforma - materiales, ensamblaje; excavación y procesamiento

de minerales; uso de energía; reciclaje; obsolescencia programada (*planned obsolescence*). 2. Esclavitud Moderna (Modern Slavery): cadenas de suministro; trabajo forzado y derechos civiles; condiciones laborales; detección y remisión (*remission*); organismos de certificación y esfuerzos benéficos.

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Explicar por qué las lesiones en la parte superior del cuerpo son comunes en los eSports [Explicar]
2. Modificar una política para un evento de LAN party para incluir mitigaciones que reduzcan el riesgo de incendio [Diseñar]
3. Diseñar una estrategia de gamificación para motivar el juego serio (*serious play*) para un juego de concientización [Diseñar]
4. Evaluar la idoneidad de dos fabricantes para producir una nueva consola de juegos [Evaluar (valorar)]
5. Discutir si los juegos son adictivos [Debatir]

2.17.27. SPD/Representación, Inclusión e Igualdad

Cubre la representación en los medios y la industria de los juegos, la inclusión de audiencias diversas, la igualdad y la movilidad social en la industria, y el uso de juegos para la defensa de causas sociales.

Temas:
Non Core

- Representación en los Medios y la Industria 1. Inclusión: identidad e identificación; inclusión de una amplia gama de personajes para audiencias diversas; representación en los medios y sus efectos; alfabetización mediática (*media literacy*); análisis de contenido; estereotipos; sexualización 2. Igualdad (Equality): historias y controversias, como gamergate, calidad de vida en la industria, discurso y conducta profesional en contextos empresariales, trayectorias hacia carreras en desarrollo de juegos, movilidad social, la experiencia de desarrolladores de diferentes orígenes e identidades, género y tecnología

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Discutir cómo reformar personajes y diálogos en una escena para reducir la amenaza del estereotipo (*stereotype threat*) [Debatir]
2. Ilustrar cómo la representación de la raza en un juego puede influir en el riesgo de exclusión social en la comunidad en línea asociada al juego [Aplicar]
3. Analizar el papel de las políticas de contratación de empresas y la defensa (*advocacy*) en la movilidad social [Analizar]
4. Sugerir cómo la representación de las mujeres en los videojuegos influye en cómo los jugadores perciben a los miembros de esos grupos [Evaluar]
5. Crear un videojuego que defienda con éxito la ciencia del clima [Crear]

2.17.28. SPD/SPD-SEP/Robótica

Temas:
Non Core

- Imparcialidad (*fairness*), transparencia y rendición de cuentas (*accountability*) en algoritmos robóticos
- Mitigar sesgos (biases) en la toma de decisiones de robots
- Seguridad pública en espacios compartidos con robots
- Cumplimiento con las leyes de protección de datos
- Consentimiento del paciente y confianza en los robots médicos

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Identificar instancias de sesgo en algoritmos robóticos y proponer estrategias para mitigarlos [Analizar]

2. Evaluar y criticar sistemas robóticos por consideraciones éticas y de imparcialidad, sugiriendo mejoras donde sea necesario [Evaluar]
3. Analizar ejemplos del mundo real de sesgos en la toma de decisiones de robots y desarrollar estrategias para reducir el sesgo en sistemas robóticos [Analizar]
4. Evaluar los riesgos potenciales asociados con robots en espacios públicos y proponer medidas de seguridad para mitigar esos riesgos [Evaluar (valorar)]
5. Evaluar el impacto del consentimiento del paciente y la confianza en la efectividad de los despliegues de robots médicos en contextos de atención médica [Evaluar]

2.17.29. SPD/SPD-SEP/Interactiva

Temas:
Non Core

- Pautas éticas al usar modelos de IA para ayudar en el periodismo y la creación de contenido
- Rendición de cuentas (*accountability*) por las salidas generadas por IA
- Comportamiento entre programadores de prompts (*prompt programmers*) y desarrolladores de IA
- Confianza con el público al usar modelos de IA

Aprendizaje esperado (*Learning Outcomes*):
NonCore:

1. Indicar un marco para la rendición de cuentas (*accountability*) en el despliegue de modelos de IA, incluyendo documentación clara y atribución [Evaluar]
2. Discutir códigos de conducta éticos y estándares profesionales relevantes para la programación de prompts (*prompt programming*) y el desarrollo de IA [Debatir]
3. Crear planes y materiales de comunicación para educar al público sobre las capacidades, limitaciones y salvaguardas éticas de la IA [Crear]

Capítulo 3

Cuerpo del conocimiento de Ciencias Básicas para Computación

Las 19 áreas de conocimiento en Ciencias Básicas para Computación son:

3.1 Cálculo y Análisis (CAN) (Pág. 156)
- 3.1.1 Integrales de Línea y Campos Vectoriales (Pág. 156)
- 3.1.2 Integrales Múltiples (Pág. 157)
- 3.1.3 Transformaciones Multivariables (Pág. 157)
- 3.1.4 Integrales de Superficie y Teoremas Fundamentales (Pág. 157)
- 3.1.5 Funciones Vectoriales de una Variable Real (Pág. 158)
- 3.1.6 Ecuaciones Paramétricas y Coordenadas Polares (Pág. 158)
- 3.1.7 Límites y Continuidad (Pág. 158)
- 3.1.8 Cálculo Diferencial (Pág. 159)
- 3.1.9 Antiderivadas y Técnicas de Integración (Pág. 159)
- 3.1.10 Integrales Impropias (Pág. 159)
- 3.1.11 Sucesiones, Series y Series de Potencias (Pág. 160)
- 3.1.12 Cálculo Diferencial Multivariable (Pág. 160)
- 3.1.13 Análisis Armónico y Teoría de Fourier (Pág. 160)

3.2 Álgebra y Teoría de Números (ANT) (Pág. 161)
- 3.2.1 Matrices y Sistemas de Ecuaciones Lineales (Pág. 161)
- 3.2.2 Espacios Vectoriales y Transformaciones Lineales (Pág. 161)
- 3.2.3 Valores Propios, Producto Interno y Formas Cuadráticas (Pág. 162)
- 3.2.4 Teoría de Grupos (Pág. 162)
- 3.2.5 Anillos, Cuerpos y Teoría de Galois (Pág. 162)
- 3.2.6 Criptografía y Teoría de Códigos (Pág. 163)

3.3 Matemática Discreta y Combinatoria (DMC) (Pág. 163)
- 3.3.1 Combinatoria Enumerativa (Pág. 163)
- 3.3.2 Teoría de Grafos (Pág. 164)
- 3.3.3 Optimización Discreta (Pág. 164)
- 3.3.4 Lógica Matemática y Teoría de Conjuntos (Pág. 164)

3.4 Probabilidad y Estadística (PST) (Pág. 165)
- 3.4.1 Axiomas de Probabilidad y Distribuciones (Pág. 165)
- 3.4.2 Teoremas Límite (Pág. 165)
- 3.4.3 Procesos Estocásticos (Pág. 166)
- 3.4.4 Teoría de la Estimación (Pág. 166)
- 3.4.5 Inferencia Bayesiana (Pág. 166)
- 3.4.6 Análisis de Series Temporales (Pág. 167)

3.5 Análisis Numérico y Científico (NSA) (Pág. 167)
- 3.5.1 Teoría de Aproximación e Interpolación (Pág. 168)
- 3.5.2 Integración Numérica y Cuadratura (Pág. 168)
- 3.5.3 Análisis de Errores y Aritmética de Punto Flotante (Pág. 168)
- 3.5.4 Búsqueda de Raíces de Ecuaciones No Lineales (Pág. 169)
- 3.5.5 Álgebra Lineal Numérica (Pág. 169)
- 3.5.6 Métodos Numéricos para Ecuaciones Diferenciales (Pág. 169)
- 3.5.7 Algoritmos de Optimización (Pág. 170)
- 3.5.8 Método de Elementos Finitos (Pág. 170)
- 3.5.9 Computación Paralela y de Alto Rendimiento (Pág. 171)

3.6 Modelado Matemático y Simulación (MMS) (Pág. 171)
- 3.6.1 Ecuaciones Diferenciales Ordinarias de Primer Orden (Pág. 171)
- 3.6.2 Ecuaciones Diferenciales de Orden Superior y Sistemas (Pág. 172)
- 3.6.3 Sistemas Dinámicos y Caos (Pág. 172)
- 3.6.4 Teoría Matemática del Control (Pág. 172)
- 3.6.5 Ecuaciones Diferenciales Estocásticas (Pág. 173)

3.7 Fundamentos Matemáticos de la Ciencia de Datos (FDS) (Pág. 173)
- 3.7.1 Teoría del Aprendizaje Estadístico (Pág. 173)
- 3.7.2 Optimización para el Aprendizaje Automático (Pág. 174)
- 3.7.3 Teoría de la Información (Pág. 174)
- 3.7.4 Reducción de Dimensionalidad (Pág. 175)
- 3.7.5 Fundamentos Matemáticos del Aprendizaje Profundo (Pág. 175)

3.8 Información Cuántica y Computación (QIC) (Pág. 175)
- 3.8.1 Fundamentos de la Mecánica Cuántica para la Computación (Pág. 176)
- 3.8.2 Circuitos y Puertas Cuánticas (Pág. 176)
- 3.8.3 Algoritmos Cuánticos (Pág. 176)
- 3.8.4 Entrelazamiento y Desigualdades de Bell (Pág. 177)

3.9 Electromagnetismo (ELM) (Pág. 177)
- 3.9.1 Electrostática: Campo Eléctrico y Ley de Gauss (Pág. 178)
- 3.9.2 Corriente Eléctrica, Circuitos y Leyes de Kirchhoff (Pág. 178)
- 3.9.3 Ecuaciones de Maxwell y Transformaciones de Norma (Pág. 178)

3.10 Termodinámica y Mecánica Estadística (TSM) (Pág. 179)
- 3.10.1 Leyes de la Termodinámica y Funciones de Estado (Pág. 179)
- 3.10.2 Teoría Cinética de los Gases (Pág. 179)
- 3.10.3 Máquinas de Calor, Entropía y la Segunda Ley (Pág. 180)

3.11 Física Computacional y Basada en Datos (CDP) (Pág. 180)
- 3.11.1 Métodos Numéricos para Ecuaciones Diferenciales en Física (Pág. 180)
- 3.11.2 Métodos de Monte Carlo y Simulaciones (Pág. 181)
- 3.11.3 Aprendizaje Automático para el Descubrimiento Físico (Pág. 181)

3.12 Mecánica Clásica (CME) (Pág. 181)
- 3.12.1 Movimiento Rotacional y Momento Angular (Pág. 182)
- 3.12.2 Sistemas de Partículas y Momento Lineal (Pág. 183)
- 3.12.3 Cinemática y Dinámica de Partículas (Pág. 183)

- 3.12.4 Trabajo, Energía y Leyes de Conservación (Pág. 184)
- 3.12.5 Dinámica de Cuerpos Rígidos (Pág. 184)
- 3.12.6 Oscilaciones y Ondas (Pág. 185)
- 3.12.7 Mecánica Lagrangiana y Hamiltoniana (Pág. 185)
- 3.12.8 Elasticidad (Pág. 185)
- 3.12.9 Fluidos: Hidrostática, Hidrodinámica y Viscosidad (Pág. 186)
- 3.12.10 Vibraciones y Ondas (Pág. 186)
- 3.12.11 Fenómeno de Transporte (Pág. 187)

3.13 Química General (GCH) (Pág. 187)
- 3.13.1 Estructura Atómica (Pág. 188)
- 3.13.2 Enlace Químico (Pág. 188)
- 3.13.3 Materia y Energía (Pág. 189)
- 3.13.4 Estados de la Materia (Pág. 189)
- 3.13.5 Estequiometría (Pág. 189)
- 3.13.6 Química Ambiental y del Agua (Pág. 190)
- 3.13.7 Polímeros y Geosintéticos (Pág. 190)
- 3.13.8 Equilibrio Químico (Pág. 191)
- 3.13.9 Equilibrio Iónico e Hidrólisis de Sales (Pág. 191)
- 3.13.10 Sales, Solubilidad y Equilibrios de Precipitación (Pág. 191)
- 3.13.11 Electroquímica y Electrólisis (Pág. 191)
- 3.13.12 Química del Cemento (Pág. 192)

3.14 Química Computacional y Teórica (CTC) (Pág. 192)
- 3.14.1 Modelado Molecular y Simulación (Pág. 192)
- 3.14.2 Aprendizaje Automático en el Descubrimiento Químico (Pág. 193)
- 3.14.3 Diseño Computacional de Fármacos (Pág. 193)

3.15 Bioquímica y Biología Molecular (BMB) (Pág. 194)
- 3.15.1 Estructura y Función de las Biomoléculas (Pág. 194)
- 3.15.2 Genética Molecular y Enzimología (Pág. 195)
- 3.15.3 Bioinformática y Biología Computacional (Pág. 195)

3.16 Teoría de la Probabilidad (PRO) (Pág. 196)
- 3.16.1 Espacios Muestrales y Eventos (Pág. 196)
- 3.16.2 Axiomas y Propiedades de la Probabilidad (Pág. 196)
- 3.16.3 Probabilidad Condicional e Independencia (Pág. 197)
- 3.16.4 Variables Aleatorias Discretas y Distribuciones (Pág. 197)
- 3.16.5 Variables Aleatorias Continuas y Distribuciones (Pág. 197)
- 3.16.6 Distribuciones Conjuntas, Marginales y Condicionales (Pág. 198)
- 3.16.7 Teoremas Límite: LGN y TCL (Pág. 198)

3.17 Estadística Descriptiva y Análisis de Datos (DSC) (Pág. 199)
- 3.17.1 Resúmenes Numéricos: Posición, Dispersión y Forma (Pág. 199)

3.18 Inferencia Estadística (INF) (Pág. 199)
- 3.18.1 Estimación Puntual (Pág. 199)
- 3.18.2 Estimación por Intervalos e Intervalos de Confianza (Pág. 200)
- 3.18.3 Fundamentos de las Pruebas de Hipótesis (Pág. 200)
- 3.18.4 Inferencia Bayesiana (Pág. 200)

3.19 Regresión y Modelos Estadísticos (REG) (Pág. 201)
- 3.19.1 Regresión Lineal Simple (Pág. 201)

3.1. Cálculo y Análisis (CAN)

Esta área cubre el estudio fundamental del cambio, los límites y la integración. Abarca desde el cálculo diferencial e integral de una variable hasta el análisis multivariable y los métodos armónicos, proporcionando el instrumental analítico esencial para el análisis de algoritmos, el procesamiento de señales y el aprendizaje automático.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.1.1 Integrales de Línea y Campos Vectoriales (Pág. 156)	1	1
⭐ 3.1.2 Integrales Múltiples (Pág. 157)	1	1
⭐ 3.1.3 Transformaciones Multivariables (Pág. 157)	1	1
⭐ 3.1.4 Integrales de Superficie y Teoremas Fundamentales (Pág. 157)	1	1
⭐ 3.1.5 Funciones Vectoriales de una Variable Real (Pág. 158)	1	1
⭐ 3.1.6 Ecuaciones Paramétricas y Coordenadas Polares (Pág. 158)	1	1
⭐ 3.1.7 Límites y Continuidad (Pág. 158)	1	1
⭐ 3.1.8 Cálculo Diferencial (Pág. 159)	1	1
⭐ 3.1.9 Antiderivadas y Técnicas de Integración (Pág. 159)	1	1
⭐ 3.1.10 Integrales Impropias (Pág. 159)	1	1
⭐ 3.1.11 Sucesiones, Series y Series de Potencias (Pág. 160)	1	1
⭐ 3.1.12 Cálculo Diferencial Multivariable (Pág. 160)	1	1
⭐ 3.1.13 Análisis Armónico y Teoría de Fourier (Pág. 160)	1	1

Cuadro 3.1: Lista de KUs del área de Cálculo y Análisis.

3.1.1. CAN/Integrales de Línea y Campos Vectoriales (CS Core: 1 hr, KA Core: 1 hr)

Integrales de línea escalares y vectoriales, campos conservativos, funciones potencial, independencia de la trayectoria y el Teorema de Green.

Temas:

Core

- Campos vectoriales: definición, visualización, divergencia y rotacional
- Integrales de línea de funciones escalares: longitud de arco y masa

- Integrales de línea de campos vectoriales: trabajo y circulación
- Campos conservativos, funciones potencial e independencia de la trayectoria
- Teorema de Green y sus aplicaciones en el plano

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Calcular integrales de línea escalares y vectoriales sobre curvas orientadas [Usar]
2. Determinar si un campo vectorial es conservativo y construir su función potencial [Usar]
3. Aplicar el Teorema de Green para evaluar integrales de línea mediante integrales dobles [Evaluar]

3.1.2. CAN/Integrales Múltiples (CS Core: 1 hr, KA Core: 1 hr)

Integración de funciones de varias variables: integrales dobles y triples en coordenadas cartesianas, polares, cilíndricas y esféricas, con aplicaciones.
Temas:
Core

- Integrales dobles sobre rectángulos y regiones generales; integrales iteradas (teorema de Fubini)
- Integrales dobles en coordenadas polares y fórmula de cambio de variable
- Integrales triples en coordenadas cartesianas y aplicaciones a volumen y masa
- Integrales triples en coordenadas cilíndricas y esféricas
- Aplicaciones: área, volumen, centros de masa y momentos de inercia

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Evaluar integrales dobles y triples sobre regiones generales usando integrales iteradas [Usar]
2. Aplicar cambios de coordenadas en sistemas polares, cilíndricos y esféricos para simplificar integrales múltiples [Usar]
3. Calcular áreas, volúmenes y centros de masa usando integrales múltiples [Evaluar]

3.1.3. CAN/Transformaciones Multivariables (CS Core: 1 hr, KA Core: 1 hr)

Funciones vectoriales de variables vectoriales: diferenciabilidad, matriz jacobiana, cambio de variable y los teoremas de la función inversa y de la función implícita.
Temas:
Core

- Transformaciones de $\mathbb{R}^n$ a $\mathbb{R}^m$ y su diferenciabilidad
- Matriz jacobiana y determinante jacobiano
- Cambio de variable en coordenadas polares, cilíndricas y esféricas
- Teorema de la función inversa
- Teorema de la función implícita en contextos multivariables

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir transformaciones entre sistemas de coordenadas usando la matriz jacobiana [Familiarizarse]
2. Calcular el jacobiano y aplicar cambio de variable para evaluar integrales múltiples [Usar]
3. Aplicar los teoremas de la función inversa y de la función implícita para analizar invertibilidad local [Evaluar]

3.1.4. CAN/Integrales de Superficie y Teoremas Fundamentales (CS Core: 1 hr, KA Core: 1 hr)

Superficies paramétricas, integrales de superficie escalares y de flujo, rotacional, divergencia y los teoremas fundamentales de Stokes y Gauss.
Temas:
Core

- Superficies paramétricas, plano tangente y área de superficie
- Integrales de superficie de funciones escalares

- Integrales de flujo: integrales de superficie de campos vectoriales
- Teorema de Stokes: relación entre integrales de línea e integrales de superficie
- Teorema de la Divergencia (Gauss): relación entre flujo e integral de volumen

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Parametrizar superficies y evaluar integrales de superficie escalares y de flujo [Usar]
2. Aplicar los Teoremas de Stokes y de la Divergencia para reducir integrales complejas [Evaluar]
3. Relacionar propiedades diferenciales locales (rotacional, divergencia) con el comportamiento global mediante teoremas integrales [Evaluar]

3.1.5. CAN/Funciones Vectoriales de una Variable Real (CS Core: 1 hr, KA Core: 1 hr)

Curvas en el espacio como funciones vectoriales de una variable real, incluyendo geometría diferencial de curvas: triedro de Frenet, curvatura y torsión.
Temas:
Core

- Definición, dominio y rango de funciones vectoriales de una variable real
- Límites, continuidad y derivadas de funciones vectoriales
- Longitud de arco y parametrización natural de una curva
- Vectores tangente, normal y binormal; triedro móvil de Frenet
- Curvatura, torsión y fórmulas de Frenet-Serret

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar límites, continuidad y derivadas de funciones vectoriales de una variable real [Familiarizarse]
2. Calcular vectores tangente, normal y binormal y reparametrizar una curva por longitud de arco [Usar]
3. Calcular curvatura y torsión usando las fórmulas de Frenet-Serret [Evaluar]

3.1.6. CAN/Ecuaciones Paramétricas y Coordenadas Polares (CS Core: 1 hr, KA Core: 1 hr)

Curvas paramétricas y coordenadas polares, usadas para describir trayectorias y geometrías en cinemática y diseño de ingeniería.
Temas:
Core

- Curvas paramétricas y su cálculo diferencial, aplicadas a trayectorias en cinemática
- Coordenadas polares y su conversión a coordenadas cartesianas

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Calcular la velocidad y aceleración de una trayectoria descrita paramétricamente [Usar]
2. Convertir entre coordenadas polares y cartesianas al modelar geometrías de ingeniería [Usar]

3.1.7. CAN/Límites y Continuidad (CS Core: 1 hr, KA Core: 1 hr)

Estudio riguroso de los límites y la continuidad de funciones de una variable, incluyendo la definición épsilon-delta, las leyes de los límites y los teoremas fundamentales.
Temas:
Core

- Definición épsilon-delta de un límite y técnicas de demostración formal
- Leyes de los límites, límites laterales y límites en el infinito
- Continuidad: tipos, propiedades y el Teorema del Valor Intermedio
- Regla de L'Hôpital y formas indeterminadas
- Límites de sucesiones: convergencia, Teorema del Sándwich y Bolzano-Weierstrass

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enunciar la definición épsilon-delta de un límite y explicar su significado geométrico [Familiarizarse]
2. Evaluar límites usando leyes de límites, la regla de L'Hôpital y análisis asintótico [Usar]
3. Demostrar continuidad o discontinuidad de una función y aplicar el Teorema del Valor Intermedio [Evaluar]

3.1.8. CAN/Cálculo Diferencial (CS Core: 1 hr, KA Core: 1 hr)

Derivadas de funciones de una variable: definición, reglas, derivadas de orden superior y aplicaciones a la optimización y al análisis de algoritmos.
Temas:
Core

- La derivada como límite; interpretaciones geométricas y físicas; diferenciabilidad
- Reglas del producto, cociente y cadena; diferenciación implícita; tasas relacionadas
- Teorema de Rolle, Teorema del Valor Medio y criterios de monotonía
- Derivadas de orden superior, polinomios de Taylor y estimación del residuo
- Extremos: pruebas de la primera y segunda derivada, trazado de curvas y optimización aplicada

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Interpretar la derivada como tasa de cambio y como pendiente de la recta tangente [Familiarizarse]
2. Aplicar las reglas de diferenciación y la diferenciación implícita para calcular derivadas de funciones compuestas e implícitas [Usar]
3. Resolver problemas de optimización y trazar curvas usando el Teorema del Valor Medio y las pruebas de derivadas [Evaluar]

3.1.9. CAN/Antiderivadas y Técnicas de Integración (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- El Teorema Fundamental del Cálculo y la antiderivada
- Técnicas de integración: sustitución, partes, fracciones parciales y sustitución trigonométrica

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enunciar el Teorema Fundamental del Cálculo y explicar la relación entre la diferenciación y la integración [Familiarizarse]
2. Calcular integrales definidas e indefinidas usando técnicas estándar, incluyendo sustitución e integración por partes [Usar]

3.1.10. CAN/Integrales Impropias (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Integrales impropias: criterios de convergencia y pruebas de comparación

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Determinar la convergencia de integrales impropias usando criterios de comparación y otras pruebas de convergencia [Evaluar]

3.1.11. CAN/Sucesiones, Series y Series de Potencias (CS Core: 1 hr, KA Core: 1 hr)

Temas:
Core

- Sucesiones y series: pruebas de convergencia (cociente, raíz, comparación, integral)
- Series de potencias, radio de convergencia, series de Taylor y Maclaurin

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Determinar la convergencia de series de potencias y construir series de Taylor para funciones estándar [Evaluar]

3.1.12. CAN/Cálculo Diferencial Multivariable (CS Core: 1 hr, KA Core: 1 hr)

Cálculo diferencial de funciones escalares de varias variables: límites, derivadas parciales, gradiente, plano tangente, optimización y el Teorema de Taylor.
Temas:
Core

- Límites y continuidad de funciones de varias variables en $\mathbb{R}^n$
- Derivadas parciales, derivadas direccionales y el vector gradiente
- Plano tangente, diferencial total y aproximación lineal
- Extremos de funciones multivariables y multiplicadores de Lagrange
- Teorema de Taylor para funciones de varias variables

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Evaluar límites y determinar la continuidad de funciones de varias variables [Familiarizarse]
2. Calcular gradientes, derivadas direccionales y planos tangentes para funciones multivariables [Usar]
3. Aplicar los multiplicadores de Lagrange para resolver problemas de optimización con restricciones en computación y ciencia de datos [Usar]
4. Expandir funciones de varias variables usando el Teorema de Taylor y estimar cotas de error [Evaluar]

3.1.13. CAN/Análisis Armónico y Teoría de Fourier (CS Core: 1 hr, KA Core: 1 hr)

Series y transformadas de Fourier con aplicaciones al procesamiento de señales, compresión y análisis espectral en computación.
Temas:
Core

- Series de Fourier: convergencia, Teorema de Parseval y completitud en L^2
- La transformada de Fourier: propiedades, inversión y convolución
- Teorema de Plancherel y distribuciones temperadas (espacio de Schwartz)
- Teoría de wavelets: análisis multirresolución y la transformada discreta de wavelets
- Aplicaciones al procesamiento de señales, resolución de EDPs y compresión de datos

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Descomponer funciones periódicas en series de Fourier y evaluar la convergencia puntual y en L^2 [Familiarizarse]
2. Aplicar la transformada de Fourier para resolver sistemas lineales y calcular convoluciones en procesamiento digital de señales [Usar]
3. Analizar la tasa de decaimiento de los coeficientes de Fourier en relación con la suavidad de una función [Evaluar]

3.2. Álgebra y Teoría de Números (ANT)

Esta área cubre las estructuras algebraicas esenciales para la computación: álgebra lineal para el aprendizaje automático y gráficos, álgebra abstracta para criptografía y teoría de códigos, y los fundamentos numéricos de los protocolos modernos de seguridad.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.2.1 Matrices y Sistemas de Ecuaciones Lineales (Pág. 161)	1	1
⭐ 3.2.2 Espacios Vectoriales y Transformaciones Lineales (Pág. 161)	1	1
⭐ 3.2.3 Valores Propios, Producto Interno y Formas Cuadráticas (Pág. 162)	1	1
⭐ 3.2.4 Teoría de Grupos (Pág. 162)	1	1
⭐ 3.2.5 Anillos, Cuerpos y Teoría de Galois (Pág. 162)	1	1
⭐ 3.2.6 Criptografía y Teoría de Códigos (Pág. 163)	1	1

Cuadro 3.2: Lista de KUs del área de Álgebra y Teoría de Números.

3.2.1. ANT/Matrices y Sistemas de Ecuaciones Lineales (CS Core: 1 hr, KA Core: 1 hr)

Álgebra de matrices, sistemas de ecuaciones lineales y sus métodos de resolución, determinantes y descomposiciones matriciales, con aplicaciones en ciencia de datos y gráficos computacionales.

Temas:

Core

- Álgebra de matrices: suma, multiplicación por escalar, producto de matrices, transpuesta, tipos de matrices e inversa de una matriz
- Resolución de sistemas de ecuaciones lineales mediante eliminación gaussiana y Gauss-Jordan; forma escalonada, rango y consistencia de sistemas
- Determinantes: definición, propiedades, regla de Cramer e interpretación geométrica
- Descomposiciones matriciales: LU, QR y descomposición en valores singulares (SVD)

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Identificar tipos de matrices y realizar operaciones básicas de álgebra matricial, incluyendo el cálculo de la inversa de una matriz [Familiarizarse]
2. Resolver sistemas de ecuaciones lineales mediante eliminación gaussiana o Gauss-Jordan, determinando el rango y la consistencia del sistema [Usar]
3. Aplicar la regla de Cramer y las descomposiciones LU, QR y SVD para resolver sistemas lineales y factorizar matrices [Evaluar]

3.2.2. ANT/Espacios Vectoriales y Transformaciones Lineales (CS Core: 1 hr, KA Core: 1 hr)

Espacios vectoriales, subespacios, bases y aplicaciones lineales entre espacios vectoriales, con aplicaciones en aprendizaje automático.

Temas:

Core

- Espacios y subespacios vectoriales; bases, dimensión e independencia lineal
- Transformaciones lineales, núcleo e imagen, teorema rango-nulidad y representaciones matriciales

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Identificar si un conjunto con operaciones dadas forma un espacio vectorial y determinar una base [Familiarizarse]
2. Determinar el núcleo, la imagen y la representación matricial de una aplicación lineal, verificando el teorema de rango-nulidad [Usar]

3. Aplicar transformaciones lineales para modelar y resolver problemas de cambio de base y composición de aplicaciones [Evaluar]

3.2.3. ANT/Valores Propios, Producto Interno y Formas Cuadráticas (CS Core: 1 hr, KA Core: 1 hr)

Valores propios, diagonalización, espacios con producto interno y formas cuadráticas, con aplicaciones en aprendizaje automático y ciencia de datos.

Temas:

Core

- Valores y vectores propios; diagonalización y el teorema espectral
- Espacios con producto interno, ortogonalidad, proceso de Gram-Schmidt y mínimos cuadrados
- Formas cuadráticas y definitud: clasificación en positiva, negativa e indefinida mediante valores propios

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Identificar los valores propios y vectores propios de una matriz y determinar si es diagonalizable [Familiarizarse]
2. Calcular la diagonalización de matrices mediante el teorema espectral y aplicar el proceso de Gram-Schmidt para obtener bases ortogonales [Usar]
3. Aplicar los métodos de mínimos cuadrados y la clasificación de formas cuadráticas mediante valores propios para resolver problemas de aproximación y optimización [Evaluar]

3.2.4. ANT/Teoría de Grupos (CS Core: 1 hr, KA Core: 1 hr)

Grupos, subgrupos, homomorfismos, grupos cociente y clasificación de grupos finitos, con aplicaciones a la simetría en computación y criptografía.

Temas:

Core

- Grupos, subgrupos, grupos cíclicos y orden de los elementos
- Clases laterales, Teorema de Lagrange y subgrupos normales
- Homomorfismos e isomorfismos de grupos y los teoremas de isomorfismo
- Acciones de grupos, órbitas, estabilizadores y Lema de Burnside

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Enunciar el Teorema de Lagrange y explicar la relación entre subgrupos y clases laterales [Familiarizarse]
2. Aplicar los teoremas de isomorfismo para analizar imágenes homomórficas y grupos cociente [Usar]
3. Analizar grupos de simetría que surgen en contextos de computación, como grupos de permutaciones sobre estructuras de datos [Evaluar]

3.2.5. ANT/Anillos, Cuerpos y Teoría de Galois (CS Core: 1 hr, KA Core: 1 hr)

Anillos, ideales, anillos de polinomios, extensiones de cuerpos y cuerpos finitos, fundamentales para la teoría de códigos y criptografía.

Temas:

Core

- Anillos, ideales, anillos cociente y los teoremas de isomorfismo de anillos
- Anillos de polinomios: irreducibilidad, factorización y dominios de factorización única
- Extensiones de cuerpos: elementos algebraicos y trascendentes, grado y cuerpos de escisión
- Cuerpos finitos: estructura, existencia y aplicaciones en teoría de códigos

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Clasificar anillos como dominios integrales, DIP o DFU y justificar la clasificación [Familiarizarse]

2. Construir y analizar cuerpos finitos utilizados en AES y otros algoritmos criptográficos [Usar]
3. Aplicar la teoría de extensiones de cuerpos para analizar códigos correctores de errores sobre cuerpos finitos [Evaluar]

3.2.6. ANT/Criptografía y Teoría de Códigos (CS Core: 1 hr, KA Core: 1 hr)

Fundamentos matemáticos de la criptografía moderna y los códigos correctores de errores, incluyendo RSA, curvas elípticas y códigos lineales.

Temas:

Core

- Aritmética modular, el criptosistema RSA y el problema del logaritmo discreto
- Curvas elípticas sobre cuerpos finitos y criptografía de curvas elípticas (ECC)
- Códigos lineales: matrices generadoras y de paridad, distancia de Hamming y corrección de errores
- Códigos cíclicos: representación polinomial, códigos BCH y Reed-Solomon

Non Core

- Criptografía postcuántica: esquemas basados en reticulados y el problema Learning With Errors (LWE)

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar las hipótesis de dificultad matemática que sustentan la criptografía RSA y de curvas elípticas [Familiarizarse]
2. Construir un código lineal, calcular sus parámetros y determinar su capacidad de corrección de errores [Usar]

NonCore:

3. Analizar las hipótesis de seguridad de los esquemas criptográficos basados en reticulados en el contexto postcuántico [Evaluar]

3.3. Matemática Discreta y Combinatoria (DMC)

Esta área cubre la matemática de las estructuras finitas y contables, incluyendo la enumeración combinatoria, la teoría de grafos, la optimización discreta y los fundamentos lógicos de la informática. Es fundamental para el diseño de algoritmos, los lenguajes formales, los sistemas de tipos y la teoría de compiladores.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.3.1 Combinatoria Enumerativa (Pág. 163)	1	1
⭐ 3.3.2 Teoría de Grafos (Pág. 164)	1	1
⭐ 3.3.3 Optimización Discreta (Pág. 164)	1	1
⭐ 3.3.4 Lógica Matemática y Teoría de Conjuntos (Pág. 164)	1	1

Cuadro 3.3: Lista de KUs del área de Matemática Discreta y Combinatoria.

3.3.1. DMC/Combinatoria Enumerativa (CS Core: 1 hr, KA Core: 1 hr)

Conteo sistemático de estructuras discretas mediante biyecciones, recurrencias, funciones generadoras y el principio de inclusión-exclusión.

Temas:

Core

- Principios básicos de conteo: permutaciones, combinaciones y el teorema del binomio
- Principio de inclusión-exclusión y aplicaciones a desarreglos y la función totiente de Euler
- Funciones generadoras ordinarias y exponenciales; resolución de recurrencias
- Particiones de enteros, tableaux de Young y la fórmula de la longitud del gancho

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Aplicar los principios básicos de conteo y la inclusión-exclusión a problemas de enumeración [Familiarizarse]
2. Construir funciones generadoras ordinarias y exponenciales para enumerar estructuras combinatorias [Usar]
3. Derivar fórmulas cerradas para sucesiones combinatorias mediante biyecciones y recurrencias [Evaluar]

3.3.2. DMC/Teoría de Grafos (CS Core: 1 hr, KA Core: 1 hr)

Estructura y propiedades de los grafos: conectividad, coloraciones, emparejamientos, planaridad y teoría espectral de grafos, con aplicaciones a redes y algoritmos.

Temas:

Core

- Grafos: definiciones, isomorfismo, sucesiones de grados, árboles y árboles de expansión
- Conectividad, Teorema de Menger y flujos en redes (flujo máximo-corte mínimo)
- Emparejamientos (Teorema de Hall), coloraciones de grafos y el polinomio cromático
- Grafos planos, fórmula de Euler, Teorema de Kuratowski y el teorema de los cuatro colores
- Matrices de adyacencia y laplaciana, valores propios y grafos expansores

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Identificar propiedades estructurales de grafos (conectividad, planaridad, bipartición) y aplicar la fórmula de Euler [Familiarizarse]
2. Aplicar el Teorema de Hall y algoritmos de flujo en redes a problemas de emparejamiento y enrutamiento [Usar]
3. Analizar el espectro de un grafo para acotar su número cromático y sus propiedades de conectividad [Evaluar]

3.3.3. DMC/Optimización Discreta (CS Core: 1 hr, KA Core: 1 hr)

Programación entera, flujos en redes, matroidesTEORIA, algoritmos de emparejamiento y la teoría de NP-completitud, fundamental para el diseño de algoritmos en computación.

Temas:

Core

- Programación lineal entera: ramificación y acotación, planos de corte y matrices totalmente unimodulares
- Algoritmos de flujo en redes: caminos más cortos, flujo máximo, flujo de costo mínimo
- Teoría de matroidesTEORIA: conjuntos independientes, algoritmo greedy e intersección de matroidesTEORIA
- Teoría de complejidad: P vs. NP, NP-completitud y reducciones
- Algoritmos de aproximación y el esquema de aproximación en tiempo polinomial (PTAS)

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar la relación entre las relajaciones de programación lineal y las cotas de programación entera [Familiarizarse]
2. Aplicar algoritmos de flujo en redes y matroidesTEORIA para resolver problemas de optimización combinatoria [Usar]
3. Demostrar la NP-completitud de un problema combinatorio mediante reducción polinomial [Evaluar]

3.3.4. DMC/Lógica Matemática y Teoría de Conjuntos (CS Core: 1 hr, KA Core: 1 hr)

Lógica proposicional y de primer orden, sistemas de prueba, teoremas de Gödel y computabilidad, que forman el fundamento lógico de los lenguajes de programación y la verificación formal.

Temas:

Core

- Lógica proposicional: sintaxis, semántica, satisfacibilidad y resolución
- Lógica de primer orden: modelos, corrección y el Teorema de Completitud de Gödel
- Teoremas de Incompletitud de Gödel y sus implicaciones para los sistemas formales
- Teoría axiomática de conjuntos (ZFC): ordinales, cardinales y el axioma de elección
- Computabilidad: máquinas de Turing, decidibilidad y el problema de la parada

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enunciar los Teoremas de Completitud e Incompletitud de Gödel y explicar su importancia para la computación [Familiarizarse]
2. Construir pruebas formales en lógica de primer orden y determinar la satisfacibilidad de fórmulas [Usar]
3. Analizar la indecidibilidad de problemas mediante reducción al problema de la parada [Evaluar]

3.4. Probabilidad y Estadística (PST)

Esta área cubre la teoría matemática de la incertidumbre y la inferencia, desde los axiomas de probabilidad y los procesos estocásticos hasta la inferencia bayesiana y el aprendizaje estadístico, que son fundamentos esenciales para la IA, la ciencia de datos y el aprendizaje automático.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.4.1 Axiomas de Probabilidad y Distribuciones (Pág. 165)	1	1
⭐ 3.4.2 Teoremas Límite (Pág. 165)	1	1
⭐ 3.4.3 Procesos Estocásticos (Pág. 166)	1	1
⭐ 3.4.4 Teoría de la Estimación (Pág. 166)	1	1
⭐ 3.4.5 Inferencia Bayesiana (Pág. 166)	1	1
⭐ 3.4.6 Análisis de Series Temporales (Pág. 167)	1	1

Cuadro 3.4: Lista de KUs del área de Probabilidad y Estadística.

3.4.1. PST/Axiomas de Probabilidad y Distribuciones (CS Core: 1 hr, KA Core: 1 hr)

Fundamento axiomático de la probabilidad de Kolmogorov, variables aleatorias, distribuciones y esperanzas, con aplicaciones en análisis de algoritmos e IA.
Temas:
Core

- Axiomas de Kolmogorov, sigma-álgebras y espacios de probabilidad
- Variables aleatorias: distribuciones discretas y continuas, FDA, FDP y FMP
- Esperanza, varianza, momentos y funciones generadoras de momentos
- Distribuciones conjuntas, marginales y condicionales; independencia y covarianza
- Familias estándar: Binomial, Poisson, Normal, Exponencial, Gamma y Beta

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enunciar los axiomas de Kolmogorov y construir un espacio de probabilidad para un experimento estándar [Familiarizarse]
2. Calcular esperanzas, varianzas y funciones generadoras de momentos para distribuciones estándar [Usar]
3. Determinar distribuciones conjuntas y condicionales y evaluar la independencia de variables aleatorias [Evaluar]

3.4.2. PST/Teoremas Límite (CS Core: 1 hr, KA Core: 1 hr)

Leyes de los grandes números, teorema central del límite, modos de convergencia y grandes desviaciones, fundamentales para el análisis de algoritmos y la inferencia estadística.

Temas:
Core
- Modos de convergencia: casi seguro, en probabilidad, en L^p y en distribución
- Leyes débil y fuerte de los grandes números y sus aplicaciones
- Teorema Central del Límite: enunciado, demostración mediante funciones características y la cota de Berry-Esseen
- Funciones características y el teorema de continuidad (Lévy)

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Distinguir entre los modos de convergencia de variables aleatorias y proporcionar contraejemplos [Familiarizarse]
2. Aplicar el Teorema Central del Límite para aproximar distribuciones de sumas y medias [Usar]
3. Demostrar la ley fuerte de los grandes números y estimar probabilidades de grandes desviaciones [Evaluar]

3.4.3. PST/Procesos Estocásticos (CS Core: 1 hr, KA Core: 1 hr)

Cadenas de Markov, procesos de Poisson, movimiento browniano, martingalas y cálculo estocástico, con aplicaciones a colas, simulación y aprendizaje por refuerzo.
Temas:
Core
- Cadenas de Markov en tiempo discreto: clasificación de estados, distribuciones estacionarias y teorema ergódico
- Procesos de Poisson: definición, propiedades y generalizaciones (compuesto, inhomogéneo)
- Movimiento browniano: definición, propiedades de las trayectorias y el principio de reflexión
- Martingalas, tiempos de parada, teorema de muestreo opcional y las desigualdades de Doob

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Clasificar los estados de una cadena de Markov y determinar su distribución estacionaria [Familiarizarse]
2. Aplicar la teoría de martingalas y el teorema de muestreo opcional para resolver problemas de tiempo de parada [Usar]
3. Modelar un sistema de colas usando procesos de llegada de Poisson y analizar su distribución en estado estacionario [Evaluar]

3.4.4. PST/Teoría de la Estimación (CS Core: 1 hr, KA Core: 1 hr)

Estimación puntual y por intervalos: suficiencia, máxima verosimilitud y teoría asintótica, usados en todo el aprendizaje estadístico y la ciencia de datos.
Temas:
Core
- Estadísticos suficientes, el teorema de factorización y completitud
- Estimadores insesgados, la cota de Cramér-Rao y el UMVUE
- Estimación por máxima verosimilitud: propiedades, consistencia y normalidad asintótica
- Intervalos de confianza: cantidades pivotales, razón de verosimilitud y métodos bootstrap
- Teoría asintótica: el método delta, M-estimadores y la matriz de información de Fisher

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Explicar la cota de Cramér-Rao e identificar cuando existe un UMVUE [Familiarizarse]
2. Derivar estimadores de máxima verosimilitud y calcular su varianza asintótica [Usar]
3. Construir intervalos de confianza usando cantidades pivotales y remuestreo bootstrap [Evaluar]

3.4.5. PST/Inferencia Bayesiana (CS Core: 1 hr, KA Core: 1 hr)

Paradigma bayesiano: distribuciones a priori y a posteriori, familias conjugadas, modelos jerárquicos y métodos MCMC, centrales para el aprendizaje automático probabilístico e IA.

Temas:
Core
- Teorema de Bayes en contextos continuos; distribuciones a priori y a posteriori
- Familias de priori conjugadas: Beta-Binomial, Dirichlet-Multinomial, Normal-Normal
- Modelos bayesianos jerárquicos y Bayes empírico
- Cadenas de Markov Monte Carlo (MCMC): Metropolis-Hastings y muestreo de Gibbs
- Inferencia variacional y la cota inferior de la evidencia (ELBO)

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Aplicar el Teorema de Bayes para actualizar una distribución a priori con datos observados y obtener la distribución a posteriori [Familiarizarse]
2. Construir un modelo bayesiano jerárquico y derivar sus distribuciones conjuntas y condicionales [Usar]
3. Implementar el muestreo de Gibbs o Metropolis-Hastings para aproximar una distribución a posteriori [Evaluar]

3.4.6. PST/Análisis de Series Temporales (CS Core: 1 hr, KA Core: 1 hr)

Procesos estacionarios, modelos ARMA, análisis espectral, modelos de espacio de estados y predicción.
Temas:
Core
- Estacionariedad, función de autocovarianza y función de autocorrelación (ACF)
- Modelos ARMA: identificación, estimación (Yule-Walker, MLE) y diagnóstico
- Densidad espectral, el periodograma y el teorema de Wiener-Khinchin
- Modelos de espacio de estados y el filtro de Kalman
- Modelos ARIMA, ajuste estacional y predicción a múltiples pasos

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Identificar la estacionariedad y determinar el orden de un modelo ARMA a partir de gráficos ACF y PACF [Familiarizarse]
2. Ajustar modelos ARIMA a datos de series temporales, validar residuos y producir predicciones [Usar]
3. Aplicar el filtro de Kalman para estimar estados ocultos en un modelo lineal gaussiano de espacio de estados [Evaluar]

3.5. Análisis Numérico y Científico (NSA)

Esta área cubre los algoritmos numéricos para la computación: análisis de errores, solucionadores de álgebra lineal, integración numérica, ecuaciones diferenciales y métodos de optimización, esenciales para implementar sistemas computacionales confiables y software científico.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.5.1 Teoría de Aproximación e Interpolación (Pág. 168)	1	1
⭐ 3.5.2 Integración Numérica y Cuadratura (Pág. 168)	1	1
⭐ 3.5.3 Análisis de Errores y Aritmética de Punto Flotante (Pág. 168)	1	1
⭐ 3.5.4 Búsqueda de Raíces de Ecuaciones No Lineales (Pág. 169)	1	1
⭐ 3.5.5 Álgebra Lineal Numérica (Pág. 169)	1	1
⭐ 3.5.6 Métodos Numéricos para Ecuaciones Diferenciales (Pág. 169)	1	1
⭐ 3.5.7 Algoritmos de Optimización (Pág. 170)	1	1
⭐ 3.5.8 Método de Elementos Finitos (Pág. 170)	1	1
⭐ 3.5.9 Computación Paralela y de Alto Rendimiento (Pág. 171)	1	1

Cuadro 3.5: Lista de KUs del área de Análisis Numérico y Científico.

3.5.1. NSA/Teoría de Aproximación e Interpolación (CS Core: 1 hr, KA Core: 1 hr)

Interpolación polinomial y por splines, mejor aproximación, polinomios de Chebyshev y aproximación racional.

Temas:

Core

- Interpolación polinomial: Lagrange, diferencias divididas de Newton y fenómeno de Runge
- Interpolación por splines: splines cúbicos, B-splines y métodos de funciones polinómicas a trozos
- Mejor aproximación en espacios normados: Chebyshev (minimax) y mínimos cuadrados
- Polinomios de Chebyshev: propiedades, ortogonalidad y convergencia espectral
- Aproximación trigonométrica y transformada rápida de Fourier (FFT)

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar el fenómeno de Runge y justificar la elección de nodos de Chebyshev para mitigarlo [Familiarizarse]
2. Construir interpolantes por splines cúbicos y de Chebyshev para datos dados y estimar el error de interpolación [Usar]
3. Aplicar la FFT para calcular eficientemente aproximaciones trigonométricas de una función muestreada [Evaluar]

3.5.2. NSA/Integración Numérica y Cuadratura (CS Core: 1 hr, KA Core: 1 hr)

Reglas de cuadratura, integración adaptativa, cuadratura gaussiana e integración multidimensional.

Temas:

Core

- Reglas de Newton-Cotes: trapezoidal, Simpson y reglas compuestas; análisis de error
- Cuadratura adaptativa y control automático del error
- Cuadratura gaussiana: nodos y pesos óptimos, conexión con polinomios ortogonales
- Tratamiento numérico de integrales impropias y singulares
- Integración de Monte Carlo y métodos cuasi-Monte Carlo para integrales de alta dimensión

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Derivar la fórmula de error para la regla compuesta de Simpson e identificar su orden de precisión [Familiarizarse]
2. Seleccionar y aplicar una regla de cuadratura apropiada (Gaussiana, adaptativa) según la regularidad del integrando [Usar]
3. Aplicar integración de Monte Carlo para estimar integrales de alta dimensión y cuantificar el error estadístico [Evaluar]

3.5.3. NSA/Análisis de Errores y Aritmética de Punto Flotante (CS Core: 1 hr, KA Core: 1 hr)

Fuentes de error numérico, aritmética de punto flotante IEEE 754, redondeo, cancelación y análisis de error directo e inverso, críticos para la ingeniería de software confiable.

Temas:

Core

- Representación en punto flotante IEEE 754: épsilon de máquina, desbordamiento, subdesbordamiento y modos de redondeo
- Errores de redondeo, cancelación catastrófica y pérdida de significado
- Análisis de error directo e inverso; número de condición de un problema
- Estabilidad numérica: algoritmos estables vs. inestables; análisis de error inverso de Wilkinson
- Propagación de errores en operaciones aritméticas y evaluación de funciones

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar el estándar IEEE 754 de punto flotante e identificar las fuentes de errores de redondeo y cancelación [Familiarizarse]
2. Estimar el número de condición de un problema y predecir la precisión de un resultado numérico [Usar]
3. Analizar un algoritmo para determinar su estabilidad numérica usando cotas de error directo e inverso [Evaluar]

3.5.4. NSA/Búsqueda de Raíces de Ecuaciones No Lineales (CS Core: 1 hr, KA Core: 1 hr)

Métodos iterativos para aproximar raíces de ecuaciones no lineales y sistemas de ecuaciones no lineales, base de los solucionadores usados en optimización y aprendizaje automático.

Temas:

Core

- Método de bisección: acotamiento del intervalo, criterio de parada y cota del error
- Método de Newton-Raphson: derivación, convergencia cuadrática y casos de falla
- Método de la secante como alternativa al método de Newton sin derivada explícita
- Orden de convergencia y comparación de la eficiencia entre métodos iterativos
- Extensión del método de Newton a sistemas de ecuaciones no lineales mediante la matriz Jacobiana, base de solucionadores en aprendizaje automático

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Comparar las garantías de convergencia de los métodos de bisección, Newton-Raphson y secante [Familiarizarse]
2. Implementar el método de Newton-Raphson para aproximar raíces de una función no lineal [Usar]
3. Analizar el orden de convergencia de un método iterativo y justificar su elección según el costo computacional [Evaluar]

3.5.5. NSA/Álgebra Lineal Numérica (CS Core: 1 hr, KA Core: 1 hr)

Algoritmos para resolver sistemas lineales, mínimos cuadrados, problemas de valores propios y factorizaciones matriciales, aplicados en aprendizaje automático y computación científica.

Temas:

Core

- Eliminación gaussiana con pivoteo parcial, factorización LU y análisis de complejidad
- Factorización QR vía Householder y Gram-Schmidt; problemas de mínimos cuadrados
- Algoritmos de valores propios: iteración de la potencia, algoritmo QR y el método de Lanczos
- Descomposición en valores singulares: cálculo, truncamiento y aplicaciones (ACP, pseudoinversa)
- Métodos iterativos para sistemas dispersos grandes: GC, GMRES y precondicionamiento

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Comparar solucionadores directos e iterativos y seleccionar el método adecuado para una estructura matricial dada [Familiarizarse]
2. Aplicar la factorización QR y la SVD para resolver problemas de mínimos cuadrados y calcular aproximaciones de bajo rango [Usar]
3. Analizar la estabilidad numérica y el costo computacional de los algoritmos de valores propios [Evaluar]

3.5.6. NSA/Métodos Numéricos para Ecuaciones Diferenciales (CS Core: 1 hr, KA Core: 1 hr)

Métodos de diferencias finitas y Runge-Kutta para EDOs, análisis de estabilidad y métodos de diferencias finitas para EDPs, usados en simulación científica y computación basada en física.

Temas:
Core
- Métodos de Runge-Kutta: Euler, RK4 y métodos embebidos para control de errores
- Métodos de pasos múltiples: Adams-Bashforth, Adams-Moulton y métodos BDF
- Análisis de estabilidad: cero-estabilidad, regiones de estabilidad absoluta y EDOs rígidas
- Métodos de diferencias finitas para EDPs parabólicas y elípticas: estabilidad y convergencia

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Comparar solucionadores de EDOs explícitos e implícitos y explicar su estabilidad relativa para problemas rígidos [Familiarizarse]
2. Implementar un método de Runge-Kutta con control adaptativo del paso y aplicarlo a un sistema de EDOs [Usar]
3. Analizar la estabilidad y convergencia de un esquema de diferencias finitas para una EDP parabólica [Evaluar]

3.5.7. NSA/Algoritmos de Optimización (CS Core: 1 hr, KA Core: 1 hr)

Métodos de gradiente, Newton y cuasi-Newton, optimización con restricciones y algoritmos de optimización convexa, fundamentales para el entrenamiento de modelos de aprendizaje automático.
Temas:
Core
- Descenso de gradiente y métodos de búsqueda lineal: condiciones de Armijo-Wolfe y tasas de convergencia
- Método de Newton y métodos cuasi-Newton (BFGS, L-BFGS)
- Optimización con restricciones: condiciones KKT, métodos de penalización y programación cuadrática secuencial
- Algoritmos de optimización convexa: métodos de punto interior y ADMM
- Descenso de gradiente estocástico (SGD), reducción de varianza y el optimizador Adam

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Explicar las garantías de convergencia del descenso de gradiente para funciones convexas suaves [Familiarizarse]
2. Aplicar los métodos de Newton y cuasi-Newton a problemas de optimización sin restricciones [Usar]
3. Formular un problema de optimización con restricciones como un sistema KKT y aplicar un método de punto interior [Evaluar]

3.5.8. NSA/Método de Elementos Finitos (CS Core: 1 hr, KA Core: 1 hr)

Formulación variacional, métodos de Galerkin, espacios de elementos finitos, estimaciones de error y aplicaciones a EDPs elípticas.
Temas:
Core
- Formulación débil (variacional) de problemas de valor en la frontera y espacios de Sobolev
- Método de Galerkin: espacios de aproximación de dimensión finita y matriz de rigidez
- Espacios de elementos finitos triangulares y cuadriláteros: elementos de Lagrange y conformidad
- Estimaciones de error a priori: lema de Céa y cotas de error de interpolación

Non Core
- Estimaciones de error a posteriori y refinamiento adaptativo de malla

Aprendizaje esperado (*Learning Outcomes*):
Core:
1. Derivar la formulación débil de un PVE elíptico y mostrar su equivalencia con la forma fuerte [Familiarizarse]
2. Ensamblar la matriz de rigidez global y el vector de carga para una discretización lineal por elementos finitos [Usar]

3. Estimar el error en H^1 de una solución por elementos finitos usando el lema de Céa y la teoría de interpolación [Evaluar]

3.5.9. NSA/Computación Paralela y de Alto Rendimiento (CS Core: 1 hr, KA Core: 1 hr)

Arquitecturas paralelas, modelos de rendimiento, algoritmos paralelos para álgebra lineal y computación con GPUs.

Temas:

Core

- Arquitecturas paralelas: memoria compartida (OpenMP), memoria distribuida (MPI) y GPU (CUDA)
- Modelos de rendimiento: ley de Amdahl, modelo de línea techada y complejidad de comunicación
- Algoritmos paralelos para álgebra lineal densa y dispersa (ScaLAPACK, PETSc)
- Descomposición de dominio y solucionadores paralelos de EDPs

Non Core

- Diferenciación automática (modo directo y reverso) y su rol en ML científico

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar la ley de Amdahl e identificar los cuellos de botella que limitan la aceleración paralela en un algoritmo dado [Familiarizarse]
2. Implementar una rutina paralela de álgebra lineal usando MPI o OpenMP y medir la eficiencia paralela [Usar]

NonCore:

3. Aplicar diferenciación automática para calcular gradientes de funciones científicas complejas con fines de optimización [Evaluar]

3.6. Modelado Matemático y Simulación (MMS)

Esta área cubre las herramientas matemáticas para describir y simular sistemas computacionales y físicos, abarcando ecuaciones diferenciales ordinarias, sistemas dinámicos, teoría de control y ecuaciones diferenciales estocásticas, con aplicaciones directas en ingeniería de simulación, robótica y sistemas de IA.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.6.1 Ecuaciones Diferenciales Ordinarias de Primer Orden (Pág. 171)	1	1
⭐ 3.6.2 Ecuaciones Diferenciales de Orden Superior y Sistemas (Pág. 172)	1	1
⭐ 3.6.3 Sistemas Dinámicos y Caos (Pág. 172)	1	1
⭐ 3.6.4 Teoría Matemática del Control (Pág. 172)	1	1
⭐ 3.6.5 Ecuaciones Diferenciales Estocásticas (Pág. 173)	1	1

Cuadro 3.6: Lista de KUs del área de Modelado Matemático y Simulación.

3.6.1. MMS/Ecuaciones Diferenciales Ordinarias de Primer Orden (CS Core: 1 hr, KA Core: 1 hr)

Teoría y métodos de solución para EDOs de primer orden y clasificación de tipos de ecuaciones.

Temas:

Core

- EDOs de primer orden: separables, lineales, exactas e integrales

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Clasificar las EDOs por orden, linealidad y tipo, y seleccionar el método de solución apropiado [Familiarizarse]

3.6.2. MMS/Ecuaciones Diferenciales de Orden Superior y Sistemas (CS Core: 1 hr, KA Core: 1 hr)

Teoría y métodos de solución para EDOs de orden superior y sistemas lineales: ecuaciones características, sistemas de valores propios, teoremas de existencia y unicidad, y métodos de transformada de Laplace usados en el modelado de sistemas.

Temas:

Core

- EDOs lineales de orden superior: ecuaciones características, variación de parámetros y coeficientes indeterminados
- Sistemas de EDOs lineales: exponencial matricial, métodos de valores propios y retratos de fase
- Teoremas de existencia y unicidad (Picard-Lindelöf) y continuación de soluciones
- Métodos de la transformada de Laplace para resolver EDOs con forzamiento discontinuo

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Resolver sistemas lineales de EDOs usando métodos de valores propios y trazar sus retratos de fase [Usar]
2. Aplicar la transformada de Laplace para resolver EDOs con funciones de forzamiento en tramos e impulsivas [Evaluar]

3.6.3. MMS/Sistemas Dinámicos y Caos (CS Core: 1 hr, KA Core: 1 hr)

Análisis del plano de fase, estabilidad, bifurcaciones, ciclos límite y caos en sistemas continuos y discretos, relevante para la dinámica de redes, los modelos de computación biológica y los sistemas de agentes.

Temas:

Core

- Retratos de fase de sistemas 2D: equilibrios, estabilidad y ciclos límite
- Teoría de estabilidad de Lyapunov y el teorema de linealización (Hartman-Grobman)
- Teoría de bifurcaciones: nodo de silla, horquilla, Hopf y bifurcaciones globales
- Caos: sistema de Lorenz, sensibilidad a las condiciones iniciales y exponentes de Lyapunov

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Trazar retratos de fase de sistemas autónomos 2D y clasificar los equilibrios por estabilidad [Familiarizarse]
2. Aplicar el método de Lyapunov y el teorema de Hartman-Grobman para evaluar la estabilidad de los equilibrios [Usar]
3. Identificar los tipos de bifurcación en sistemas parametrizados y trazar la evolución de los conjuntos invariantes [Evaluar]

3.6.4. MMS/Teoría Matemática del Control (CS Core: 1 hr, KA Core: 1 hr)

Modelos en espacio de estado, controlabilidad y observabilidad, diseño de realimentación, estabilidad de Lyapunov y control óptimo, con aplicaciones en robótica, sistemas empotrados y agentes autónomos.

Temas:

Core

- Representación en espacio de estado de sistemas lineales: funciones de transferencia y realizaciones
- Controlabilidad y observabilidad: condiciones de rango y descomposición de Kalman
- Estabilización por realimentación: asignación de polos, diseño LQR y el principio de separación
- Estabilidad de Lyapunov para sistemas no lineales y estabilidad entrada-salida
- Control óptimo: el Principio del Máximo de Pontryagin y la ecuación de Hamilton-Jacobi-Bellman

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Determinar la controlabilidad y observabilidad de un sistema lineal a partir de sus matrices en espacio de estado [Familiarizarse]
2. Diseñar un controlador de realimentación de estado usando la asignación de polos o la optimización LQR [Usar]
3. Aplicar el Principio del Máximo de Pontryagin para resolver un problema básico de control óptimo [Evaluar]

3.6.5. MMS/Ecuaciones Diferenciales Estocásticas (CS Core: 1 hr, KA Core: 1 hr)

Cálculo de Itô, EDEs impulsadas por movimiento browniano, la ecuación de Fokker-Planck y la simulación numérica de EDEs, fundamentales para el aprendizaje automático probabilístico y los modelos de difusión.

Temas:

Core

- Integral de Itô, variación cuadrática y la fórmula de Itô
- Existencia y unicidad de soluciones para EDEs bajo condiciones de Lipschitz
- Ecuación de Fokker-Planck: derivación y conexión con las EDEs
- Fórmula de Feynman-Kac y conexiones con las EDPs parabólicas
- Esquemas de Euler-Maruyama y Milstein para la simulación numérica de EDEs

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Enunciar la fórmula de Itô y aplicarla para calcular el diferencial de una función de un proceso estocástico [Familiarizarse]
2. Derivar la ecuación de Fokker-Planck a partir de una EDE dada e interpretarla como un flujo de probabilidad [Usar]
3. Implementar el esquema de Euler-Maruyama para simular trayectorias de una ecuación diferencial estocástica [Evaluar]

3.7. Fundamentos Matemáticos de la Ciencia de Datos (FDS)

Esta área cubre la teoría matemática que subyace a la ciencia de datos moderna y el aprendizaje automático: teoría del aprendizaje estadístico, optimización, teoría de la información, reducción de dimensionalidad y la matemática del aprendizaje profundo, todos esenciales para estudiantes de Computación en IA, ciencia de datos e ingeniería de software.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.7.1 Teoría del Aprendizaje Estadístico (Pág. 173)	1	1
⭐ 3.7.2 Optimización para el Aprendizaje Automático (Pág. 174)	1	1
⭐ 3.7.3 Teoría de la Información (Pág. 174)	1	1
⭐ 3.7.4 Reducción de Dimensionalidad (Pág. 175)	1	1
⭐ 3.7.5 Fundamentos Matemáticos del Aprendizaje Profundo (Pág. 175)	1	1

Cuadro 3.7: Lista de KUs del área de Fundamentos Matemáticos de la Ciencia de Datos.

3.7.1. FDS/Teoría del Aprendizaje Estadístico (CS Core: 1 hr, KA Core: 1 hr)

Marco PAC de aprendizaje, dimensión VC, complejidad de Rademacher, cotas de generalización y la compensación sesgo-varianza.

Temas:

Core

- Marco de aprendizaje PAC: complejidad muestral, clases de hipótesis y aprendizabilidad
- Dimensión VC: definición, ejemplos y el teorema fundamental del aprendizaje
- Complejidad de Rademacher y cotas de convergencia uniforme
- Descomposición sesgo-varianza y el fenómeno de doble descenso
- Regularización: regularización de Tikhonov y minimización del riesgo estructural

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enunciar la definición de aprendizaje PAC y determinar si una clase de hipótesis es PAC-aprendible [Familiarizarse]
2. Calcular la dimensión VC de clases de hipótesis estándar y derivar cotas de complejidad muestral [Usar]
3. Analizar el error de generalización usando la complejidad de Rademacher y explicar el fenómeno de doble descenso [Evaluar]

3.7.2. FDS/Optimización para el Aprendizaje Automático (CS Core: 1 hr, KA Core: 1 hr)

Teoría de optimización convexa, SGD y sus variantes, métodos adaptativos y los paisajes de optimización no convexos para el entrenamiento de modelos de aprendizaje.
Temas:
Core

- Convexidad, convexidad fuerte y dualidad de Fenchel en optimización
- Descenso de gradiente estocástico: tasas de convergencia para objetivos convexos y no convexos
- Métodos de gradiente adaptativos: AdaGrad, RMSProp y Adam
- Optimización no convexa: puntos de silla, mínimos locales y geometría del paisaje de pérdida de redes neuronales
- Métodos de gradiente proximal y descenso de coordenadas para problemas estructurados

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar las garantías de convergencia del SGD para funciones convexas suaves e identificar el papel de la tasa de aprendizaje [Familiarizarse]
2. Aplicar Adam y los métodos de gradiente proximal para entrenar modelos de aprendizaje automático regularizados [Usar]
3. Analizar el paisaje de pérdida de una red sobreparametrizada y explicar por qué el SGD encuentra buenos mínimos [Evaluar]

3.7.3. FDS/Teoría de la Información (CS Core: 1 hr, KA Core: 1 hr)

Entropía de Shannon, información mutua, capacidad del canal, teoría de la distorsión y aplicaciones al aprendizaje automático y la compresión de datos en computación.
Temas:
Core

- Entropía de Shannon, entropía conjunta y condicional, y la regla de la cadena
- Información mutua, divergencia KL y la desigualdad de procesamiento de datos
- Capacidad del canal: el teorema de codificación de canal ruidoso de Shannon y el canal binario simétrico
- Teoría de la tasa-distorsión y la compensación entre compresión y fidelidad
- Fundamentos de la teoría de la información en ML: MDL, inferencia variacional (ELBO) y el cuello de botella de la información

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Calcular la entropía de Shannon y la información mutua para distribuciones discretas e interpretar su significado [Familiarizarse]
2. Aplicar el teorema de capacidad del canal de Shannon para determinar la tasa máxima de transmisión confiable [Usar]
3. Analizar un algoritmo de aprendizaje usando el principio del cuello de botella de la información y relacionar la compresión con la generalización [Evaluar]

3.7.4. FDS/Reducción de Dimensionalidad (CS Core: 1 hr, KA Core: 1 hr)

Reducción de dimensionalidad lineal y no lineal: ACP, proyecciones aleatorias, aprendizaje de variedades y métodos modernos (t-SNE, UMAP), ampliamente usados en ciencia de datos y visualización.
Temas:
Core

- ACP: descomposición en valores singulares, varianza explicada y aproximación óptima de bajo rango
- Lema de Johnson-Lindenstrauss y proyecciones aleatorias para reducción de dimensionalidad
- Aprendizaje de variedades: Isomap, embedding lineal local (LLE) y embedding espectral
- Métodos no lineales modernos: t-SNE y UMAP; teoría, parámetros y limitaciones
- Autoencoders y autoencoders variacionales (VAE) como reducción de dimensionalidad no lineal

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar el Lema de Johnson-Lindenstrauss y justificar por qué las proyecciones aleatorias preservan las distancias [Familiarizarse]
2. Aplicar ACP y t-SNE a datos de alta dimensionalidad e interpretar los embeddings de baja dimensión resultantes [Usar]
3. Comparar los métodos de reducción de dimensionalidad lineal y no lineal respecto a la preservación de la geometría y el costo computacional [Evaluar]

3.7.5. FDS/Fundamentos Matemáticos del Aprendizaje Profundo (CS Core: 1 hr, KA Core: 1 hr)

Aproximación universal, el núcleo tangente neuronal, sesgo implícito del SGD, sobreparametrización y aprendizaje profundo geométrico, la matemática de frontera de la IA moderna.
Temas:
Core

- Teoremas de aproximación universal: compensaciones de anchura y profundidad para redes ReLU
- Núcleo tangente neuronal (NTK) y el límite de ancho infinito de las redes neuronales
- Regularización implícita y sesgo implícito del SGD hacia soluciones de norma mínima
- Sobreajuste benigno y doble descenso en modelos sobreparametrizados
- Aprendizaje profundo geométrico: equivarianza, simetría de grupos y redes neuronales de grafos

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enunciar el teorema de aproximación universal e identificar sus hipótesis y limitaciones [Familiarizarse]
2. Explicar el núcleo tangente neuronal y su papel en la comprensión de la dinámica de entrenamiento de redes anchas [Usar]
3. Analizar el sesgo implícito del descenso de gradiente y relacionarlo con la generalización en regímenes sobreparametrizados [Evaluar]

3.8. Información Cuántica y Computación (QIC)

Esta área cubre la teoría matemática de la información y la computación cuántica, incluyendo el formalismo algebraico lineal de la mecánica cuántica, los algoritmos cuánticos, la teoría del entrelazamiento, la corrección de errores y los métodos de redes tensoriales. Esta área del conocimiento de frontera es cada vez más esencial para los estudiantes de Computación que trabajan con software cuántico y seguridad postcuántica.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.8.1 Fundamentos de la Mecánica Cuántica para la Computación (Pág. 176)	1	1
⭐ 3.8.2 Circuitos y Puertas Cuánticas (Pág. 176)	1	1
⭐ 3.8.3 Algoritmos Cuánticos (Pág. 176)	1	1
⭐ 3.8.4 Entrelazamiento y Desigualdades de Bell (Pág. 177)	1	1

Cuadro 3.8: Lista de KUs del área de Información Cuántica y Computación.

3.8.1. QIC/Fundamentos de la Mecánica Cuántica para la Computación (CS Core: 1 hr, KA Core: 1 hr)

Postulados de la mecánica cuántica formulados para la computación: espacios de estados, observables, medición y evolución unitaria.

Temas:

Core

- Formalismo del espacio de Hilbert: vectores de estado, notación de Dirac y productos internos
- Observables como operadores hermíticos: valores propios, estados propios y el teorema espectral
- Postulado de medición: regla de Born, mediciones proyectivas y colapso del estado
- Evolución unitaria: la ecuación de Schrödinger y el operador de evolución temporal
- Matrices de densidad: estados mezclados, traza parcial y la representación en la esfera de Bloch

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Enunciar los postulados de la mecánica cuántica y representar los estados de un qubit en la esfera de Bloch [Familiarizarse]
2. Calcular probabilidades de medición y estados post-medición usando la regla de Born y los operadores de proyección [Usar]
3. Distinguir estados puros de estados mezclados usando matrices de densidad y la traza parcial sobre un subsistema [Evaluar]

3.8.2. QIC/Circuitos y Puertas Cuánticas (CS Core: 1 hr, KA Core: 1 hr)

Qubits, puertas cuánticas, circuitos cuánticos, universalidad y el modelo de computación por circuitos.

Temas:

Core

- Puertas de un qubit: Pauli, Hadamard, fase y puertas de rotación
- Puertas de múltiples qubits: CNOT, Toffoli y puertas controladas-U
- Circuitos cuánticos: diagramas de circuitos, profundidad y reversibilidad
- Universalidad: el teorema de Solovay-Kitaev y los conjuntos de puertas universales
- Paralelismo cuántico, interferencia y el papel de la superposición en la computación

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Construir circuitos cuánticos usando conjuntos de puertas estándar y representar su acción sobre estados base [Familiarizarse]
2. Analizar la acción de un circuito cuántico sobre un estado de entrada usando multiplicación matricial y notación de Dirac [Usar]
3. Justificar la universalidad de un conjunto de puertas usando el teorema de Solovay-Kitaev y aproximar un unitario objetivo [Evaluar]

3.8.3. QIC/Algoritmos Cuánticos (CS Core: 1 hr, KA Core: 1 hr)

Transformada de Fourier cuántica, estimación de fase, el algoritmo de factorización de Shor, la búsqueda de Grover y el algoritmo HHL para sistemas lineales.

Temas:

Core

- Transformada de Fourier cuántica: construcción del circuito y análisis de complejidad
- Algoritmo de estimación de fase cuántica y su papel como subrutina
- Algoritmo de factorización de Shor: búsqueda de período y reducción a factorización
- Algoritmo de búsqueda de Grover: aceleración cuadrática y amplificación de amplitud

Non Core

- Algoritmo HHL para sistemas lineales y aceleración cuántica para el aprendizaje automático

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Describir la transformada de Fourier cuántica y explicar cómo la estimación de fase extrae información de valores propios [Familiarizarse]
2. Seguir paso a paso los algoritmos de Shor y Grover e identificar la fuente de la aceleración cuántica [Usar]

NonCore:

3. Analizar los requisitos de recursos del algoritmo HHL e identificar las condiciones bajo las cuales logra aceleración exponencial [Evaluar]

3.8.4. QIC/Entrelazamiento y Desigualdades de Bell (CS Core: 1 hr, KA Core: 1 hr)

Entrelazamiento cuántico: estados de Bell, desigualdades de Bell, la desigualdad CHSH, el teorema de no clonación y medidas de entrelazamiento.

Temas:

Core

- Estados de Bell, estados entrelazados vs. separables y la descomposición de Schmidt
- Desigualdades de Bell y la desigualdad CHSH: teorías de variables ocultas locales vs. mecánica cuántica
- Teorema de no clonación y teorema de no comunicación
- Medidas de entrelazamiento: entropía de entrelazamiento, concurrencia y negatividad
- Teleportación cuántica y protocolos de codificación superdensa

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar qué es el entrelazamiento y por qué las violaciones de la desigualdad de Bell descartan las teorías de variables ocultas locales [Familiarizarse]
2. Construir circuitos de estados de Bell y verificar el entrelazamiento usando la desigualdad CHSH [Usar]
3. Analizar el protocolo de teleportación cuántica y verificar que no viaja información más rápida que la luz [Evaluar]

3.9. Electromagnetismo (ELM)

Esta área cubre la física de los campos eléctricos y magnéticos, los circuitos y las ondas electromagnéticas al nivel necesario para que los estudiantes de Computación comprendan la electrónica de hardware, los circuitos digitales, la transmisión de señales y los fundamentos físicos de los dispositivos de cómputo.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.9.1 Electrostática: Campo Eléctrico y Ley de Gauss (Pág. 178)	1	1
⭐ 3.9.2 Corriente Eléctrica, Circuitos y Leyes de Kirchhoff (Pág. 178)	1	1
⭐ 3.9.3 Ecuaciones de Maxwell y Transformaciones de Norma (Pág. 178)	1	1

Cuadro 3.9: Lista de KUs del área de Electromagnetismo.

3.9.1. ELM/Electrostática: Campo Eléctrico y Ley de Gauss (CS Core: 1 hr, KA Core: 1 hr)

Análisis de cargas en reposo, la naturaleza vectorial del campo eléctrico y la aplicación de la Ley de Gauss, proporcionando la base para entender los capacitores y el almacenamiento de carga en hardware de cómputo.

Temas:

Core

- Ley de Coulomb para cargas puntuales y el principio de superposición lineal
- El vector campo eléctrico $\vec{E}$ para distribuciones de carga discretas y continuas
- La Ley de Gauss en forma integral y su relación con la carga encerrada
- Forma diferencial de la Ley de Gauss: $\nabla \cdot \vec{E} = \rho/\epsilon_0$
- Aplicaciones de la Ley de Gauss a simetría esférica y cilíndrica

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Calcular la fuerza y el campo electrostático para distribuciones de carga usando superposición y la Ley de Gauss [Usar]
2. Evaluar el flujo eléctrico a través de superficies y relacionarlo con la carga encerrada [Evaluar]
3. Determinar la densidad de carga volumétrica a partir de una expresión dada del campo eléctrico usando la forma diferencial de la Ley de Gauss [Usar]

3.9.2. ELM/Corriente Eléctrica, Circuitos y Leyes de Kirchhoff (CS Core: 1 hr, KA Core: 1 hr)

Estudio del flujo de carga eléctrica, la resistencia y el análisis de circuitos DC usando las leyes de Kirchhoff, esenciales para comprender el hardware de lógica digital y los sistemas empotrados.

Temas:

Core

- Definición de corriente, densidad de corriente y la ecuación de continuidad
- Ley de Ohm, resistividad y dependencia de la resistencia con la temperatura
- Ley de Corriente de Kirchhoff (LCK) para el análisis de nodos
- Ley de Voltaje de Kirchhoff (LVK) para el análisis de mallas
- Simplificación de circuitos en serie y paralelo y resistencia equivalente
- Transitorios en circuitos RC y RL y constantes de tiempo

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Aplicar la LCK y la LVK para resolver circuitos DC de múltiples mallas relevantes para el diseño de hardware digital [Usar]
2. Calcular la potencia disipada y determinar las constantes de tiempo para circuitos RC y RL [Evaluar]
3. Analizar la respuesta transitoria de circuitos RC y RL y relacionarla con la temporización de señales en sistemas digitales [Usar]

3.9.3. ELM/Ecuaciones de Maxwell y Transformaciones de Norma (CS Core: 1 hr, KA Core: 1 hr)

El conjunto unificado de ecuaciones que gobiernan todos los fenómenos electromagnéticos clásicos y el vector de Poynting, proporcionando la base física para las comunicaciones inalámbricas y la compatibilidad electromagnética.

Temas:

Core

- Corriente de desplazamiento de Maxwell y su papel en completar la Ley de Ampère
- El conjunto completo de cuatro ecuaciones de Maxwell en forma integral y diferencial
- Teorema de Poynting y la conservación de la energía en campos electromagnéticos
- El vector de Poynting $\vec{S} = \vec{E} \times \vec{H}$ como densidad de flujo de energía
- Invarianza de norma y la formulación de potenciales en la electrodinámica

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enunciar las cuatro ecuaciones de Maxwell e identificar su contenido físico [Familiarizarse]
2. Calcular la corriente de desplazamiento y el vector de Poynting para una configuración electromagnética [Usar]
3. Analizar el flujo de energía y el transporte de potencia en sistemas electromagnéticos incluyendo líneas de transmisión [Evaluar]

3.10. Termodinámica y Mecánica Estadística (TSM)

Esta área cubre las leyes de la transferencia de energía, la entropía, los estados de equilibrio y los fundamentos estadísticos del comportamiento macroscópico. Para los estudiantes de Computación, proporciona la base física para el diseño de sistemas energéticamente eficientes, la gestión térmica del hardware y la mecánica estadística subyacente a los algoritmos de aprendizaje automático.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.10.1 Leyes de la Termodinámica y Funciones de Estado (Pág. 179)	1	1
⭐ 3.10.2 Teoría Cinética de los Gases (Pág. 179)	1	1
⭐ 3.10.3 Máquinas de Calor, Entropía y la Segunda Ley (Pág. 180)	1	1

Cuadro 3.10: Lista de KUs del área de Termodinámica y Mecánica Estadística.

3.10.1. TSM/Leyes de la Termodinámica y Funciones de Estado (CS Core: 1 hr, KA Core: 1 hr)

Principios fundamentales que gobiernan la energía, el calor y el trabajo en los sistemas termodinámicos, y la definición matemática de las variables de estado.
Temas:
Core

- Ley Cero de la Termodinámica y el equilibrio térmico
- Primera Ley de la Termodinámica: $dU = dQ - dW$ para sistemas de ingeniería
- Entalpía $H = U + PV$ y su papel en los procesos de flujo
- Calores específicos C_V y C_P y su medición y aplicación

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enunciar las tres leyes de la termodinámica y dar una interpretación física de cada una [Familiarizarse]
2. Calcular el trabajo realizado por un gas durante expansiones isotérmicas y adiabáticas [Usar]
3. Evaluar cambios de energía interna y aplicar la Primera Ley a procesos termodinámicos en sistemas de cómputo [Evaluar]

3.10.2. TSM/Teoría Cinética de los Gases (CS Core: 1 hr, KA Core: 1 hr)

Modelo microscópico de los gases que relaciona el movimiento molecular con propiedades macroscópicas como la presión, la temperatura y los coeficientes de transporte.
Temas:
Core

- Derivación de la presión macroscópica a partir de la transferencia de momento molecular
- Distribución de velocidades de Maxwell-Boltzmann y sus implicaciones
- Teorema de equipartición y grados de libertad para gases monoatómicos y poliatómicos
- Recorrido libre medio y su relevancia para el flujo de gases y los sistemas de vacío

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir la base microscópica de la presión y la temperatura en un gas ideal [Familiarizarse]

2. Calcular las velocidades molecular cuadrática media y promedio usando la distribución de Maxwell-Boltzmann [Usar]
3. Aplicar el teorema de equipartición para estimar los calores específicos de los gases y relacionarlos con el almacenamiento de energía en entornos de cómputo [Evaluar]

3.10.3. TSM/Máquinas de Calor, Entropía y la Segunda Ley (CS Core: 1 hr, KA Core: 1 hr)

Límites de eficiencia térmica, generación de entropía y las restricciones termodinámicas, relevantes para la eficiencia energética en centros de datos y hardware de cómputo.

Temas:

Core

- Enunciados de Kelvin-Planck y Clausius de la Segunda Ley y sus implicaciones
- El ciclo de Carnot y la eficiencia máxima $\eta = 1 - T_c/T_h$
- Desigualdad de Clausius y la definición termodinámica de la entropía
- Fórmula de entropía de Boltzmann $S = k_B \ln \Omega$ y su interpretación estadística
- Principio de Landauer: costo energético de la computación irreversible y el borrado de bits

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar la Segunda Ley de la Termodinámica y su conexión con la dirección de los procesos espontáneos [Familiarizarse]
2. Calcular la eficiencia térmica máxima de una máquina de calor que opera entre dos temperaturas [Usar]
3. Aplicar el principio de Landauer para analizar el costo termodinámico de las operaciones de cómputo y el borrado de memoria [Evaluar]

3.11. Física Computacional y Basada en Datos (CDP)

Esta área cubre los métodos de simulación numérica, las técnicas de Monte Carlo, el aprendizaje automático para el descubrimiento físico y las redes neuronales informadas por la física. Conecta la física con la computación, equipando a los estudiantes para construir simulaciones, analizar datos científicos y construir modelos híbridos de física-AM.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.11.1 Métodos Numéricos para Ecuaciones Diferenciales en Física (Pág. 180)	1	1
⭐ 3.11.2 Métodos de Monte Carlo y Simulaciones (Pág. 181)	1	1
⭐ 3.11.3 Aprendizaje Automático para el Descubrimiento Físico (Pág. 181)	1	1

Cuadro 3.11: Lista de KUs del área de Física Computacional y Basada en Datos.

3.11.1. CDP/Métodos Numéricos para Ecuaciones Diferenciales en Física (CS Core: 1 hr, KA Core: 1 hr)

Aproximaciones discretas para resolver ecuaciones fundamentales de movimiento y teorías de campos, esenciales para construir simuladores físicos en computación.

Temas:

Core

- Métodos de Runge-Kutta (RK2, RK4) para problemas de valor inicial
- Dominio del tiempo por diferencias finitas (FDTD) para las ecuaciones de Maxwell
- Problemas de valor de frontera: métodos de disparo y técnicas de relajación
- Integradores simplécticos: algoritmos de Verlet y leapfrog para sistemas hamiltonianos
- Análisis de estabilidad y la condición de Courant-Friedrichs-Lewy (CFL)

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar los requisitos de estabilidad de los integradores numéricos de EDOs e identificar la condición CFL [Familiarizarse]
2. Implementar un algoritmo de Verlet de velocidad para simular un sistema físico con energía conservada [Usar]
3. Evaluar la escala de los errores de discretización para diferentes órdenes de integración numérica [Evaluar]

3.11.2. CDP/Métodos de Monte Carlo y Simulaciones (CS Core: 1 hr, KA Core: 1 hr)

Técnicas de muestreo estocástico para la integración multidimensional y la física estadística, relevantes para la inferencia probabilística y la simulación en computación.

Temas:

Core

- Algoritmos de generación de números pseudoaleatorios y cuasialeatorios
- El algoritmo de Metropolis-Hastings para la Cadena de Markov Monte Carlo (MCMC)
- Simulación Monte Carlo del modelo de Ising y transiciones de fase
- Muestreo por importancia y técnicas de reducción de varianza
- Estimación de error estadístico y tiempo de autocorrelación en cadenas MCMC

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Describir el criterio de aceptación de Metropolis-Hastings y explicar su conexión con el balance detallado [Familiarizarse]
2. Desarrollar un esquema de muestreo por importancia para integrar una función multidimensional con un pico pronunciado [Usar]
3. Analizar la calidad estadística de una estimación de Monte Carlo y cuantificar el sesgo inducido por la autocorrelación [Evaluar]

3.11.3. CDP/Aprendizaje Automático para el Descubrimiento Físico (CS Core: 1 hr, KA Core: 1 hr)

Aplicación del aprendizaje estadístico para identificar patrones y modelos sustitutos en datos, incluyendo redes neuronales informadas por la física y operadores neuronales para resolver EDPs.

Temas:

Core

- Aprendizaje supervisado: regresión lineal, LASSO y Ridge para modelos físicos
- Aprendizaje no supervisado: agrupamiento y ACP para detectar transiciones de fase
- Regresión por Procesos Gaussianos para el modelado sustituto con cuantificación de incertidumbre
- Regresión simbólica para descubrir leyes físicas analíticas a partir de datos
- Redes Neuronales Informadas por la Física (PINNs): incorporación de restricciones de EDP en la función de pérdida de redes neuronales

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar cómo se pueden codificar las restricciones físicas en la función de pérdida de una red neuronal [Familiarizarse]
2. Entrenar un modelo de regresión o de Procesos Gaussianos para predecir una cantidad física y cuantificar la incertidumbre [Usar]
3. Evaluar y comparar modelos sustitutos basados en datos con solucionadores numéricos tradicionales en cuanto a precisión y costo computacional [Evaluar]

3.12. Mecánica Clásica (CME)

La Mecánica Clásica proporciona los principios fundamentales que gobiernan el movimiento de partículas y cuerpos rígidos. Este subconjunto orientado a la computación enfatiza la cinemática,

los métodos energéticos, la rotación de cuerpos rígidos, los fenómenos oscilatorios y la mecánica variacional, relevantes para robótica, motores de juegos, simulaciones físicas y aprendizaje automático informado por la física.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.12.1 Movimiento Rotacional y Momento Angular (Pág. 182)	1	1
⭐ 3.12.2 Sistemas de Partículas y Momento Lineal (Pág. 183)	1	1
⭐ 3.12.3 Cinemática y Dinámica de Partículas (Pág. 183)	1	1
⭐ 3.12.4 Trabajo, Energía y Leyes de Conservación (Pág. 184)	1	1
⭐ 3.12.5 Dinámica de Cuerpos Rígidos (Pág. 184)	1	1
⭐ 3.12.6 Oscilaciones y Ondas (Pág. 185)	1	1
⭐ 3.12.7 Mecánica Lagrangiana y Hamiltoniana (Pág. 185)	1	1
⭐ 3.12.8 Elasticidad (Pág. 185)	1	1
⭐ 3.12.9 Fluidos: Hidrostática, Hidrodinámica y Viscosidad (Pág. 186)	1	1
⭐ 3.12.10 Vibraciones y Ondas (Pág. 186)	1	1
⭐ 3.12.11 Fenómeno de Transporte (Pág. 187)	1	1

Cuadro 3.12: Lista de KUs del área de Mecánica Clásica.

3.12.1. CME/Movimiento Rotacional y Momento Angular (CS Core: 1 hr, KA Core: 1 hr)

Cinemática y dinámica rotacional, torque, momento de inercia, energía rotacional y conservación del momento angular para partículas y sistemas.

Temas:

Core

- Cinemática rotacional incluyendo desplazamiento angular, velocidad y aceleración (θ, ω, α)
- Definición de torque: $\vec{\tau} = \vec{r} \times \vec{F}$ y el cálculo de su magnitud
- Momento angular de una partícula: $\vec{L} = \vec{r} \times \vec{p}$
- Forma rotacional de la segunda ley de Newton: $\vec{\tau}_{net} = \frac{d\vec{L}}{dt}$
- Momento de inercia $I = \sum m_i r_i^2$ y el teorema de los ejes paralelos
- Energía cinética rotacional: $K_{rot} = \frac{1}{2} I\omega^2$
- Conservación del momento angular para sistemas aislados
- Movimiento giroscópico y el fenómeno de la precesión
- Dinámica de rodadura sin deslizamiento: $v_{CM} = R\omega$

Electivo

- El efecto Coriolis como manifestación de la dinámica rotacional en marcos no inerciales

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Relacionar las magnitudes cinemáticas lineales y angulares para el movimiento circular [Usar]
2. Calcular el torque respecto a un eje dado para una fuerza aplicada en un punto específico [Usar]
3. Determinar el momento de inercia para geometrías simples como una varilla, un aro, un disco o una esfera [Usar]
4. Aplicar la conservación del momento angular para analizar cambios en la velocidad de rotación cuando cambia el momento de inercia [Evaluar]
5. Calcular la energía cinética rotacional de un cuerpo rígido en rotación [Usar]
6. Analizar el movimiento de objetos que ruedan sin deslizar por un plano inclinado [Evaluar]
7. Predecir la dirección de la precesión para una peonza o un giroscopio en rotación [Evaluar]

Elective:

8. Explicar el efecto Coriolis en términos de marcos de referencia rotatorios y su efecto sobre los proyectiles [Familiarizarse]

3.12.2. CME/Sistemas de Partículas y Momento Lineal (CS Core: 1 hr, KA Core: 1 hr)

Dinámica de sistemas de múltiples partículas: movimiento del centro de masa, conservación del momento lineal, fuerzas internas y externas, y sistemas de masa variable.

Temas:

Core

- Definición de centro de masa: $\vec{R}_{CM} = \frac{1}{M}\sum m_i \vec{r}_i$
- Momento lineal de una partícula y de un sistema: $\vec{p} = m\vec{v}$ y $\vec{P} = M\vec{V}_{CM}$
- Segunda ley de Newton para sistemas: $\vec{F}_{ext} = \frac{d\vec{P}}{dt}$
- Conservación del momento lineal para sistemas aislados
- Distinción entre fuerzas internas y externas en un sistema
- Colisiones de dos cuerpos, incluyendo casos elásticos, inelásticos y completamente inelásticos
- Coeficiente de restitución y su significado físico en las colisiones
- Sistemas con masa variable, como la ecuación del cohete: $M\frac{dV}{dt} = v_{ex}\frac{dM}{dt}$
- Sistemas de muchas partículas y su dinámica del centro de masa

Electivo

- Distribuciones continuas de masa y el cálculo de su centro de masa mediante integración

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Localizar el centro de masa para un conjunto discreto de partículas en Una, Dos y Tres dimensiones [Usar]
2. Determinar el momento lineal total de un sistema a partir del movimiento de su centro de masa [Usar]
3. Aplicar la conservación del momento lineal para resolver problemas de colisión en Una y Dos dimensiones [Usar]
4. Clasificar colisiones como elásticas, inelásticas o completamente inelásticas basándose en la conservación de la energía cinética [Familiarizarse]
5. Calcular el coeficiente de restitución a partir de las velocidades antes y después de la colisión [Usar]
6. Resolver la ecuación del cohete para determinar la velocidad en función de la masa expulsada [Usar]
7. Analizar cómo las fuerzas internas afectan a las partículas individuales pero no al momento total de un sistema [Evaluar]

Elective:

8. Integrar para hallar el centro de masa de cuerpos continuos simétricos como varillas, discos y esferas [Usar]

3.12.3. CME/Cinemática y Dinámica de Partículas (CS Core: 1 hr, KA Core: 1 hr)

Descripción matemática del movimiento de partículas y las fuerzas que lo gobiernan, con énfasis en el análisis del espacio de fases y los métodos de integración numérica utilizados en robótica y simulación.

Temas:

Core

- Descripción vectorial de posición, velocidad y aceleración; las tres leyes de Newton y marcos de referencia inerciales
- Formulación y solución analítica de ecuaciones de movimiento para fuerzas constantes y variables
- Diagramas del espacio de fases (posición vs. velocidad) para visualizar el estado mecánico, la estabilidad y la periodicidad
- Integración numérica de ecuaciones de movimiento usando métodos de Euler, Verlet y Runge-Kutta
- Movimiento con restricciones y ligaduras holonómicas relevantes para eslabones robóticos y cuerpos articulados

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar todas las fuerzas que actúan sobre una partícula en un escenario mecánico dado y trazar el diagrama correcto de cuerpo libre [Familiarizarse]
2. Implementar un algoritmo de integración numérica como el método de Runge-Kutta para calcular la trayectoria de una partícula bajo fuerzas variables [Usar]
3. Construir e interpretar diagramas del espacio de fases para caracterizar la estabilidad y periodicidad de un sistema mecánico simulado [Evaluar]

3.12.4. CME/Trabajo, Energía y Leyes de Conservación (CS Core: 1 hr, KA Core: 1 hr)

Métodos energéticos y principios de conservación que sustentan los motores de física, la resolución de colisiones y la optimización eficiente en sistemas computacionales.
Temas:
Core

- Teorema trabajo-energía que relaciona el trabajo neto realizado con el cambio en la energía cinética
- Funciones de energía potencial y la relación fuerza-gradiente para campos de fuerza conservativos
- Conservación de la energía mecánica total y condiciones para su aplicabilidad
- Diagramas de energía para identificar puntos de equilibrio, puntos de retorno y estabilidad
- Modelos de colisiones elásticas e inelásticas: intercambio de momento y energía en implementaciones de motores de física

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar la distinción entre fuerzas conservativas y no conservativas y las condiciones bajo las cuales se conserva la energía mecánica [Familiarizarse]
2. Aplicar métodos de conservación de energía para calcular velocidades y posiciones en problemas de movimiento unidimensional comunes en implementaciones de motores de física [Usar]
3. Analizar eventos de colisión elástica e inelástica, calculando velocidades post-colisión y cuantificando la disipación de energía [Evaluar]

3.12.5. CME/Dinámica de Cuerpos Rígidos (CS Core: 1 hr, KA Core: 1 hr)

Dinámica rotacional de cuerpos rígidos incluyendo torque, momento de inercia y momento angular, directamente aplicable a la física de motores de juegos y cinemática de robots.
Temas:
Core

- Definición del torque y momento de inercia para geometrías estándar: barra, disco, esfera y caja
- Vector de momento angular y su conservación para sistemas rotativos aislados
- Ecuaciones de movimiento rotacional y acoplamiento entre grados de libertad traslacionales y rotacionales
- Ángulos de Euler y representaciones en cuaterniones para el seguimiento de orientación tridimensional
- Condición de rodadura sin deslizamiento y fuerzas de restricción en sistemas de cuerpos rígidos

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar cómo el momento de inercia gobierna la respuesta rotacional de un cuerpo rígido ante los torques aplicados [Familiarizarse]
2. Calcular la velocidad angular y la orientación de un cuerpo rígido dado un historial de torques aplicados y condiciones iniciales [Usar]
3. Diseñar una simulación de cuerpo rígido para un escenario de motor de juegos, seleccionando integradores numéricos apropiados y representaciones de orientación [Evaluar]

3.12.6. CME/Oscilaciones y Ondas (CS Core: 1 hr, KA Core: 1 hr)

Movimiento armónico simple, amortiguamiento, resonancia y propagación de ondas, que proporcionan las bases mecánicas para el procesamiento de señales, la síntesis de audio y el análisis de vibraciones en aplicaciones computacionales.

Temas:

Core

- Oscilador armónico simple: frecuencia, período, amplitud, fase y energía
- Osciladores amortiguados: regímenes subamortiguado, críticamente amortiguado y sobreamortiguado y sus respuestas en el dominio temporal
- Oscilaciones forzadas, frecuencia de resonancia, respuesta en amplitud y factor de calidad Q
- Osciladores acoplados y descomposición en modos normales para sistemas de múltiples grados de libertad
- Propagación de ondas mecánicas: ecuación de onda, relación de dispersión y velocidades de grupo y de fase

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Describir el comportamiento físico de los osciladores subamortiguado, críticamente amortiguado y sobreamortiguado, y relacionar cada régimen con aplicaciones en sistemas de control y audio [Familiarizarse]
2. Resolver la ecuación del oscilador armónico amortiguado de forma analítica y numérica para condiciones iniciales y funciones de forzado dadas [Usar]
3. Analizar la respuesta en amplitud-frecuencia de un oscilador forzado y cuantificar el efecto del amortiguamiento sobre el ancho de banda de resonancia y la ganancia máxima [Evaluar]

3.12.7. CME/Mecánica Lagrangiana y Hamiltoniana (CS Core: 1 hr, KA Core: 1 hr)

Formulaciones variacionales de la mecánica que sustentan los integradores simplécticos, las simulaciones de física diferenciable y el aprendizaje automático informado por la física.

Temas:

Core

- Principio de acción estacionaria de Hamilton y su papel en la derivación de ecuaciones de movimiento
- Ecuaciones de Euler-Lagrange en coordenadas generalizadas para sistemas mecánicos con ligaduras
- Formulación hamiltoniana: coordenadas canónicas, flujo en el espacio de fases e interpretación energética
- Integradores simplécticos (Verlet, leapfrog) y sus propiedades de conservación de energía para simulaciones de larga duración
- Teorema de Noether que relaciona simetrías continuas con cantidades conservadas

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar el principio variacional que subyace a la mecánica lagrangiana y contrastarlo con el enfoque newtoniano basado en fuerzas [Familiarizarse]
2. Construir el lagrangiano de un sistema mecánico con ligaduras holonómicas y derivar sus ecuaciones de movimiento mediante las ecuaciones de Euler-Lagrange [Usar]
3. Evaluar la conservación de energía a largo plazo de los integradores simplécticos frente a los no simplécticos mediante experimentos numéricos en un sistema hamiltoniano [Evaluar]

3.12.8. CME/Elasticidad (CS Core: 1 hr, KA Core: 1 hr)

Propiedades elásticas de sólidos bajo carga axial, de corte y volumétrica: esfuerzo, deformación unitaria y módulos elásticos escalares.

Temas:

Core

- Definición de esfuerzo (fuerza/área) y deformación unitaria ($\Delta L/L$)
- Módulo de Young y ley de Hooke para deformación longitudinal: $\sigma = E\epsilon$
- Módulo de corte (rigidez) y deformación por cizalladura
- Módulo de volumen (compresibilidad) y deformación volumétrica
- Límite elástico, región plástica y punto de ruptura en el diagrama esfuerzo-deformación

Electivo

- Energía elástica almacenada en un material deformado

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Aplicar la ley de Hooke escalar para calcular la deformación de una barra bajo carga axial [Usar]
2. Calcular el módulo de Young, de corte o de volumen a partir de datos experimentales esfuerzo-deformación [Usar]
3. Identificar la región elástica, el límite elástico y el punto de fractura en una curva esfuerzo-deformación [Familiarizarse]

Elective:

4. Calcular la energía potencial elástica almacenada en un resorte o barra deformada [Usar]

3.12.9. CME/Fluidos: Hidrostática, Hidrodinámica y Viscosidad (CS Core: 1 hr, KA Core: 1 hr)

Estática y dinámica de fluidos incompresibles: presión, empuje, ecuaciones de continuidad y de Bernoulli, e introducción al flujo viscoso.

Temas:

Core

- Densidad, presión y su variación con la profundidad en un fluido en reposo: $p = p_0 + \rho gh$
- Principio de Pascal y sus aplicaciones (prensa hidráulica)
- Principio de Arquímedes, empuje y flotación
- Medición de presión con manómetros y barómetros
- Ecuación de continuidad para flujo incompresible: $A_1 v_1 = A_2 v_2$
- Ecuación de Bernoulli y conservación de energía en flujo ideal
- Aplicaciones: tubo de Venturi, teorema de Torricelli, tubo de Pitot

Electivo

- Viscosidad y ley de Poiseuille para flujo laminar en tuberías
- Número de Reynolds y transición a flujo turbulento
- Ley de Stokes para la fuerza de arrastre sobre una esfera en un fluido viscoso

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Calcular la presión en un punto de un fluido en reposo dado la profundidad y densidad [Usar]
2. Aplicar el principio de Arquímedes para determinar si un objeto flota, se hunde o permanece en equilibrio [Usar]
3. Aplicar las ecuaciones de continuidad y de Bernoulli para resolver problemas de flujo en tuberías de sección variable [Usar]
4. Interpretar lecturas de manómetros y barómetros para determinar presiones absolutas y manométricas [Evaluar]

Elective:

5. Aplicar la ley de Poiseuille para calcular el caudal de un fluido viscoso en un tubo cilíndrico [Usar]
6. Estimar el número de Reynolds de un flujo y predecir si es laminar o turbulento [Evaluar]

3.12.10. CME/Vibraciones y Ondas (CS Core: 1 hr, KA Core: 1 hr)

Movimiento armónico simple, oscilaciones amortiguadas y forzadas, y propagación, superposición y resonancia de ondas mecánicas.

Temas:

Core

- Movimiento armónico simple (MAS): ecuación de movimiento, posición, velocidad y aceleración
- Sistemas oscilantes: resorte-masa y péndulo simple/físico
- Energía en el movimiento armónico simple
- Oscilaciones amortiguadas y forzadas; introducción a la resonancia
- Ondas transversales y longitudinales; velocidad de propagación
- Ecuación de onda y velocidad de una onda en una cuerda tensa: $v = \sqrt{T/\mu}$
- Principio de superposición y ondas estacionarias
- Ondas sonoras, intensidad y nivel de intensidad sonora (decibeles)

Electivo

- Efecto Doppler para ondas sonoras
- Pulsaciones (beats) e interferencia de ondas de frecuencias cercanas

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Deducir la ecuación de movimiento y el período de un sistema resorte-masa y de un péndulo simple [Usar]
2. Calcular la energía cinética, potencial y total de un oscilador armónico simple en cualquier instante [Usar]
3. Analizar la respuesta en amplitud de un oscilador forzado cerca de la frecuencia de resonancia [Evaluar]
4. Calcular la velocidad de propagación de una onda en una cuerda a partir de la tensión y la densidad lineal [Usar]
5. Determinar las frecuencias y longitudes de onda de los modos normales de una onda estacionaria en una cuerda fija [Usar]
6. Calcular el nivel de intensidad sonora en decibeles a partir de la intensidad de una onda sonora [Usar]

Elective:

7. Aplicar la fórmula del efecto Doppler para fuente y/o observador en movimiento [Usar]

3.12.11. CME/Fenómeno de Transporte (CS Core: 1 hr, KA Core: 1 hr)

Introducción a los fenómenos de transporte en medios continuos: conducción de calor y difusión de masa como procesos de flujo impulsados por gradientes.

Temas:

Core

- Conducción de calor y la ley de Fourier: $\vec{q} = -k\nabla T$
- Conductividad térmica de materiales y resistencia térmica en paredes compuestas
- Difusión molecular y la primera ley de Fick: $\vec{J} = -D\nabla C$

Electivo

- Analogía entre el transporte de calor, masa y momento (números de Prandtl y Schmidt)

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Aplicar la ley de Fourier para calcular el flujo de calor a través de una pared plana o compuesta [Usar]
2. Aplicar la primera ley de Fick para calcular el flujo difusivo de una especie en un gradiente de concentración [Usar]

Elective:

3. Identificar analogías matemáticas entre los procesos de transporte de calor, masa y momento [Familiarizarse]

3.13. Química General (GCH)

Esta área de conocimiento cubre los fundamentos introductorios de la química: la naturaleza de la materia y la energía, la estructura atómica, el enlace químico, la estequiometría y los estados de la materia. Constituye la base conceptual sobre la cual se construyen las demás áreas de la química

y sus aplicaciones en ingeniería y ciencias de la computación (ciencia de materiales, nanotecnología, procesos industriales).

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.13.1 Estructura Atómica (Pág. 188)	1	1
⭐ 3.13.2 Enlace Químico (Pág. 188)	1	1
⭐ 3.13.3 Materia y Energía (Pág. 189)	1	1
⭐ 3.13.4 Estados de la Materia (Pág. 189)	1	1
⭐ 3.13.5 Estequiometría (Pág. 189)	1	1
⭐ 3.13.6 Química Ambiental y del Agua (Pág. 190)	1	1
⭐ 3.13.7 Polímeros y Geosintéticos (Pág. 190)	1	1
⭐ 3.13.8 Equilibrio Químico (Pág. 191)	1	1
⭐ 3.13.9 Equilibrio Iónico e Hidrólisis de Sales (Pág. 191)	1	1
⭐ 3.13.10 Sales, Solubilidad y Equilibrios de Precipitación (Pág. 191)	1	1
⭐ 3.13.11 Electroquímica y Electrólisis (Pág. 191)	1	1
⭐ 3.13.12 Química del Cemento (Pág. 192)	1	1

Cuadro 3.13: Lista de KUs del área de Química General.

3.13.1. GCH/Estructura Atómica (CS Core: 1 hr, KA Core: 1 hr)

Evolución del modelo atómico, partículas subatómicas, configuración electrónica y organización de la tabla periódica, y las tendencias periódicas que de ella se derivan.

Temas:

Core

- Evolución histórica del modelo atómico: Dalton, Thomson, Rutherford y Bohr
- Partículas subatómicas, número atómico, número de masa e isótopos
- Configuración electrónica, números cuánticos y el principio de Aufbau
- Organización de la tabla periódica: grupos, periodos y bloques
- Tendencias periódicas: radio atómico, energía de ionización, afinidad electrónica y electronegatividad

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Describir la evolución histórica del modelo atómico y las contribuciones de cada modelo a la comprensión actual del átomo [Familiarizarse]
2. Determinar la configuración electrónica de un átomo o ion a partir de su posición en la tabla periódica [Usar]
3. Predecir tendencias periódicas (radio atómico, energía de ionización) para comparar dos o más elementos [Evaluar]

3.13.2. GCH/Enlace Químico (CS Core: 1 hr, KA Core: 1 hr)

Tipos de enlace químico, estructuras de Lewis, geometría molecular mediante la teoría RPECV y las fuerzas intermoleculares que determinan las propiedades físicas de las sustancias.

Temas:

Core

- Enlace iónico: formación, energía reticular y propiedades de los compuestos iónicos
- Enlace covalente, estructuras de Lewis y el concepto de resonancia
- Geometría molecular mediante la teoría de repulsión de pares de electrones de valencia (RPECV)
- Electronegatividad, polaridad de enlaces y polaridad molecular
- Fuerzas intermoleculares: dipolo-dipolo, puente de hidrógeno y fuerzas de dispersión de London

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Dibujar estructuras de Lewis correctas, incluyendo casos de resonancia, para moléculas e iones poliatómicos [Usar]

2. Predecir la geometría molecular y la polaridad de una molécula aplicando la teoría RPECV [Usar]
3. Relacionar el tipo de fuerza intermolecular presente en una sustancia con sus propiedades físicas observadas (punto de ebullición, solubilidad) [Evaluar]

3.13.3. GCH/Materia y Energía (CS Core: 1 hr, KA Core: 1 hr)

Clasificación de la materia, sus propiedades y cambios, el sistema de unidades y cifras significativas, y las formas de energía involucradas en los procesos físicos y químicos.

Temas:

Core

- Clasificación de la materia: elementos, compuestos, mezclas homogéneas y heterogéneas
- Propiedades físicas y químicas, y cambios físicos y químicos de la materia
- Sistema Internacional de unidades, cifras significativas y análisis dimensional
- Formas de energía, calor y temperatura; ley de conservación de la masa y la energía
- El método científico aplicado a la química: observación, hipótesis y experimentación

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Clasificar una muestra de materia como elemento, compuesto o mezcla, y distinguir mezclas homogéneas de heterogéneas [Familiarizarse]
2. Aplicar el análisis dimensional y las reglas de cifras significativas en cálculos de mediciones químicas [Usar]
3. Relacionar los cambios de energía observados con procesos físicos y químicos específicos, aplicando la ley de conservación de la energía [Evaluar]

3.13.4. GCH/Estados de la Materia (CS Core: 1 hr, KA Core: 1 hr)

Las leyes de los gases y la teoría cinético-molecular, las propiedades de líquidos y sólidos, los cambios de fase, y las propiedades coligativas de las disoluciones.

Temas:

Core

- Leyes de los gases: Boyle, Charles, Avogadro y la ecuación del gas ideal
- Teoría cinético-molecular de los gases y sus aplicaciones
- Cambios de fase, curvas de calentamiento y diagramas de fase
- Sólidos cristalinos y amorfos; tipos de sólidos cristalinos
- Propiedades coligativas de las disoluciones: disminución de la presión de vapor, elevación ebulloscópica, descenso crioscópico y presión osmótica

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Aplicar la ecuación del gas ideal y las leyes de los gases para resolver problemas de presión, volumen y temperatura [Usar]
2. Interpretar un diagrama de fase para identificar los estados y transiciones de una sustancia a distintas condiciones [Familiarizarse]
3. Calcular las propiedades coligativas de una disolución (presión de vapor, punto de ebullición, punto de congelación) a partir de su concentración [Evaluar]

3.13.5. GCH/Estequiometría (CS Core: 1 hr, KA Core: 1 hr)

El concepto de mol, el balanceo de ecuaciones químicas y los cálculos estequiométricos, incluyendo reactivo limitante, rendimiento porcentual y estequiometría de disoluciones.

Temas:

Core

- Fórmulas químicas, masa molar y el concepto de mol
- Balanceo de ecuaciones químicas y su interpretación cuantitativa
- Cálculos estequiométricos mol-mol, masa-masa y masa-mol
- Reactivo limitante, reactivo en exceso y rendimiento porcentual

- Molaridad, preparación de disoluciones y estequiometría de disoluciones

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Balancear ecuaciones químicas por el método de tanteo, verificando la conservación de la masa [Familiarizarse]
2. Calcular las cantidades de reactivos y productos en una reacción química usando relaciones molares [Usar]
3. Determinar el reactivo limitante y el rendimiento porcentual de una reacción a partir de datos experimentales [Evaluar]

3.13.6. GCH/Química Ambiental y del Agua (CS Core: 1 hr, KA Core: 1 hr)

Composición y calidad del agua como recurso, contaminantes ambientales comunes y fundamentos de química ambiental aplicados a la ingeniería.

Temas:
Core

- Composición química del agua natural y parámetros de calidad (pH, dureza, oxígeno disuelto, DBO)
- Fundamentos de tratamiento de agua: coagulación, floculación y desinfección
- Principales contaminantes del aire y del agua y sus fuentes
- Ciclos biogeoquímicos del carbono, nitrógeno y fósforo

Electivo

- Química atmosférica y el efecto invernadero

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Interpretar parámetros de calidad del agua para evaluar su aptitud de uso [Usar]
2. Describir los procesos básicos de tratamiento de agua potable y residual [Familiarizarse]
3. Identificar las principales fuentes y efectos de los contaminantes del aire y del agua [Familiarizarse]

Elective:

4. Explicar el mecanismo químico del efecto invernadero [Usar]

3.13.7. GCH/Polímeros y Geosintéticos (CS Core: 1 hr, KA Core: 1 hr)

Clasificación y propiedades de los polímeros, con énfasis en los geosintéticos usados en ingeniería civil.

Temas:
Core

- Clasificación de polímeros: termoplásticos, termoestables y elastómeros
- Reacciones de polimerización: adición y condensación
- Propiedades mecánicas y químicas de los polímeros relevantes para ingeniería
- Geosintéticos: geotextiles, geomembranas y geomallas, y sus aplicaciones

Electivo

- Degradación y durabilidad de polímeros en condiciones ambientales

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Clasificar un polímero según su estructura y comportamiento térmico [Usar]
2. Identificar si una reacción de polimerización es por adición o por condensación [Familiarizarse]
3. Seleccionar el tipo de geosintético adecuado según la aplicación de ingeniería civil requerida [Usar]

Elective:

4. Evaluar la durabilidad de un polímero frente a la exposición ambiental [Evaluar]

3.13.8. GCH/Equilibrio Químico (CS Core: 1 hr, KA Core: 1 hr)

Fundamentos del equilibrio químico: la constante de equilibrio, el cociente de reacción y el principio de Le Chatelier.

Temas:

Core

- La constante de equilibrio K_c y K_p, y su relación con la estequiometría de la reacción
- El cociente de reacción Q y la predicción del sentido de una reacción
- Principio de Le Chatelier: efecto de la concentración, la presión y la temperatura
- Tablas ICE para el cálculo de concentraciones en el equilibrio

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Escribir la expresión de la constante de equilibrio para una reacción dada [Usar]
2. Calcular concentraciones en el equilibrio usando tablas ICE [Usar]
3. Predecir el desplazamiento de un equilibrio ante cambios de concentración, presión o temperatura usando el principio de Le Chatelier [Evaluar]

3.13.9. GCH/Equilibrio Iónico e Hidrólisis de Sales (CS Core: 1 hr, KA Core: 1 hr)

Equilibrio ácido-base, el concepto de pH, y la hidrólisis de sales en disolución acuosa.

Temas:

Core

- Teorías ácido-base de Arrhenius, Brønsted-Lowry y Lewis
- Cálculo de pH y pOH para ácidos y bases fuertes
- Equilibrio de ácidos y bases débiles: K_a, K_b y grado de disociación
- Hidrólisis de sales y predicción del carácter ácido, básico o neutro de una disolución
- Disoluciones amortiguadoras (buffer) y la ecuación de Henderson-Hasselbalch

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Calcular el pH de disoluciones de ácidos y bases fuertes y débiles [Usar]
2. Predecir el carácter ácido-base de una disolución de una sal a partir de la hidrólisis de sus iones [Evaluar]
3. Calcular el pH de una disolución amortiguadora usando la ecuación de Henderson-Hasselbalch [Usar]

3.13.10. GCH/Sales, Solubilidad y Equilibrios de Precipitación (CS Core: 1 hr, KA Core: 1 hr)

El producto de solubilidad K_sp y su aplicación a la predicción de la precipitación y disolución de sales poco solubles.

Temas:

Core

- El producto de solubilidad K_{sp} y su relación con la solubilidad molar
- Predicción de precipitación mediante la comparación del cociente de solubilidad con K_{sp}
- Efecto del ion común sobre la solubilidad

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Calcular K_{sp} a partir de la solubilidad molar de una sal, y viceversa [Usar]
2. Predecir si se formará un precipitado al mezclar dos disoluciones dadas sus concentraciones y el K_{sp} correspondiente [Evaluar]

3.13.11. GCH/Electroquímica y Electrólisis (CS Core: 1 hr, KA Core: 1 hr)

Reacciones de oxidación-reducción, celdas galvánicas y electrolíticas, y sus aplicaciones a la corrosión de metales y del concreto armado.

Temas:

Core

- Reacciones de oxidación-reducción y balanceo por el método del ion-electrón
- Celdas galvánicas, potenciales estándar de electrodo y la ecuación de Nernst
- Celdas electrolíticas y electrólisis; leyes de Faraday
- Corrosión electroquímica de metales y métodos de protección
- Corrosión del acero de refuerzo en concreto armado

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Balancear ecuaciones redox por el método del ion-electrón en medio ácido o básico [Usar]
2. Calcular el potencial de una celda galvánica y predecir la espontaneidad de una reacción redox [Usar]
3. Aplicar las leyes de Faraday para calcular la masa depositada o el tiempo en un proceso de electrólisis [Usar]
4. Analizar las condiciones que favorecen la corrosión de un metal o del acero de refuerzo en concreto, y proponer métodos de protección [Evaluar]

3.13.12. GCH/Química del Cemento (CS Core: 1 hr, KA Core: 1 hr)

Composición química del cemento Portland, sus reacciones de hidratación y las propiedades resultantes del concreto.
Temas:
Core

- Composición química del cemento Portland: silicatos, aluminatos y ferroaluminatos de calcio
- Reacciones de hidratación del cemento y desarrollo de resistencia
- Rol de cada compuesto (C_3S, C_2S, C_3A, C_4AF) en las propiedades del cemento

Electivo

- Materiales cementantes suplementarios: puzolanas, escoria y cenizas volantes

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir la composición química del cemento Portland y sus principales compuestos [Familiarizarse]
2. Explicar el proceso de hidratación del cemento y su relación con el desarrollo de resistencia [Usar]

Elective:

3. Evaluar el efecto de materiales cementantes suplementarios sobre las propiedades del concreto [Evaluar]

3.14. Química Computacional y Teórica (CTC)

La Química Computacional y Teórica proporciona fundamentos teóricos y herramientas computacionales para predecir y simular el comportamiento químico. Este subconjunto orientado a la computación cubre el modelado molecular, el aprendizaje automático para la predicción de propiedades químicas y el diseño computacional de fármacos, directamente relevantes para bioinformática, IA y ciencia de datos.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.14.1 Modelado Molecular y Simulación (Pág. 192)	1	1
⭐ 3.14.2 Aprendizaje Automático en el Descubrimiento Químico (Pág. 193)	1	1
⭐ 3.14.3 Diseño Computacional de Fármacos (Pág. 193)	1	1

Cuadro 3.14: Lista de KUs del área de Química Computacional y Teórica.

3.14.1. CTC/Modelado Molecular y Simulación (CS Core: 1 hr, KA Core: 1 hr)

Principios de la mecánica y la dinámica molecular para estudiar la estructura y el movimiento de sistemas moleculares, con conexiones a la simulación y los métodos de muestreo basados en física en

computación.
Temas:
Core

- Funciones de energía potencial de campo de fuerza: términos de enlace, ángulo, torsión e interacciones no enlazantes
- Dinámica molecular: algoritmos de integración numérica, condiciones de frontera periódicas y termostatos
- Métodos de Monte Carlo para el muestreo conformacional: criterio de Metropolis y muestreo por importancia
- Modelos de solvatación implícitos y explícitos para simulaciones moleculares en fase condensada
- Perturbación de energía libre e integración termodinámica para la estimación de afinidades de unión relativas

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir los componentes de un campo de fuerza de mecánica molecular estándar y la interpretación física de cada término de energía [Familiarizarse]
2. Configurar y ejecutar una simulación de dinámica molecular, monitoreando la conservación de energía, la equilibración de temperatura y los observables estructurales [Usar]
3. Evaluar la estabilidad y convergencia de una trayectoria de dinámica molecular usando RMSD, perfiles de energía y observables termodinámicos [Evaluar]

3.14.2. CTC/Aprendizaje Automático en el Descubrimiento Químico (CS Core: 1 hr, KA Core: 1 hr)

Aplicación del aprendizaje estadístico y el aprendizaje profundo para predecir propiedades moleculares, acelerar el descubrimiento químico y generar nuevas estructuras moleculares relevantes para IA y ciencia de datos.
Temas:
Core

- Representaciones moleculares: notación SMILES, huellas digitales moleculares y codificaciones de grafos moleculares
- Modelos de relación cuantitativa estructura-actividad (QSAR) usando regresión supervisada y clasificación
- Redes Neuronales de Grafos (GNN) para aprender propiedades moleculares directamente de estructuras de grafos químicos
- Modelos generativos (VAE, difusión, basados en transformadores) para el diseño molecular de novo
- Validación de modelos: validación cruzada, dominio de aplicabilidad y diversidad del espacio químico en conjuntos de entrenamiento

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Definir y comparar diferentes representaciones moleculares y explicar su idoneidad para tareas específicas de aprendizaje automático [Familiarizarse]
2. Entrenar un modelo de regresión QSAR para predecir una propiedad molecular a partir de descriptores y evaluar su desempeño con métricas estándar de regresión [Usar]
3. Evaluar la validez, diversidad y novedad de las moléculas generadas por un modelo de aprendizaje profundo usando métricas de referencia estándar [Evaluar]

3.14.3. CTC/Diseño Computacional de Fármacos (CS Core: 1 hr, KA Core: 1 hr)

Enfoques computacionales basados en estructura y en ligando para identificar y optimizar moléculas candidatas a fármacos, relevantes para las cadenas de descubrimiento de fármacos impulsadas por IA.
Temas:
Core

- Acoplamiento molecular: algoritmos de búsqueda conformacional y funciones de puntuación para la predicción de poses de unión
- Cribado virtual de bibliotecas de compuestos contra dianas proteicas para priorizar candidatos a fármacos
- Predicción de propiedades ADMET: absorción, distribución, metabolismo, excreción y toxicidad

- Modelado de farmacóforos y enfoques de cribado virtual basados en ligando
- Identificación y caracterización de sitios de unión en proteínas para el diseño de fármacos basado en estructura

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Identificar las interacciones intermoleculares clave entre un ligando y el sitio activo de una proteína a partir de un complejo acoplado computacionalmente [Familiarizarse]
2. Realizar un estudio de acoplamiento molecular, clasificar las poses de unión predichas según la función de puntuación y seleccionar candidatos para análisis posterior [Usar]
3. Diseñar y ejecutar un flujo de trabajo de cribado virtual para identificar posibles inhibidores de enzimas a partir de una base de datos de compuestos [Evaluar]

3.15. Bioquímica y Biología Molecular (BMB)

La Bioquímica y Biología Molecular explora los procesos químicos dentro de los organismos vivos a nivel molecular. Este subconjunto orientado a la computación enfatiza la estructura biomolecular, la cinética enzimática, la genética molecular y los métodos de bioinformática directamente relevantes para la predicción de estructura de proteínas, AlphaFold, el análisis de secuencias, la genómica y las aplicaciones de IA en biología.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.15.1 Estructura y Función de las Biomoléculas (Pág. 194)	1	1
⭐ 3.15.2 Genética Molecular y Enzimología (Pág. 195)	1	1
⭐ 3.15.3 Bioinformática y Biología Computacional (Pág. 195)	1	1

Cuadro 3.15: Lista de KUs del área de Bioquímica y Biología Molecular.

3.15.1. BMB/Estructura y Función de las Biomoléculas (CS Core: 1 hr, KA Core: 1 hr)

Arquitectura química y roles biológicos de aminoácidos, proteínas y ácidos nucleicos, con énfasis en el plegamiento de proteínas y la predicción de estructura tridimensional relevante para AlphaFold y la bioinformática estructural.
Temas:
Core

- Aminoácidos: propiedades químicas, quiralidad, clases de cadenas laterales y formación del enlace peptídico
- Jerarquía de la estructura proteica: niveles primario, secundario, terciario y cuaternario y las fuerzas estabilizadoras
- Problema del plegamiento de proteínas, perspectiva del paisaje energético, mal plegamiento y relevancia para AlphaFold
- Estructura de ADN y ARN: reglas de apareamiento de bases, geometría de la doble hélice y motivos de estructura secundaria
- Banco de Datos de Proteínas (PDB) y bases de datos estructurales como fuentes de datos para cadenas de bioinformática

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir las propiedades químicas de los aminoácidos y nucleótidos como bloques de construcción de proteínas y ácidos nucleicos [Familiarizarse]

2. Analizar estructuras de proteínas a partir de archivos PDB, identificando elementos de estructura secundaria, sitios activos y residuos funcionales clave [Usar]
3. Evaluar métodos de predicción de estructura de proteínas incluyendo enfoques basados en energía y herramientas de aprendizaje profundo como AlphaFold [Evaluar]

3.15.2. BMB/Genética Molecular y Enzimología (CS Core: 1 hr, KA Core: 1 hr)

Mecanismos del flujo de información genética, cinética enzimática y regulación genética que proporcionan las bases moleculares para la genómica, la transcriptómica y la biología de sistemas en computación.

Temas:

Core

- El dogma central de la biología molecular: mecanismos de replicación, transcripción y traducción
- Cinética enzimática de Michaelis-Menten: Km, Vmax, número de recambio y modos de inhibición
- Regulación genética: factores de transcripción, operones, mecanismos epigenéticos y motivos de redes regulatorias
- Técnicas de ADN recombinante: PCR, CRISPR-Cas9 y sistemas de expresión génica usados en biotecnología
- Tecnologías ómicas de alto rendimiento: generación de datos de genómica, transcriptómica, proteómica y metabolómica

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Describir el dogma central de la biología molecular e identificar excepciones conocidas como la transcripción inversa y la edición de ARN [Familiarizarse]
2. Analizar datos cinéticos enzimáticos usando Michaelis-Menten y gráficas de Lineweaver-Burk para determinar Km y Vmax [Usar]
3. Interpretar conjuntos de datos ómicos para identificar genes o vías biológicas diferencialmente expresados o enriquecidos en una condición dada [Evaluar]

3.15.3. BMB/Bioinformática y Biología Computacional (CS Core: 1 hr, KA Core: 1 hr)

Métodos computacionales para el análisis de secuencias biológicas, estructuras y datos a escala genómica, directamente relevantes para el alineamiento de secuencias, el modelado por homología, la bioinformática estructural y las aplicaciones de IA en biología.

Temas:

Core

- Algoritmos de alineamiento de secuencias: Needleman-Wunsch (global), Smith-Waterman (local) y BLAST para búsqueda en bases de datos
- Construcción de árboles filogenéticos, medidas de distancia evolutiva y alineamiento múltiple de secuencias
- Modelado comparativo por homología y predicción de estructura de proteínas basada en plantillas
- Bioinformática estructural: acoplamiento molecular, métricas de comparación de estructuras (RMSD, puntuación TM) y predicción de sitios de unión
- Análisis de datos a escala genómica: llamado de variantes, expresión diferencial de RNA-seq y enriquecimiento de vías
- Aprendizaje automático en bioinformática: predicción de función proteica, interacción fármaco-diana y análisis de datos de célula única

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar las diferencias algorítmicas entre el alineamiento de secuencias global y local e identificar los casos de uso apropiados para cada uno [Familiarizarse]

2. Realizar búsquedas en bases de datos de secuencias usando BLAST e interpretar los resultados en términos de homología, valores e y conservación evolutiva [Usar]
3. Evaluar la precisión de las predicciones computacionales de estructura de proteínas usando métricas estándar como GDT-TS y puntuación TM [Evaluar]

3.16. Teoría de la Probabilidad (PRO)

Esta área de conocimiento cubre los fundamentos matemáticos de la probabilidad, incluyendo espacios muestrales, variables aleatorias, distribuciones de probabilidad, distribuciones conjuntas, momentos, transformaciones y los teoremas límite fundamentales. Proporciona la base teórica para toda la inferencia estadística y el modelado.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.16.1 Espacios Muestrales y Eventos (Pág. 196)	1	1
⭐ 3.16.2 Axiomas y Propiedades de la Probabilidad (Pág. 196)	1	1
⭐ 3.16.3 Probabilidad Condicional e Independencia (Pág. 197)	1	1
⭐ 3.16.4 Variables Aleatorias Discretas y Distribuciones (Pág. 197)	1	1
⭐ 3.16.5 Variables Aleatorias Continuas y Distribuciones (Pág. 197)	1	1
⭐ 3.16.6 Distribuciones Conjuntas, Marginales y Condicionales (Pág. 198)	1	1
⭐ 3.16.7 Teoremas Límite: LGN y TCL (Pág. 198)	1	1

Cuadro 3.16: Lista de KUs del área de Teoría de la Probabilidad.

3.16.1. PRO/Espacios Muestrales y Eventos (CS Core: 1 hr, KA Core: 1 hr)

Definición formal de experimentos aleatorios, espacios muestrales y eventos. Cubre las operaciones de teoría de conjuntos sobre eventos y la construcción de sigma-álgebras como fundamento de la teoría de la probabilidad.

Temas:

Core

- Experimentos aleatorios, resultados y espacios muestrales: finitos, contables e incontables
- Operaciones de teoría de conjuntos sobre eventos: unión, intersección, complemento y leyes de De Morgan
- Sigma-álgebras y espacios medibles como fundamento de la probabilidad rigurosa
- Conjuntos de Borel y la sigma-álgebra de Borel sobre la recta real

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Definir espacios muestrales, eventos y sigma-álgebras para un experimento aleatorio dado [Familiarizarse]
2. Aplicar operaciones de teoría de conjuntos para calcular probabilidades de eventos compuestos [Usar]
3. Construir sigma-álgebras apropiadas para espacios medibles que surgen en problemas estadísticos [Evaluar]

3.16.2. PRO/Axiomas y Propiedades de la Probabilidad (CS Core: 1 hr, KA Core: 1 hr)

Axiomas de probabilidad de Kolmogorov y propiedades derivadas. Cubre las interpretaciones clásica, frecuentista y subjetiva de la probabilidad y sus implicaciones.

Temas:

Core

- Los tres axiomas de probabilidad de Kolmogorov y sus consecuencias
- Propiedades derivadas: regla del complemento, regla de la adición, monotonicidad y continuidad de la probabilidad
- Interpretaciones clásica, frecuentista y subjetiva (bayesiana) de la probabilidad

- Métodos de conteo: permutaciones, combinaciones y el principio de multiplicación

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Enunciar los axiomas de Kolmogorov y derivar propiedades fundamentales de la probabilidad [Familiarizarse]
2. Calcular probabilidades de eventos utilizando la regla de la adición y la regla del complemento [Usar]
3. Aplicar métodos de conteo para calcular probabilidades en espacios muestrales finitos [Usar]

3.16.3. PRO/Probabilidad Condicional e Independencia (CS Core: 1 hr, KA Core: 1 hr)

Probabilidad condicional, regla de la multiplicación, ley de la probabilidad total, teorema de Bayes y el concepto de independencia entre eventos y colecciones de eventos.

Temas:

Core

- Definición e interpretación de la probabilidad condicional P(A|B) y sus propiedades
- Teorema de Bayes y ley de la probabilidad total: derivación y aplicaciones
- Independencia de eventos: independencia por pares e independencia mutua
- Independencia condicional y su papel en los modelos gráficos probabilísticos

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Explicar la probabilidad condicional y la independencia y su relación [Familiarizarse]
2. Aplicar el teorema de Bayes y la ley de la probabilidad total para resolver problemas de probabilidad [Usar]
3. Analizar si los eventos son independientes o condicionalmente independientes en un modelo de probabilidad dado [Evaluar]

3.16.4. PRO/Variables Aleatorias Discretas y Distribuciones (CS Core: 1 hr, KA Core: 1 hr)

Variables aleatorias como funciones medibles, funciones de masa de probabilidad, funciones de distribución acumulada y las distribuciones de probabilidad discretas más importantes.

Temas:

Core

- Funciones de masa de probabilidad (FMP) y funciones de distribución acumulada (FDA) para variables aleatorias discretas
- Distribuciones discretas clave: Bernoulli, Binomial, Geométrica, Binomial Negativa, Poisson, Hipergeométrica
- Propiedades de las distribuciones discretas: media, varianza, asimetría y curtosis
- Aproximación de Poisson a la Binomial y argumentos de distribución límite

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Identificar y describir las principales distribuciones de probabilidad discretas y sus parámetros [Familiarizarse]
2. Calcular probabilidades, medias y varianzas utilizando las FMP de distribuciones discretas estándar [Usar]
3. Seleccionar una distribución discreta apropiada para modelar un fenómeno aleatorio dado y justificar la elección [Evaluar]

3.16.5. PRO/Variables Aleatorias Continuas y Distribuciones (CS Core: 1 hr, KA Core: 1 hr)

Funciones de densidad de probabilidad, FDA y las distribuciones de probabilidad continuas más importantes utilizadas en la teoría y práctica estadística.

Temas:
Core

- Funciones de densidad de probabilidad (FDP) y FDA para variables aleatorias continuas
- Distribuciones continuas clave: Uniforme, Normal, Exponencial, Gamma, Beta, Chi-cuadrado, t, F
- La distribución Normal: propiedades, normal estándar y su papel central en la inferencia estadística
- La familia exponencial de distribuciones: forma canónica y estadísticos suficientes

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir las propiedades y parámetros de las principales distribuciones de probabilidad continuas [Familiarizarse]
2. Calcular probabilidades y cuantiles para distribuciones continuas estándar [Usar]
3. Identificar distribuciones pertenecientes a la familia exponencial y derivar sus estadísticos suficientes [Evaluar]

3.16.6. PRO/Distribuciones Conjuntas, Marginales y Condicionales (CS Core: 1 hr, KA Core: 1 hr)

Distribuciones conjuntas de múltiples variables aleatorias, distribuciones marginales y condicionales, covarianza, correlación y la distribución normal bivariada.
Temas:
Core

- FMP y FDP conjuntas para dos o más variables aleatorias
- Distribuciones marginales y condicionales derivadas de distribuciones conjuntas
- Covarianza, correlación y sus propiedades; la independencia implica covarianza cero (el recíproco no siempre es cierto)
- La distribución normal bivariada y sus distribuciones condicionales

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Derivar distribuciones marginales y condicionales a partir de una distribución conjunta dada [Usar]
2. Calcular la covarianza y correlación entre dos variables aleatorias [Usar]
3. Analizar la estructura condicional de la distribución normal bivariada [Evaluar]

3.16.7. PRO/Teoremas Límite: LGN y TCL (CS Core: 1 hr, KA Core: 1 hr)

Las leyes de los grandes números y el teorema central del límite como justificación matemática de los procedimientos de estimación e inferencia estadística basados en muestras grandes.
Temas:
Core

- Ley débil de los grandes números: convergencia en probabilidad y sus implicaciones para la estimación
- Teorema central del límite: enunciado, condiciones y papel en la justificación de aproximaciones normales
- Aplicaciones del TCL: aproximación normal a la binomial y distribuciones muestrales
- Ley fuerte de los grandes números: convergencia casi segura
- El método delta: distribución asintótica de funciones suaves de estimadores

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Enunciar la LGN y el TCL y explicar las condiciones bajo las cuales se cumplen [Familiarizarse]
2. Aplicar el TCL para derivar aproximaciones normales y distribuciones muestrales [Usar]
3. Aplicar el método delta para derivar distribuciones asintóticas de estimadores no lineales [Evaluar]

3.17. Estadística Descriptiva y Análisis de Datos (DSC)

Esta área de conocimiento cubre los métodos para resumir, visualizar y explorar datos. Incluye resúmenes numéricos y gráficos, análisis exploratorio de datos, evaluación de la calidad de los datos y comunicación de hallazgos estadísticos.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.17.1 Resúmenes Numéricos: Posición, Dispersión y Forma (Pág. 199)	1	1

Cuadro 3.17: Lista de KUs del área de Estadística Descriptiva y Análisis de Datos.

3.17.1. DSC/Resúmenes Numéricos: Posición, Dispersión y Forma (CS Core: 1 hr, KA Core: 1 hr)

Medidas de tendencia central, dispersión y forma distribucional. Incluye alternativas robustas e interpretación de los resúmenes numéricos en contexto.

Temas:

Core

- Medidas de posición: media, mediana, moda, media recortada y media ponderada
- Medidas de dispersión: varianza, desviación estándar, RIQ, MAD y rango
- Medidas de forma: asimetría y curtosis; cuantiles y percentiles
- Estadísticas de resumen robustas: resistencia a valores atípicos y distribuciones de colas pesadas

Aprendizaje esperado (*Learning Outcomes*):

Core:

1. Calcular e interpretar medidas de posición, dispersión y forma para un conjunto de datos univariado [Usar]
2. Seleccionar estadísticas de resumen apropiadas según las propiedades distribucionales de los datos [Usar]
3. Comparar estadísticas de resumen clásicas y robustas y justificar la elección para un conjunto de datos dado [Evaluar]

3.18. Inferencia Estadística (INF)

Esta área de conocimiento cubre los fundamentos y métodos de la inferencia estadística: estimación puntual e intervalar, pruebas de hipótesis, teoría de la verosimilitud, inferencia bayesiana, métodos no paramétricos, pruebas múltiples y teoría de gran muestra. Es el núcleo de la estadística clásica y moderna.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.18.1 Estimación Puntual (Pág. 199)	1	1
⭐ 3.18.2 Estimación por Intervalos e Intervalos de Confianza (Pág. 200)	1	1
⭐ 3.18.3 Fundamentos de las Pruebas de Hipótesis (Pág. 200)	1	1
⭐ 3.18.4 Inferencia Bayesiana (Pág. 200)	1	1

Cuadro 3.18: Lista de KUs del área de Inferencia Estadística.

3.18.1. INF/Estimación Puntual (CS Core: 1 hr, KA Core: 1 hr)

Métodos para estimar parámetros desconocidos a partir de datos. Cubre propiedades deseables de los estimadores, método de momentos, estimación de máxima verosimilitud y la cota inferior de Cramér-Rao.

Temas:

Core

- Propiedades deseables: insesgamiento, consistencia, eficiencia y suficiencia

- Estimación por método de momentos: derivación y propiedades asintóticas
- Estimación de máxima verosimilitud: derivación, propiedades y principio de invarianza
- Cota inferior de Cramér-Rao y el concepto de información de Fisher

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Describir las propiedades clave de los estimadores y explicar las concesiones entre ellas [Familiarizarse]
2. Derivar estimadores por método de momentos y de máxima verosimilitud para distribuciones estándar [Usar]
3. Evaluar la eficiencia de un estimador usando la cota inferior de Cramér-Rao [Evaluar]

3.18.2. INF/Estimación por Intervalos e Intervalos de Confianza (CS Core: 1 hr, KA Core: 1 hr)

Construcción e interpretación de intervalos de confianza. Cubre el método de la cantidad pivote, intervalos para muestras grandes y la relación entre intervalos de confianza y pruebas de hipótesis.
Temas:
Core

- Construcción de intervalos de confianza mediante cantidades pivote y distribuciones exactas
- Interpretación correcta de los intervalos de confianza: probabilidad de cobertura y errores conceptuales comunes
- Intervalos de confianza para muestras grandes basados en normalidad asintótica
- Dualidad entre intervalos de confianza y pruebas de hipótesis

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Construir intervalos de confianza para medias, proporciones y varianzas usando métodos exactos y de gran muestra [Usar]
2. Interpretar correctamente los intervalos de confianza e identificar interpretaciones erróneas comunes [Usar]
3. Explicar la dualidad entre intervalos de confianza y pruebas de hipótesis y utilizarla para extraer inferencias [Evaluar]

3.18.3. INF/Fundamentos de las Pruebas de Hipótesis (CS Core: 1 hr, KA Core: 1 hr)

El marco de Neyman-Pearson para las pruebas de hipótesis: hipótesis nula y alternativa, errores tipo I y II, potencia, valores p y las pruebas más poderosas.
Temas:
Core

- Marco de Neyman-Pearson: hipótesis nula y alternativa, regiones de rechazo, errores tipo I y II
- Valores p: definición, interpretación correcta y usos incorrectos comunes; potencia y tamaño de muestra
- Pruebas paramétricas comunes: prueba z, prueba t, prueba F y prueba ji-cuadrada
- Lema de Neyman-Pearson y pruebas uniformemente más poderosas

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar la lógica de las pruebas de hipótesis, los errores tipo I y II y el significado de un valor p [Familiarizarse]
2. Aplicar pruebas paramétricas estándar e interpretar los resultados en contexto [Usar]
3. Derivar la prueba más poderosa para una hipótesis simple usando el lema de Neyman-Pearson [Evaluar]

3.18.4. INF/Inferencia Bayesiana (CS Core: 1 hr, KA Core: 1 hr)

El enfoque bayesiano de la inferencia estadística: distribuciones a priori, verosimilitud, distribuciones a posteriori, intervalos de credibilidad y teoría de decisión bayesiana.

Temas:
Core

- Marco bayesiano: a priori, verosimilitud, a posteriori y el papel del teorema de Bayes
- Distribuciones a priori: informativas, no informativas y conjugadas
- Inferencia a posteriori: estimaciones puntuales, intervalos de credibilidad y distribuciones predictivas a posteriori
- Comparación de modelos bayesianos: factores de Bayes y probabilidades a posteriori de modelos

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Explicar el enfoque bayesiano de la inferencia y contrastarlo con el enfoque frecuentista [Familiarizarse]
2. Derivar la distribución a posteriori para modelos conjugados estándar [Usar]
3. Aplicar factores de Bayes para comparar modelos estadísticos e interpretar la evidencia [Evaluar]

3.19. Regresión y Modelos Estadísticos (REG)

Esta área de conocimiento cubre la construcción, estimación, interpretación y validación de modelos estadísticos para relaciones entre variables. Abarca la regresión lineal simple y múltiple, modelos lineales generalizados, regresión no paramétrica, modelos mixtos e inferencia causal.

Área de Conocimiento (KA)	CS Core	KA Core
⭐ 3.19.1 Regresión Lineal Simple (Pág. 201)	1	1

Cuadro 3.19: Lista de KUs del área de Regresión y Modelos Estadísticos.

3.19.1. REG/Regresión Lineal Simple (CS Core: 1 hr, KA Core: 1 hr)

El modelo de regresión lineal simple: estimación por mínimos cuadrados, inferencia sobre coeficientes, bondad de ajuste y los supuestos del modelo lineal clásico.
Temas:
Core

- Estimación por mínimos cuadrados ordinarios (MCO): derivación de coeficientes y sus propiedades
- Inferencia sobre coeficientes de regresión: pruebas t, intervalos de confianza y valores p
- Coeficiente de determinación R-cuadrado y descomposición de la varianza mediante análisis de varianza
- Teorema de Gauss-Markov: propiedad MELI del MCO bajo supuestos clásicos

Aprendizaje esperado (*Learning Outcomes*):
Core:

1. Ajustar un modelo de regresión lineal simple e interpretar los coeficientes estimados [Usar]
2. Realizar pruebas de hipótesis y construir intervalos de confianza para parámetros de regresión [Usar]
3. Verificar los supuestos de Gauss-Markov y evaluar las consecuencias cuando se violan [Evaluar]

Capítulo 4

Plan de estudios

4.1. Codificación de los cursos

Los cursos son clasificados en 4 niveles:

- **Introductorios** Código 1.. ejemplo: CS100
- **Intermedios** Código 2.. ejemplo: MA200
- **Avanzados** Código 3.. y ejemplo: IS370
- **Trabajo de final de carrera** Código 4.. ejemplo: CS403

Los cursos se encuentran codificados bajo el esquema que se muestra en la Figura 4.1.

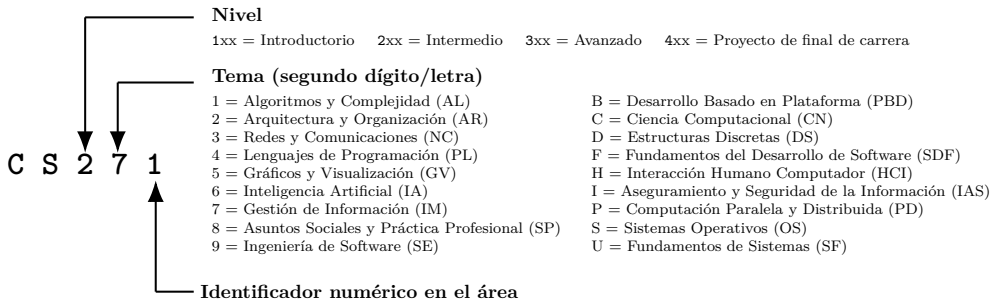

Figura 4.1: Esquema de codificación para los cursos.

El tipo de curso esta determinado por las letras iniciales. Los posibles códigos para estos tipos son:

1. **CS** Área de Ciencia de la Computación (*Computer Science* – CS);
2. **BIC** Área de Computación (*Computer Science* – CS);
3. **AI** Área de Inteligencia Artificial (*Artificial Intelligence* – AI);
4. **MA** Matemáticas (MA);
5. **BMA** Matemáticas (BMA);
6. **STA** Estadística (ST);
7. **CB** Área de Ciencias Básicas (Física, Biología);
8. **FI** Área de Física;
9. **BFI** Ciencias Físicas
10. **BCH** Ciencias Químicas (BCH);
11. **FG** Área de Formación General;
12. **EX** Actividades extracurriculares
13. **ET** Área de Formación de Empresas de Base Tecnológica;
14. **BEI** Área de Idiomas

Al mismo tiempo, los cursos están clasificados en:

- Estudios generales
- Específicos
- Especialización

4.2. Estructura Curricular

La relación de cursos se muestra a continuación:

HT = Horas de teoría, HP = Horas de práctica, CR = Créditos, T = Tipo de curso.

O = Obligatorio, E = Electivo.

Primer Semestre							
Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
BIC101	Introducción a la Computación (Pág 257)	Estudios generales	2	4	4	O	
BMA101	Algebra Lineal (Pág 262)	Estudios generales	3	2	4	O	
BMA102	Cálculo Diferencial (Pág 264)	Estudios generales	4	2	5	O	
BCH101	Química I (Pág 267)	Estudios generales	4	2	5	O	
BEI101	Inglés I (Pág 272)	Estudios generales		4	2	O	
					20		

Segundo Semestre							
Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
CS100	Introducción a la Ciencia de la Computación (Pág 276)	Específicos	2	2	3	O	
CS112	Programación Orientada a Objetos I (Pág 287)	Específicos	2	4	4	O	BIC101 (1er Sem)
CS1D1	Estructuras Discretas (Pág 292)	Específicos	2	4	4	O	
BMA103	Cálculo Integral (Pág 295)	Estudios generales	4	2	5	O	BMA102 (1er Sem)
BFI101	Física I (Pág 297)	Estudios generales	4	4	5	O	
BEI102	Inglés II (Pág 302)	Estudios generales		4	2	O	BEI101 (1er Sem)
					23		

	Tercer Semestre							
	Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
	CS113	Programación Orientada a Objetos II (Pág 306)	Específicos	2	4	4	O	CS112 (2do Sem)
	CS2B1	Desarrollo Basado en Plataformas (Pág 312)	Específicos	2	4	4	O	CS112 (2do Sem)
	AI161	IA Aplicada (Pág 316)	Específicos	2	4	4	O	BIC101 (1er Sem)
	BMA104	Cálculo Diferencial e Integral Avanzado (Pág 320)	Estudios generales	4	2	5	O	BMA102 (1er Sem)
	STA251	Estadística y Probabilidades (Pág 324)	Específicos	2	2	3	O	BMA102 (1er Sem)
Electivo	BEI201	Inglés III (Pág 329)	Estudios generales		4	2	E	BEI102 (2do Sem)
						20		

	Cuarto Semestre							
	Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
	CS210	Algoritmos y Estructuras de Datos (Pág 333)	Específicos	2	4	4	O	CS113 (3er Sem)
	CS211	Teoría de la Computación (Pág 338)	Específicos	2	4	4	O	CS1D1 (2do Sem)
	CS221	Arquitectura de Computadores (Pág 343)	Específicos	2	4	4	O	CS1D1 (2do Sem)
	CS271	Bases de Datos I (Pág 348)	Específicos	2	4	4	O	CS112 (2do Sem), CS1D1 (2do Sem)
	CS401	Metodología de la Investigación (Pág 353)	Especialización	1	2	2	O	CS100 (2do Sem)
	MA202	Métodos Numéricos (Pág 357)	Específicos	2	2	3	O	BMA103 (2do Sem)
Electivo	BEI202	Inglés IV (Pág 362)	Estudios generales		4	2	E	BEI201 (3er Sem)
						21		

	Quinto Semestre							
	Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
	CS212	Análisis y Diseño de Algoritmos (Pág 366)	Específicos	2	4	4	O	CS210 (4to Sem), CS211 (4to Sem)
	CS261	Inteligencia Artificial (Pág 371)	Específicos	2	4	4	O	STA251 (3er Sem)
	CS272	Bases de Datos II (Pág 378)	Específicos	1	4	3	O	CS271 (4to Sem)
	CS291	Ingeniería de Software I (Pág 382)	Específicos	2	4	4	O	CS113 (3er Sem), CS271 (4to Sem)
	CS2S1	Sistemas Operativos (Pág 388)	Específicos	2	4	4	O	CS221 (4to Sem)
	FG106	Teatro (Pág 393)	Específicos	1	2	2	O	
Electivo	BEI203	Inglés V (Pág 397)	Estudios generales		4	2	E	BEI202 (4to Sem)
						21		

Sexto Semestre							
Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
CS231	Redes y Comunicación (Pág 401)	Específicos	1	4	3	O	CS2S1 (5to Sem)
CS311	Algoritmos para Problemas Complejos (Pág 406)	Específicos	2	2	3	O	CS212 (5to Sem)
CS312	Estructuras de Datos Avanzadas (Pág 409)	Específicos	2	4	4	O	CS212 (5to Sem)
CS342	Compiladores (Pág 412)	Específicos	2	4	4	O	CS211 (4to Sem)
AI263	Introducción al Aprendizaje de Máquina (Pág 415)	Especialización	2	4	4	O	CS261 (5to Sem)
FI201	Física Computacional (Pág 418)	Específicos	2	2	3	O	BFI101 (2do Sem)
					21		

Séptimo Semestre							
Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
CS251	Computación Gráfica (Pág 421)	Especialización	2	4	4	O	MA202 (4to Sem)
CS292	Ingeniería de Software II (Pág 425)	Específicos	2	4	4	O	CS291 (5to Sem)
CS2H1	Experiencia de Usuario (UX) (Pág 430)	Específicos	2	4	4	O	CS291 (5to Sem)
AI264	Aprendizaje Profundo (Pág 435)	Especialización	2	4	4	O	AI263 (6to Sem)
FG211	Ética Profesional (Pág 438)	Específicos	1	2	2	O	
FG350	Liderazgo y Desempeño (Pág 440)	Específicos	2		2	O	FG106 (5to Sem)
					20		

Octavo Semestre

	Código	Curso	Área	HT	HP	CR	T	Requisitos
	CS281	Computación en la Sociedad (Pág 443)	Específicos	2		2	O	
	CS3I1	Seguridad en Computación (Pág 449)	Específicos	1	4	3	O	CS231 (6to Sem)
Electivo	CS3P1	Computación Paralela y Distribuída (Pág 455)	Específicos	2	4	4	O	CS212 (5to Sem), CS231 (6to Sem)
	CS402	Proyecto de tesis 1 (Pág 465)	Especialización	2	4	4	O	CS401 (4to Sem)
	EX301	Actividades Extracurriculares (Pág 469)	Estudios generales	1	2	2	O	
	ET201	Formación de Empresas de Base Tecnológica I (Pág 471)	Específicos	2	2	3	O	FG350 (7mo Sem)
	AI268	Visión Computacional (Pág 475)	Especialización	2	2	3	E	AI264 (7mo Sem)
	CS391	Ingeniería de Software III (Pág 481)	Específicos	2	2	3	E	CS292 (7mo Sem)
	CS393	Sistemas de Información (Pág 486)	Específicos	2	2	3	E	CS291 (5to Sem)
	CS394	Arquitectura de Software y DevOps para Software en la Nube (Pág 489)	Específicos	2	2	3	E	CS292 (7mo Sem)
	CS395	Calidad de Software (Pág 495)	Específicos	2	2	3	E	CS292 (7mo Sem)
						21		

Noveno Semestre

	Código	Curso	Área	HT	HP	CR	T	Requisitos
	CS370	Big Data (Pág 503)	Especialización	1	4	3	O	CS3P1 (8vo Sem)
	CS400	Prácticas Pre-profesionales (Pág 507)	Especialización	1	2	2	O	110Cr
	CS403	Proyecto de tesis 2 (Pág 509)	Especialización	2	4	3	O	CS402 (8vo Sem)
	AI365	Modelos Generativos Avanzados en IA (Pág 513)	Específicos	2	4	4	O	AI264 (7mo Sem)
	CB309	Bioinformática (Pág 515)	Especialización	2	4	4	O	
	ET301	Formación de Empresas de Base Tecnológica II (Pág 519)	Específicos	2	2	3	O	ET201 (8vo Sem)
	AI369	Robótica (Pág 522)	Especialización	2	2	3	E	AI268 (8vo Sem)
	CS351	Tópicos en Computación Gráfica (Pág 526)	Especialización	2	2	3	E	CS251 (7mo Sem)
	CS353	Computación Cuántica (Pág 530)	Especialización	2	2	3	E	CS3I1 (8vo Sem)
	CS392	Tópicos en Ingeniería de Software (Pág 534)	Especialización	2	2	3	E	CS391 (8vo Sem)
	CS3P3	Internet de las Cosas (Pág 541)	Especialización	1	4	3	E	CS3P1 (8vo Sem)
						22		

Décimo Semestre							
Código	**Curso**	**Área**	**HT**	**HP**	**CR**	**T**	**Requisitos**
CS3P2	Cloud Computing (Pág 544)	Específicos	1	4	3	O	CS370 (9no Sem)
CS404	Taller de Investigación (Pág 548)	Especialización	2	4	4	O	CS403 (9no Sem)
AI367	Tópicos en Inteligencia Artificial (Pág 552)	Especialización	2	4	4	O	AI264 (7mo Sem)
AI368	Computación Evolutiva (Pág 559)	Específicos	2	4	4	O	AI264 (7mo Sem)
					11		

Total de créditos de la carrera: 200 .

4.3. Créditos por Actividades Diversas

Las Actividades Extracurriculares tendrán un mínimo de 80 horas en 3 actividades distintas (Art. 27, RR 544, 2017).

Ítem	Créditos	Actividades Extracurriculares (Art. 31, RR 1817, 2019)
01	01	▪ Actividades artísticas: música, teatro, literatura, artes visuales, folklore, etc. ▪ Actividades física y deporte. ▪ Participación en cursos, conferencias, talleres. ▪ Participación en Actividades Científicas: concursos científicos y tecnológicos. ▪ Proyección social (ferias y difusión vocacional en colegios). ▪ Cursos de desarrollo personal: comunicación interpersonal, liderazgo, coaching, emprendedurismo, marketing personal, etc. ▪ Ayudantía de estudiantes en talleres de reforzamiento de asignaturas, para estudiantes que se encuentran en riesgo académico, de acuerdo al inciso d) del Art. 26° del Reglamento de Tutoría para Estudiantes de Pregrado en Riesgo Académico de la UNI, aprobado con R.R. N° 0477 del 21.04.2017. ▪ Ayudantía de estudiantes en laboratorios y gabinetes.

4.4. Condición de Egreso

Tipo	Escuela de Ciencia de la Computación	N6
Obligatorios	Créditos en cursos obligatorios	180
	BIE01 Idioma Extranjero o Lengua Nativa en el Nivel Intermedio*	2
	BAE01 Actividades Extracurriculares	1
	YA100 Ayudantía Académica I	1
	XP200 Prácticas Pre-Profesionales II	2
	Total créditos obligatorios	**186**
Electivos	Mínimo de créditos en cursos electivos (de especialidad)	0
	Mínimo de suma en créditos en cursos electivos (de especialidad) y en cursos electivos complementarios	**22**
Total	**Mínimo de créditos para egresar**	**208**

*Los alumnos que soliciten la exoneración del curso BIE01 en los años académicos 2020, 2021 y 2022, incluyendo el 2023-2, están exceptuados de llevar el curso según la excepcionalidad contenida en la Ley N° 31359.

4.5. Tópicos distribuídos por curso

4.5.1. Tópicos de Ciencia de la Computación–*Computer Science*/curso

		Primer Sem					Segundo Sem						.
	Unidades de Conocimiento de Ciencia de la Computación–*Computer Science* **Cursos →**	BIC101	BMA101	BMA102	BCH101	BEI101	CS100	CS112	CS1D1	BMA103	BFI101	BEI102	Total
★	AI/Fundamentos de Aprendizaje Automático (1 hora Core-Tier1, 2 horas Core-Tier2)						1						1
★	AI/Redes Neuronales y Ética (1 hora Core-Tier1, 1 hora Core-Tier2)						1						1
★	AL/Estructuras de Datos Fundamentales (5 horas Core-Tier1, 3 horas Core-Tier2)						3						3
★	AL/Algoritmos Fundamentales (4 horas Core-Tier1, 2 horas Core-Tier2)						3						3
★	AL/Algoritmos Avanzados (3 horas Core-Tier1, 2 horas Core-Tier2)						2						2
★	AL/Estrategias Algorítmicas (6 horas Core-Tier1)	6											6
★	AL/Marco de Análisis de Complejidad (2 horas Core-Tier1, 1 hora Core-Tier2)						2						2
★	AL/Notación Asintótica y Clases de Complejidad (1 hora Core-Tier1, 1 hora Core-Tier2)						1						1
★	AL/Análisis de Complejidad: Recursión, Amortización y Cotas Ajustadas (2 horas Core-Tier1, 1 hora Core-Tier2)						1						1
★	AR/Representación de Datos a Nivel de Máquina (1 hora Core-Tier1)						4						4
★	DM/Conceptos Fundamentales de Sistemas de Bases de Datos (2 horas Core-Tier1, 1 hora Core-Tier2)						4						4
★	FPL/Fundamentos de Programación Orientada a Objetos (2 horas Core-Tier1, 1 hora Core-Tier2)	24					4	14					42
★	FPL/Programación Orientada a Objetos: Encapsulación, Subtipado y Reflexión (2 horas Core-Tier1, 1 hora Core-Tier2)							8					8
★	FPL/Programación Funcional (4 horas Core-Tier1, 3 horas Core-Tier2)							10					10
★	FPL/Scripting de Shell (2 horas Core-Tier1)							4					4
★	FPL/Fundamentos de Sistemas de Tipos (2 horas Core-Tier1, 2 horas Core-Tier2)	6						8					14
★	FPL/Modelo de Ejecución y Memoria de Sistemas (2 horas Core-Tier1)							12					12
★	FPL/Traducción y Ejecución de Lenguajes (2 horas Core-Tier1, 3 horas Core-Tier2)							4					4
★	FPL/Abstracción y Representación de Programas (3 horas Core-Tier2)						2						2
	FPL/Construcciones de Programación Avanzadas							6					6
	FPL/Pragmática del Lenguaje							2					2
★	GIT/Conceptos Fundamentales de Gráficos y Técnicas Interactivas (4 horas Core-Tier1, 3 horas Core-Tier2)						2						2
★	GIT/Renderizado Aplicado y Técnicas (15 horas Core-Tier2)						2						2

Unidades de Conocimiento de Ciencia de la Computación–*Computer Science*	Primer Sem							Segundo Sem							Total
Cursos →	BIC101	BMA101	BMA102	BCH101	BEI101	CS100	CS112	CS1D1	BMA103	BFI101	BEI102				
NC/Fundamentos de Redes y Comunicaciones (3 horas Core-Tier1)						4									4
NC/Aplicaciones en Red (4 horas Core-Tier1)						4									4
OS/Rol y Propósito de los Sistemas Operativos (2 horas Core-Tier1)						2									2
OS/Principios del Sistema Operativo (2 horas Core-Tier1)						2									2
OS/Gestión de Memoria (1 hora Core-Tier2)						4									4
SE/Control de Versiones y CI/CD (1 hora Core-Tier1, 2 horas Core-Tier2)						2									2
SE/Ingeniería de Requisitos (2 horas Core-Tier2)						2									2
SEC/Fundamentos de Criptografía (1 hora Core-Tier1, 2 horas Core-Tier2)						2									2
Total	36	0	0	0	0	54	68	48	0	0	0				

Cuadro 4.2: Tópicos de Ciencia de la Computación–*Computer Science* por curso del 1er al 2do Semestre (Parte 2 de 2)

		Tercer Sem						Cuarto Sem							.
	Unidades de Conocimiento de Ciencia de la Computación– *Computer Science* Cursos →	CS113	CS2B1	AI161	BMA104	STA251	BEI201	CS210	CS211	CS221	CS271	CS401	MA202	BEI202	Total
★	AL/Estructuras de Datos Fundamentales (5 horas Core-Tier1, 3 horas Core-Tier2)							32							32
★	AL/Marco de Análisis de Complejidad (2 horas Core-Tier1, 1 hora Core-Tier2)								3						3
★	AL/Notación Asintótica y Clases de Complejidad (1 hora Core-Tier1, 1 hora Core-Tier2)								4						4
★	AL/Análisis de Complejidad: Recursión, Amortización y Cotas Ajustadas (2 horas Core-Tier1, 1 hora Core-Tier2)								7						7
★	AL/Teoría de la Complejidad Computacional (2 horas Core-Tier1, 1 hora Core-Tier2)								6						6
★	AL/Lenguajes Formales y Autómatas (4 horas Core-Tier1, 10 horas Core-Tier2)								62						62
★	AR/Organización de Máquina a Nivel de Ensamblador (1 hora Core-Tier1, 2 horas Core-Tier2)									20					20
★	AR/Jerarquía de Memoria (6 horas Core-Tier1)									18					18
★	AR/Interconexión y Comunicación (1 hora Core-Tier1)									8					8
★	AR/Organización Funcional (2 horas Core-Tier2)									8					8
★	AR/Arquitecturas Heterogéneas (2 horas Core-Tier2)									10					10
★	AR/Arquitecturas de Procesador Seguras (2 horas Core-Tier2)									8					8
★	DM/Conceptos Fundamentales de Sistemas de Bases de Datos (2 horas Core-Tier1, 1 hora Core-Tier2)										8				8
★	DM/Modelado de Datos (2 horas Core-Tier1, 3 horas Core-Tier2)										8				8
★	DM/Bases de Datos Relacionales (1 hora Core-Tier1, 3 horas Core-Tier2)										16				16
★	DM/Construcción de Consultas (2 horas Core-Tier1, 4 horas Core-Tier2)										16				16
★	DM/Internos de DBMS (4 horas Core-Tier2)										8				8
★	DM/Seguridad y Privacidad de Datos (1 hora Core-Tier1, 2 horas Core-Tier2)										8				8
★	FPL/Programación Orientada a Objetos: Encapsulación, Subtipado y Reflexión (2 horas Core-Tier1, 1 hora Core-Tier2)	8													8
★	FPL/Computación Paralela y Distribuida (2 horas Core-Tier1, 2 horas Core-Tier2)	12													12
★	FPL/Sistemas de Tipos: Polimorfismo y Tipado Complementario (1 hora Core-Tier1, 1 hora Core-Tier2)	12													12
★	FPL/Modelo de Ejecución y Memoria de Sistemas (2 horas Core-Tier1)	8													8
★	FPL/Gestión de Memoria (1 hora Core-Tier1)	8													8

Unidades de Conocimiento de Ciencia de la Computación–*Computer Science* Cursos →	Tercer Sem CS113	CS2B1	AI161	BMA104	STA251	BEI201	Cuarto Sem CS210	CS211	CS221	CS271	CS401	MA202	BEI202	Total
★ SPD/Aspectos Comunes: Lenguajes de Plataforma y Patrones Web (2 horas Core-Tier1, 1 hora Core-Tier2)		3												3
★ SPD/Frontend, Frameworks y Estándares Web (3 horas Core-Tier2)		12												12
★ SPD/Plataformas Móviles: Fundamentos del Desarrollo (2 horas Core-Tier2)		6												6
★ SPD/Plataformas Móviles: Arquitectura y Restricciones (2 horas Core-Tier2)		3												3
★ SPD/Plataformas Móviles: Capacidades y Computación Asíncrona (1 hora Core-Tier2)		3												3
★ SPD/Plataformas Interactivas de Datos y Análisis (2 horas Core-Tier2)		6												6
SPD/SPD-SEP/Interactiva		10	0	0	0	0	32	82	72	64	0	0	0	10
Total	72	48												

Cuadro 4.4: Tópicos de Ciencia de la Computación–*Computer Science* por curso del 3er al 4to Semestre (Parte 2 de 2)

		Quinto Sem							Sexto Sem						.
	Unidades de Conocimiento de Ciencia de la Computación– *Computer Science* Cursos →	CS212	CS261	CS272	CS291	CS2S1	FG106	BEI203	CS231	CS311	CS312	CS342	AI263	FI201	Total
⭐	AI/Introducción a la IA: Conceptos y Comportamiento Inteligente (2 horas Core-Tier1, 1 hora Core-Tier2)		4												4
⭐	AI/Características del Problema y Naturaleza de los Agentes (1 hora Core-Tier1, 1 hora Core-Tier2)		4												4
⭐	AI/Estado y Búsqueda No Informada (1 hora Core-Tier1, 2 horas Core-Tier2)		5												5
⭐	AI/Búsqueda Heurística y CSP (1 hora Core-Tier1, 2 horas Core-Tier2)		6												6
⭐	AI/Búsqueda Adversarial y Optimización Estocástica (1 hora Core-Tier1, 3 horas Core-Tier2)		5												5
⭐	AI/Representación del Conocimiento y Razonamiento Probabilístico Básico (1 hora Core-Tier1, 1 hora Core-Tier2)		6												6
⭐	AI/Independencia, Modelos de Markov y Toma de Decisiones (1 hora Core-Tier1, 1 hora Core-Tier2)		6												6
⭐	AI/Fundamentos de Aprendizaje Automático (1 hora Core-Tier1, 2 horas Core-Tier2)		7												7
⭐	AI/Aprendizaje Automático: Datos, Representaciones y Evaluación (1 hora Core-Tier1, 2 horas Core-Tier2)		7												7
⭐	AI/Redes Neuronales y Ética (1 hora Core-Tier1, 1 hora Core-Tier2)		6												6
⭐	AI/Formulación del Aprendizaje Automático y Evaluación (1 hora Core-Tier1, 2 horas Core-Tier2)		3												3
⭐	AI/Aprendizaje Profundo, RL y Ética (1 hora Core-Tier1, 2 horas Core-Tier2)		3												3
⭐	AI/Aplicaciones e Impacto Social de la IA (2 horas Core-Tier1, 2 horas Core-Tier2)		4												4
⭐	AI/Aplicaciones Adicionales e Impacto Social de la IA (2 horas Core-Tier1, 2 horas Core-Tier2)		4												4
⭐	AL/Estructuras de Datos Fundamentales (5 horas Core-Tier1, 3 horas Core-Tier2)	6													6
⭐	AL/Algoritmos Fundamentales (4 horas Core-Tier1, 2 horas Core-Tier2)	6								36					42
⭐	AL/Algoritmos Avanzados (3 horas Core-Tier1, 2 horas Core-Tier2)	4													4
⭐	AL/Estrategias Algorítmicas (6 horas Core-Tier1)	24								12					36
⭐	AL/Marco de Análisis de Complejidad (2 horas Core-Tier1, 1 hora Core-Tier2)	6													6
⭐	AL/Notación Asintótica y Clases de Complejidad (1 hora Core-Tier1, 1 hora Core-Tier2)	6													6
⭐	AL/Análisis de Complejidad: Recursión, Amortización y Cotas Ajus-	12													12

		Quinto Sem							Sexto Sem						.
	Unidades de Conocimiento de Ciencia de la Computación–*Computer Science* **Cursos →**	CS212	CS261	CS272	CS291	CS2S1	FG106	BEI203	CS231	CS311	CS312	CS342	AI263	FI201	Total
	DM/Bases de Datos Semiestructuradas y No Estructuradas			12											12
★	FPL/Programación Orientada a Objetos: Encapsulación, Subtipado y Reflexión (2 horas Core-Tier1, 1 hora Core-Tier2)										12				12
★	FPL/Scripting de Shell (2 horas Core-Tier1)					8									8
★	FPL/Traducción y Ejecución de Lenguajes (2 horas Core-Tier1, 3 horas Core-Tier2)											24			24
	FPL/Análisis de Programas y Analizadores											20			20
	FPL/Comportamiento en Tiempo de Ejecución y Sistemas											20			20
★	NC/Fundamentos de Redes y Comunicaciones (3 horas Core-Tier1)								6						6
★	NC/Aplicaciones en Red (4 horas Core-Tier1)								6						6
★	NC/Soporte de Fiabilidad (6 horas Core-Tier2)								6						6
★	NC/Enrutamiento y Reenvío (4 horas Core-Tier2)								6						6
★	NC/Comunicación de un solo salto (3 horas Core-Tier2)								6						6
★	NC/Seguridad de Red (3 horas Core-Tier2)								6						6
★	NC/Soporte de Movilidad (4 horas Core-Tier2)								6						6
★	NC/Temas Emergentes (4 horas Core-Tier2)								6						6
★	OS/Rol y Propósito de los Sistemas Operativos (2 horas Core-Tier1)					8									8
★	OS/Protección y Seguridad (2 horas Core-Tier1, 1 hora Core-Tier2)					12									12
★	OS/Planificación (2 horas Core-Tier2)					16									16
★	OS/Gestión de Memoria (1 hora Core-Tier2)					16									16
★	OS/API de Sistemas de Archivos e Implementación (2 horas Core-Tier2)					12									12
★	SE/Colaboración y Dinámica de Equipo (2 horas Core-Tier1, 2 horas Core-Tier2)				14										14
★	SE/Control de Versiones y CI/CD (1 hora Core-Tier1, 2 horas Core-Tier2)				12										12
★	SE/Ingeniería de Requisitos (2 horas Core-Tier2)				14										14
★	SE/Principios y Arquitectura de Software (1 hora Core-Tier1, 2 horas Core-Tier2)				12										12
★	SE/Conceptos de verificación y validación (1 hora Core-Tier1, 1 hora Core-Tier2)				12										12
★	SE/Herramientas y Análisis de Pruebas (1 hora Core-Tier1, 1 hora Core-Tier2)				4										4
★	SE/Pruebas de Rendimiento y Evaluación Comparativa (1 hora Core-Tier1, 1 hora Core-Tier2)				4										4

Cuadro 4.6: Tópicos de Ciencia de la Computación–*Computer Science* por curso del 5to al 6to Semestre (Parte 2 de 3)

Cuadro 4.7: Tópicos de Ciencia de la Computación–*Computer Science* por curso del 5to al 6to Semestre (Parte 3 de 3)

		Séptimo Sem						.
	Unidades de Conocimiento de Ciencia de la Computación–*Computer Science* **Cursos →**	CS251	CS262	CS2H1	AI264	FG211	FG350	Total
⭐	AI/Entrenamiento y Optimización de Redes Neuronales (1 hora Core-Tier1, 1 hora Core-Tier2)				10			10
⭐	AI/Técnicas de Regularización para Redes Profundas (1 hora Core-Tier1, 1 hora Core-Tier2)				5			5
⭐	AI/Arquitecturas de Redes Convolucionales y Residuales (1 hora Core-Tier1, 1 hora Core-Tier2)				9			9
⭐	AI/Arquitecturas Transformer y Modelos Generativos (1 hora Core-Tier1, 1 hora Core-Tier2)				16			16
	AI/Agentes y Sistemas Cognitivos			8				8
	AI/Procesamiento del Lenguaje Natural			8				8
⭐	GIT/Conceptos Fundamentales de Gráficos y Técnicas Interactivas (4 horas Core-Tier1, 3 horas Core-Tier2)	16						16
⭐	GIT/Visualización (6 horas Core-Tier2)	16						16
⭐	GIT/Renderizado Aplicado y Técnicas (15 horas Core-Tier2)	16						16
⭐	GIT/Modelado Geométrico (6 horas Core-Tier2)	16						16
⭐	GIT/Sombreado y Renderizado Avanzado (6 horas Core-Tier2)	12						12
⭐	HCI/Comprender al Usuario: Objetivos individuales e interacciones con otros (2 horas Core-Tier1, 5 horas Core-Tier2)			8				8
⭐	HCI/Responsabilidad y Rendición de Cuentas en el Diseño (2 horas Core-Tier1, 2 horas Core-Tier2)			8				8
⭐	HCI/Accesibilidad y Diseño Inclusivo (2 horas Core-Tier1, 2 horas Core-Tier2)			8				8
⭐	HCI/Evaluación del Diseño (1 hora Core-Tier1, 2 horas Core-Tier2)			8				8
⭐	HCI/Diseño de Sistemas (1 hora Core-Tier1, 5 horas Core-Tier2)			8				8
	HCI/Sociedad, ética y Profesión			8				8
⭐	SE/Colaboración y Dinámica de Equipo (2 horas Core-Tier1, 2 horas Core-Tier2)		14					14
⭐	SE/Principios y Arquitectura de Software (1 hora Core-Tier1, 2 horas Core-Tier2)		16					16
⭐	SE/Prácticas de Codificación (1 hora Core-Tier1, 2 horas Core-Tier2)		14					14
⭐	SE/Conceptos de verificación y validación (1 hora Core-Tier1, 1 hora Core-Tier2)		10					10
⭐	SE/Herramientas y Análisis de Pruebas (1 hora Core-Tier1, 1 hora Core-Tier2)		4					4
⭐	SE/Pruebas de Rendimiento y Evaluación Comparativa (1 hora Core-Tier1, 1 hora Core-Tier2)		4					4
⭐	SE/Conceptos de Confiabilidad (2 horas Core-Tier2)		10					10
	Total	76	72	64	40	0	0	

Cuadro 4.8: Tópicos de Ciencia de la Computación–*Computer Science* por curso del 7mo Semestre

		Octavo Sem											.
	Unidades de Conocimiento de Ciencia de la Computación–*Computer Science* **Cursos →**	CS281	CS3I1	CS3P1	CS402	EX301	ET201	AI268	CS391	CS393	CS394	CS395	Total
★	AI/Introducción a la IA: Conceptos y Comportamiento Inteligente (2 horas Core-Tier1, 1 hora Core-Tier2)							5					5
★	AI/Características del Problema y Naturaleza de los Agentes (1 hora Core-Tier1, 1 hora Core-Tier2)							5					5
★	AI/Fundamentos de Aprendizaje Automático (1 hora Core-Tier1, 2 horas Core-Tier2)							5					5
★	AI/Aprendizaje Automático: Datos, Representaciones y Evaluación (1 hora Core-Tier1, 2 horas Core-Tier2)							5					5
★	AI/Redes Neuronales y Ética (1 hora Core-Tier1, 1 hora Core-Tier2)							4					4
★	AI/Formulación del Aprendizaje Automático y Evaluación (1 hora Core-Tier1, 2 horas Core-Tier2)							3					3
★	AI/Aprendizaje Profundo, RL y Ética (1 hora Core-Tier1, 2 horas Core-Tier2)							3					3
	AI/Percepción y Visión por Computadora							24					24
★	NC/Aplicaciones en Red (4 horas Core-Tier1)										6		6
★	OS/Virtualización (1 hora Core-Tier2)										6		6
★	PDC/Programas: Paralelismo Declarativo (1 hora Core-Tier1, 1 hora Core-Tier2)			4									4
★	PDC/Programas: Inicio de Actividades (1 hora Core-Tier1, 1 hora Core-Tier2)			3									3
★	PDC/Programas: Propiedades de Ejecución (1 hora Core-Tier1, 1 hora Core-Tier2)			3									3
★	PDC/Programas: Distribución (1 hora Core-Tier1, 1 hora Core-Tier2)			2									2
★	PDC/Programas: Mapeos de Implementación (1 hora Core-Tier1, 1 hora Core-Tier2)			2									2
	PDC/Programación GPU			4									4
★	PDC/Comunicación (2 horas Core-Tier1, 4 horas Core-Tier2)			8									8
★	PDC/Comunicación: Propiedades y Extensiones (1 hora Core-Tier1, 1 hora Core-Tier2)			3									3
★	PDC/Memoria y Consistencia (1 hora Core-Tier1, 1 hora Core-Tier2)			3									3
★	PDC/Coordinación (1 hora Core-Tier1, 3 horas Core-Tier2)			7							6		13
★	PDC/Coordinación: Sincronización y Atomicidad (1 hora Core-Tier1, 2 horas Core-Tier2)			3									3
★	PDC/Coordinación: Propiedades Avanzadas (1 hora Core-Tier1, 2 horas Core-Tier2)			4									4
★	PDC/Evaluación (1 hora Core-Tier1, 3 horas Core-Tier2)			7									7
★	PDC/Evaluación: Métodos Formales (1 hora Core-Tier1, 1 hora Core-Tier2)			3									3
★	PDC/Evaluación: Pruebas y Medición (1 hora Core-Tier1, 1 hora Core-Tier2)			2									2
★	PDC/Algoritmos (1 hora Core-Tier1, 3 horas Core-Tier2)			4									4

Cuadro 4.9: Tópicos de Ciencia de la Computación–*Computer Science* por curso del 8vo Semestre (Parte 1 de 3)

	Unidades de Conocimiento de Ciencia de la Computación–*Computer Science* / Cursos →	Octavo Sem. CS281	CS3I1	CS3P1	CS402	EX301	ET201	AI268	CS391	CS393	CS394	CS395	Total
⭐	PDC/Algoritmos: Dominios de Aplicación (1 hora Core-Tier1, 3 horas Core-Tier2)			4									4
⭐	PDC/Algoritmos: Dominios Algorítmicos (1 hora Core-Tier1, 3 horas Core-Tier2)			4									4
⭐	SE/Control de Versiones y CI/CD (1 hora Core-Tier1, 2 horas Core-Tier2)								12		12	12	36
⭐	SE/Principios y Arquitectura de Software (1 hora Core-Tier1, 2 horas Core-Tier2)								12		8		20
⭐	SE/Diseño de API y Modelado (1 hora Core-Tier1, 1 hora Core-Tier2)										6		6
⭐	SE/Calidad y Evaluación del Diseño (1 hora Core-Tier1, 2 horas Core-Tier2)										8	4	12
⭐	SE/Construcción a Escala y Proceso (1 hora Core-Tier1, 2 horas Core-Tier2)											4	4
⭐	SE/Conceptos de verificación y validación (1 hora Core-Tier1, 1 hora Core-Tier2)								12				12
⭐	SE/Planificación y Tipos de Pruebas (1 hora Core-Tier1, 1 hora Core-Tier2)											8	8
⭐	SE/Prácticas Avanzadas de Pruebas (1 hora Core-Tier1, 1 hora Core-Tier2)											4	4
⭐	SE/Herramientas y Análisis de Pruebas (1 hora Core-Tier1, 1 hora Core-Tier2)								3			8	11
⭐	SE/Pruebas de Rendimiento y Evaluación Comparativa (1 hora Core-Tier1, 1 hora Core-Tier2)								3			4	7
⭐	SE/Compatibilidad y Versionado de API (2 horas Core-Tier2)								12			4	16
⭐	SE/Técnicas de Refactorización (1 hora Core-Tier2)											4	4
⭐	SE/Ingeniería de Confiabilidad (1 hora Core-Tier2)										6	4	10
⭐	SEC/Principios y Mentalidad de Seguridad (1 hora Core-Tier1, 3 horas Core-Tier2)		6										6
⭐	SEC/Ética, Ley y Sociedad en Seguridad (1 hora Core-Tier1, 2 horas Core-Tier2)		6										6
⭐	SEC/Inyección y Validación de Entrada (1 hora Core-Tier1, 2 horas Core-Tier2)		8										8
⭐	SEC/Análisis de Malware y Seguridad Avanzada (1 hora Core-Tier1, 3 horas Core-Tier2)		6										6
⭐	SEC/Fundamentos de Criptografía (1 hora Core-Tier1, 2 horas Core-Tier2)		8										8
⭐	SEC/Análisis de Amenazas e Ingeniería de Seguridad (1 hora Core-Tier1, 4 horas Core-Tier2)		8										8
⭐	SEC/Técnicas de Informática Forense (4 horas Core-Tier2)		6										6
⭐	SEC/Gobierno, Gestión y Política de Seguridad (2 horas Core-Tier2)		6										6
⭐	SEP/Impacto Social e IA (2 horas Core-Tier1, 1 hora Core-Tier2)	2											2
⭐	SEP/Fundamentos y Marcos del Análisis Ético (2 horas Core-Tier1, 1 hora Core-Tier2)	4											4
⭐	SEP/Valores, Naturaleza y Actualización Profesional (1 hora Core-Tier1, 1 hora Core-Tier2)	4											4

Cuadro 4.10: Tópicos de Ciencia de la Computación–*Computer Science* por curso del 8vo Semestre (Parte 2 de 3)

Octavo Sem.	CS281	CS311	CS3P1	CS402	EX301	ET201	AI268	CS391	CS393	CS394	CS395	Total
Unidades de Conocimiento de Ciencia de la Computación–*Computer Science*												
Cursos →												
SEP/Fundamentos de la Propiedad Intelectual (1 hora Core-Tier1, 1 hora Core-Tier2)	4											4
SEP/Derechos de PI Digital y Derechos de Autor (1 hora Core-Tier1, 1 hora Core-Tier2)	6											6
SEP/Datos, Vigilancia y Tecnología de Privacidad (1 hora Core-Tier1, 1 hora Core-Tier2)	4											4
SEP/Comunicación Técnica y Profesional (1 hora Core-Tier1, 1 hora Core-Tier2)	2											2
SEP/Huella Ambiental de los Sistemas Informáticos (1 hora Core-Tier1, 1 hora Core-Tier2)	2											2
SEP/Fundamentos e Historia Temprana de la Informática (1 hora Core-Tier1, 1 hora Core-Tier2)	2											2
SEP/Modelos y Estrategias Económicas (1 hora Core-Tier2)	2											2
SEP/Delitos y Ataques Informáticos (1 hora Core-Tier1, 1 hora Core-Tier2)	2											2
SEP/Identidad y Diversidad en la Informática (1 hora Core-Tier1, 1 hora Core-Tier2)	2											2
SF/Confiabilidad del Sistema (2 horas Core-Tier1, 1 hora Core-Tier2)										6		6
Total	36	54	70	0	0	0	54	54	0	64	56	

Cuadro 4.11: Tópicos de Ciencia de la Computación–*Computer Science* por curso del 8vo Semestre (Parte 3 de 3)

	Unidades de Conocimiento de Ciencia de la Computación–*Computer Science* Cursos →	Noveno Sem CS370	CS400	CS403	AI365	CB309	ET301	AI369	CS351	CS353	CS392	CS3P3	Décimo Sem CS3P2	CS404	AI367	AI368	. Total
⭐	AI/Fundamentos de Aprendizaje Automático (1 hora Core-Tier1, 2 horas Core-Tier2)														12		12
⭐	AI/Aprendizaje Automático: Datos, Representaciones y Evaluación (1 hora Core-Tier1, 2 horas Core-Tier2)														10		10
⭐	AI/Redes Neuronales y Ética (1 hora Core-Tier1, 1 hora Core-Tier2)														10		10
⭐	AI/Formulación del Aprendizaje Automático y Evaluación (1 hora Core-Tier1, 2 horas Core-Tier2)														3		3
⭐	AI/Aprendizaje Profundo, RL y Ética (1 hora Core-Tier1, 2 horas Core-Tier2)														3		3
⭐	AI/Aplicaciones e Impacto Social de la IA (2 horas Core-Tier1, 2 horas Core-Tier2)														6		6
⭐	AI/Aplicaciones Adicionales e Impacto Social de la IA (2 horas Core-Tier1, 2 horas Core-Tier2)														6		6
	AI/Procesamiento del Lenguaje Natural														20		20
	AI/Robótica							36									36
	AI/Percepción y Visión por Computadora							12									12
⭐	AL/Estrategias Algorítmicas (6 horas Core-Tier1)									12							12
⭐	AL/Lenguajes Formales y Autómatas (4 horas Core-Tier1, 10 horas Core-Tier2)									10							10
⭐	AR/Arquitecturas Cuánticas (2 horas Core-Tier2)									10							10
⭐	DM/Conceptos Fundamentales de Sistemas de Bases de Datos (2 horas Core-Tier1, 1 hora Core-Tier2)	12															12
⭐	DM/Sistemas NoSQL (2 horas Core-Tier2)	12															12
⭐	DM/Análisis de Datos (3 horas Core-Tier2)	12															12
	DM/Bases de Datos Distribuidas/Computación en la Nube	12											10				22
⭐	FPL/Computación Paralela y Distribuida (2 horas Core-Tier1, 2 horas Core-Tier2)												8				8
⭐	GIT/Modelado Geométrico (6 horas Core-Tier2)								10								10
⭐	GIT/Sombreado y Renderizado Avanzado (6 horas Core-Tier2)								12								12
⭐	GIT/Animación por Computadora (6 horas Core-Tier2)								8								8
⭐	GIT/Simulación (6 horas Core-Tier2)								8								8
⭐	GIT/Inmersión (6 horas Core-Tier2)								5								5
⭐	GIT/Interacción (4 horas Core-Tier2)								5								5
⭐	GIT/Computación Tangible/Física (6 horas Core-Tier2)								8								8
	GIT/Sociedad, ética y Profesión								8								8

		Noveno Sem											Décimo Sem				.
	Unidades de Conocimiento de Ciencia de la Computación–*Computer Science* **Cursos →**	CS370	CS400	CS403	AI365	CB309	ET301	AI369	CS351	CS353	CS392	CS3P3	CS3P2	CS404	AI367	AI368	Total
★	NC/Aplicaciones en Red (4 horas Core-Tier1)											24	6				30
★	NC/Soporte de Movilidad (4 horas Core-Tier2)											12					12
★	PDC/Algoritmos: Dominios de Aplicación (1 hora Core-Tier1, 3 horas Core-Tier2)												4				4
★	SE/Colaboración y Dinámica de Equipo (2 horas Core-Tier1, 2 horas Core-Tier2)										4						4
★	SE/Control de Versiones y CI/CD (1 hora Core-Tier1, 2 horas Core-Tier2)										4						4
★	SE/Ingeniería de Requisitos (2 horas Core-Tier2)										6						6
★	SE/Principios y Arquitectura de Software (1 hora Core-Tier1, 2 horas Core-Tier2)										6						6
★	SE/Prácticas de Codificación (1 hora Core-Tier1, 2 horas Core-Tier2)										6						6
★	SE/Conceptos de verificación y validación (1 hora Core-Tier1, 1 hora Core-Tier2)										6						6
★	SE/Herramientas y Análisis de Pruebas (1 hora Core-Tier1, 1 hora Core-Tier2)										3						3
★	SE/Pruebas de Rendimiento y Evaluación Comparativa (1 hora Core-Tier1, 1 hora Core-Tier2)										3						3
★	SE/Compatibilidad y Versionado de API (2 horas Core-Tier2)										6						6
★	SE/Conceptos de Confiabilidad (2 horas Core-Tier2)										5						5
	SE/Métodos Formales										5						5
★	SEC/Principios y Mentalidad de Seguridad (1 hora Core-Tier1, 3 horas Core-Tier2)												8				8
★	SEC/Ética, Ley y Sociedad en Seguridad (1 hora Core-Tier1, 2 horas Core-Tier2)											12					12
★	SPD/Arquitectura Cloud y Gestión de Datos (3 horas Core-Tier2)												12				12
	Total	48	0	0	0	0	0	48	64	32	54	48	48	0	70	0	

Cuadro 4.13: Tópicos de Ciencia de la Computación–*Computer Science* por curso del 9no al 10mo Semestre (Parte 2 de 2)

4.5.2. Tópicos de Ciencias Básicas para Computación/curso

		Primer Sem					Segundo Sem						
	Unidades de Conocimiento de Ciencias Básicas para Computación Cursos →	BIC101	BMA101	BMA102	BCH101	BEI101	CS100	CS112	CS1D1	BMA103	BFI101	BEI102	Total
★	CAN/Ecuaciones Paramétricas y Coordenadas Polares (1 hora Core-Tier1, 1 hora Core-Tier2)			6									6
★	CAN/Límites y Continuidad (1 hora Core-Tier1, 1 hora Core-Tier2)			12									12
★	CAN/Cálculo Diferencial (1 hora Core-Tier1, 1 hora Core-Tier2)			36									36
★	CAN/Antiderivadas y Técnicas de Integración (1 hora Core-Tier1, 1 hora Core-Tier2)									20			20
★	CAN/Integrales Impropias (1 hora Core-Tier1, 1 hora Core-Tier2)									8			8
★	CAN/Sucesiones, Series y Series de Potencias (1 hora Core-Tier1, 1 hora Core-Tier2)									20			20
★	ANT/Matrices y Sistemas de Ecuaciones Lineales (1 hora Core-Tier1, 1 hora Core-Tier2)		19										19
★	ANT/Espacios Vectoriales y Transformaciones Lineales (1 hora Core-Tier1, 1 hora Core-Tier2)		14										14
★	ANT/Valores Propios, Producto Interno y Formas Cuadráticas (1 hora Core-Tier1, 1 hora Core-Tier2)		15										15
★	CME/Movimiento Rotacional y Momento Angular (1 hora Core-Tier1, 1 hora Core-Tier2)										12		12
★	CME/Sistemas de Partículas y Momento Lineal (1 hora Core-Tier1, 1 hora Core-Tier2)										8		8
★	CME/Cinemática y Dinámica de Partículas (1 hora Core-Tier1, 1 hora Core-Tier2)										22		22
★	CME/Trabajo, Energía y Leyes de Conservación (1 hora Core-Tier1, 1 hora Core-Tier2)										12		12
★	CME/Elasticidad (1 hora Core-Tier1, 1 hora Core-Tier2)										2		2
★	CME/Fluidos: Hidrostática, Hidrodinámica y Viscosidad (1 hora Core-Tier1, 1 hora Core-Tier2)										10		10
★	CME/Vibraciones y Ondas (1 hora Core-Tier1, 1 hora Core-Tier2)										14		14
★	CME/Fenómeno de Transporte (1 hora Core-Tier1, 1 hora Core-Tier2)										4		4
★	GCH/Estructura Atómica (1 hora Core-Tier1, 1 hora Core-Tier2)				10								10
★	GCH/Enlace Químico (1 hora Core-Tier1, 1 hora Core-Tier2)				10								10
★	GCH/Materia y Energía (1 hora Core-Tier1, 1 hora Core-Tier2)				8								8
★	GCH/Estados de la Materia (1 hora Core-Tier1, 1 hora Core-Tier2)				10								10
★	GCH/Estequiometría (1 hora Core-Tier1, 1 hora Core-Tier2)				10								10
★	GCH/Química Ambiental y del Agua (1 hora Core-Tier1, 1 hora Core-Tier2)				4								4
★	GCH/Polímeros y Geosintéticos (1 hora Core-Tier1, 1 hora Core-Tier2)				4								4

Unidades de Conocimiento de Ciencias Básicas para Computación Cursos →	BIC101	BMA101	BMA102	BCH101	BEI101	CS100	CS112	CS1D1	BMA103	BFI101	BEI102	Total
	Primer Sem					Segundo Sem						
GCH/Equilibrio Químico (1 hora Core-Tier1, 1 hora Core-Tier2)				4								4
GCH/Equilibrio Iónico e Hidrólisis de Sales (1 hora Core-Tier1, 1 hora Core-Tier2)				12								12
GCH/Sales, Solubilidad y Equilibrios de Precipitación (1 hora Core-Tier1, 1 hora Core-Tier2)				6								6
GCH/Electroquímica y Electrólisis (1 hora Core-Tier1, 1 hora Core-Tier2)				8								8
GCH/Química del Cemento (1 hora Core-Tier1, 1 hora Core-Tier2)				4								4
Total	0	48	54	90	0	0	0	0	48	84	0	

Cuadro 4.15: Tópicos de Ciencias Básicas para Computación por curso del 1er al 2do Semestre (Parte 2 de 2)

		Tercer Sem						Cuarto Sem							
	Unidades de Conocimiento de Ciencias Básicas para Computación / **Cursos →**	CS113	CS2B1	AI161	BMA104	STA251	BEI201	CS210	CS211	CS221	CS271	CS401	MA202	BEI202	Total
★	CAN/Integrales de Línea y Campos Vectoriales (1 hora Core-Tier1, 1 hora Core-Tier2)				10										10
★	CAN/Integrales Múltiples (1 hora Core-Tier1, 1 hora Core-Tier2)				18										18
★	CAN/Transformaciones Multivariables (1 hora Core-Tier1, 1 hora Core-Tier2)				12										12
★	CAN/Integrales de Superficie y Teoremas Fundamentales (1 hora Core-Tier1, 1 hora Core-Tier2)				12										12
★	CAN/Funciones Vectoriales de una Variable Real (1 hora Core-Tier1, 1 hora Core-Tier2)				14										14
★	CAN/Cálculo Diferencial Multivariable (1 hora Core-Tier1, 1 hora Core-Tier2)				18										18
★	DMC/Teoría de Grafos (1 hora Core-Tier1, 1 hora Core-Tier2)												4		4
★	PST/Análisis de Series Temporales (1 hora Core-Tier1, 1 hora Core-Tier2)												2		2
★	NSA/Teoría de Aproximación e Interpolación (1 hora Core-Tier1, 1 hora Core-Tier2)												10		10
★	NSA/Integración Numérica y Cuadratura (1 hora Core-Tier1, 1 hora Core-Tier2)												10		10
★	NSA/Análisis de Errores y Aritmética de Punto Flotante (1 hora Core-Tier1, 1 hora Core-Tier2)												6		6
★	NSA/Búsqueda de Raíces de Ecuaciones No Lineales (1 hora Core-Tier1, 1 hora Core-Tier2)												10		10
★	NSA/Álgebra Lineal Numérica (1 hora Core-Tier1, 1 hora Core-Tier2)												12		12
★	NSA/Métodos Numéricos para Ecuaciones Diferenciales (1 hora Core-Tier1, 1 hora Core-Tier2)												12		12
★	NSA/Algoritmos de Optimización (1 hora Core-Tier1, 1 hora Core-Tier2)												4		4
★	NSA/Método de Elementos Finitos (1 hora Core-Tier1, 1 hora Core-Tier2)												8		8
★	NSA/Computación Paralela y de Alto Rendimiento (1 hora Core-Tier1, 1 hora Core-Tier2)												4		4
★	PRO/Espacios Muestrales y Eventos (1 hora Core-Tier1, 1 hora Core-Tier2)					4									4
★	PRO/Axiomas y Propiedades de la Probabilidad (1 hora Core-Tier1, 1 hora Core-Tier2)					4									4
★	PRO/Probabilidad Condicional e Independencia (1 hora Core-Tier1, 1					4									4

Tercer Sem										Cuarto Sem										.
Unidades de Conocimiento de Ciencias Básicas para Computación Cursos →	CS113	CS2B1	AI161	BMA104	STA251	BEI201	CS210	CS211	CS221	CS271	CS401	MA202	BEI202	Total
INF/Estimación por Intervalos e Intervalos de Confianza (1 hora Core-Tier1, 1 hora Core-Tier2)					4									4
INF/Fundamentos de las Pruebas de Hipótesis (1 hora Core-Tier1, 1 hora Core-Tier2)					4									4
INF/Inferencia Bayesiana (1 hora Core-Tier1, 1 hora Core-Tier2)					4									4
REG/Regresión Lineal Simple (1 hora Core-Tier1, 1 hora Core-Tier2)					4									4
Total	0	0	0	84	52	0	0	0	0	0	0	82	0	

Cuadro 4.17: Tópicos de Ciencias Básicas para Computación por curso del 3er al 4to Semestre (Parte 2 de 2)

Quinto Sem										Sexto Sem										.
Unidades de Conocimiento de Ciencias Básicas para Computación Cursos →	CS212	CS261	CS272	CS291	CS2S1	FG106	BEI203	CS231	CS311	CS312	CS342	AI263	FI201	Total
Total	0	0	0	0	0	0	0	0	0	0	0	0	0	

Cuadro 4.18: Tópicos de Ciencias Básicas para Computación por curso del 5to al 6to Semestre

		Séptimo Sem						
Unidades de Conocimiento de Ciencias Básicas para Computación	**Cursos →**	CS251	CS292	CS2H1	AI264	FG211	FG350	Total
Total		0	0	0	0	0	0	

Cuadro 4.19: Tópicos de Ciencias Básicas para Computación por curso del 7mo Semestre

		Octavo Sem										
Unidades de Conocimiento de Ciencias Básicas para Computación	**Cursos →**	CS281	CS3I1	CS3P1	CS402	EX301	ET201	AI268	CS391	CS393	CS394	CS395	Total
Total		0	0	0	0	0	0	0	0	0	0	0	

Cuadro 4.20: Tópicos de Ciencias Básicas para Computación por curso del 8vo Semestre

		Noveno Sem										Décimo Sem				
Unidades de Conocimiento de Ciencias Básicas para Computación	**Cursos →**	CS370	CS400	CS403	AI365	CB309	ET301	AI369	CS351	CS353	CS392	CS3P3	CS3P2	CS404	AI367	AI368	Total
Total		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Cuadro 4.21: Tópicos de Ciencias Básicas para Computación por curso del 9no al 10mo Semestre

4.6. Resultados esperados distribuídos por curso

Las siquientes tablas nos muestras una visión global de los resultados que se esperan lograr en cada curso de la presente malla curricular. La lista completa de resultados esperados se encuentra en la Sección: 1.5.1.

		Primer Sem						Segundo Sem					
	Competencia/Curso	BIC101	BMA101	BMA102	BCH101	BEI101	CS100	CS112	CS1D1	BMA103	BFI101	BEI102
AG-C01)	Impacto social de soluciones computacionales.											
AG-C02)	Ética profesional en computación.						1					
AG-C03)	Trabajo individual y en equipo.					1						1
AG-C04)	Comunicación efectiva en computación.					2						2
AG-C05)	Gestión de proyectos en computación.											
AG-C06)	Aprendizaje continuo en computación.											
AG-C07)	Conocimientos de computación.		1	1	1				2	1	1	
AG-C08)	Análisis de problemas computacionales.	1					1					
AG-C09)	Diseño y desarrollo de soluciones.							2				
AG-C10)	Indagación en computación.											
AG-C11)	Uso de herramientas computacionales.											
AG-C12)	Gestión de tecnologías seguras de computación.	2					1	2	2			

Cuadro 4.22: Resultados esperados por curso 1er al 2do Semestre

		Tercer Sem						Cuarto Sem						
	Competencia/Curso	CS113	CS2B1	AI161	BMA104	STA251	BEI201	CS210	CS211	CS221	CS271	CS401	MA202	BEI202
AG-C01)	Impacto social de soluciones computacionales.			1										
AG-C02)	Ética profesional en computación.											3		
AG-C03)	Trabajo individual y en equipo.						1			2				1
AG-C04)	Comunicación efectiva en computación.		2				2					3		2
AG-C05)	Gestión de proyectos en computación.											3		
AG-C06)	Aprendizaje continuo en computación.													
AG-C07)	Conocimientos de computación.				3	1								
AG-C08)	Análisis de problemas computacionales.							2	2			3	2	
AG-C09)	Diseño y desarrollo de soluciones.	2	2					2		2	2	3	2	
AG-C10)	Indagación en computación.													
AG-C11)	Uso de herramientas computacionales.			2									2	
AG-C12)	Gestión de tecnologías seguras de computación.	3							2		2			

Cuadro 4.23: Resultados esperados por curso 3er al 4to Semestre

		Quinto Sem								Sexto Sem						
	Competencia/Curso	CS212	CS261	CS272	CS291	CS251	FG1106	BE1203	CS231	CS311	CS312	CS342	AI263	FI201
AG-C01)	Impacto social de soluciones computacionales.													
AG-C02)	Ética profesional en computación.						2							
AG-C03)	Trabajo individual y en equipo.				2	2		1	2					
AG-C04)	Comunicación efectiva en computación.						2	2						
AG-C05)	Gestión de proyectos en computación.													
AG-C06)	Aprendizaje continuo en computación.													
AG-C07)	Conocimientos de computación.									2	2	2	2	2
AG-C08)	Análisis de problemas computacionales.	2	2	2									2	
AG-C09)	Diseño y desarrollo de soluciones.		2		2	2			2					
AG-C10)	Indagación en computación.													
AG-C11)	Uso de herramientas computacionales.													2
AG-C12)	Gestión de tecnologías seguras de computación.	2		2						2	2	2		

Cuadro 4.24: Resultados esperados por curso 5to al 6to Semestre

Séptimo Sem						
Competencia/Curso	CS251	CS292	CS2H1	AI264	FG211	FG350
AG-C01) Impacto social de soluciones computacionales.						
AG-C02) Ética profesional en computación.					2	2
AG-C03) Trabajo individual y en equipo.		2	2		2	
AG-C04) Comunicación efectiva en computación.						2
AG-C05) Gestión de proyectos en computación.						
AG-C06) Aprendizaje continuo en computación.						
AG-C07) Conocimientos de computación.	2					
AG-C08) Análisis de problemas computacionales.	2					
AG-C09) Diseño y desarrollo de soluciones.	3	2	2	2		
AG-C10) Indagación en computación.						
AG-C11) Uso de herramientas computacionales.				3		
AG-C12) Gestión de tecnologías seguras de computación.						

Cuadro 4.25: Resultados esperados por curso 7mo Semestre

Octavo Sem

Competencia/Curso	CS281	CS311	CS3P1	CS402	EX301	ET201	AI268	CS391	CS393	CS394	CS395
AG-C01) Impacto social de soluciones computacionales.	2				1						
AG-C02) Ética profesional en computación.	2			3							
AG-C03) Trabajo individual y en equipo.								2	2		
AG-C04) Comunicación efectiva en computación.				3							
AG-C05) Gestión de proyectos en computación.				3							
AG-C06) Aprendizaje continuo en computación.					1						
AG-C07) Conocimientos de computación.											
AG-C08) Análisis de problemas computacionales.				3			2				
AG-C09) Diseño y desarrollo de soluciones.		2	2	3			3	2	2	3	3
AG-C10) Indagación en computación.											
AG-C11) Uso de herramientas computacionales.		2	2							3	3
AG-C12) Gestión de tecnologías seguras de computación.							2			2	2

Cuadro 4.26: Resultados esperados por curso 8vo Semestre

		Noveno Sem												Décimo Sem			
	Competencia/Curso	CS370	CS400	CS403	AI365	CB309	ET301	AI369	CS351	CS353	CS392	CS3P3	CS3P2	CS404	AI367	AI368
AG-C01)	Impacto social de soluciones computacionales.		2													
AG-C02)	Ética profesional en computación.		3	3										3		
AG-C03)	Trabajo individual y en equipo.		3								2					
AG-C04)	Comunicación efectiva en computación.		3	3										3		
AG-C05)	Gestión de proyectos en computación.		2	3										3		
AG-C06)	Aprendizaje continuo en computación.		3												2	
AG-C07)	Conocimientos de computación.															
AG-C08)	Análisis de problemas computacionales.			3										3		2
AG-C09)	Diseño y desarrollo de soluciones.	2	3	3	3			2	2		2	2	2	3		
AG-C10)	Indagación en computación.					2				2						
AG-C11)	Uso de herramientas computacionales.	2			3	2		3	2	2		2	2			
AG-C12)	Gestión de tecnologías seguras de computación.															2

Cuadro 4.27: Resultados esperados por curso 9no al 10mo Semestre

4.7. Cursos por resultado (*Outcome*)

4.7.1. Outcome: AG-C01) Impacto social de soluciones computacionales.

- AI161 IA Aplicada (3er Sem-Pág 316)
- CS281 Computación en la Sociedad (8vo Sem-Pág 443)
- EX301 Actividades Extracurriculares (8vo Sem-Pág 469)
- CS400 Prácticas Pre-profesionales (9no Sem-Pág 507)

4.7.2. Outcome: AG-C02) Ética profesional en computación.

- CS100 Introducción a la Ciencia de la Computación (2do Sem-Pág 276)
- CS401 Metodología de la Investigación (4to Sem-Pág 353)
- FG106 Teatro (5to Sem-Pág 393)
- FG211 Ética Profesional (7mo Sem-Pág 438)
- FG350 Liderazgo y Desempeño (7mo Sem-Pág 440)
- CS281 Computación en la Sociedad (8vo Sem-Pág 443)
- CS402 Proyecto de tesis 1 (8vo Sem-Pág 465)
- CS400 Prácticas Pre-profesionales (9no Sem-Pág 507)
- CS403 Proyecto de tesis 2 (9no Sem-Pág 509)
- CS404 Taller de Investigación (10mo Sem-Pág 548)

4.7.3. Outcome: AG-C03) Trabajo individual y en equipo.

- BEI101 Inglés I (1er Sem-Pág 272)
- BEI102 Inglés II (2do Sem-Pág 302)
- BEI201 Inglés III (3er Sem-Pág 329)
- CS221 Arquitectura de Computadores (4to Sem-Pág 343)
- BEI202 Inglés IV (4to Sem-Pág 362)
- CS291 Ingeniería de Software I (5to Sem-Pág 382)
- CS2S1 Sistemas Operativos (5to Sem-Pág 388)
- BEI203 Inglés V (5to Sem-Pág 397)
- CS231 Redes y Comunicación (6to Sem-Pág 401)
- CS292 Ingeniería de Software II (7mo Sem-Pág 425)
- CS2H1 Experiencia de Usuario (UX) (7mo Sem-Pág 430)
- CS391 Ingeniería de Software III (8vo Sem-Pág 481)
- CS393 Sistemas de Información (8vo Sem-Pág 486)
- CS400 Prácticas Pre-profesionales (9no Sem-Pág 507)
- CS392 Tópicos en Ingeniería de Software (9no Sem-Pág 534)

4.7.4. Outcome: AG-C04) Comunicación efectiva en computación.

- BEI101 Inglés I (1er Sem-Pág 272)
- BEI102 Inglés II (2do Sem-Pág 302)
- CS2B1 Desarrollo Basado en Plataformas (3er Sem-Pág 312)
- BEI201 Inglés III (3er Sem-Pág 329)
- CS401 Metodología de la Investigación (4to Sem-Pág 353)
- BEI202 Inglés IV (4to Sem-Pág 362)
- FG106 Teatro (5to Sem-Pág 393)
- BEI203 Inglés V (5to Sem-Pág 397)
- FG211 Ética Profesional (7mo Sem-Pág 438)

- FG350 Liderazgo y Desempeño (7mo Sem-Pág 440)
- CS402 Proyecto de tesis 1 (8vo Sem-Pág 465)
- CS400 Prácticas Pre-profesionales (9no Sem-Pág 507)
- CS403 Proyecto de tesis 2 (9no Sem-Pág 509)
- CS404 Taller de Investigación (10mo Sem-Pág 548)

4.7.5. Outcome: AG-C05) Gestión de proyectos en computación.

- CS401 Metodología de la Investigación (4to Sem-Pág 353)
- CS402 Proyecto de tesis 1 (8vo Sem-Pág 465)
- CS400 Prácticas Pre-profesionales (9no Sem-Pág 507)
- CS403 Proyecto de tesis 2 (9no Sem-Pág 509)
- CS404 Taller de Investigación (10mo Sem-Pág 548)

4.7.6. Outcome: AG-C06) Aprendizaje continuo en computación.

- EX301 Actividades Extracurriculares (8vo Sem-Pág 469)
- CS400 Prácticas Pre-profesionales (9no Sem-Pág 507)
- AI367 Tópicos en Inteligencia Artificial (10mo Sem-Pág 552)

4.7.7. Outcome: AG-C07) Conocimientos de computación.

- BMA101 Algebra Lineal (1er Sem-Pág 262)
- BMA102 Cálculo Diferencial (1er Sem-Pág 264)
- BCH101 Química I (1er Sem-Pág 267)
- CS1D1 Estructuras Discretas (2do Sem-Pág 292)
- BMA103 Cálculo Integral (2do Sem-Pág 295)
- BFI101 Física I (2do Sem-Pág 297)
- BMA104 Cálculo Diferencial e Integral Avanzado (3er Sem-Pág 320)
- STA251 Estadística y Probabilidades (3er Sem-Pág 324)
- CS251 Computación Gráfica (7mo Sem-Pág 421)

4.7.8. Outcome: AG-C08) Análisis de problemas computacionales.

- BIC101 Introducción a la Computación (1er Sem-Pág 257)
- CS100 Introducción a la Ciencia de la Computación (2do Sem-Pág 276)
- CS210 Algoritmos y Estructuras de Datos (4to Sem-Pág 333)
- CS211 Teoría de la Computación (4to Sem-Pág 338)
- CS401 Metodología de la Investigación (4to Sem-Pág 353)
- MA202 Métodos Numéricos (4to Sem-Pág 357)
- CS212 Análisis y Diseño de Algoritmos (5to Sem-Pág 366)
- CS261 Inteligencia Artificial (5to Sem-Pág 371)
- CS272 Bases de Datos II (5to Sem-Pág 378)
- CS311 Algoritmos para Problemas Complejos (6to Sem-Pág 406)
- CS312 Estructuras de Datos Avanzadas (6to Sem-Pág 409)
- CS342 Compiladores (6to Sem-Pág 412)
- AI263 Introducción al Aprendizaje de Máquina (6to Sem-Pág 415)
- FI201 Física Computacional (6to Sem-Pág 418)
- CS251 Computación Gráfica (7mo Sem-Pág 421)
- CS402 Proyecto de tesis 1 (8vo Sem-Pág 465)
- AI268 Visión Computacional (8vo Sem-Pág 475)

- CS403 Proyecto de tesis 2 (9no Sem-Pág 509)
- CS404 Taller de Investigación (10mo Sem-Pág 548)
- AI368 Computación Evolutiva (10mo Sem-Pág 559)

4.7.9. Outcome: AG-C09) Diseño y desarrollo de soluciones.

- CS112 Programación Orientada a Objetos I (2do Sem-Pág 287)
- CS113 Programación Orientada a Objetos II (3er Sem-Pág 306)
- CS2B1 Desarrollo Basado en Plataformas (3er Sem-Pág 312)
- CS210 Algoritmos y Estructuras de Datos (4to Sem-Pág 333)
- CS221 Arquitectura de Computadores (4to Sem-Pág 343)
- CS271 Bases de Datos I (4to Sem-Pág 348)
- CS401 Metodología de la Investigación (4to Sem-Pág 353)
- MA202 Métodos Numéricos (4to Sem-Pág 357)
- CS261 Inteligencia Artificial (5to Sem-Pág 371)
- CS291 Ingeniería de Software I (5to Sem-Pág 382)
- CS2S1 Sistemas Operativos (5to Sem-Pág 388)
- CS231 Redes y Comunicación (6to Sem-Pág 401)
- AI263 Introducción al Aprendizaje de Máquina (6to Sem-Pág 415)
- CS251 Computación Gráfica (7mo Sem-Pág 421)
- CS292 Ingeniería de Software II (7mo Sem-Pág 425)
- CS2H1 Experiencia de Usuario (UX) (7mo Sem-Pág 430)
- AI264 Aprendizaje Profundo (7mo Sem-Pág 435)
- CS3I1 Seguridad en Computación (8vo Sem-Pág 449)
- CS3P1 Computación Paralela y Distribuída (8vo Sem-Pág 455)
- CS402 Proyecto de tesis 1 (8vo Sem-Pág 465)
- AI268 Visión Computacional (8vo Sem-Pág 475)
- CS391 Ingeniería de Software III (8vo Sem-Pág 481)
- CS393 Sistemas de Información (8vo Sem-Pág 486)
- CS394 Arquitectura de Software y DevOps para Software en la Nube (8vo Sem-Pág 489)
- CS395 Calidad de Software (8vo Sem-Pág 495)
- CS370 Big Data (9no Sem-Pág 503)
- CS400 Prácticas Pre-profesionales (9no Sem-Pág 507)
- CS403 Proyecto de tesis 2 (9no Sem-Pág 509)
- AI365 Modelos Generativos Avanzados en IA (9no Sem-Pág 513)
- AI369 Robótica (9no Sem-Pág 522)
- CS351 Tópicos en Computación Gráfica (9no Sem-Pág 526)
- CS392 Tópicos en Ingeniería de Software (9no Sem-Pág 534)
- CS3P3 Internet de las Cosas (9no Sem-Pág 541)
- CS3P2 Cloud Computing (10mo Sem-Pág 544)
- CS404 Taller de Investigación (10mo Sem-Pág 548)

4.7.10. Outcome: AG-C10) Indagación en computación.

- CB309 Bioinformática (9no Sem-Pág 515)
- CS353 Computación Cuántica (9no Sem-Pág 530)

4.7.11. Outcome: AG-C11) Uso de herramientas computacionales.

- AI161 IA Aplicada (3er Sem-Pág 316)

- MA202 Métodos Numéricos (4to Sem-Pág 357)
- FI201 Física Computacional (6to Sem-Pág 418)
- AI264 Aprendizaje Profundo (7mo Sem-Pág 435)
- CS3I1 Seguridad en Computación (8vo Sem-Pág 449)
- CS3P1 Computación Paralela y Distribuída (8vo Sem-Pág 455)
- CS394 Arquitectura de Software y DevOps para Software en la Nube (8vo Sem-Pág 489)
- CS395 Calidad de Software (8vo Sem-Pág 495)
- CS370 Big Data (9no Sem-Pág 503)
- AI365 Modelos Generativos Avanzados en IA (9no Sem-Pág 513)
- CB309 Bioinformática (9no Sem-Pág 515)
- AI369 Robótica (9no Sem-Pág 522)
- CS351 Tópicos en Computación Gráfica (9no Sem-Pág 526)
- CS353 Computación Cuántica (9no Sem-Pág 530)
- CS3P3 Internet de las Cosas (9no Sem-Pág 541)
- CS3P2 Cloud Computing (10mo Sem-Pág 544)

4.7.12. Outcome: AG-C12) Gestión de tecnologías seguras de computación.

- BIC101 Introducción a la Computación (1er Sem-Pág 257)
- CS100 Introducción a la Ciencia de la Computación (2do Sem-Pág 276)
- CS112 Programación Orientada a Objetos I (2do Sem-Pág 287)
- CS1D1 Estructuras Discretas (2do Sem-Pág 292)
- CS113 Programación Orientada a Objetos II (3er Sem-Pág 306)
- CS211 Teoría de la Computación (4to Sem-Pág 338)
- CS271 Bases de Datos I (4to Sem-Pág 348)
- CS212 Análisis y Diseño de Algoritmos (5to Sem-Pág 366)
- CS272 Bases de Datos II (5to Sem-Pág 378)
- CS311 Algoritmos para Problemas Complejos (6to Sem-Pág 406)
- CS312 Estructuras de Datos Avanzadas (6to Sem-Pág 409)
- CS342 Compiladores (6to Sem-Pág 412)
- AI268 Visión Computacional (8vo Sem-Pág 475)
- CS394 Arquitectura de Software y DevOps para Software en la Nube (8vo Sem-Pág 489)
- CS395 Calidad de Software (8vo Sem-Pág 495)
- AI368 Computación Evolutiva (10mo Sem-Pág 559)

4.8. Distribución de cursos en la carrera

Esta propuesta puede ser analizada por el número de créditos dedicados a cada área, por niveles de cursos (Introductorios, Intermedios, Avanzados y Proyectos).

	Estudios generales	Específicos	Especialización	
1er Sem	20			20
2do Sem	12	11		23
3er Sem	7	15		22
4to Sem	2	19	2	23
5to Sem	2	21		23
6to Sem		17	4	21
7mo Sem		12	8	20
8vo Sem	2	15	4	21
9no Sem		7	15	22
10mo Sem		7	8	15
Total	45	124	41	210
	21.4 %	59 %	19.5 %	

Cuadro 4.28: Distribución de créditos por área por semestre

Figura 4.2: Créditos por área por semestre

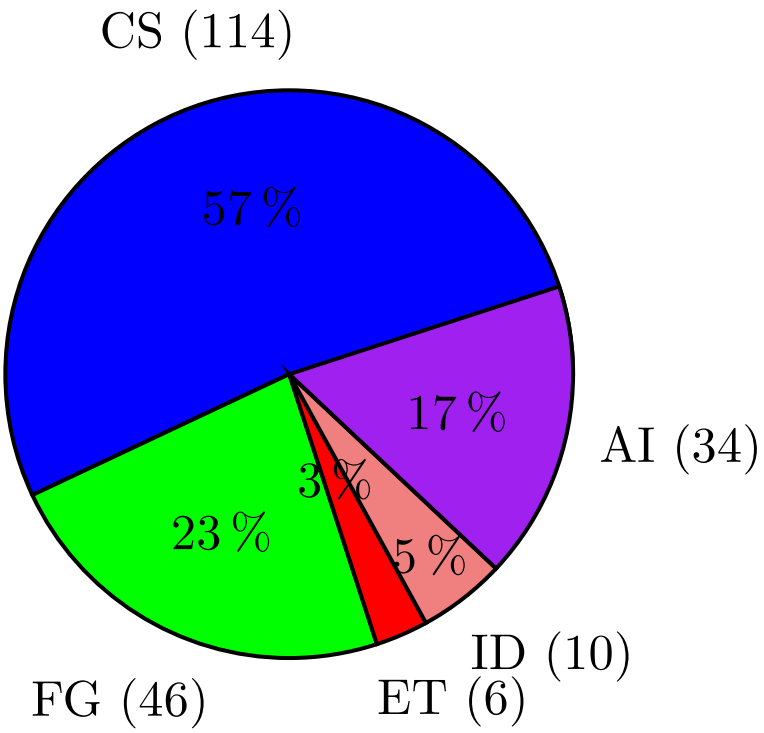

Figura 4.3: Distribución de cursos por áreas considerando creditaje.

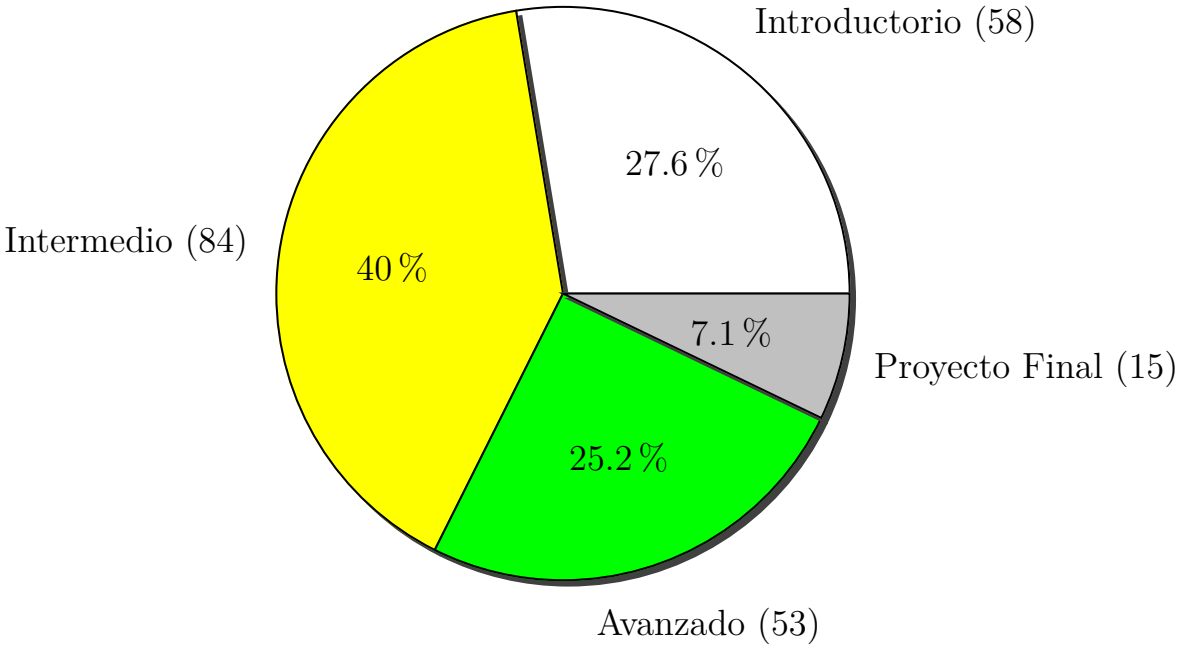

Figura 4.4: Distribución de créditos por niveles de cursos.

4.9. Departamentos

4.9.1. DAISI

La lista de cursos de este departamento es:

- CB309 Bioinformática (9no Sem)

- ET301 Formación de Empresas de Base Tecnológica II (9no Sem)
- AI367 Tópicos en Inteligencia Artificial (10mo Sem)
- AI368 Computación Evolutiva (10mo Sem)
- CS3P2 Cloud Computing (10mo Sem)

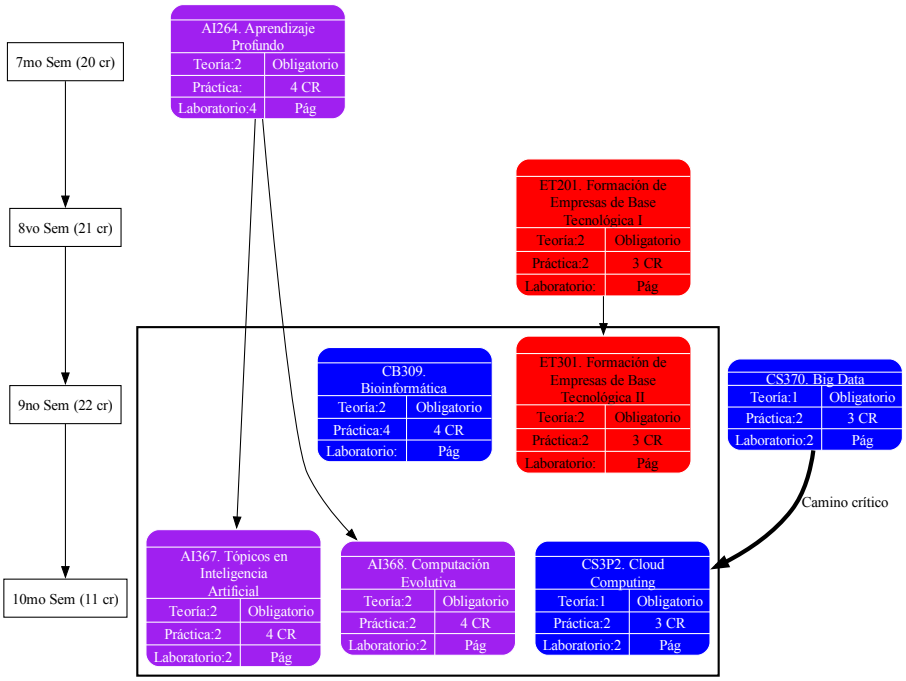

Figura 4.5: DAISI

4.10. Compatibilidad de la carrera con relación a estandares internacionales

En esta sección presentamos la distribución de cursos por áreas de concentración en contraste con las propuestas internacionales de las carreras de la *Computing Curricula* de IEEE-CS/ACM.

Es necesario notar que en algunos casos las materias podrían aparecen en más de un eje pues tienen contenido de más de una área. Por ejemplo, la materia de sistemas operativos contiene unidades de aplicación que pueden ser clasificadas en Tecnología de Información pero al mismo tiempo contiene fundamentos de como está estructurado un Sistema Operativo que es del eje de Ciencia de la Computación. En estos casos el creditaje ha sido divido entre los ejes correspondientes.

1. Hardware y Arquitectura (9.50 Créditos)
   - CS221 Arquitectura de Computadores (4to Sem-Pág 343)
   - AI369 Robótica (9no Sem-Pág 522)
2. Ciencia de la Computación (134.17 Créditos)
   - BIC101 Introducción a la Computación (1er Sem-Pág 257)
   - CS100 Introducción a la Ciencia de la Computación (2do Sem-Pág 276)
   - CS112 Programación Orientada a Objetos I (2do Sem-Pág 287)
   - CS113 Programación Orientada a Objetos II (3er Sem-Pág 306)
   - CS210 Algoritmos y Estructuras de Datos (4to Sem-Pág 333)
   - CS211 Teoría de la Computación (4to Sem-Pág 338)
   - CS401 Metodología de la Investigación (4to Sem-Pág 353)

- CS212 Análisis y Diseño de Algoritmos (5to Sem-Pág 366)
- CS2S1 Sistemas Operativos (5to Sem-Pág 388)
- CS231 Redes y Comunicación (6to Sem-Pág 401)
- CS312 Estructuras de Datos Avanzadas (6to Sem-Pág 409)
- CS342 Compiladores (6to Sem-Pág 412)
- CS251 Computación Gráfica (7mo Sem-Pág 421)
- CS2H1 Experiencia de Usuario (UX) (7mo Sem-Pág 430)
- CS281 Computación en la Sociedad (8vo Sem-Pág 443)
- CS3P1 Computación Paralela y Distribuída (8vo Sem-Pág 455)
- CS402 Proyecto de tesis 1 (8vo Sem-Pág 465)
- CS403 Proyecto de tesis 2 (9no Sem-Pág 509)
- CB309 Bioinformática (9no Sem-Pág 515)
- CS351 Tópicos en Computación Gráfica (9no Sem-Pág 526)
- CS353 Computación Cuántica (9no Sem-Pág 530)
- CS3P3 Internet de las Cosas (9no Sem-Pág 541)
- CS3P2 Cloud Computing (10mo Sem-Pág 544)
- CS404 Taller de Investigación (10mo Sem-Pág 548)

3. Inteligencia Artificial (60.50 Créditos)
   - AI161 IA Aplicada (3er Sem-Pág 316)
   - CS261 Inteligencia Artificial (5to Sem-Pág 371)
   - AI263 Introducción al Aprendizaje de Máquina (6to Sem-Pág 415)
   - AI264 Aprendizaje Profundo (7mo Sem-Pág 435)
   - AI268 Visión Computacional (8vo Sem-Pág 475)
   - AI365 Modelos Generativos Avanzados en IA (9no Sem-Pág 513)
   - AI369 Robótica (9no Sem-Pág 522)
   - AI367 Tópicos en Inteligencia Artificial (10mo Sem-Pág 552)
   - AI368 Computación Evolutiva (10mo Sem-Pág 559)
4. Ingeniería de Software (41.67 Créditos)
   - CS112 Programación Orientada a Objetos I (2do Sem-Pág 287)
   - CS113 Programación Orientada a Objetos II (3er Sem-Pág 306)
   - CS291 Ingeniería de Software I (5to Sem-Pág 382)
   - CS292 Ingeniería de Software II (7mo Sem-Pág 425)
   - CS2H1 Experiencia de Usuario (UX) (7mo Sem-Pág 430)
   - CS391 Ingeniería de Software III (8vo Sem-Pág 481)
   - CS393 Sistemas de Información (8vo Sem-Pág 486)
   - CS394 Arquitectura de Software y DevOps para Software en la Nube (8vo Sem-Pág 489)
   - CS395 Calidad de Software (8vo Sem-Pág 495)
   - CS392 Tópicos en Ingeniería de Software (9no Sem-Pág 534)
5. Sistemas de Información
   - Ninguno
6. Tecnología de Información (34.17 Créditos)
   - CS2B1 Desarrollo Basado en Plataformas (3er Sem-Pág 312)
   - CS271 Bases de Datos I (4to Sem-Pág 348)
   - CS272 Bases de Datos II (5to Sem-Pág 378)
   - CS2S1 Sistemas Operativos (5to Sem-Pág 388)
   - CS231 Redes y Comunicación (6to Sem-Pág 401)
   - CS2H1 Experiencia de Usuario (UX) (7mo Sem-Pág 430)
   - CS281 Computación en la Sociedad (8vo Sem-Pág 443)
   - CS370 Big Data (9no Sem-Pág 503)
   - CS3P3 Internet de las Cosas (9no Sem-Pág 541)

- CS3P2 Cloud Computing (10mo Sem-Pág 544)

7. Ciencia de Datos (10.00 Créditos)
   - CS271 Bases de Datos I (4to Sem-Pág 348)
   - CS272 Bases de Datos II (5to Sem-Pág 378)
   - CS370 Big Data (9no Sem-Pág 503)
8. Ciberseguridad (6.00 Créditos)
   - CS3I1 Seguridad en Computación (8vo Sem-Pág 449)
9. Matemática para Computación (14.00 Créditos)
   - CS1D1 Estructuras Discretas (2do Sem-Pág 292)
   - CS311 Algoritmos para Problemas Complejos (6to Sem-Pág 406)
10. Aspectos Organizacionales
    - Ninguno
11. Humanidades
    - Ninguno
12. Ciencias Básicas (76.00 Créditos)
    - BMA101 Algebra Lineal (1er Sem-Pág 262)
    - BMA102 Cálculo Diferencial (1er Sem-Pág 264)
    - BCH101 Química I (1er Sem-Pág 267)
    - BMA103 Cálculo Integral (2do Sem-Pág 295)
    - BFI101 Física I (2do Sem-Pág 297)
    - BMA104 Cálculo Diferencial e Integral Avanzado (3er Sem-Pág 320)
    - STA251 Estadística y Probabilidades (3er Sem-Pág 324)
    - MA202 Métodos Numéricos (4to Sem-Pág 357)
    - FI201 Física Computacional (6to Sem-Pág 418)
13. Contenido Empresarial (12.00 Créditos)
    - ET201 Formación de Empresas de Base Tecnológica I (8vo Sem-Pág 471)
    - ET301 Formación de Empresas de Base Tecnológica II (9no Sem-Pág 519)
14. Formación General (34.00 Créditos)
    - BEI101 Inglés I (1er Sem-Pág 272)
    - BEI102 Inglés II (2do Sem-Pág 302)
    - BEI201 Inglés III (3er Sem-Pág 329)
    - BEI202 Inglés IV (4to Sem-Pág 362)
    - FG106 Teatro (5to Sem-Pág 393)
    - BEI203 Inglés V (5to Sem-Pág 397)
    - FG211 Ética Profesional (7mo Sem-Pág 438)
    - FG350 Liderazgo y Desempeño (7mo Sem-Pág 440)
    - EX301 Actividades Extracurriculares (8vo Sem-Pág 469)
    - CS400 Prácticas Pre-profesionales (9no Sem-Pág 507)

Considerando esta distribución, las figuras 4.6 a 4.18 nos permiten tener una vista gráfica de este currículo en comparación con las propuestas de programas presentadas por IEEE-CS/ACM en la *Computing Curricula*.

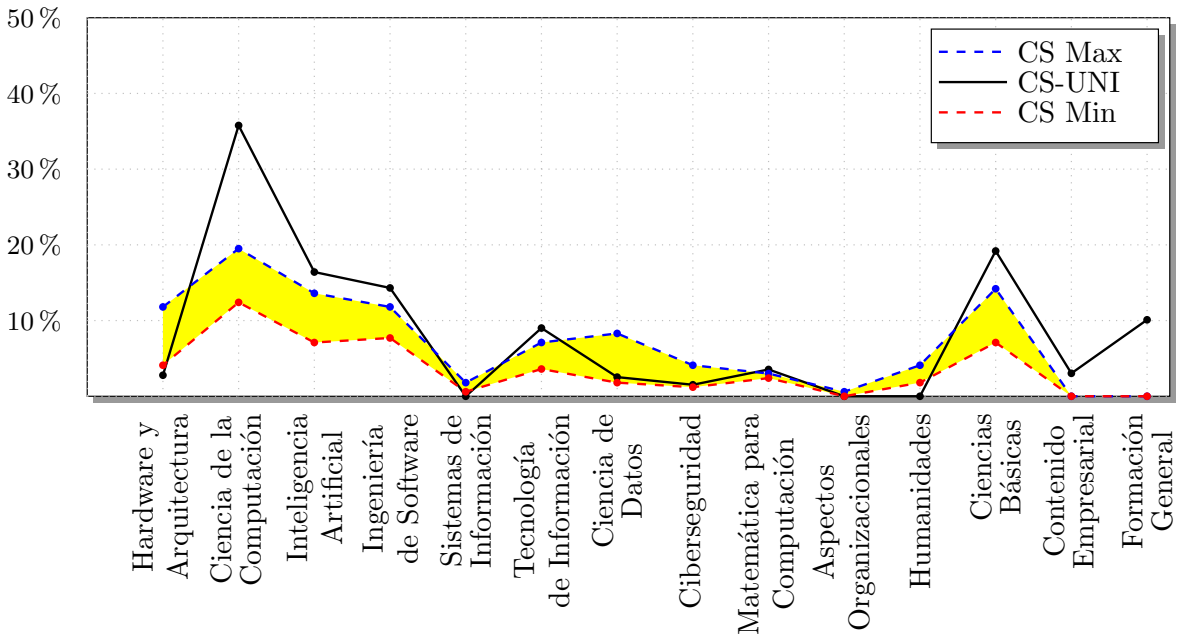

Figura 4.6: Comparación de CS-UNI con CS de ACM/IEEE-CS (Gráfico de curvas).

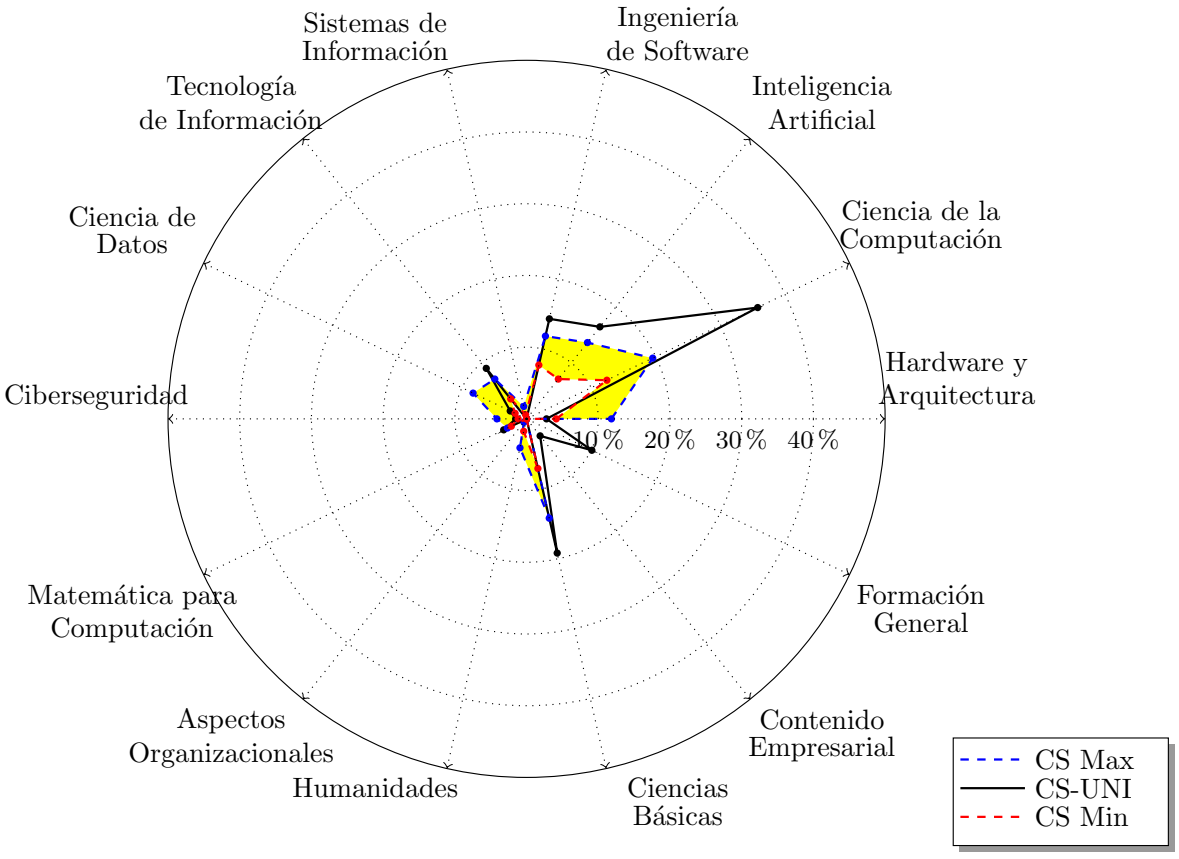

Figura 4.7: Comparación de CS-UNI con CS de ACM/IEEE-CS (Gráfico de araña).

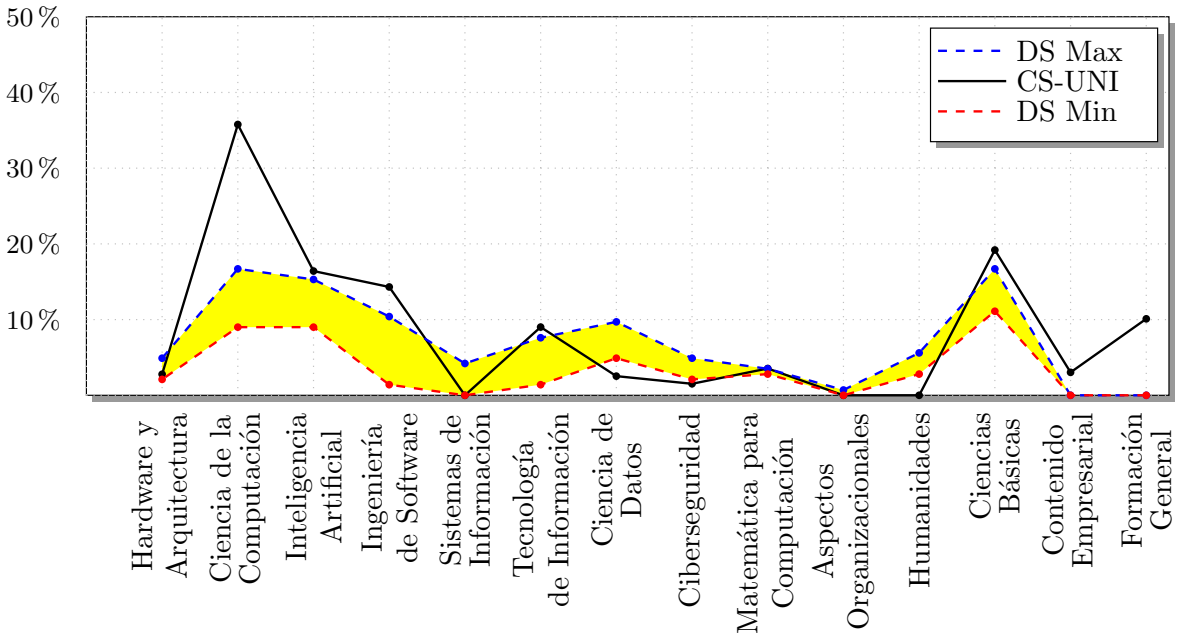

Figura 4.8: Comparación de CS-UNI con DS de ACM/IEEE-CS (Gráfico de curvas).

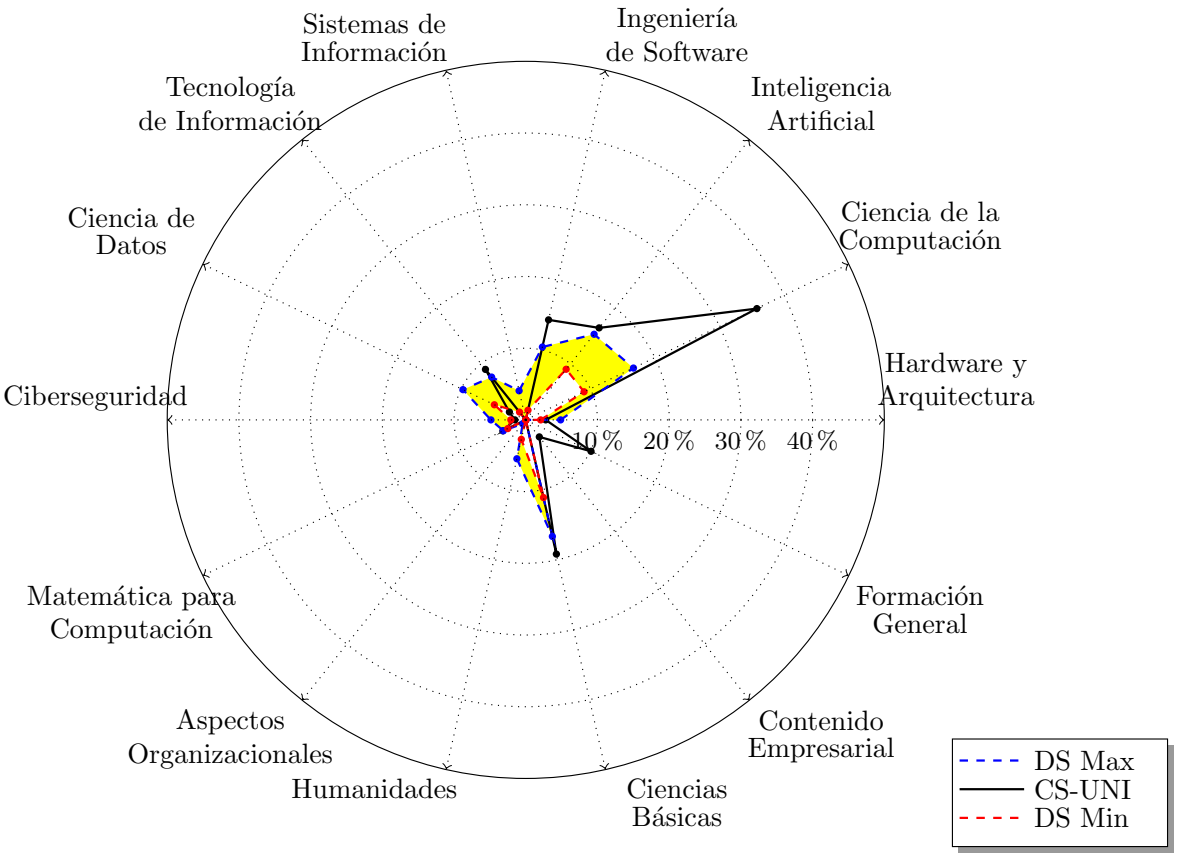

Figura 4.9: Comparación de CS-UNI con DS de ACM/IEEE-CS (Gráfico de araña).

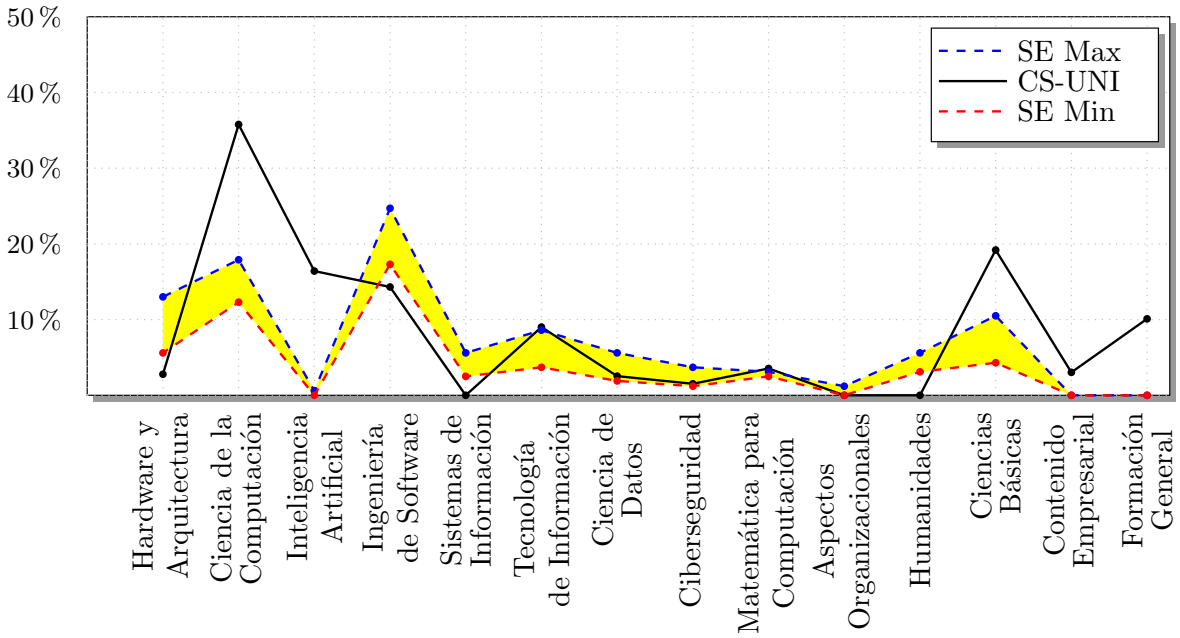

Figura 4.10: Comparación de CS-UNI con SE de ACM/IEEE-CS (Gráfico de curvas).

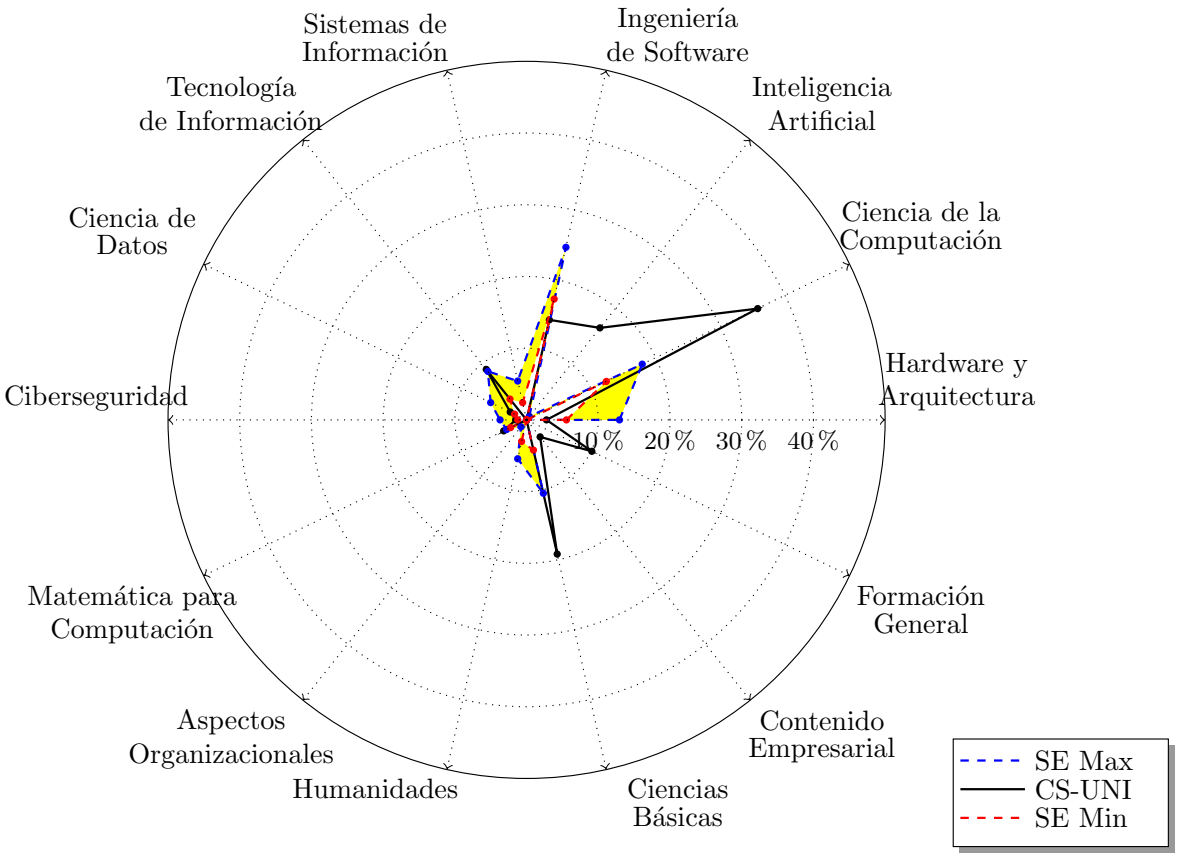

Figura 4.11: Comparación de CS-UNI con SE de ACM/IEEE-CS (Gráfico de araña).

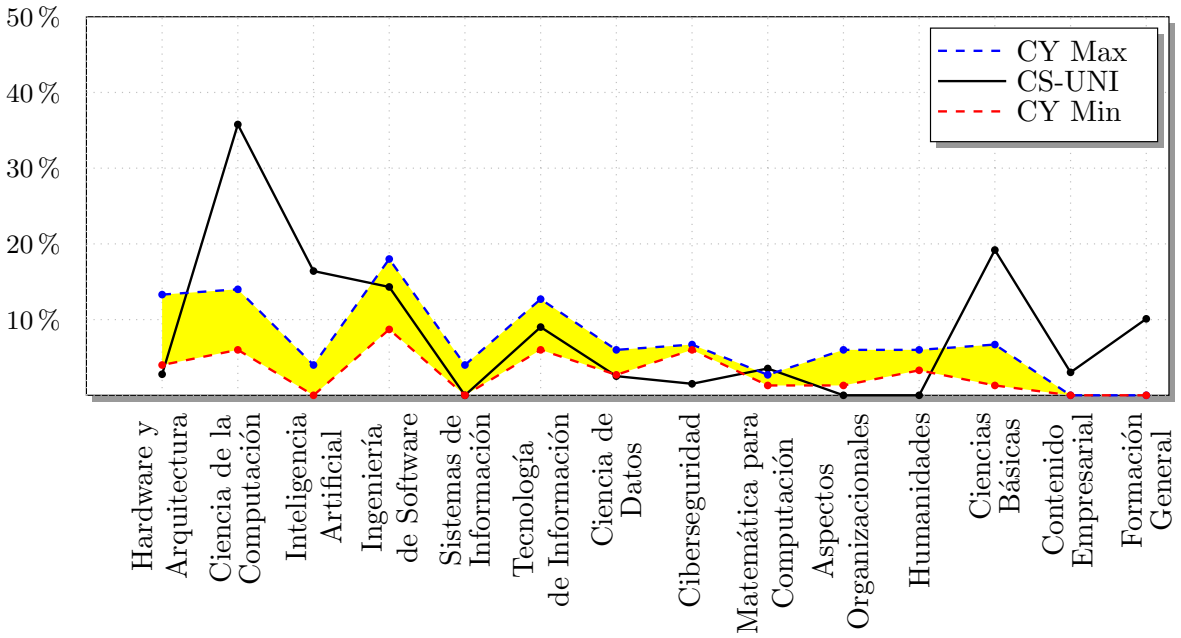

Figura 4.12: Comparación de CS-UNI con CY de ACM/IEEE-CS (Gráfico de curvas).

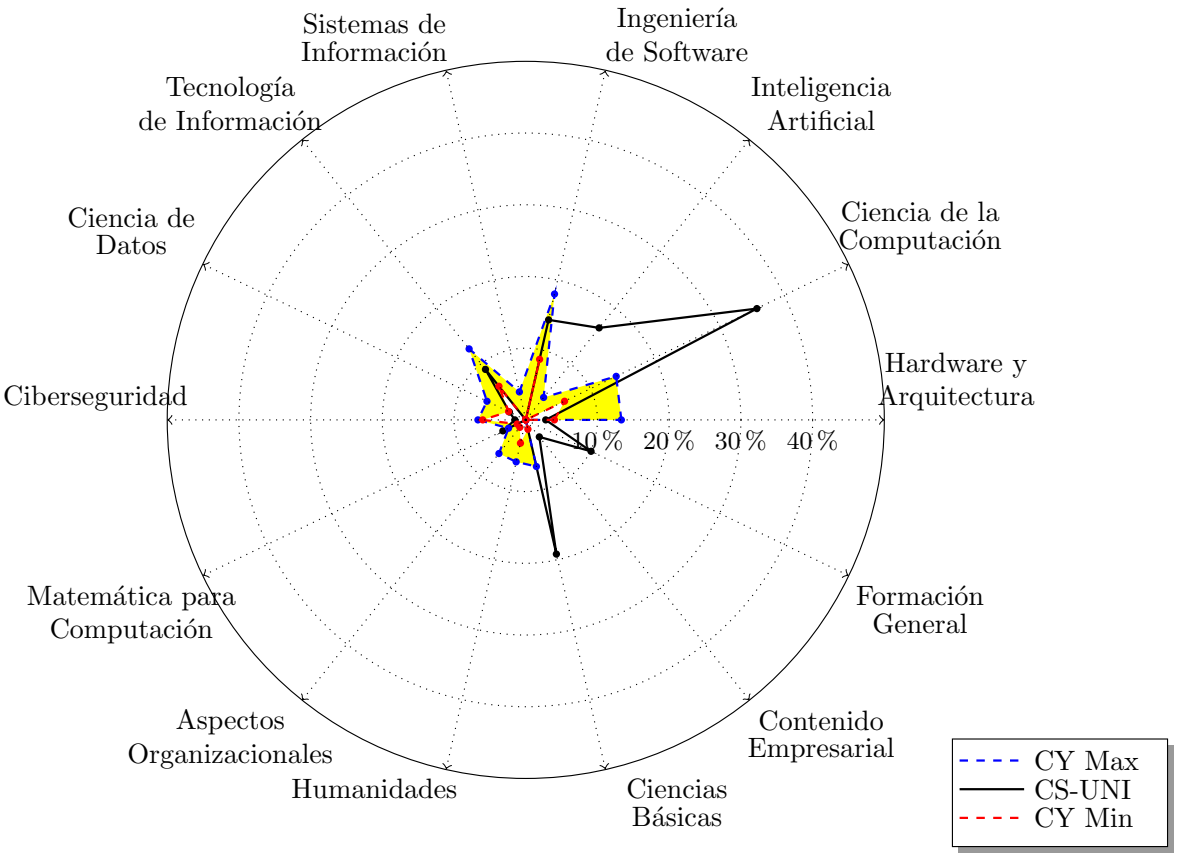

Figura 4.13: Comparación de CS-UNI con CY de ACM/IEEE-CS (Gráfico de araña).

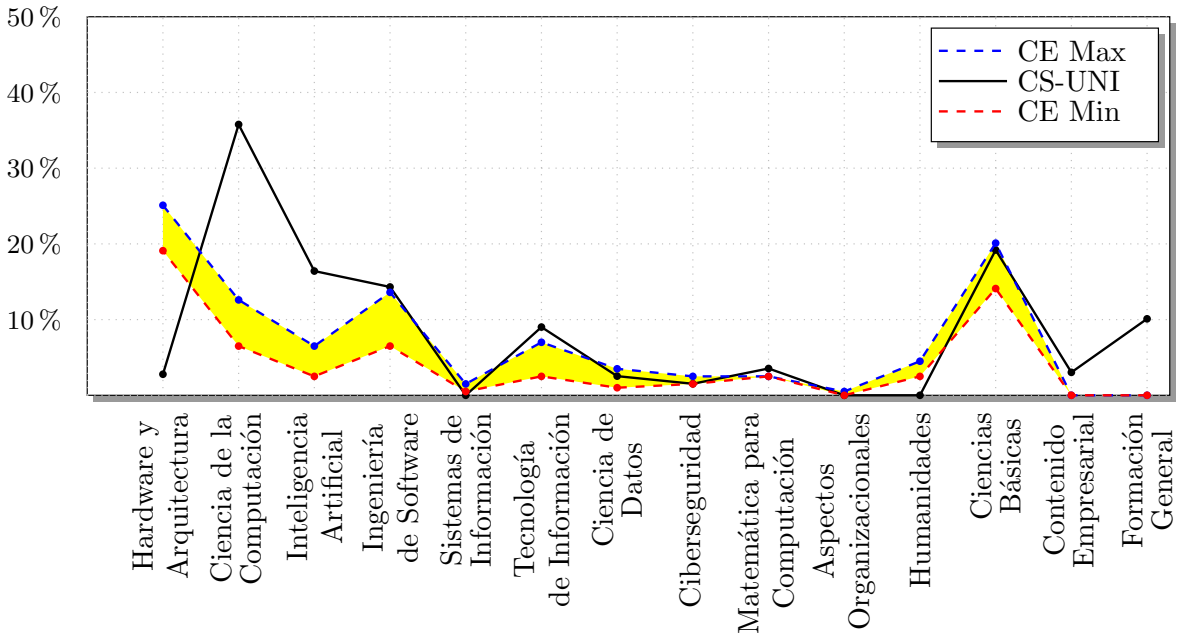

Figura 4.14: Comparación de CS-UNI con CE de ACM/IEEE-CS (Gráfico de curvas).

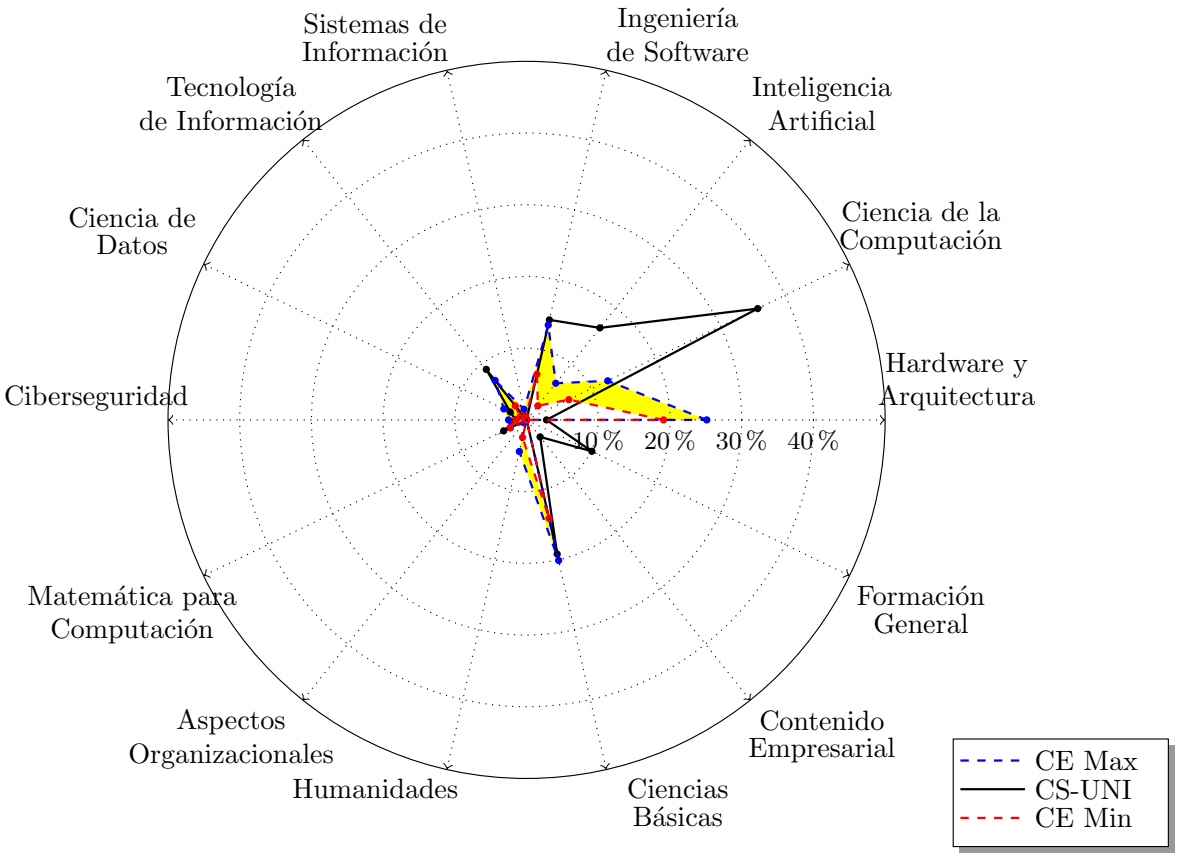

Figura 4.15: Comparación de CS-UNI con CE de ACM/IEEE-CS (Gráfico de araña).

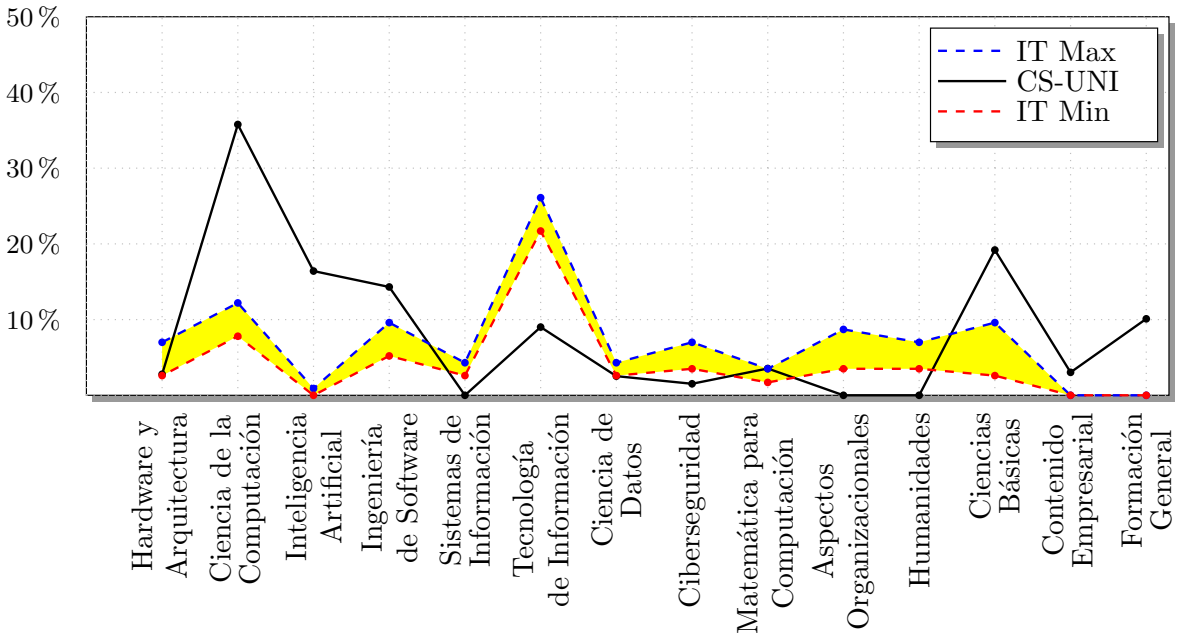

Figura 4.16: Comparación de CS-UNI con IT de ACM/IEEE-CS (Gráfico de curvas).

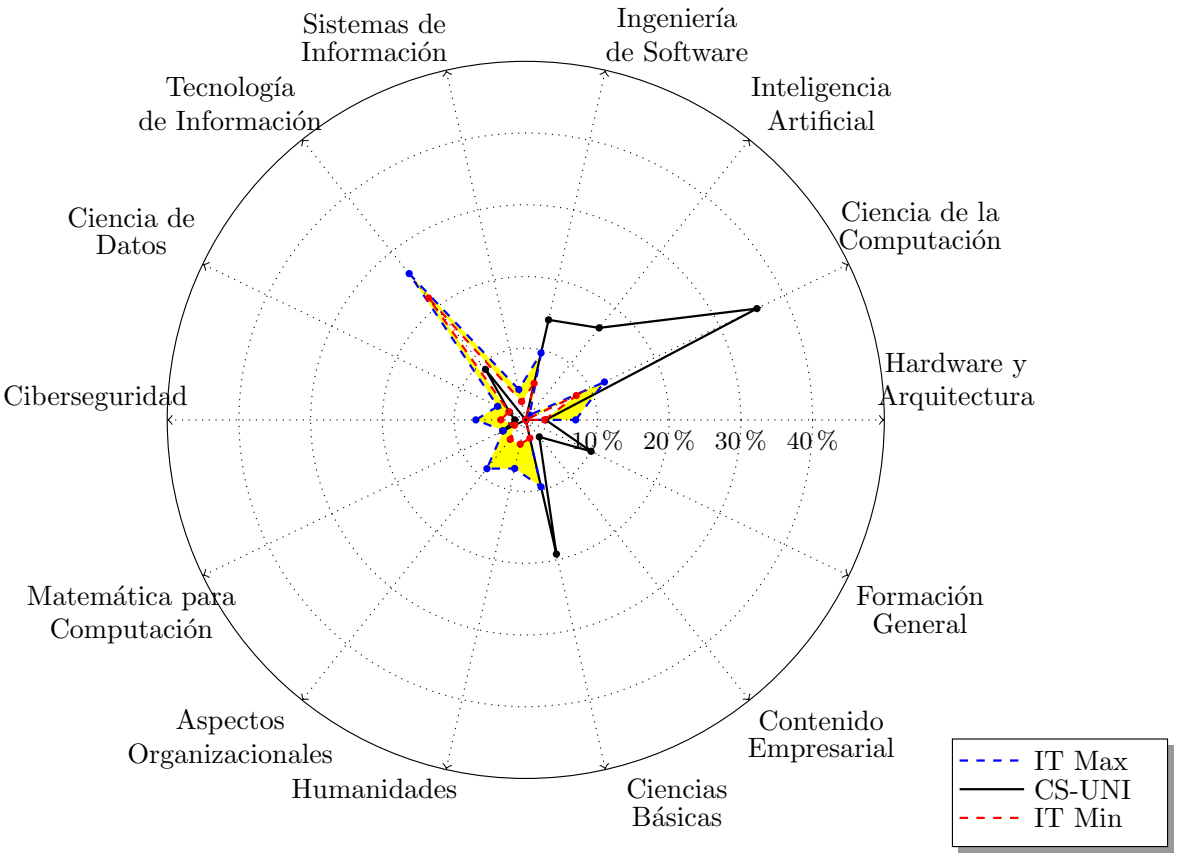

Figura 4.17: Comparación de CS-UNI con IT de ACM/IEEE-CS (Gráfico de araña).

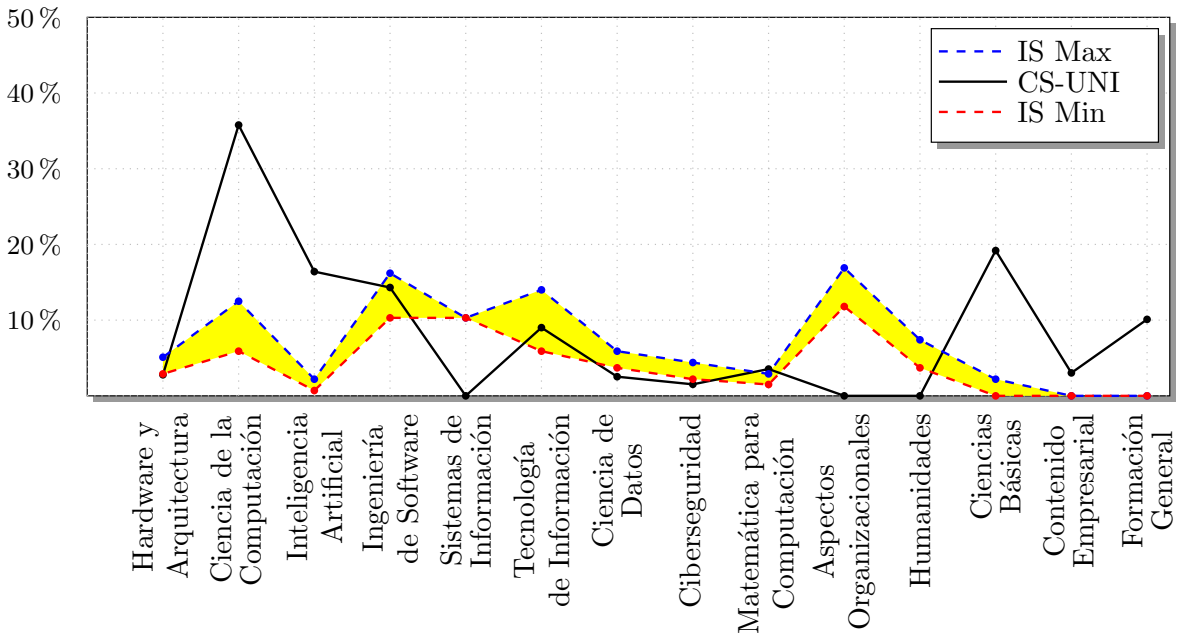

Figura 4.18: Comparación de CS-UNI con IS de ACM/IEEE-CS (Gráfico de curvas).

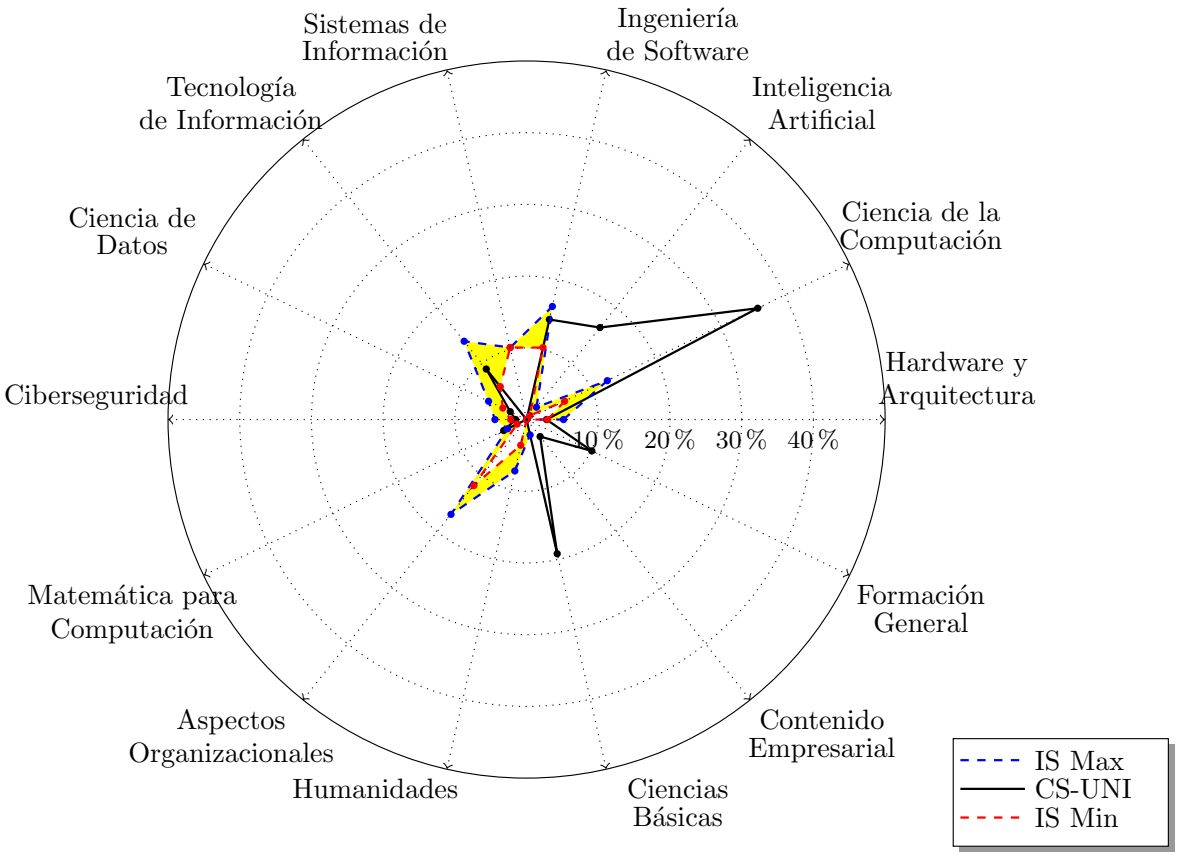

Figura 4.19: Comparación de CS-UNI con IS de ACM/IEEE-CS (Gráfico de araña).

Referencias Bibliográficas

[ACM and IEEE-CS, 2017] ACM and IEEE-CS (2017). Information technology curricula 2017. Technical report, ACM and IEEE-CS.

[ACM Data Science Task Force, 2021] ACM Data Science Task Force (2021). Computing competencies for undergraduate data science curricula. Technical report, Association for Computing Machinery (ACM).

[ACM/IEEE-CS, 2020] ACM/IEEE-CS (2020). Computing curricula 2020. Technical report, ACM Press and IEEE Computer Society Press.

[ACM/IEEE-CS Joint Task Force on Cybersecurity Education, 2017] ACM/IEEE-CS Joint Task Force on Cybersecurity Education (2017). Cybersecurity curricula 2017. Technical report, ACM Press and IEEE Computer Society Press and Association for Information Systems Special Interest Group on Information Security and Privacy (AIS SIGSEC) and International Federation for Information Processing Technical Committee on Information Security Education (IFIP WG 11.8).

[ACM/IEEE-CS/AAAI Joint Task Force, 2023] ACM/IEEE-CS/AAAI Joint Task Force (2023). Cs2023: Acm/ieee-cs/aaai computer science curricula. Technical report, ACM Press and IEEE Computer Society Press and AAAI Press.

[Díaz-Herrera and Hilburn, 2004] Díaz-Herrera, J. L. and Hilburn, T. B. (2004). Software engineering: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. Technical report, ACM, IEEE.

[LeBlanc et al., 2015] LeBlanc, R., Sobel, A., Ben-Menachem, M., Lethbridge, T. C., Díaz-Herrera, J. L., Hilburn, T. B., McGettrick, A., Atlee, J. M., Hawthorne, E. K., Leaney, J., Budgen, D., Matsumoto, Y., Thompson, J. B., Ardis, M., Hislop, G., Offutt, J., Sebern, M., and Visser, W. (2015). Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. Technical report, IEEE Computer Society, ACM.

[Shakelford et al., 2005] Shakelford, R., Cross, J. H., Davies, G., Impagliazzo, J., Kamali, R., LeBlanc, R., Lunt, B., McGettrick, A., Sloan, R., and Topi, H. (2005). Computing curricula 2005. Technical report, ACM/IEEE.

[Soldan et al., 2016] Soldan, D., Aylor, J., Clements, A., Engel, G., Hoelzeman, R., Hughes, E. A., Hughes, J. L., Impagliazzo, J., Jaeger, R. C., Klenke, R., Lyon, D. A., McGettrick, A., Nelson, V. P., Neebel, D. J., Page, I., Peterson, G. D., Ranganathan, N., Sloan, R., Srimani, P. K., Theys, M. D., Wolf, W., and Varanasi, M. (2016). Computer engineering curricula 2016. Technical report, ACM and IEEE-CS.

[The Joint ACM/AIS IS2020 Task Force, 2020] The Joint ACM/AIS IS2020 Task Force (2020). A competency model for undergraduate programs in information systems. Technical report, ACM Press and AIS Press.

Capítulo 5

Contenido detallado por curso

Primer Semestre

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BIC101 Introducción a la Computación (Obligatorio)

1. Información general

- **Semestre**: 1er Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**: Ninguno

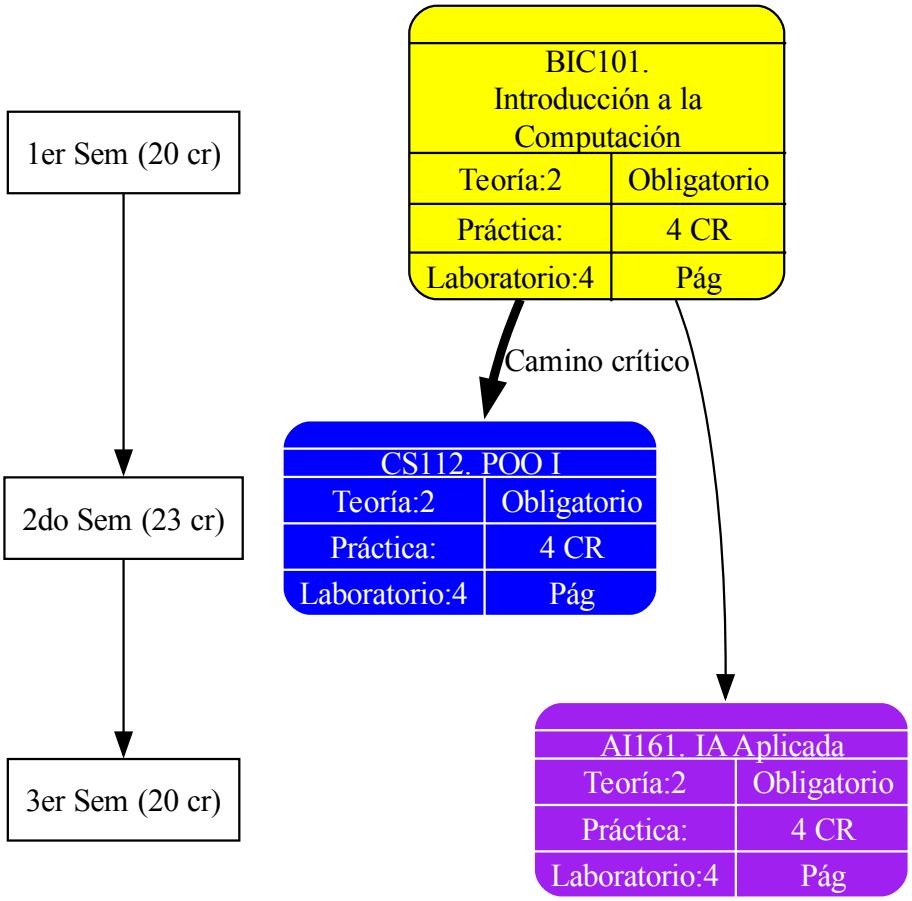

Figura 5.1: Mapa de Conexión. BIC101 Introducción a la Computación

2. Fundamentación

Este es el primer curso de la secuencia de cursos introductorios a la Ciencia de la Computación. Este curso introduce a los participantes a los conceptos fundamentales de programación usando Python. Los temas incluyen tipos de datos, estructuras de control, funciones, listas, recursión, y la mecánica de ejecución, prueba y depuración. La programación es uno de los pilares de la Ciencia de la Computación; cualquier profesional en el área necesitará programar para materializar sus modelos y propuestas.

3. Objetivos del curso.

- Introducir los conceptos fundamentales de programación.
- Desarrollar la capacidad de abstracción usando lenguaje de programación.
- Desarrollar habilidades de resolución de problemas mediante el pensamiento algorítmico.
- Comprender estructuras de datos básicas y sus aplicaciones.
- Dominar el lenguaje de programación Python para la resolución de problemas.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Familiarizarse)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Introducción a la Programación y su Historia, 4 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Breve historia de la computación y los lenguajes de programación. **II:** Visión general del lenguaje de programación Python. **III:** Entornos y herramientas de programación. **IV:** Estructura básica y ejecución de programas. **V:** Escritura, ejecución y depuración de programas simples.

Objetivos de Aprendizaje I: Identificar tendencias importantes en la historia de la computación. [Familiarizarse] **II:** Discutir el contexto histórico de los paradigmas de lenguajes de programación. [Familiarizarse] **III:** Comparar la vida diaria antes y después de las computadoras personales e Internet. [Evaluar] **IV:** Escribir, compilar y ejecutar un programa simple en Python. [Usar]

Bibliografía: [Zelle, 2010, Guttag, 2013, Brookshear and Brylow, 2019]

5.2 Fundamentos de Sistemas de Tipos, 6 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Un tipo como un conjunto de valores junto con un conjunto de operaciones: *a*) Tipos primitivos (por ejemplo, números, booleanos) *b*) Tipos compuestos construidos a partir de otros tipos (por ejemplo, registros/estructuras, uniones, arreglos, listas, funciones, referencias usando operaciones de conjuntos) **II:** Seguridad de tipos y errores causados por usar valores inconsistentemente dados sus tipos pretendidos **III:** Objetivos y limitaciones del tipado estático y dinámico: detectar y eliminar errores lo antes posible. **IV:** Tipos de datos primitivos en Python (int, float, str, bool). **V:** Conversión de tipos y casting.

Objetivos de Aprendizaje I: Describir, tanto para un tipo primitivo como para uno compuesto, los valores que tienen ese tipo [Familiarizarse] **II:** Describir ejemplos de errores de programa detectados por un sistema de tipos [Familiarizarse] **III:** Usar tipos y mensajes de error de tipos para escribir y depurar programas [Usar] **IV:** Explicar el sistema de tipado dinámico de Python. [Familiarizarse]

Bibliografía: [Zelle, 2010, Guttag, 2013]

5.3 Fundamentos de Programación Orientada a Objetos, 14 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Programación imperativa como subconjunto de la programación orientada a objetos. **II:** Definición de clases: campos, métodos y constructores. **III:** Propiedades dinámicas vs estáticas. **IV:** Variables, expresiones y asignaciones. **V:** Operaciones de entrada y salida. **VI:** Sentencias condicionales (if, elif, else). **VII:** Estructuras de iteración (bucles for, bucles while). **VIII:** Bucles anidados y sentencias de control de bucle. **IX:** Fundamentos del manejo de excepciones.

Objetivos de Aprendizaje I: Componer una clase a través del diseño, implementación y prueba para cumplir con los requisitos de comportamiento [Usar] **II:** Predecir y validar el flujo de control en un programa usando despacho dinámico [Evaluar] **III:** Usar mecanismos de encapsulación orientados a objetos como interfaces y miembros privados [Usar] **IV:** Implementar programas usando estructuras condicionales e iterativas. [Usar] **V:** Depurar programas con errores de sintaxis y lógicos. [Usar] **VI:** Diseñar e implementar algoritmos simples usando bucles. [Usar]

Bibliografía: [Zelle, 2010, Guttag, 2013, Brookshear and Brylow, 2019]

5.4 Funciones y paso de parámetros, 10 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Definición e invocación de funciones. **II:** Mecanismos de paso de parámetros (por valor, por referencia). **III:** Valores de retorno y múltiples retornos. **IV:** Alcance y tiempo de vida de variables (local, global, nonlocal). **V:** Fundamentos de recursión. **VI:** Expresiones lambda y funciones de orden superior. **VII:** Funciones como objetos de primera clase. **VIII:** Documentación y prueba de funciones.

Objetivos de Aprendizaje **I:** Definir e implementar funciones con paso de parámetros adecuado. [Usar] **II:** Explicar la diferencia entre paso por valor y paso por referencia. [Familiarizarse] **III:** Implementar soluciones recursivas a problemas simples. [Usar] **IV:** Usar expresiones lambda para construcciones de programación funcional. [Usar] **V:** Diseñar funciones con alcance y documentación apropiados. [Evaluar]

Bibliografía: [Zelle, 2010, Guttag, 2013, Hunt and Thomas, 1999]

5.5 Estructuras de datos básicas, 10 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Listas y operaciones con listas (indexación, segmentación, mutabilidad). **II:** Tuplas e inmutabilidad. **III:** Diccionarios y mapeo clave-valor. **IV:** Conjuntos y operaciones con conjuntos. **V:** Cadenas como secuencias. **VI:** Comprensión de listas y expresiones generadoras. **VII:** Algoritmos básicos de búsqueda y ordenamiento. **VIII:** Introducción a la complejidad algorítmica para operaciones básicas.

Objetivos de Aprendizaje **I:** Implementar programas usando listas, diccionarios y conjuntos. [Usar] **II:** Elegir estructuras de datos apropiadas para problemas dados. [Evaluar] **III:** Explicar las compensaciones entre diferentes estructuras de datos. [Familiarizarse] **IV:** Implementar operaciones básicas de búsqueda y ordenamiento en listas. [Usar]

Bibliografía: [Zelle, 2010, Guttag, 2013]

5.6 Fundamentos de Programación Orientada a Objetos, 10 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Subclases, herencia (incluyendo herencia múltiple) y sobrescritura de métodos. **II:** Despacho dinámico: definición de llamada a método. **III:** Idiomas orientados a objetos para encapsulación: *a*) Privacidad, ocultación de datos y visibilidad de miembros de clase. *b*) Interfaces que revelan solo firmas de métodos. *c*) Clases base abstractas, rasgos (*traits*) y mixins. **IV:** Clases y objetos en Python. **V:** Atributos y métodos. **VI:** Fundamentos de herencia y polimorfismo. **VII:** Encapsulamiento y ocultación de información.

Objetivos de Aprendizaje **I:** Construir una jerarquía de clases simple utilizando subclases que permita reutilizar código para subclases distintas [Usar] **II:** Explicar la relación entre la herencia orientada a objetos (compartición de código y sobrescritura) y el subtipado (la idea de que un subtipo sea usable en un contexto que espera el supertipo) [Familiarizarse] **III:** Comparar y contrastar los beneficios y costos/impacto de usar herencia (subclases) y composición (específicamente, cómo basar la composición en funciones de orden superior) [Evaluar] **IV:** Diseñar e implementar clases simples en Python. [Usar] **V:** Explicar los beneficios de la POO para la organización del código. [Familiarizarse]

Bibliografía: [Zelle, 2010, Guttag, 2013]

5.7 Estrategias Algorítmicas, 6 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Paradigmas: *a*) Fuerza Bruta (por ejemplo, búsqueda lineal, ordenamiento por selección, viajante de comercio, mochila) *b*) Disminuir y Vencer: 1) Por una Constante (por ejemplo, ordenamiento por inserción, ordenamiento topológico) 2) Por un Factor Constante (por ejemplo, búsqueda binaria) 3) Por un Tamaño Variable (por ejemplo, Euclides) *c*) Dividir y Vencer (por ejemplo, búsqueda binaria, quicksort, mergesort, Strassen) *d*) Voraz (por ejemplo, Dijkstra, Kruskal, Mochila) *e*) Transformar y Vencer: 1) Simplificación de instancia (por ejemplo, encontrar duplicados mediante preordenamiento de lista) 2) Cambio de representación (por ejemplo, heapsort) 3) Reducción de problema (por ejemplo, mínimo común múltiplo, programación lineal) 4) Programación dinámica (por ejemplo, Floyd, Marshall, Bellman-Ford) *f*) Compromisos espacio vs tiempo (por ejemplo, hash) **II:** Iteración vs recursión (por ejemplo, factorial, búsqueda en árbol) **III:** Estrategias de descomposición de problemas. **IV:** Técnicas de diseño de algoritmos. **V:** Estrategias de depuración y prueba. **VI:** Seguimiento de la ejecución del programa.

Objetivos de Aprendizaje I: Para cada uno de los paradigmas en esta unidad: *a*) Explicar sus características definitorias *b*) Explicar un ejemplo que demuestre el paradigma incluyendo cómo este ejemplo satisface las características del paradigma. [Familiarizarse] **II:** Dar ejemplos de algoritmos iterativos y recursivos que resuelvan el mismo problema, explicar los beneficios y desventajas de cada enfoque [Usar] **III:** Evaluar si un enfoque voraz conduce a una solución óptima [Evaluar] **IV:** Aplicar técnicas sistemáticas de depuración para localizar y corregir errores. [Usar] **V:** Analizar la complejidad algorítmica para problemas simples. [Familiarizarse]

Bibliografía: [Guttag, 2013, Cormen et al., 2009]

5.8 Métodos Modernos de Desarrollo y Uso de Herramientas, 4 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Uso de entornos de programación modernos (IDEs, notebooks). **II:** Mejores prácticas de búsqueda y reutilización de código. **III:** Uso de bibliotecas estándar y APIs. **IV:** Técnicas de depuración y prueba. **V:** Fundamentos de control de versiones. **VI:** Documentación de código y guías de estilo. **VII:** Entornos virtuales y gestión de paquetes.

Objetivos de Aprendizaje I: Usar herramientas de desarrollo modernas efectivamente. [Usar] **II:** Aplicar principios de reutilización de código mediante bibliotecas y APIs. [Usar] **III:** Depurar y probar programas sistemáticamente. [Usar] **IV:** Seguir estándares de codificación y prácticas de documentación. [Familiarizarse] **V:** Gestionar paquetes y entornos de Python. [Familiarizarse]

Bibliografía: [Zelle, 2010, Hunt and Thomas, 1999]

6. Bibliografía

[Brookshear and Brylow, 2019] Brookshear, J. G. and Brylow, D. (2019). *Computer Science: An Overview*. Pearson, global edition edition.

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, 3rd edition.

[Guttag, 2013] Guttag, J. V. (2013). *Introduction to Computation and Programming Using Python*. MIT Press, 1st edition.

[Hunt and Thomas, 1999] Hunt, A. and Thomas, D. (1999). *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley, 1st edition.

[Zelle, 2010] Zelle, J. M. (2010). *Python Programming: An Introduction to Computer Science*. Franklin, Beedle & Associates Inc, 2nd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BMA101 Algebra Lineal (Obligatorio)

1. Información general

- **Semestre**: 1er Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 3 horas; **Práctica**: 2 horas;
- **Prerrequisitos**: Ninguno

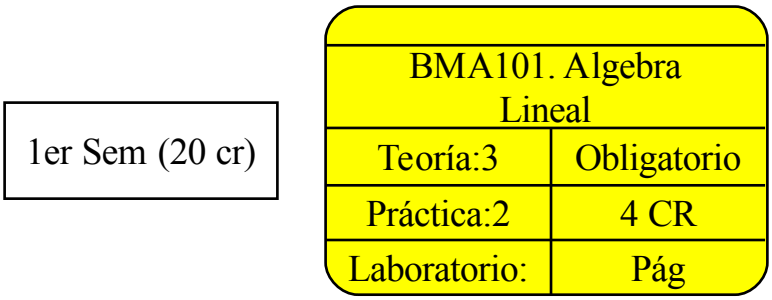

Figura 5.2: Mapa de Conexión. BMA101 Algebra Lineal

2. Fundamentación

El álgebra lineal es una herramienta matemática fundamental compartida por la ciencia de la computación y las disciplinas de ingeniería, proporcionando las bases para el análisis estructural, las redes de circuitos, los sistemas de control, el procesamiento de datos y el diseño de algoritmos. Este curso proporciona una base sólida en los conceptos y técnicas del álgebra lineal elemental y avanzada, con énfasis en su aplicación en contextos científicos y de ingeniería.

3. Objetivos del curso.

- Comprender los conceptos fundamentales del álgebra lineal, incluyendo espacios vectoriales, transformaciones lineales y teoría de matrices.
- Aplicar las técnicas del álgebra lineal para resolver sistemas de ecuaciones y problemas de valores/vectores propios provenientes de contextos científicos y de ingeniería.
- Desarrollar habilidades de razonamiento abstracto y pensamiento lógico para abordar problemas matemáticos.

4. Competencias Generales (*Outcomes*)

AG-C07) Conocimientos de Computación: Aplica conocimientos de matemáticas, ciencias y computación. (Familiarizarse)

5. Contenido

5.1 Matrices y Sistemas de Ecuaciones Lineales, 19 hr(s)

Competencias: AG-C07

Tópicos: **I:** Álgebra de matrices: suma, multiplicación por escalar, producto de matrices, transpuesta, tipos de matrices e inversa de una matriz **II:** Resolución de sistemas de ecuaciones lineales mediante eliminación gaussiana y Gauss-Jordan; forma escalonada, rango y consistencia de sistemas **III:** Determinantes: definición, propiedades, regla de Cramer e interpretación geométrica **IV:** Descomposiciones matriciales: LU, QR y descomposición en valores singulares (SVD)

Objetivos de Aprendizaje **I:** Identificar tipos de matrices y realizar operaciones básicas de álgebra matricial, incluyendo el cálculo de la inversa de una matriz [Familiarizarse] **II:** Resolver sistemas de ecuaciones lineales mediante eliminación gaussiana o Gauss-Jordan, determinando el rango y la consistencia del sistema [Usar] **III:** Aplicar la regla de Cramer y las descomposiciones LU, QR y SVD para resolver sistemas lineales y factorizar matrices [Evaluar]

Bibliografía: [Strang, 2016, Lay et al., 2016]

5.2 Espacios Vectoriales y Transformaciones Lineales, 14 hr(s)

Competencias: AG-C07

Tópicos: **I:** Espacios y subespacios vectoriales; bases, dimensión e independencia lineal **II:** Transformaciones lineales, núcleo e imagen, teorema rango-nulidad y representaciones matriciales

Objetivos de Aprendizaje **I:** Identificar si un conjunto con operaciones dadas forma un espacio vectorial y determinar una base [Familiarizarse] **II:** Determinar el núcleo, la imagen y la representación matricial de una aplicación lineal, verificando el teorema de rango-nulidad [Usar] **III:** Aplicar transformaciones lineales para modelar y resolver problemas de cambio de base y composición de aplicaciones [Evaluar]

Bibliografía: [Strang, 2016, Lay et al., 2016]

5.3 Valores Propios, Producto Interno y Formas Cuadráticas, 15 hr(s)

Competencias: AG-C07

Tópicos: **I:** Valores y vectores propios; diagonalización y el teorema espectral **II:** Espacios con producto interno, ortogonalidad, proceso de Gram-Schmidt y mínimos cuadrados **III:** Formas cuadráticas y definitud: clasificación en positiva, negativa e indefinida mediante valores propios

Objetivos de Aprendizaje **I:** Identificar los valores propios y vectores propios de una matriz y determinar si es diagonalizable [Familiarizarse] **II:** Calcular la diagonalización de matrices mediante el teorema espectral y aplicar el proceso de Gram-Schmidt para obtener bases ortogonales [Usar] **III:** Aplicar los métodos de mínimos cuadrados y la clasificación de formas cuadráticas mediante valores propios para resolver problemas de aproximación y optimización [Evaluar]

Bibliografía: [Strang, 2016, Lay et al., 2016]

6. Bibliografía

[Lay et al., 2016] Lay, D. C., Lay, S. R., and McDonald, J. J. (2016). *Linear Algebra and Its Applications*. Pearson.

[Strang, 2016] Strang, G. (2016). *Introduction to Linear Algebra*. Wellesley-Cambridge Press.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BMA102 Cálculo Diferencial (Obligatorio)

1. Información general

- **Semestre**: 1er Sem. **Créditos**: 5
- **Horas del curso**: **Teoría**: 4 horas; **Práctica**: 2 horas;
- **Prerrequisitos**: Ninguno

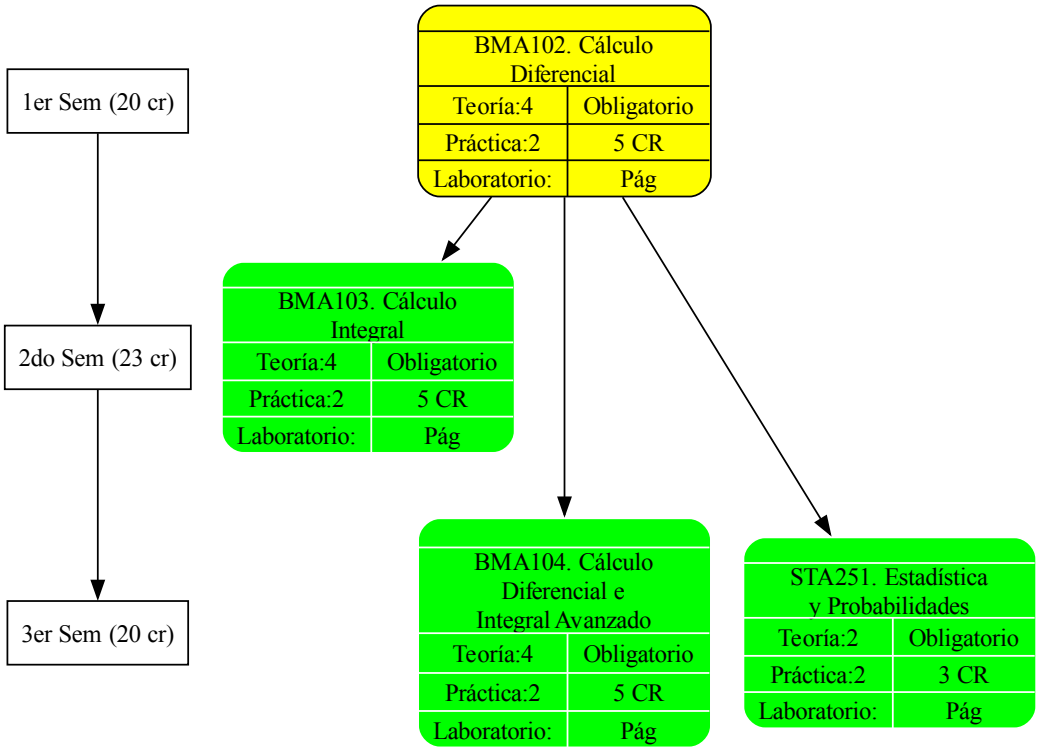

Figura 5.3: Mapa de Conexión. BMA102 Cálculo Diferencial

2. Fundamentación

El cálculo diferencial es una herramienta matemática fundamental compartida por la ciencia de la computación y las disciplinas de ingeniería, proporcionando las bases para el estudio riguroso del cambio, el análisis de razones de cambio, la optimización de sistemas y funciones, y el modelado de fenómenos continuos. Este curso proporciona una base sólida en los conceptos y técnicas del cálculo diferencial de una variable, con énfasis en su aplicación en contextos científicos y de ingeniería.

3. Objetivos del curso.

- Comprender rigurosamente los conceptos de límite, continuidad y derivada de funciones de una variable.
- Aplicar las reglas de diferenciación para calcular derivadas de funciones algebraicas, compuestas e implícitas provenientes de contextos científicos y de ingeniería.
- Utilizar el cálculo diferencial para resolver problemas de optimización, razones de cambio relacionadas y análisis del comportamiento de funciones.

4. Competencias Generales (*Outcomes*)

AG-C07) Conocimientos de Computación: Aplica conocimientos de matemáticas, ciencias y computación. (Familiarizarse)

5. Contenido

5.1 Límites y Continuidad, 12 hr(s)

Competencias: AG-C07

Tópicos: **I:** Definición épsilon-delta de un límite y técnicas de demostración formal **II:** Leyes de los límites, límites laterales y límites en el infinito **III:** Continuidad: tipos, propiedades y el Teorema del Valor Intermedio **IV:** Regla de L'Hôpital y formas indeterminadas **V:** Límites de sucesiones: convergencia, Teorema del Sándwich y Bolzano-Weierstrass

Objetivos de Aprendizaje **I:** Enunciar la definición épsilon-delta de un límite y explicar su significado geométrico [Familiarizarse] **II:** Evaluar límites usando leyes de límites, la regla de L'Hôpital y análisis asintótico [Usar] **III:** Demostrar continuidad o discontinuidad de una función y aplicar el Teorema del Valor Intermedio [Evaluar]

Bibliografía: [Stewart, 2015, Larson and Edwards, 2014]

5.2 Cálculo Diferencial, 36 hr(s)

Competencias: AG-C07

Tópicos: **I:** La derivada como límite; interpretaciones geométricas y físicas; diferenciabilidad **II:** Reglas del producto, cociente y cadena; diferenciación implícita; tasas relacionadas **III:** Teorema de Rolle, Teorema del Valor Medio y criterios de monotonía **IV:** Derivadas de orden superior, polinomios de Taylor y estimación del residuo **V:** Extremos: pruebas de la primera y segunda derivada, trazado de curvas y optimización aplicada

Objetivos de Aprendizaje **I:** Interpretar la derivada como tasa de cambio y como pendiente de la recta tangente [Familiarizarse] **II:** Aplicar las reglas de diferenciación y la diferenciación implícita para calcular derivadas de funciones compuestas e implícitas [Usar] **III:** Resolver problemas de optimización y trazar curvas usando el Teorema del Valor Medio y las pruebas de derivadas [Evaluar]

Bibliografía: [Stewart, 2015, Larson and Edwards, 2014]

5.3 Ecuaciones Paramétricas y Coordenadas Polares, 6 hr(s)

Competencias: AG-C07

Tópicos: **I:** Curvas paramétricas y su cálculo diferencial, aplicadas a trayectorias en cinemática **II:** Coordenadas polares y su conversión a coordenadas cartesianas

Objetivos de Aprendizaje **I:** Calcular la velocidad y aceleración de una trayectoria descrita paramétricamente [Usar] **II:** Convertir entre coordenadas polares y cartesianas al modelar geometrías de ingeniería [Usar]

Bibliografía: [Stewart, 2016, Thomas et al., 2014]

6. Bibliografía

[Larson and Edwards, 2014] Larson, R. and Edwards, B. H. (2014). *Calculus*. Cengage Learning.

[Stewart, 2015] Stewart, J. (2015). *Calculus: Early Transcendentals*. Cengage Learning.

[Stewart, 2016] Stewart, J. (2016). *Cálculo de una variable: Trascendentes tempranas*. Cengage Learning, 8th edition.

[Thomas et al., 2014] Thomas, G. B., Weir, M. D., and Hass, J. (2014). *Cálculo de una variable*. Pearson, 13th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BCH101 Química I (Obligatorio)

1. Información general

- **Semestre**: 1er Sem. **Créditos**: 5
- **Horas del curso**: **Teoría**: 4 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**: Ninguno

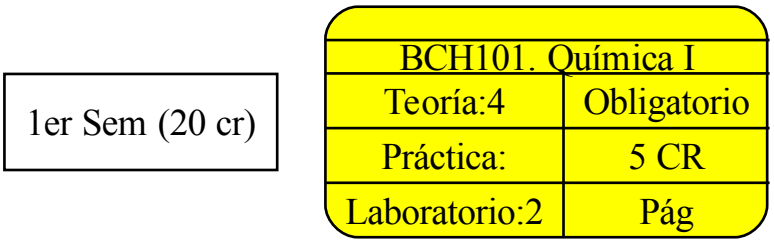

Figura 5.4: Mapa de Conexión. BCH101 Química I

2. Fundamentación

La química proporciona la base para comprender la composición, estructura y propiedades de la materia – conocimiento requerido en los programas de computación e ingeniería por igual, como parte de una sólida formación en ciencias naturales. Este curso introduce los principios básicos de la química general y los extiende al equilibrio químico y la electroquímica, junto con la química aplicada del agua, los polímeros y el cemento, temas que reaparecen en la ciencia de materiales, el análisis ambiental y la práctica de la ingeniería.

3. Objetivos del curso.

- Comprender la estructura de la materia a nivel atómico y molecular.
- Aplicar los principios de estequiometría para realizar cálculos químicos.
- Comprender los diferentes tipos de enlaces químicos y su influencia en las propiedades de las sustancias.
- Aplicar el equilibrio químico, incluidos los equilibrios ácido-base y de solubilidad, para predecir el comportamiento de los sistemas acuosos.
- Aplicar principios electroquímicos para analizar la corrosión de metales y del concreto armado.
- Relacionar la composición y propiedades del agua, los polímeros y el cemento con sus aplicaciones en la ciencia de materiales y en la ingeniería civil.

4. Competencias Generales (*Outcomes*)

AG-C07) Conocimientos de Computación: Aplica conocimientos de matemáticas, ciencias y computación. (Familiarizarse)

5. Contenido

5.1 Materia y Energía, 8 hr(s)

Competencias: AG-C07

Tópicos: **I:** Clasificación de la materia: elementos, compuestos, mezclas homogéneas y heterogéneas **II:** Propiedades físicas y químicas, y cambios físicos y químicos de la materia **III:** Sistema Internacional de unidades, cifras significativas y análisis dimensional **IV:** Formas de energía, calor y temperatura; ley de conservación de la masa y la energía **V:** El método científico aplicado a la química: observación, hipótesis y experimentación

Objetivos de Aprendizaje **I:** Clasificar una muestra de materia como elemento, compuesto o mezcla, y distinguir mezclas homogéneas de heterogéneas [Familiarizarse] **II:** Aplicar el análisis dimensional y las reglas de cifras significativas en cálculos de mediciones químicas [Usar] **III:** Relacionar los cambios de energía observados con procesos físicos y químicos específicos, aplicando la ley de conservación de la energía [Evaluar]

Bibliografía: [Brown et al., 2017, Chang and Goldsby, 2016]

5.2 Fundamentos de la Estructura Atómica, 10 hr(s)

Competencias: AG-C07

Tópicos: **I:** Evolución histórica del modelo atómico: Dalton, Thomson, Rutherford y Bohr **II:** Partículas subatómicas, número atómico, número de masa e isótopos **III:** Configuración electrónica, números cuánticos y el principio de Aufbau **IV:** Organización de la tabla periódica: grupos, periodos y bloques **V:** Tendencias periódicas: radio atómico, energía de ionización, afinidad electrónica y electronegatividad

Objetivos de Aprendizaje **I:** Describir la evolución histórica del modelo atómico y las contribuciones de cada modelo a la comprensión actual del átomo [Familiarizarse] **II:** Determinar la configuración electrónica de un átomo o ion a partir de su posición en la tabla periódica [Usar] **III:** Predecir tendencias periódicas (radio atómico, energía de ionización) para comparar dos o más elementos [Evaluar]

Bibliografía: [Brown et al., 2017, Chang and Goldsby, 2016]

5.3 Enlace Químico, 10 hr(s)

Competencias: AG-C07

Tópicos: **I:** Enlace iónico: formación, energía reticular y propiedades de los compuestos iónicos **II:** Enlace covalente, estructuras de Lewis y el concepto de resonancia **III:** Geometría molecular mediante la teoría de repulsión de pares de electrones de valencia (RPECV) **IV:** Electronegatividad, polaridad de enlaces y polaridad molecular **V:** Fuerzas intermoleculares: dipolo-dipolo, puente de hidrógeno y fuerzas de dispersión de London

Objetivos de Aprendizaje **I:** Dibujar estructuras de Lewis correctas, incluyendo casos de resonancia, para moléculas e iones poliatómicos [Usar] **II:** Predecir la geometría molecular y la polaridad de una molécula aplicando la teoría RPECV [Usar] **III:** Relacionar el tipo de fuerza intermolecular presente en una sustancia con sus propiedades físicas observadas (punto de ebullición, solubilidad) [Evaluar]

Bibliografía: [Brown et al., 2017, Chang and Goldsby, 2016]

5.4 Estequiometría, 10 hr(s)

Competencias: AG-C07

Tópicos: **I:** Fórmulas químicas, masa molar y el concepto de mol **II:** Balanceo de ecuaciones químicas y su interpretación cuantitativa **III:** Cálculos estequiométricos mol-mol, masa-masa y masa-mol **IV:** Reactivo limitante, reactivo en exceso y rendimiento porcentual **V:** Molaridad, preparación de disoluciones y estequiometría de disoluciones

Objetivos de Aprendizaje I: Balancear ecuaciones químicas por el método de tanteo, verificando la conservación de la masa [Familiarizarse] **II:** Calcular las cantidades de reactivos y productos en una reacción química usando relaciones molares [Usar] **III:** Determinar el reactivo limitante y el rendimiento porcentual de una reacción a partir de datos experimentales [Evaluar]

Bibliografía: [Brown et al., 2017, Chang and Goldsby, 2016]

5.5 Estados de la Materia, 10 hr(s)

Competencias: AG-C07

Tópicos: I: Leyes de los gases: Boyle, Charles, Avogadro y la ecuación del gas ideal **II:** Teoría cinético-molecular de los gases y sus aplicaciones **III:** Cambios de fase, curvas de calentamiento y diagramas de fase **IV:** Sólidos cristalinos y amorfos; tipos de sólidos cristalinos **V:** Propiedades coligativas de las disoluciones: disminución de la presión de vapor, elevación ebulloscópica, descenso crioscópico y presión osmótica

Objetivos de Aprendizaje I: Aplicar la ecuación del gas ideal y las leyes de los gases para resolver problemas de presión, volumen y temperatura [Usar] **II:** Interpretar un diagrama de fase para identificar los estados y transiciones de una sustancia a distintas condiciones [Familiarizarse] **III:** Calcular las propiedades coligativas de una disolución (presión de vapor, punto de ebullición, punto de congelación) a partir de su concentración [Evaluar]

Bibliografía: [Brown et al., 2017, Chang and Goldsby, 2016]

5.6 Química Ambiental del Agua, 4 hr(s)

Competencias: AG-C07

Tópicos: I: Composición química del agua natural y parámetros de calidad (pH, dureza, oxígeno disuelto, DBO) **II:** Fundamentos de tratamiento de agua: coagulación, floculación y desinfección **III:** Principales contaminantes del aire y del agua y sus fuentes **IV:** Ciclos biogeoquímicos del carbono, nitrógeno y fósforo **V:** Química atmosférica y el efecto invernadero

Objetivos de Aprendizaje I: Interpretar parámetros de calidad del agua para evaluar su aptitud de uso [Usar] **II:** Describir los procesos básicos de tratamiento de agua potable y residual [Familiarizarse] **III:** Identificar las principales fuentes y efectos de los contaminantes del aire y del agua [Familiarizarse] **IV:** Explicar el mecanismo químico del efecto invernadero [Usar]

Bibliografía: [Brown et al., 2017, Chang and Goldsby, 2016]

5.7 Polímeros y Geosintéticos, 4 hr(s)

Competencias: AG-C07

Tópicos: I: Clasificación de polímeros: termoplásticos, termoestables y elastómeros **II:** Reacciones de polimerización: adición y condensación **III:** Propiedades mecánicas y químicas de los polímeros relevantes para ingeniería **IV:** Geosintéticos: geotextiles, geomembranas y geomallas, y sus aplicaciones **V:** Degradación y durabilidad de polímeros en condiciones ambientales

Objetivos de Aprendizaje I: Clasificar un polímero según su estructura y comportamiento térmico [Usar] **II:** Identificar si una reacción de polimerización es por adición o por condensación [Familiarizarse] **III:** Seleccionar el tipo de geosintético adecuado según la aplicación de ingeniería civil requerida [Usar] **IV:** Evaluar la durabilidad de un polímero frente a la exposición ambiental [Evaluar]

Bibliografía: [Chang and Goldsby, 2016, Mamlouk and Zaniewski, 2017]

5.8 Equilibrio Químico, 4 hr(s)

Competencias: AG-C07

Tópicos: **I:** La constante de equilibrio K_c y K_p, y su relación con la estequiometría de la reacción **II:** El cociente de reacción Q y la predicción del sentido de una reacción **III:** Principio de Le Chatelier: efecto de la concentración, la presión y la temperatura **IV:** Tablas ICE para el cálculo de concentraciones en el equilibrio

Objetivos de Aprendizaje **I:** Escribir la expresión de la constante de equilibrio para una reacción dada [Usar] **II:** Calcular concentraciones en el equilibrio usando tablas ICE [Usar] **III:** Predecir el desplazamiento de un equilibrio ante cambios de concentración, presión o temperatura usando el principio de Le Chatelier [Evaluar]

Bibliografía: [Brown et al., 2017, Chang and Goldsby, 2016]

5.9 Equilibrio Ácido-Base, 12 hr(s)

Competencias: AG-C07

Tópicos: **I:** Teorías ácido-base de Arrhenius, Brønsted-Lowry y Lewis **II:** Cálculo de pH y pOH para ácidos y bases fuertes **III:** Equilibrio de ácidos y bases débiles: K_a, K_b y grado de disociación **IV:** Hidrólisis de sales y predicción del carácter ácido, básico o neutro de una disolución **V:** Disoluciones amortiguadoras (buffer) y la ecuación de Henderson-Hasselbalch

Objetivos de Aprendizaje **I:** Calcular el pH de disoluciones de ácidos y bases fuertes y débiles [Usar] **II:** Predecir el carácter ácido-base de una disolución de una sal a partir de la hidrólisis de sus iones [Evaluar] **III:** Calcular el pH de una disolución amortiguadora usando la ecuación de Henderson-Hasselbalch [Usar]

Bibliografía: [Brown et al., 2017, Chang and Goldsby, 2016]

5.10 Equilibrio de Solubilidad, 6 hr(s)

Competencias: AG-C07

Tópicos: **I:** El producto de solubilidad K_{sp} y su relación con la solubilidad molar **II:** Predicción de precipitación mediante la comparación del cociente de solubilidad con K_{sp} **III:** Efecto del ion común sobre la solubilidad

Objetivos de Aprendizaje **I:** Calcular K_{sp} a partir de la solubilidad molar de una sal, y viceversa [Usar] **II:** Predecir si se formará un precipitado al mezclar dos disoluciones dadas sus concentraciones y el K_{sp} correspondiente [Evaluar]

Bibliografía: [Brown et al., 2017, Chang and Goldsby, 2016]

5.11 Electroquímica, 8 hr(s)

Competencias: AG-C07

Tópicos: **I:** Reacciones de oxidación-reducción y balanceo por el método del ion-electrón **II:** Celdas galvánicas, potenciales estándar de electrodo y la ecuación de Nernst **III:** Celdas electrolíticas y electrólisis; leyes de Faraday **IV:** Corrosión electroquímica de metales y métodos de protección **V:** Corrosión del acero de refuerzo en concreto armado

Objetivos de Aprendizaje **I:** Balancear ecuaciones redox por el método del ion-electrón en medio ácido o básico [Usar] **II:** Calcular el potencial de una celda galvánica y predecir la espontaneidad de una reacción redox [Usar] **III:** Aplicar las leyes de Faraday para calcular la masa depositada o el tiempo en un proceso de electrólisis [Usar] **IV:** Analizar las condiciones que favorecen la corrosión de un metal o del acero de refuerzo en concreto, y proponer métodos de protección [Evaluar]

Bibliografía: [Brown et al., 2017, Chang and Goldsby, 2016]

5.12 Química del Cemento, 4 hr(s)

Competencias: AG-C07

Tópicos: I: Composición química del cemento Portland: silicatos, aluminatos y ferroaluminatos de calcio **II:** Reacciones de hidratación del cemento y desarrollo de resistencia **III:** Rol de cada compuesto (C_3S, C_2S, C_3A, C_4AF) en las propiedades del cemento **IV:** Materiales cementantes suplementarios: puzolanas, escoria y cenizas volantes

Objetivos de Aprendizaje I: Describir la composición química del cemento Portland y sus principales compuestos [Familiarizarse] **II:** Explicar el proceso de hidratación del cemento y su relación con el desarrollo de resistencia [Usar] **III:** Evaluar el efecto de materiales cementantes suplementarios sobre las propiedades del concreto [Evaluar]

Bibliografía: [Mamlouk and Zaniewski, 2017, Chang and Goldsby, 2016]

6. Bibliografía

[Brown et al., 2017] Brown, T. L., Jr., H. E. L., Bursten, B. E., Murphy, C. J., and Woodward, P. M. (2017). *Chemistry: The Central Science*. Pearson.

[Chang and Goldsby, 2016] Chang, R. and Goldsby, K. A. (2016). *Chemistry*. McGraw-Hill Education.

[Mamlouk and Zaniewski, 2017] Mamlouk, M. S. and Zaniewski, J. P. (2017). *Materials for Civil and Construction Engineers*. Pearson, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BEI101 Inglés I (Obligatorio)

1. Información general

- **Semestre**: 1er Sem. **Créditos**: 2
- **Horas del curso**: **Práctica**: 4 horas;
- **Prerrequisitos**: Ninguno

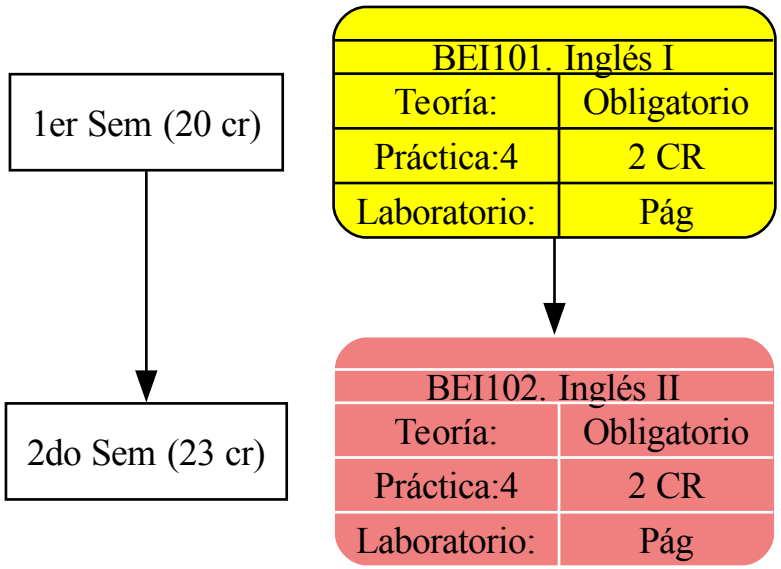

Figura 5.5: Mapa de Conexión. BEI101 Inglés I

2. Fundamentación

El inglés es el idioma principal de la ciencia, la ingeniería y la tecnología en el mundo. Para los estudiantes de computación e ingeniería, dominarlo significa acceder directamente a documentación técnica, artículos de investigación y colaboración profesional sin barreras de traducción. Este curso desarrolla las habilidades fundamentales del inglés utilizando contextos extraídos de sistemas computacionales, hardware y el trabajo cotidiano de ingenieros y científicos.

3. Objetivos del curso.

- Adquirir estructuras gramaticales básicas del inglés aplicadas a contextos de computación e ingeniería.
- Identificar y utilizar vocabulario técnico relacionado con hardware, software y conceptos científicos.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Familiarizarse)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Usar)

5. Contenido

5.1 Bienvenido a la Computación, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Verbo To Be: Afirmativo, Negativo e Interrogativo. **II:** Pronombres personales y adjetivos posesivos. **III:** Expresiones numéricas (1-100): Puertos, direcciones y fundamentos binarios. **IV:** Expresiones sociales en entornos académicos y de laboratorio.

Objetivos de Aprendizaje **I:** Al finalizar la unidad, el estudiante utiliza el verbo 'to be' para presentarse como estudiante de computación y describir propiedades básicas de sistemas computacionales (e.g., "The processor is fast. It is a 64-bit CPU.").

Bibliografía: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.2 Partes y Sistemas, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Posesivo 's y sustantivos plurales aplicados a componentes (the CPU's speed, the system's memory). **II:** Componentes de hardware y sus relaciones. **III:** Estructuras Adjetivo + Sustantivo para descripciones técnicas. **IV:** Interacciones en el laboratorio: Preguntar sobre equipos y herramientas.

Objetivos de Aprendizaje **I:** El estudiante describe componentes de hardware y sus relaciones utilizando posesivos y frases nominales (e.g., "the motherboard's slots", "the server's processors").

Bibliografía: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.3 Cómo Funciona, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Presente Simple I (Tercera persona: He/She/It) para describir procesos. **II:** Vocabulario: Cómo operan programas, algoritmos y dispositivos. **III:** Expresar la hora e interpretar relojes y horarios de sistemas. **IV:** Verbos de operación: runs, compiles, executes, stores, transmits.

Objetivos de Aprendizaje **I:** El estudiante explica cómo funciona un programa o dispositivo utilizando el presente simple en tercera persona (e.g., "The compiler translates source code. It generates an executable file.").

Bibliografía: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.4 Lo que Hacen los Ingenieros, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Presente Simple II (I/You/We/They) para describir rutinas profesionales. **II:** Preguntas y negaciones con 'Do/Does' en contextos técnicos. **III:** Roles en ingeniería: developer, analyst, researcher, systems engineer. **IV:** Adverbios de frecuencia para describir ciclos de trabajo y pruebas.

Objetivos de Aprendizaje **I:** El estudiante describe las tareas diarias y los roles profesionales de ingenieros y científicos computacionales utilizando el presente simple con precisión.

Bibliografía: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.5 Dentro de una Computadora, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Existencial: There is / There are aplicado a descripciones de arquitecturas. **II:** Preposiciones de lugar: componentes en una placa madre, archivos en un directorio. **III:** Vocabulario: memoria, almacenamiento, buses, ranuras, puertos, capas. **IV:** Descripción de topología de red e infraestructura física.

Objetivos de Aprendizaje I: El estudiante describe la organización espacial de componentes de hardware y sistemas de archivos, permitiéndole interpretar y producir descripciones técnicas de arquitecturas.

Bibliografía: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.6 ¿Pueden Pensar las Computadoras?, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Verbo modal Can/Can't para describir capacidades y limitaciones computacionales. **II:** Adverbios de modo aplicados al rendimiento de sistemas. **III:** Pasado del verbo 'To Be' (Was/Were): sistemas computacionales tempranos y pioneros. **IV:** Expresiones básicas para soporte técnico y resolución de problemas.

Objetivos de Aprendizaje I: El estudiante describe lo que los sistemas computacionales pueden y no pueden hacer, e identifica hechos básicos sobre la historia temprana de la computación utilizando estructuras de pasado simple.

Bibliografía: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.7 Historia de la Computación, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Pasado Simple: Verbos regulares e irregulares para narrar la historia tecnológica. **II:** Expresiones de tiempo pasado: invented, developed, published, launched. **III:** Figuras clave: Turing, Babbage, Lovelace, von Neumann. **IV:** Vocabulario: Generaciones de computadoras, hitos, inventos.

Objetivos de Aprendizaje I: El estudiante narra eventos clave en la historia de la computación y la ciencia utilizando el pasado simple correctamente, identificando las contribuciones de figuras fundacionales.

Bibliografía: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

5.8 Mis Metas Profesionales en Computación, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Introducción al inglés técnico para computación e ingeniería. **II:** Lectura y comprensión de especificaciones técnicas simples. **III:** Trabajo en equipos de ingeniería multiculturales: normas de comunicación. **IV:** Presentación oral: "Mi rol en el campo de la computación".

Objetivos de Aprendizaje I: El estudiante presenta sus metas profesionales en computación utilizando vocabulario técnico básico y las relaciona con los roles y responsabilidades de profesionales en software y hardware.

Bibliografía: [Soars and John, 2002d, Cambridge, 2006, MacGrew, 1999]

6. Bibliografía

[Cambridge, 2006] Cambridge (2006). *Diccionario Inglés-Espanol Cambridge*. Editorial Oxford.

[MacGrew, 1999] MacGrew, J. (1999). *Focus on Grammar Basic*. Editorial Oxford.

[Soars and John, 2002d] Soars, L. and John (2002d). *American Headway N 2 Student Book*. Editorial Oxford.

Segundo Semestre

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS100 Introducción a la Ciencia de la Computación (Obligatorio)

1. Información general

- **Semestre**: 2do Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas;
- **Prerrequisitos**: Ninguno

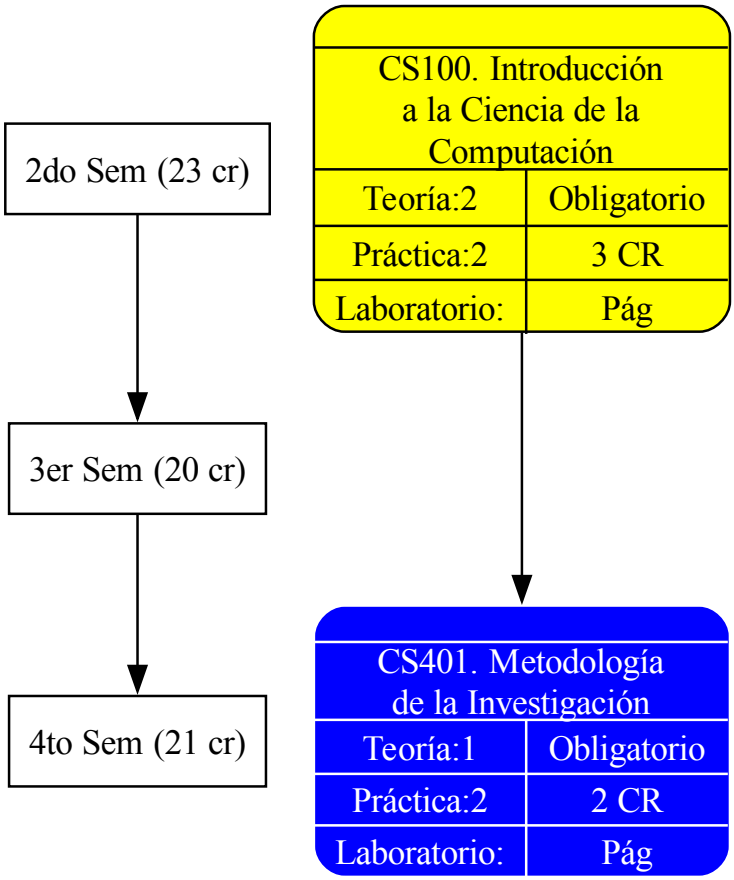

Figura 5.6: Mapa de Conexión. CS100 Introducción a la Ciencia de la Computación

2. Fundamentación

Este curso sirve como base para comprender los conceptos fundamentales del pensamiento computacional aplicables en diversas profesiones.

El curso proporciona, partiendo desde cero, una visión panorámica de: pensamiento computacional introductorio, almacenamiento de datos, arquitectura de computadoras, sistemas operativos, redes e Internet, algoritmos, métodos de ordenamiento, ingeniería de software, bases de datos, estructuras de

datos, ingeniería de software, gráficos por computadora. Además, todos los conceptos son integrados con los nuevos conceptos de la inteligencia artificial.

Diseñado como un curso introductorio a la Ciencia de la Computación, los conceptos se presentan de manera lúdica y utilizando una metodología de Aprendizaje Activo. A lo largo del curso, se fomenta la participación activa de la audiencia, similar a una representación teatral.

Las áreas de conocimiento relacionadas cubiertas están directamente alineadas con los Planes de Estudio de Computación ACM/IEEE-CS.

El curso **no requiere** conocimientos previos en temas de manejo de computadoras y puede ser tomado por estudiantes de cualquier campo.

3. Objetivos del curso.

- Introducir los conceptos fundamentales del Pensamiento Computacional y la Ciencia de la Computación a estudiantes de cualquier formación profesional.
- Desarrollar su capacidad de abstracción.
- Comprender cómo se aplica el Pensamiento Computacional en cada una de sus profesiones.
- Aplicar conceptos avanzados de manera simplificada en cualquier carrera.

4. Competencias Generales (*Outcomes*)

AG-C02) Ética: Aplica principios éticos y se compromete con la ética profesional y las normas de la práctica profesional de la computación. (Familiarizarse)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Familiarizarse)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Familiarizarse)

5. Contenido

5.1 Pensamiento Computacional (Parte I), 4 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: **I:** Explicación del sistema de evaluación. **II:** Instrucciones generales del curso. **III:** Definición de Computación. **IV:** La Computación como un binomio Humano-Computadora. **V:** Distorsiones en la definición de computación. **VI:** La Computación como la automatización de la abstracción. **VII:** Computación e Ingeniería: similitudes y diferencias. **VIII:** Resolución de problemas algorítmicos. **IX:** Dinámica: Comprender la ejecución de un algoritmo a velocidad humana.

Objetivos de Aprendizaje **I:** Aplicar los conceptos fundamentales de la computación en situaciones de la vida real. [Usar] **II:** Identificar distorsiones de la Computación en situaciones de la vida real. [Usar] **III:** Identificar claramente al menos 3 contextos de uso de la palabra .[En]gineer.[en] inglés. [Evaluar] **IV:** Identificar las limitaciones humanas para resolver problemas computacionales. [Usar]

Bibliografía: [Brookshear and Brylow, 2019]

5.2 Pensamiento Computacional (Parte II), 4 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: **I:** Descomposición (Dividir un problema en piezas más pequeñas). **II:** Abstracción (Enfocarse en lo importante). **III:** Reconocimiento de patrones (Identificar secuencias similares basadas en problemas previos). **IV:** Diseño de algoritmos.

Objetivos de Aprendizaje **I:** Aplicar el Pensamiento Computacional a problemas del mundo real. [Usar]

Bibliografía: [Brookshear and Brylow, 2019]

5.3 Conceptos Generales, 4 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: **I:** Numeración binaria vs. decimal. **II:** Representación de caracteres: la tabla ASCII. **III:** Búsqueda binaria. **IV:** Complejidad computacional de un algoritmo.

Objetivos de Aprendizaje **I:** Aplicar varios sistemas de numeración a problemas del mundo real. [Usar] **II:** Comprender la representación interna de caracteres en las tablas ASCII y UTF-8. [Familiarizarse] **III:** Aplicar la estrategia algorítmica de Divide y Vencerás. [Usar] **IV:** Determinar análisis básicos de complejidad algorítmica. [Usar]

Bibliografía: [Brookshear and Brylow, 2019]

5.4 Representación de Datos a Nivel de Máquina, 4 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: **I:** Visión general e historia de la arquitectura de computadoras **II:** Bits, bytes y palabras **III:** Representaciones sin signo, con signo y complemento a dos **IV:** Representación de datos numéricos y bases numéricas: *a*) Punto fijo *b*) Punto flotante **V:** Representación de datos no numéricos **VI:** Representación de registros, arreglos y tipos de datos UTF

Objetivos de Aprendizaje **I:** Discutir por qué todo en las computadoras son datos, incluyendo instrucciones [Debatir] **II:** Explicar cómo las representaciones numéricas de longitud fija pueden afectar la exactitud y precisión [Explicar] **III:** Describir cómo se almacenan los enteros negativos en representaciones de magnitud con signo y complemento a dos [Describir] **IV:** Discutir cómo diferentes formatos pueden representar datos numéricos [Debatir] **V:** Explicar la representación a nivel de bits de datos no numéricos, como caracteres, cadenas, registros y arreglos [Explicar] **VI:** Traducir datos numéricos de un formato a otro [Traducir] **VII:** Describir cómo un sumador único (sin detección de desbordamiento) puede manejar tanto entrada con signo (complemento a dos) como sin signo (binario) sin ”saber” qué formato está usando una entrada dada [Describir]

Bibliografía: [Brookshear and Brylow, 2019]

5.5 Abstracción y Representación de Programas, 2 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: **I:** Programas que toman (otros) programas como entrada, como intérpretes, compiladores, verificadores de tipos, generadores de documentación **II:** Componentes de un lenguaje: *a*) Definiciones de alfabetos, delimitadores, oraciones, sintaxis y semántica *b*) Sintaxis vs semántica **III:** Programa como un conjunto de oraciones significativas no ambiguas **IV:** Abstracciones básicas de programación: constantes, variables, declaraciones (incluyendo declaraciones anidadas), comando, expresión, asignación, selección, iteración definida e indefinida, iteradores, función, procedimiento, módulos, manejo de excepciones **V:** Tipos de variables: estáticas, locales, no locales (*nonlocal*), globales; necesidad y problemas con variables no locales y globales. **VI:** Reglas de ámbito (*scope*): estático vs dinámico; visibilidad de variables; efectos secundarios (*side-effects*). **VII:** Entorno (*environment*) vs almacén (*store*) y sus propiedades **VIII:** Abstracción de datos y de control **IX:** Mecanismos para intercambio de información entre unidades de programa como procedimientos, funciones y módulos: variables no locales, variables globales, paso de parámetros, importación-exportación entre módulos **X:** Representación de instrucciones de bajo nivel como instrucciones de máquina virtual, lenguaje ensamblador y representación binaria

Objetivos de Aprendizaje **I:** Ilustrar el ámbito de variables y visibilidad usando programas simples [Aplicar] **II:** Ilustrar diferentes tipos de paso de parámetros usando un lenguaje de programación pseudo simple [Aplicar] **III:** Explicar el efecto secundario usando variables globales y no locales y cómo corregir tales programas [Explicar] **IV:** Explicar cómo los programas que procesan otros

programas tratan a los otros programas como sus datos de entrada [Explicar] **V:** Describir una gramática y un árbol de sintaxis abstracta para un lenguaje pequeño [Describir]

Bibliografía: [Brookshear and Brylow, 2019]

5.6 Fundamentos de Criptografía, 2 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Basic concepts on Cryptography **II:** Public-keys, Private-keys

Objetivos de Aprendizaje I: Explicar el papel de la criptografía en el soporte de la seguridad y la privacidad [Explicar] **II:** Discutir los riesgos de inventar los propios métodos criptográficos [Debatir] **III:** Discutir la importancia de los números primos en criptografía y explicar su uso en algoritmos criptográficos [Debatir] **IV:** Implementar y criptoanalizar cifrados clásicos [Implementar]

Bibliografía: [Brookshear and Brylow, 2019]

5.7 Gestión de Memoria, 4 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Revisión de memoria física, traducción de direcciones y hardware de gestión de memoria. Ver también: Jerarquía de Memoria, Matemáticas Discretas **II:** Impacto de la jerarquía de memoria incluyendo concepto de caché, búsqueda en caché y almacenamiento en caché por CPU en mecanismos y políticas del sistema operativo Ver también: Jerarquía de Memoria, Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento **III:** Direccionamiento lógico y físico, virtualización de espacio de direcciones. Ver también: Jerarquía de Memoria, Matemáticas Discretas **IV:** Conceptos de paginación, reemplazo de páginas, trashing y asignación de páginas y marcos **V:** Técnicas de asignación/desasignación/almacenamiento (algoritmos y estructura de datos) rendimiento y flexibilidad *a*) Arenas, asignadores de losas (slab allocators), listas libres, clases de tamaño, páginas de tamaño heterogéneo (páginas enormes) **VI:** Almacenamiento en caché de memoria y coherencia de caché y el efecto de vaciar la caché para evitar vulnerabilidades de ejecución especulativaVer también: Organización Funcional, Jerarquía de Memoria, Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento) **VII:** Memoria virtual: aprovechando el hardware de memoria virtual para servicios del SO y eficiencia

Objetivos de Aprendizaje I: Explicar jerarquía de memoria y compensaciones costo-rendimiento [Explicar] **II:** Resumir los principios de memoria virtual aplicados a almacenamiento en caché y paginación [Resumir] **III:** Evaluar las compensaciones en términos de tamaño de memoria (memoria principal, memoria caché, memoria auxiliar) y velocidad del procesador [Evaluar] **IV:** Describir la razón y el uso de la memoria caché (rendimiento y proximidad, cómo las cachés complican el aislamiento y la abstracción de máquina virtual) [Describir]

Bibliografía: [Brookshear and Brylow, 2019]

5.8 Rol y Propósito de los Sistemas Operativos, 2 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Los sistemas operativos median entre el hardware de propósito general y el software específico de la aplicación. **II:** Funciones universales del sistema operativo (por ejemplo, proceso, interfaces de usuario y dispositivos, persistencia de datos) **III:** Influencias de seguridad, redes, multimedia, computación paralela y distribuida

Objetivos de Aprendizaje I: Comprender los objetivos y funciones de los sistemas operativos modernos [Explicar] **II:** Evaluar los problemas de diseño en diferentes escenarios de uso (por ejemplo, sistema operativo en tiempo real, móvil, servidor) [Evaluar] **III:** Comprender cómo la evolución y la estabilidad son deseables y mutuamente antagónicas en las funciones del sistema operativo [Explicar]

Bibliografía: [Brookshear and Brylow, 2019]

5.9 Principios del Sistema Operativo, 2 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Diseño y enfoques de software del sistema operativo (por ejemplo, monolítico, en capas, modular, micro-kernel, unikernel) **II:** Abstracciones, procesos y recursos **III:** La evolución del vínculo entre la arquitectura del hardware y las funciones del sistema operativo **IV:** Aprovechamiento de interrupciones desde el nivel de hardware: rutinas de servicio e implementaciones. Ver también: Organización de Máquina a Nivel de Ensamblador *a*) Interrupciones de temporizador para implementar segmentos de tiempo *b*) Interrupciones de E/S para poner hilos bloqueados en espera sin sondeo

Objetivos de Aprendizaje I: Comprender cómo la aplicación de enfoques de diseño de software al diseño/implementación de sistemas operativos (por ejemplo, en capas, modular, etc.) afecta la robustez y mantenibilidad de un sistema operativo [Explicar] **II:** Categorizar llamadas al sistema por propósito [Categorizar] **III:** Aplicar técnicas de SO para aislamiento, protección y rendimiento a través de las funciones del SO (por ejemplo, similitudes de inanición en planificación de procesos, planificación de solicitudes de disco, semáforos, etc.) y más allá [Aplicar]

Bibliografía: [Brookshear and Brylow, 2019]

5.10 Fundamentos de Redes y Comunicaciones, 4 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Importancia de las redes en la informática contemporánea, y desafíos asociados. Ver también: Equidad Digital y Accesibilidad, Impacto Social e IA, Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad **II:** Organización de Internet (por ejemplo, usuarios, Proveedores de Servicios de Internet, sistemas autónomos, proveedores de contenido, redes de entrega de contenido) **III:** Capas y sus roles (aplicación, transporte, red, enlace de datos y física) **IV:** Elementos de red (por ejemplo, routers, switches, hubs, puntos de acceso y hosts) **V:** Conceptos básicos de colas (por ejemplo, relación con latencia, congestión, niveles de servicio, etc.)

Objetivos de Aprendizaje I: Articular la organización de Internet [Articular] **II:** Listar y definir la terminología de red apropiada [Listar/Enumerar] **III:** Describir la estructura en capas de una arquitectura de red típica [Describir]

Bibliografía: [Brookshear and Brylow, 2019]

5.11 Aplicaciones en Red, 4 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Esquemas de denominación y direccionamiento (por ejemplo, DNS e Identificadores Uniformes de Recursos) **II:** Paradigmas de aplicaciones distribuidas (por ejemplo, cliente/servidor, peer-to-peer, nube, edge y fog). Ver también: Comunicación, Coordinación **III:** Diversidad de demandas de aplicaciones en red (por ejemplo, latencia, ancho de banda y tolerancia a pérdidas). Ver también: Comunicación, Diseño Sostenible y Computación Ubicua, Huella Ambiental de los Sistemas Informáticos, Efectos Sistémicos y Contexto Social, Equidad Digital y Accesibilidad, Impacto Social e IA **IV:** Cobertura de protocolos de capa de aplicación (por ejemplo, HTTP) **V:** Interacciones con APIs de TCP, UDP y Socket. Ver también: Programas: Distribución, Programas: Propiedades de Ejecución, Programación GPU, Programas: Mapeos de Implementación, Programas: Paralelismo Declarativo, Programas: Inicio de Actividades

Objetivos de Aprendizaje I: Definir los principios de denominación, direccionamiento y localización de recursos [Definir] **II:** Analizar las necesidades de demandas específicas de aplicaciones en red [Analizar]

Bibliografía: [Brookshear and Brylow, 2019]

5.12 Marco de Análisis de Complejidad, 2 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: **I:** Marco de Análisis de Complejidad: *a*) Rendimiento de un algoritmo en el mejor caso, caso promedio y peor caso *b*) Mediciones empíricas y relativas (Orden de Crecimiento) *c*) Tamaño de entrada y operaciones primitivas *d*) Eficiencia de tiempo y espacio **II:** Mediciones empíricas de rendimiento **III:** Compromisos tiempo-espacio en algoritmos

Objetivos de Aprendizaje **I:** Preparar una presentación que explique a estudiantes de primer año los conceptos básicos de complejidad algorítmica incluyendo comportamiento de algoritmo en mejor caso, caso promedio y peor caso, notaciones Big-O, Omega y Theta, clases de complejidad, compromisos tiempo-espacio, medición empírica e impacto en problemas prácticos [Explicar] **II:** Para cada algoritmo en la unidad Estructuras de Datos Fundamentales, Algoritmos Fundamentales, Algoritmos Avanzados, explicar su clase de complejidad de tiempo de ejecución y por qué pertenece a esta clase [Explicar] **III:** Evaluar informalmente la clase de complejidad fundamental de algoritmos simples [Evaluar] **IV:** Desarrollar estudios empíricos para determinar y validar hipótesis sobre la complejidad de tiempo de ejecución de varios algoritmos ejecutando algoritmos con entradas de varios tamaños y comparando el rendimiento real con el análisis teórico [Crear] **V:** Explicar ejemplos que ilustren los compromisos tiempo-espacio de algoritmos [Explicar] **VI:** Explicar cómo el balance del árbol afecta la eficiencia de las operaciones de árbol de búsqueda binaria [Explicar]

Bibliografía: [Brookshear and Brylow, 2019]

5.13 Notación Asintótica y Clases de Complejidad, 1 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: **I:** Análisis de complejidad asintótica (cotas promedio y peor caso): *a*) Notaciones formales Big-O, Big-Omega y Big-Theta *b*) Clases de Complejidad Fundamentales y Ejemplos/Problemas Representativos: 1) $O(1)$ Constante (por ejemplo, acceso a arreglo) 2) $O(\log_2 n)$ Logarítmica (por ejemplo, búsqueda binaria) 3) $O(n)$ Lineal (por ejemplo, búsqueda lineal) 4) $O(n \log_2 n)$ Log Lineal (por ejemplo, mergesort) 5) $O(n^2)$ Cuadrática (por ejemplo, ordenamiento por selección) 6) $O(n^c)$ Polinomial (por ejemplo, $O(n^3)$ eliminación gaussiana) 7) $O(2^n)$ Exponencial (por ejemplo, Mochila, Satisfactibilidad (SAT), Viajante de Comercio (TSP), todos los subconjuntos) 8) $O(n!)$ Factorial (por ejemplo, circuito hamiltoniano, todas las permutaciones)

Objetivos de Aprendizaje **I:** Usando ejemplos, explicar cada una de las clases de complejidad fundamentales en esta unidad [Explicar] **II:** Para cada clase de complejidad fundamental en esta unidad, explicar un algoritmo que demuestre la complejidad de tiempo de ejecución asociada [Explicar] **III:** Explicar a una audiencia no técnica la importancia de los algoritmos tratables versus intratables usando una explicación intuitiva de la complejidad Big-O [Explicar]

Bibliografía: [Brookshear and Brylow, 2019]

5.14 Análisis de Complejidad: Recursión, Amortización y Cotas Ajustadas, 1 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: **I:** Notaciones Little-o, Little-Omega y Little Theta

Objetivos de Aprendizaje **I:** Dado un problema para programar para el cual puede haber varios enfoques algorítmicos, evaluarlos y determinar cuáles son factibles, y seleccionar uno que sea óptimo en implementación y comportamiento de tiempo de ejecución [Evaluar]

Bibliografía: [Brookshear and Brylow, 2019]

5.15 Estructuras de Datos Fundamentales, 3 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Tipo de Dato Abstracto (ADT) y operaciones sobre un ADT: *a*) Operaciones de diccionario (insertar, eliminar, encontrar) **II:** Arreglos: *a*) Numéricos vs no numéricos, cadenas de caracteres *b*) Unidimensionales (vector) vs multidimensionales (matriz) **III:** Registros/Estructuras/Tuplas y Objetos **IV:** Listas enlazadas (por razones históricas): *a*) Simples vs Dobles y Lineales vs Circulares **V:** Pilas **VI:** Colas y deques: *a*) Cola de prioridad basada en montículo **VII:** Tablas/mapas hash: *a*) Resolución de colisiones y complejidad (por ejemplo, sondeo, encadenamiento, rehash) **VIII:** árboles: *a*) Binarios, n-arios y árboles de búsqueda *b*) Balanceados (por ejemplo, AVL, Rojo-Negro, Montículo)

Objetivos de Aprendizaje I: Para cada ADT/Estructura de Datos en esta unidad: *a*) Explicar su definición, propiedades, representación(es) y operaciones de ADT asociadas. *b*) Explicar paso a paso cómo las operaciones de ADT asociadas con la estructura de datos la transforman. [Explicar] **II:** Explicar cómo se maneja la evitación de colisiones y la resolución de colisiones en tablas hash [Explicar] **III:** Explicar la propiedad de montículo y el uso de montículos como una implementación de una cola de prioridad [Explicar]

Bibliografía: [Brookshear and Brylow, 2019]

5.16 Algoritmos Fundamentales, 3 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Grafos (por ejemplo, [no]dirigidos, [a]cíclicos, [no]conexos y [no]ponderados): *a*) Representación de grafos: lista de adyacencia vs matriz **II:** Algoritmos de búsqueda: *a*) Complejidad $O(n)$ (por ejemplo, búsqueda lineal/secuencial en arreglo/lista) *b*) Complejidad $O(\log_2 n)$ (por ejemplo, búsqueda binaria) *c*) Complejidad $O(\log_b n)$ (por ejemplo, búsqueda en árbol no informada en profundidad/amplitud) **III:** Algoritmos de ordenamiento (por ejemplo, estables, inestables): *a*) Complejidad $O(n^2)$ (por ejemplo, inserción, selección) *b*) Complejidad $O(n \log n)$ (por ejemplo, quicksort, merge, timesort) **IV:** Algoritmos de grafos: *a*) Camino más corto (por ejemplo, Dijkstra, Floyd) *b*) árbol de expansión mínima (por ejemplo, Prim, Kruskal) **V:** Algoritmos de ordenamiento: *a*) Complejidad $O(n \log n)$ heapsort *b*) Pseudo $O(n)$ complejidad (por ejemplo, bucket, counting, radix) **VI:** Algoritmos de grafos: *a*) Clausura transitiva (por ejemplo, Warshall) *b*) Ordenamiento topológico

Objetivos de Aprendizaje I: Para cada algoritmo en esta unidad explicar paso a paso cómo opera el algoritmo [Explicar] **II:** Para cada enfoque algorítmico (por ejemplo, ordenamiento) en esta unidad aplicar un ejemplo prototípico del enfoque (por ejemplo, ordenamiento por mezcla) [Aplicar] **III:** Dados los requisitos para un problema, desarrollar múltiples soluciones usando varias estructuras de datos y algoritmos. Posteriormente, evaluar la idoneidad, fortalezas y debilidades seleccionando un enfoque que satisfaga mejor los requisitos [Crear] **IV:** Explicar factores más allá de la eficiencia computacional que influyen en la elección de algoritmos, como el tiempo de programación, la mantenibilidad y el uso de patrones específicos de la aplicación en los datos de entrada [Explicar]

Bibliografía: [Brookshear and Brylow, 2019]

5.17 Algoritmos Avanzados, 2 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Algoritmos de criptografía (por ejemplo, SHA-256) **II:** Algoritmos paralelos

Objetivos de Aprendizaje I: Una apreciación de la computación cuántica y su aplicación a ciertos problemas [Explicar]

Bibliografía: [Brookshear and Brylow, 2019]

5.18 Conceptos Fundamentales de Sistemas de Bases de Datos, 4 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: **I:** Propósito y ventajas de los sistemas de bases de datos **II:** Componentes de los sistemas de bases de datos **III:** Diseño de funciones fundamentales de DBMS (por ejemplo, mecanismos de consulta, gestión de transacciones, gestión de búfer, métodos de acceso) **IV:** Gestión de transacciones **V:** Normalización **VI:** Enfoques para gestionar grandes volúmenes de datos (por ejemplo, sistemas de bases de datos NoSQL, uso de MapReduce) **VII:** Bases de datos distribuidas/sistemas basados en la nube **VIII:** Uso de un lenguaje de consulta declarativo

Objetivos de Aprendizaje I: Identificar al menos cuatro ventajas que proporciona el uso de un sistema de base de datos [Analizar] **II:** Enumerar los componentes de un sistema de base de datos (relacional) [Enumerar] **III:** Defender el valor de la independencia de datos [Defender] **IV:** Componer una consulta simple de selección-proyección-unión en SQL [Componer] **V:** Describir las ventajas de eliminar datos duplicados repetidos [Describir] **VI:** Esbozar cómo MapReduce usa paralelismo para procesar datos eficientemente [Esquematizar/Esbozar]

Bibliografía: [Brookshear and Brylow, 2019]

5.19 Fundamentos de Programación Orientada a Objetos, 4 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: **I:** Programación imperativa como subconjunto de la programación orientada a objetos. **II:** Diseño orientado a objetos: *a*) Descomposición en objetos que poseen estado y tienen comportamiento. *b*) Diseño de jerarquía de clases para modelado. **III:** Definición de clases: campos, métodos y constructores. **IV:** Subclases, herencia (incluyendo herencia múltiple) y sobrescritura de métodos. **V:** Despacho dinámico: definición de llamada a método. **VI:** Idiomas orientados a objetos para encapsulación: *a*) Privacidad, ocultación de datos y visibilidad de miembros de clase. *b*) Interfaces que revelan solo firmas de métodos. *c*) Clases base abstractas, rasgos (*traits*) y mixins.

Objetivos de Aprendizaje I: Enumerar las diferencias entre los paradigmas de programación imperativa y orientada a objetos [Enumerar] **II:** Componer una clase a través del diseño, implementación y prueba para cumplir con los requisitos de comportamiento [Componer] **III:** Usar mecanismos de encapsulación orientados a objetos como interfaces y miembros privados [Usar]

Bibliografía: [Brookshear and Brylow, 2019]

5.20 Control de Versiones y CI/CD, 2 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: **I:** Gestión de configuración de software y control de versiones: Prácticas de Desarrollo de Software *a*) Configuración en control de versiones, builds/configuración reproducibles. *b*) Estrategias de ramificación en control de versiones. Ramas de desarrollo vs ramas de lanzamiento. Desarrollo basado en tronco (*trunk-based development*). *c*) Estrategias de fusión/rebase, cuando sean relevantes. **II:** Gestión de lanzamientos (*release management*). **III:** Automatización de procesos de software: *a*) Sistemas de construcción (*build systems*): el valor de builds rápidos, herméticos y reproducibles, comparar/contrastar enfoques para construir un proyecto. *b*) Integración Continua (CI): el uso de automatización y pruebas automatizadas para hacer una validación preliminar de que la revisión actual del tronco se construye y pasa las pruebas (básicas). *c*) Despliegue Continuo (CD): el uso de automatización para liberar de forma automática cada cambio que supera las pruebas hacia el entorno de producción, garantizando entregas frecuentes y confiables. *d*) Gestión de dependencias: actualización de dependencias externas/aguas arriba, gestión de paquetes, SemVer. **IV:** Conceptos y mecanismos de integración de herramientas. Prácticas de Desarrollo de Software **V:** Uso de las facilidades de un IDE moderno: depuración, refactorización, búsqueda/indexación, asistentes de código basados en ML, etc. Prácticas de Desarrollo de Software

Objetivos de Aprendizaje I: Describir la diferencia entre la gestión de configuración de software centralizada y distribuida [Describir] **II:** Describir cómo el control de versiones puede usarse para ayudar en la gestión de lanzamientos de software [Describir] **III:** Identificar elementos de configuración y usar una herramienta de control de código fuente en un proyecto pequeño basado en equipo [Analizar] **IV:** Comprender el uso de sistemas de CI/CD como una fuente de verdad para el estado del código compartido del equipo (éxito en la construcción y pruebas) [Explicar] **V:** Demostrar la capacidad de usar herramientas de software para apoyar el desarrollo de un producto de software de tamaño mediano [Demostrar]

Bibliografía: [Brookshear and Brylow, 2019]

5.21 Ingeniería de Requisitos, 2 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Describir requisitos funcionales usando, por ejemplo, casos de uso o historias de usuario. *a*) Usar al menos un método para documentar y estructurar requisitos funcionales. *b*) Comprender cómo el método apoya el diseño y la implementación. *c*) Fortalezas y debilidades de usar un enfoque específico. **II:** Obtención (*elicitation*) de requisitos. *a*) Fuentes de requisitos, por ejemplo, usuarios, administradores o personal de soporte. *b*) Métodos de recopilación de requisitos, por ejemplo, encuestas, entrevistas o análisis de comportamiento. **III:** Requisitos no funcionales, por ejemplo, seguridad, usabilidad o rendimiento, también llamados Atributos de Calidad. Diseño Sostenible y Computación Ubicua, Huella Ambiental de los Sistemas Informáticos, Efectos Sistémicos y Contexto Social **IV:** Identificación y gestión de riesgos, incluyendo consideraciones éticas en torno al producto propuesto. Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo **V:** Comunicar y/o formalizar especificaciones de requisitos. **VI:** Prototipado de una herramienta tanto para obtener como para validar/confirmar requisitos. **VII:** Evolución del producto: cuando cambian los requisitos, cómo entender qué efecto tiene eso y qué cambios deben realizarse. **VIII:** Estimación de esfuerzo: *a*) Aprender técnicas para estimar mejor el esfuerzo requerido para completar una tarea; *b*) Practicar la estimación y compararla con cuánto tiempo toman las tareas; *c*) La estimación de esfuerzo es bastante difícil, por lo que es probable que los estudiantes se equivoquen en muchos casos, pero ver el proceso desarrollarse con su propio trabajo es valioso.

Objetivos de Aprendizaje I: Comparar diferentes métodos de obtención de requisitos a lo largo de múltiples ejes [Comparar] **II:** Identificar diferencias entre dos métodos de describir requisitos funcionales (por ejemplo, entrevistas con clientes, estudios de usuarios) y las situaciones donde se preferiría cada uno [Analizar] **III:** Identificar qué comportamientos son requeridos, permitidos o prohibidos a partir de un conjunto dado de requisitos y una lista de comportamientos candidatos [Analizar] **IV:** Recopilar un conjunto de requisitos para un sistema de software simple [Analizar] **V:** Identificar áreas de un sistema de software que deben cambiarse, dada una descripción del sistema y un conjunto de nuevos requisitos a implementar [Analizar] **VI:** Identificar los requisitos funcionales y no funcionales en un conjunto de requisitos [Analizar] **VII:** Estimar el tiempo para completar un conjunto de tareas, luego comparar las estimaciones con el tiempo real tomado [Estimar] **VIII:** Determinar una secuencia de implementación para un conjunto de tareas, respetando las dependencias entre ellas, con el objetivo de retirar el riesgo lo antes posible [Determinar] **IX:** Escribir una especificación de requisitos para un sistema de software simple [Escribir]

Bibliografía: [Brookshear and Brylow, 2019]

5.22 Fundamentos de Aprendizaje Automático, 1 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Definición y ejemplos de una amplia variedad de tareas de aprendizaje automático: *a*) Aprendizaje supervisado: 1) Clasificación 2) Regresión *b*) Aprendizaje por refuerzo *c*) Aprendizaje

no supervisado: 1) Agrupamiento **II:** Ideas fundamentales: *a*) Teorema de no hay almuerzo gratis: ningún aprendiz puede resolver todos los problemas; las decisiones de diseño de representación tienen consecuencias. *b*) Fuentes de error e indecidibilidad en el aprendizaje automático **III:** Un aprendizaje supervisado simple basado en estadísticas como regresión lineal o árboles de decisión: *a*) Enfocarse en cómo funcionan sin entrar en detalles matemáticos o de optimización; suficiente para entender y usar implementaciones existentes correctamente

Objetivos de Aprendizaje I: Describir las diferencias entre los tres estilos principales de aprendizaje (supervisado, por refuerzo y no supervisado) y determinar cuál es apropiado para un dominio de problema particular [Explicar] **II:** Diferenciar los términos: IA, aprendizaje automático y aprendizaje profundo [Evaluar] **III:** Explicar cómo funciona el aprendizaje automático como un proceso de optimización/búsqueda [Explicar]

Bibliografía: [Brookshear and Brylow, 2019]

5.23 Redes Neuronales y Ética, 1 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Redes neuronales básicas: *a*) Fundamentos de comprensión de cómo funcionan las redes neuronales y su proceso de entrenamiento, sin detalles de los cálculos *b*) Introducción básica a redes neuronales generativas (por ejemplo, modelos de lenguaje grandes) **II:** ética para el Aprendizaje Automático: *a*) Enfocarse en datos reales, escenarios reales y estudios de caso *b*) Sesgo de conjunto de datos/algorítmico/de evaluación y consecuencias no deseadas **III:** Aprendizaje profundo: *a*) Redes feed-forward profundas (solo intuición, sin matemáticas) *b*) Redes neuronales convolucionales (solo intuición, sin matemáticas) *c*) Visualización de representaciones de características aprendidas de redes profundas *d*) Otras arquitecturas (NN generativas, NN recurrentes, transformers, etc.) **IV:** ética para el Aprendizaje Automático: *a*) Continuar enfocándose en datos reales, escenarios reales y estudios de caso *b*) Privacidad *c*) Equidad *d*) Propiedad intelectual *e*) Explicabilidad

Objetivos de Aprendizaje I: Describir el proceso de entrenamiento de redes neuronales y las representaciones aprendidas resultantes [Explicar] **II:** Visualizar el progreso del entrenamiento de una red neuronal a través de curvas de aprendizaje en un kit de herramientas establecido (por ejemplo, TensorBoard) y visualizar las características aprendidas de la red [Aplicar] **III:** Dada una aplicación real de aprendizaje automático, describir problemas éticos respecto a las elecciones de datos, pasos de preprocesamiento, selección de algoritmo y visualización/presentación de resultados [Aplicar]

Bibliografía: [Brookshear and Brylow, 2019]

5.24 Conceptos Fundamentales de Gráficos y Técnicas Interactivas, 2 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Descripción general del pipeline de gráficos por computadora y sistemas interactivos **II:** Conceptos básicos de renderizado: rasterización, trazado de rayos (*ray tracing*), modelos de sombreado **III:** Teoría del color y espacios de color **IV:** Sistemas de coordenadas y transformaciones **V:** Técnicas básicas de interacción: dispositivos de entrada, manejo de eventos **VI:** Conceptos avanzados de renderizado: iluminación global (*global illumination*), renderizado basado en física **VII:** Hardware gráfico y técnicas de aceleración **VIII:** Percepción humana y cognición visual

Objetivos de Aprendizaje I: Explicar las etapas básicas del pipeline de gráficos por computadora [Explicar] **II:** Describir técnicas básicas de renderizado como rasterización y trazado de rayos [Describir] **III:** Aplicar teoría del color para diseñar gráficos visualmente efectivos [Aplicar] **IV:** Implementar transformaciones de coordenadas para gráficos 2D y 3D [Implementar] **V:** Diseñar aplicaciones interactivas básicas utilizando dispositivos de entrada comunes y manejo de eventos [Diseñar] **VI:** Comparar técnicas avanzadas de renderizado como iluminación global

y renderizado basado en física [Comparar] **VII:** Analizar el rol del hardware gráfico en acelerar tareas de renderizado [Analizar] **VIII:** Explicar cómo la percepción humana influye en el diseño de gráficos e interacción [Explicar]

Bibliografía: [Brookshear and Brylow, 2019]

5.25 Renderizado Aplicado y Técnicas, 2 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Trazado de rayos (*ray tracing*): estructuras de aceleración, muestreo (*sampling*), sombreado **II:** Mapeo de texturas (*texture mapping*) y filtrado

Objetivos de Aprendizaje I: Implementar algoritmos de rasterización para renderizado de primitivas [Implementar] **II:** Implementar trazado de rayos básico con estructuras de aceleración [Implementar]

Bibliografía: [Brookshear and Brylow, 2019]

5.26 Clase de cierre: ¿Cómo funciona un motor de búsqueda como Google?, 2 hr(s)

Competencias: AG-C02,AG-C08,AG-C12

Tópicos: I: Análisis del problema. **II:** El índice no crece linealmente con el tamaño de la información indexada. **III:** El tiempo de respuesta no depende del tamaño de la ”base de datos”. **IV:** El tiempo de respuesta no depende del número de ocurrencias encontradas. **V:** Combinar varias estructuras de datos para llegar a una solución. **VI:** Analizar la escalabilidad de la solución.

Objetivos de Aprendizaje I: Comprender los principios bajo los cuales se crea un motor de búsqueda [Usar] **II:** Aplicar correctamente estructuras de datos para resolver el problema [Usar] **III:** Aplicar conceptos relacionados con la complejidad algorítmica en un motor de búsqueda [Usar].

Bibliografía: [Brookshear and Brylow, 2019]

6. Bibliografía

[Brookshear and Brylow, 2019] Brookshear, J. G. and Brylow, D. (2019). *Computer Science: An Overview*. Pearson, global edition edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS112 Programación Orientada a Objetos I (Obligatorio)

1. Información general

- **Semestre**: 2do Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - BIC101 Introducción a la Computación (1er Sem-Pág 257)

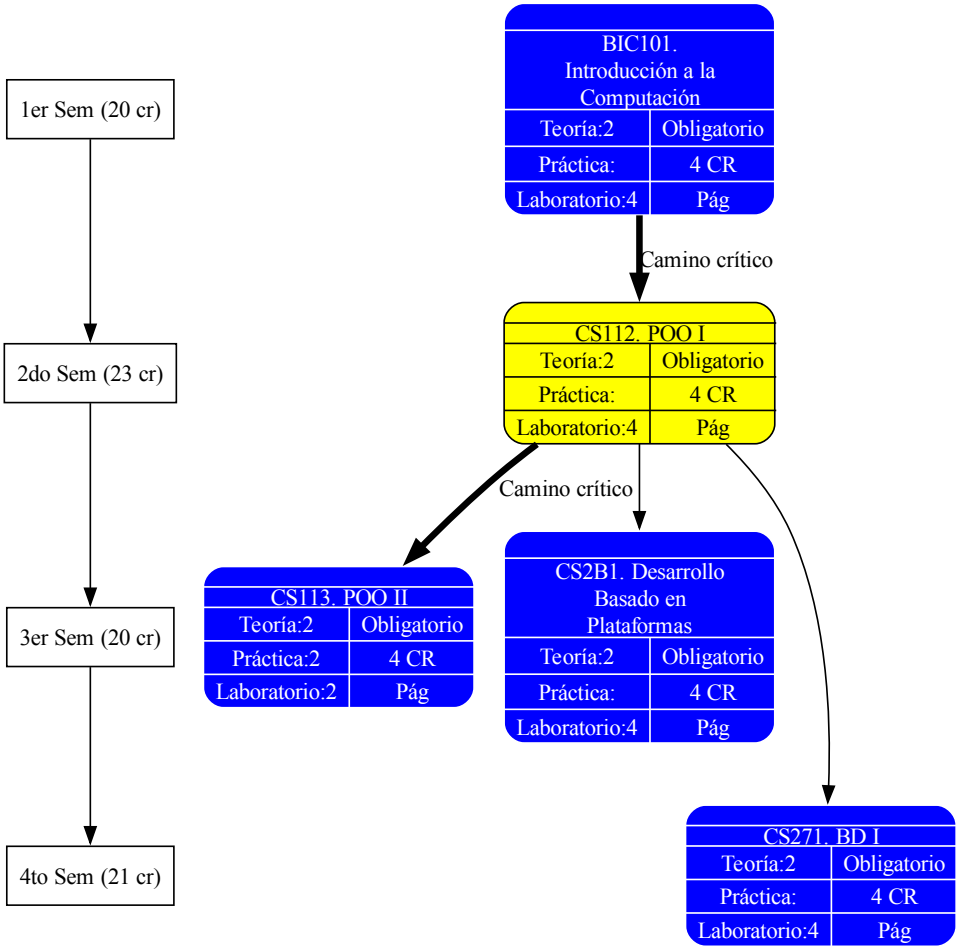

Figura 5.7: Mapa de Conexión. CS112 Programación Orientada a Objetos I

2. Fundamentación

Este es el segundo curso en la secuencia introductoria de programación en Ciencia de la Computación. Incorpora una transición de lenguaje de programación como estrategia pedagógica y se enfoca en los conceptos centrales de Programación Orientada a Objetos (POO) en C++17/C++20, expandiendo las bases del primer curso mientras introduce conceptos de programación de bajo nivel. El curso proporciona una base sólida para los temas avanzados cubiertos en el curso subsiguiente del plan de estudios.

3. Objetivos del curso.

- Introducir al estudiante a los fundamentos del paradigma de orientación a objetos en C++.
- Desarrollar habilidades en programación C++17/C++20, incluyendo plantillas y STL.
- Comprender la gestión de memoria y conceptos de programación de bajo nivel.
- Desarrollar habilidades de resolución de problemas usando principios de POO.
- Dominar técnicas de programación C++ intermedias.

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Pragmática del Lenguaje, 2 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Dominios de problemas y paradigma de programación **II:** Criterios para un buen diseño de lenguaje de programación: *a*) Principios de diseño de lenguajes como la ortogonalidad *b*) Definir construcciones de control e iteración *c*) Modularización de software grande **III:** Breve revisión de paradigmas de programación. **IV:** Comparación entre programación funcional e imperativa. **V:** Historia de los lenguajes de programación (énfasis en C y C++).

Objetivos de Aprendizaje **I:** Discutir el contexto histórico de los paradigmas de lenguajes de programación [Familiarizarse]. **II:** Evaluar las compensaciones en el diseño de lenguajes para aplicaciones específicas [Evaluar].

Bibliografía: [Stroustrup, 2013, Josuttis, 2019]

5.2 Traducción y Ejecución de Lenguajes, 4 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Modelos de ejecución para JIT (Just-In-Time), compilador, intérprete **II:** Uso de código intermedio, por ejemplo, bytecode **III:** Ejecución como código nativo o dentro de una máquina virtual **IV:** Máquinas virtuales y lenguajes intermedios. **V:** Modelos de compilación vs. interpretación.

Objetivos de Aprendizaje **I:** Explicar y comprender las diferencias entre implementaciones de lenguaje compilado, JIT e interpretado, incluyendo los beneficios y limitaciones de cada una [Familiarizarse] **II:** Diferenciar sintaxis y análisis (parsing) de semántica y evaluación [Usar] **III:** Ilustrar ambigüedad en el análisis usando if-then-else anidado/expresión aritmética y mostrar resolución usando orden de precedencia [Evaluar]

Bibliografía: [Stroustrup, 2013, Deitel and Deitel, 2017]

5.3 Fundamentos de Sistemas de Tipos, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Un tipo como un conjunto de valores junto con un conjunto de operaciones: *a*) Tipos primitivos (por ejemplo, números, booleanos) *b*) Tipos compuestos construidos a partir de otros tipos (por ejemplo, registros/estructuras, uniones, arreglos, listas, funciones, referencias usando operaciones de conjuntos) **II:** Tipos genéricos (polimorfismo paramétrico): *a*) Definición y ventajas del polimorfismo: paramétrico, de subtipado (subtyping), de sobrecarga (overloading) y de

coerción *b*) Comparación de tipos monomórficos y polimórficos *c*) Comparación con polimorfismo ad-hoc (sobrecarga) y polimorfismo de subtipos *d*) Parámetros genéricos y tipado *e*) Uso de bibliotecas genéricas como colecciones *f*) Comparación con polimorfismo ad hoc (sobrecarga) y polimorfismo de subtipos *g*) Polimorfismo prescriptivo vs descriptivo *h*) Modelos de implementación de tipos polimórficos *i*) Subtipado **III:** Modelos de declaración (enlace, visibilidad, alcance, tiempo de vida). **IV:** Visión general de la verificación de tipos en C++. **V:** Conversión de tipos y casting en C++.

Objetivos de Aprendizaje **I:** Definir y usar piezas de programa (como funciones, clases, métodos) que usen tipos genéricos, incluyendo para colecciones [Usar] **II:** Discutir las diferencias entre genéricos, subtipado y sobrecarga [Familiarizarse] **III:** Explicar múltiples beneficios y limitaciones del tipado estático al escribir, mantener y depurar software [Evaluar]

Bibliografía: [Stroustrup, 2013, Josuttis, 2019]

5.4 Fundamentos de Programación Orientada a Objetos, 14 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Definición de clases: campos, métodos y constructores. **II:** Subclases, herencia (incluyendo herencia múltiple) y sobrescritura de métodos. **III:** Despacho dinámico: definición de llamada a método. **IV:** Subtipado: *a*) Polimorfismo de subtipos; conversiones ascendentes (*upcasts*) implícitas en lenguajes tipados. *b*) Noción de reemplazo conductual: los subtipos actúan como el supertipo. *c*) Relación entre subtipado y herencia. **V:** Clases y objetos en C++. **VI:** Constructores y destructores. **VII:** Especificadores de acceso y encapsulamiento. **VIII:** Jerarquías de herencia. **IX:** Polimorfismo y funciones virtuales.

Objetivos de Aprendizaje **I:** Componer una clase a través del diseño, implementación y prueba para cumplir con los requisitos de comportamiento [Usar] **II:** Construir una jerarquía de clases simple utilizando subclases que permita reutilizar código para subclases distintas [Usar] **III:** Predecir y validar el flujo de control en un programa usando despacho dinámico [Evaluar] **IV:** Diseñar clases en C++ con encapsulamiento adecuado [Usar]. **V:** Implementar jerarquías de herencia para reutilización de código [Usar].

Bibliografía: [Stroustrup, 2013, Josuttis, 2019]

5.5 Modelo de Ejecución y Memoria de Sistemas, 12 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Acceso directo, indirecto e indexado a ubicación de memoria **II:** Representación en tiempo de ejecución de abstracciones de datos como variables, arreglos, vectores, registros, elementos de datos basados en punteros como listas enlazadas y árboles, y objetos **III:** Asignación y acceso de bajo nivel a estructuras de datos de alto nivel como tipos de datos básicos, arreglos n-dimensionales, vectores, registros y objetos **IV:** Retorno de procedimiento como mecanismo de desasignación automática para elementos de datos locales en la pila **V:** Gestión manual de memoria: asignar, desasignar y reutilizar memoria del montón **VI:** Gestión automática de memoria: recolección de basura (garbage collection) como técnica automatizada usando la noción de alcanzabilidad (reachability) **VII:** Distribución de memoria (pila, montón, estática). **VIII:** Aritmética de punteros y operaciones. **IX:** Asignación dinámica de memoria (new/delete). **X:** Fugas de memoria y punteros colgantes. **XI:** Punteros inteligentes (unique_ptr, shared_ptr).

Objetivos de Aprendizaje **I:** Explicar por qué ocurren las fugas de memoria (memory leaks) y los problemas de punteros colgantes (dangling pointers), y qué puede hacer un programador para evitarlos/corregirlos [Usar] **II:** Explicar cómo las implementaciones de lenguajes de programación típicamente organizan la memoria en secciones de datos globales, texto, montón y pila, y cómo características como recursión y gestión de memoria se mapean a este modelo de memoria [Evaluar] **III:** Explicar cómo se ejecuta una construcción central del lenguaje, como abstracciones de datos y abstracciones de control [Evaluar] **IV:** Implementar RAII para la gestión de recursos [Usar]. **V:** Depurar errores relacionados con memoria en programas C++ [Usar].

Bibliografía: [Stroustrup, 2013, Deitel and Deitel, 2017]

5.6 Programación Funcional, 10 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Expresiones lambda y evaluación: *a*) Enlace (binding) de variables y reglas de ámbito (scope). *b*) Paso de parámetros. *c*) Expresiones lambda anidadas y orden de reducción. **II:** Usar funciones de orden superior (tomar, devolver y almacenar funciones). **III:** Plantillas de funciones y plantillas de clases. **IV:** Especialización de plantillas. **V:** Expresiones lambda en C++11/14. **VI:** Conceptos de la Biblioteca de Plantillas Estándar (STL).

Objetivos de Aprendizaje **I:** Desarrollar funciones útiles que tomen y devuelvan otras funciones [Usar] **II:** Interpretar correctamente las variables y el ámbito léxico en un programa que usa clausuras de funciones [Evaluar] **III:** Usar mecanismos de encapsulación funcional como clausuras e interfaces modulares [Usar] **IV:** Implementar funciones genéricas usando plantillas [Usar]. **V:** Usar expresiones lambda para programación funcional [Usar].

Bibliografía: [Stroustrup, 2013, Josuttis, 2019]

5.7 Programación Orientada a Objetos: Encapsulación, Subtipado y Reflexión, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Clases de colecciones, iteradores y otros componentes comunes de biblioteca. **II:** Contenedores de secuencia (vector, list, deque). **III:** Contenedores asociativos (set, map, unordered_set). **IV:** Adaptadores de contenedor (stack, queue, priority_queue). **V:** Iteradores y algoritmos. **VI:** Funtores y predicados.

Objetivos de Aprendizaje **I:** Usar clases de colecciones e iteradores efectivamente para resolver un problema [Usar] **II:** Definir y usar iteradores y otras operaciones en agregados, incluyendo operaciones que toman funciones como argumentos, en múltiples lenguajes de programación, seleccionando los idiomas más naturales para cada lenguaje [Usar] **III:** Comparar y contrastar los mecanismos para definir y proteger elementos de datos dentro de los enfoques procedural, funcional y orientado a objetos [Evaluar] **IV:** Seleccionar contenedores STL apropiados para necesidades específicas [Usar].

Bibliografía: [Stroustrup, 2013, Josuttis, 2019]

5.8 Construcciones de Programación Avanzadas, 6 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Abstracciones orientadas a objetos: herencia múltiple, mixins, rasgos (traits), multimétodos (multimethods) **II:** Metaprogramación: macros, programación generativa, desarrollo basado en modelos (model-based development) **III:** Sobrecarga de operadores. **IV:** Semántica de movimiento y referencias a valor-r. **V:** Manejo de excepciones en C++. **VI:** Espacios de nombres y programación modular.

Objetivos de Aprendizaje **I:** Usar varias construcciones e idiomas de programación avanzados correctamente [Usar] **II:** Discutir cómo varias construcciones de programación avanzadas tienen como objetivo mejorar la estructura del programa, la calidad del software y la productividad del programador [Evaluar] **III:** Discutir cómo varias construcciones de programación avanzadas interactúan con la definición e implementación de otras características del lenguaje [Familiarizarse] **IV:** Implementar semántica de movimiento para optimización de rendimiento [Usar].

Bibliografía: [Stroustrup, 2013, Josuttis, 2019]

5.9 Scripting de Shell, 4 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Comandos del sistema: *a*) Interfaz con sistemas operativos **II:** Tuberías (piping) **III:** Abstracción de archivo y operadores **IV:** Programas y procesos **V:** Automatización de la compilación y pruebas de programas C++ mediante scripts (Makefiles, scripts de build/test).

Objetivos de Aprendizaje I: Crear y ejecutar scripts automatizados para gestionar varias tareas del sistema [Usar]

Bibliografía: [Robbins and Beebe, 2005]

6. Bibliografía

[Deitel and Deitel, 2017] Deitel, H. and Deitel, P. (2017). *C++ How to Program*. Pearson, Boston, MA, 10th edition.

[Josuttis, 2019] Josuttis, N. M. (2019). *C++17: The Complete Guide*. Pearson Education, Indianapolis, IN, 1st edition.

[Robbins and Beebe, 2005] Robbins, A. and Beebe, N. H. (2005). *Classic Shell Scripting*. O'Reilly Media, Sebastopol, CA, 1st edition.

[Stroustrup, 2013] Stroustrup, B. (2013). *The C++ Programming Language*. Addison-Wesley Professional, Upper Saddle River, NJ, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS1D1 Estructuras Discretas (Obligatorio)

1. Información general

- **Semestre**: 2do Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 4 horas;
- **Prerrequisitos**: Ninguno

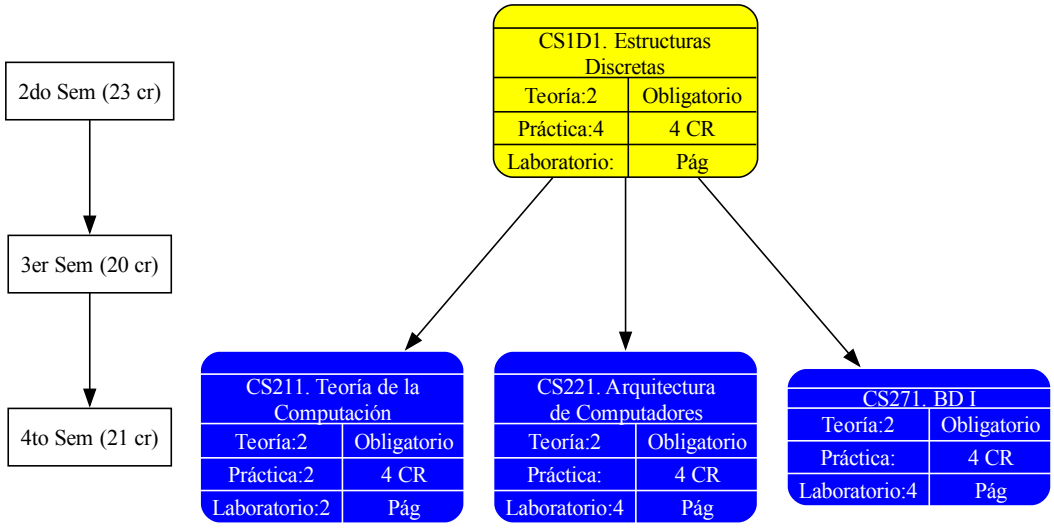

Figura 5.8: Mapa de Conexión. CS1D1 Estructuras Discretas

2. Fundamentación

Las estructuras discretas proporcionan el fundamento teórico para la ciencia de la computación. Este curso introduce las herramientas matemáticas esenciales utilizadas para modelar problemas computacionales, incluyendo lógica, teoría de conjuntos y funciones. Comprender estos conceptos es crítico para desarrollar habilidades de razonamiento riguroso y para el estudio posterior de algoritmos, bases de datos y lenguajes formales.

3. Objetivos del curso.

- Comprender los principios de la lógica matemática y las técnicas de demostración.
- Realizar operaciones con conjuntos, funciones y relaciones.
- Aplicar técnicas básicas de conteo y relaciones de recurrencia a problemas.
- Desarrollar la capacidad de pensar de manera abstracta y resolver problemas formalmente.

4. Competencias Generales (*Outcomes*)

AG-C07) Conocimientos de Computación: Aplica conocimientos de matemáticas, ciencias y computación. (Usar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Lógica y Fundamentos, 12 hr(s)

Competencias: AG-C07,AG-C12

Tópicos: **I:** Lógica: lógica proposicional y de predicados, cuantificadores, demostraciones **II:** Conjuntos: operaciones, cardinalidad, conjuntos potencia, productos cartesianos **III:** Relaciones: propiedades, relaciones de equivalencia, órdenes parciales **IV:** Funciones: inyectivas, sobreyectivas, biyectivas, composición, inversa **V:** Lógica avanzada: sistemas de demostración, completitud, decidibilidad **VI:** Teoría avanzada de conjuntos: conjuntos infinitos, argumento diagonal de Cantor

Objetivos de Aprendizaje **I:** Explicar lógica proposicional y de predicados [Explicar] **II:** Construir demostraciones lógicas simples [Aplicar] **III:** Definir operaciones y relaciones básicas de conjuntos [Explicar] **IV:** Definir propiedades de las relaciones [Explicar] **V:** Definir tipos de funciones [Explicar]

Bibliografía: [Rosen, 2019, Lehman et al., 2018]

5.2 Técnicas de Demostración y Estructuras Avanzadas, 12 hr(s)

Competencias: AG-C07,AG-C12

Tópicos: **I:** Lógica: lógica proposicional y de predicados, cuantificadores, demostraciones **II:** Lógica avanzada: sistemas de demostración, completitud, decidibilidad **III:** Teoría avanzada de conjuntos: conjuntos infinitos, argumento diagonal de Cantor **IV:** Relaciones avanzadas: clausuras, retículos **V:** Funciones avanzadas: funciones recursivas, computabilidad

Objetivos de Aprendizaje **I:** Construir demostraciones lógicas simples [Aplicar] **II:** Explicar conceptos de conjuntos infinitos y cardinalidad [Explicar] **III:** Aplicar operaciones de clausura a relaciones [Aplicar] **IV:** Determinar si una relación es una relación de equivalencia [Aplicar]

Bibliografía: [Rosen, 2019, Velleman, 2019]

5.3 Relaciones, Funciones y Grafos, 12 hr(s)

Competencias: AG-C07,AG-C12

Tópicos: **I:** Relaciones: propiedades, relaciones de equivalencia, órdenes parciales **II:** Funciones: inyectivas, sobreyectivas, biyectivas, composición, inversa **III:** Grafos: definiciones, caminos, ciclos, árboles, conectividad **IV:** Teoría avanzada de grafos: planaridad, coloración, redes

Objetivos de Aprendizaje **I:** Determinar si una relación es una relación de equivalencia [Aplicar] **II:** Componer funciones y encontrar inversas [Aplicar] **III:** Definir conceptos básicos de grafos [Explicar] **IV:** Resolver problemas básicos de grafos [Aplicar] **V:** Analizar propiedades avanzadas de grafos [Evaluar]

Bibliografía: [Rosen, 2019, Scheinerman, 2012]

5.4 Conteo y Recurrencia, 12 hr(s)

Competencias: AG-C07,AG-C12

Tópicos: **I:** Conteo: permutaciones, combinaciones, teorema del binomio **II:** Conteo avanzado: funciones generadoras, principio de inclusión-exclusión **III:** Relaciones de recurrencia: resolver recurrencias simples **IV:** Recurrencia avanzada: funciones generadoras, comportamiento asintótico

Objetivos de Aprendizaje **I:** Explicar permutaciones y combinaciones [Explicar] **II:** Aplicar técnicas de conteo para resolver problemas [Aplicar] **III:** Definir relaciones de recurrencia [Explicar] **IV:** Resolver relaciones de recurrencia simples [Aplicar] **V:** Desarrollar funciones generadoras para problemas de conteo [Crear]

Bibliografía: [Rosen, 2019, Grimaldi, 2003]

6. Bibliografía

[Grimaldi, 2003] Grimaldi, R. P. (2003). *Discrete and Combinatorial Mathematics: An Applied Introduction*. Pearson, 5th edition.

[Lehman et al., 2018] Lehman, E., Leighton, F. T., and Meyer, A. R. (2018). *Mathematics for Computer Science*. MIT OpenCourseWare.

[Rosen, 2019] Rosen, K. H. (2019). *Discrete Mathematics and Its Applications*. McGraw-Hill Education, 8th edition.

[Scheinerman, 2012] Scheinerman, E. A. (2012). *Mathematics: A Discrete Introduction*. Cengage Learning, 3rd edition.

[Velleman, 2019] Velleman, D. J. (2019). *How to Prove It: A Structured Approach*. Cambridge University Press, 3rd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BMA103 Cálculo Integral (Obligatorio)

1. Información general

- **Semestre**: 2do Sem. **Créditos**: 5
- **Horas del curso**: **Teoría**: 4 horas; **Práctica**: 2 horas;
- **Prerrequisitos**:
  - BMA102 Cálculo Diferencial (1er Sem-Pág 264)

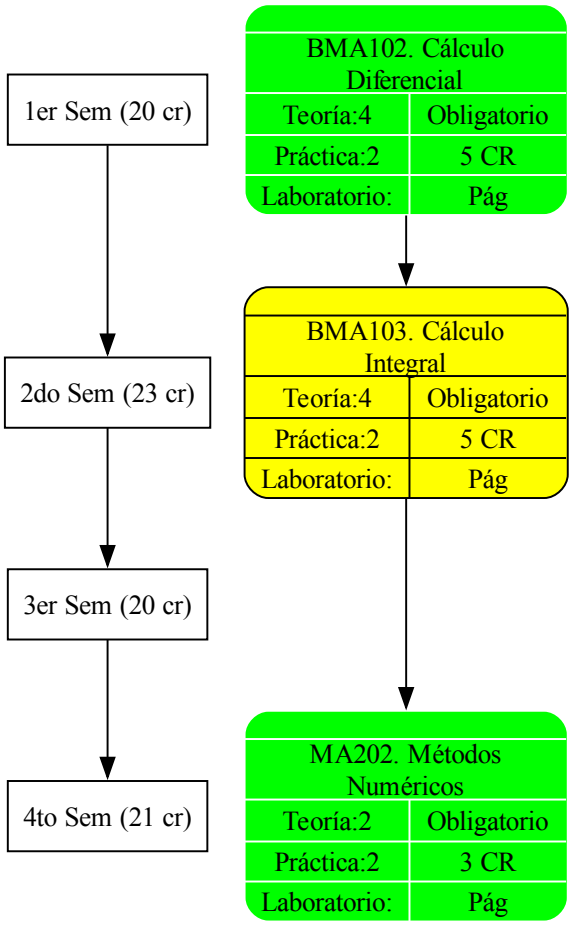

Figura 5.9: Mapa de Conexión. BMA103 Cálculo Integral

2. Fundamentación

El cálculo integral es una herramienta matemática fundamental compartida por la ciencia de la computación y las disciplinas de ingeniería, esencial para modelar y resolver problemas que involucran acumulación, cambio y áreas bajo curvas. Este curso proporciona las bases del cálculo integral, incluyendo técnicas de integración y sus aplicaciones en contextos científicos y de ingeniería.

3. Objetivos del curso.

- Comprender el concepto de integral definida e indefinida.

- Aplicar diversas técnicas de integración para resolver problemas.
- Utilizar el cálculo integral para modelar y resolver problemas en contextos científicos y de ingeniería.

4. Competencias Generales (*Outcomes*)

AG-C07) Conocimientos de Computación: Aplica conocimientos de matemáticas, ciencias y computación. (Familiarizarse)

5. Contenido

5.1 Antiderivadas y Técnicas de Integración, 20 hr(s)

Competencias: AG-C07

Tópicos: I: El Teorema Fundamental del Cálculo y la antiderivada **II:** Técnicas de integración: sustitución, partes, fracciones parciales y sustitución trigonométrica

Objetivos de Aprendizaje I: Enunciar el Teorema Fundamental del Cálculo y explicar la relación entre la diferenciación y la integración [Familiarizarse] **II:** Calcular integrales definidas e indefinidas usando técnicas estándar, incluyendo sustitución e integración por partes [Usar]

Bibliografía: [Stewart, 2015, Larson and Edwards, 2014]

5.2 Integrales Impropias, 8 hr(s)

Competencias: AG-C07

Tópicos: I: Integrales impropias: criterios de convergencia y pruebas de comparación

Objetivos de Aprendizaje I: Determinar la convergencia de integrales impropias usando criterios de comparación y otras pruebas de convergencia [Evaluar]

Bibliografía: [Stewart, 2015, Larson and Edwards, 2014]

5.3 Sucesiones, Series y Series de Potencias, 20 hr(s)

Competencias: AG-C07

Tópicos: I: Sucesiones y series: pruebas de convergencia (cociente, raíz, comparación, integral) **II:** Series de potencias, radio de convergencia, series de Taylor y Maclaurin

Objetivos de Aprendizaje I: Determinar la convergencia de series de potencias y construir series de Taylor para funciones estándar [Evaluar]

Bibliografía: [Stewart, 2015, Larson and Edwards, 2014]

6. Bibliografía

[Larson and Edwards, 2014] Larson, R. and Edwards, B. H. (2014). *Calculus.* Cengage Learning.

[Stewart, 2015] Stewart, J. (2015). *Calculus: Early Transcendentals.* Cengage Learning.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BFI101 Física I (Obligatorio)

1. Información general

- **Semestre**: 2do Sem. **Créditos**: 5
- **Horas del curso**: **Teoría**: 4 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**: Ninguno

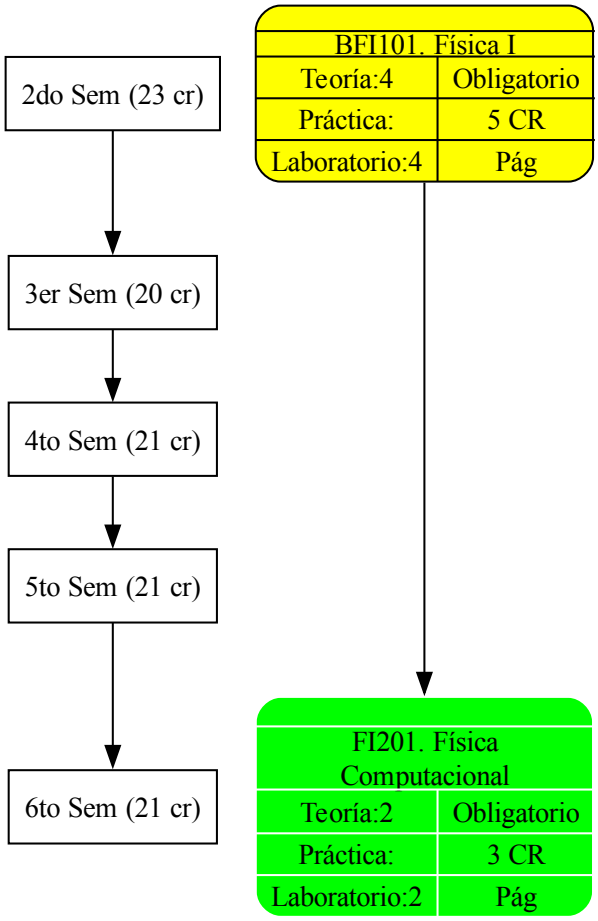

Figura 5.10: Mapa de Conexión. BFI101 Física I

2. Fundamentación

Física I introduce los fundamentos de la mecánica clásica como una disciplina matemática y experimental. A través del estudio de la cinemática, la dinámica newtoniana, los principios de conservación, el movimiento rotacional, la elasticidad, la mecánica de fluidos, las vibraciones y ondas, y una introducción a los fenómenos de transporte, los estudiantes adquieren las herramientas analíticas necesarias para modelar y resolver problemas del mundo físico. Este curso proporciona la base indispensable para el estudio de los fenómenos oscilatorios, ondulatorios y electromagnéticos que se abordan en cursos posteriores.

3. Objetivos del curso.

- Describir el movimiento de partículas en diferentes sistemas de coordenadas y formular las ecuaciones del movimiento a partir de las leyes de Newton.
- Aplicar los principios de trabajo, energía potencial y conservación de la energía mecánica a sistemas con fuerzas conservativas y no conservativas.
- Resolver problemas de dinámica de sistemas de partículas utilizando la conservación del momento lineal y el análisis de colisiones.
- Analizar el movimiento rotacional y la dinámica del momento angular en sistemas de partículas y cuerpos extendidos.
- Aplicar los módulos elásticos y la ley de Hooke para relacionar el esfuerzo y la deformación en sólidos.
- Aplicar los principios de la hidrostática y la hidrodinámica, incluyendo las ecuaciones de continuidad y de Bernoulli, a problemas de flujo de fluidos.
- Analizar el movimiento armónico simple y la propagación de ondas mecánicas, incluyendo los fenómenos de superposición y resonancia.
- Aplicar las leyes de Fourier y Fick para describir procesos básicos de conducción de calor y difusión de masa.

4. Competencias Generales (*Outcomes*)

AG-C07) Conocimientos de Computación: Aplica conocimientos de matemáticas, ciencias y computación. (Familiarizarse)

5. Contenido

5.1 Cinemática y Dinámica de Partículas, 22 hr(s)

Competencias: AG-C07

Tópicos: **I:** Descripción vectorial de posición, velocidad y aceleración; las tres leyes de Newton y marcos de referencia inerciales **II:** Formulación y solución analítica de ecuaciones de movimiento para fuerzas constantes y variables **III:** Diagramas del espacio de fases (posición vs. velocidad) para visualizar el estado mecánico, la estabilidad y la periodicidad **IV:** Integración numérica de ecuaciones de movimiento usando métodos de Euler, Verlet y Runge-Kutta **V:** Movimiento con restricciones y ligaduras holonómicas relevantes para eslabones robóticos y cuerpos articulados

Objetivos de Aprendizaje **I:** Identificar todas las fuerzas que actúan sobre una partícula en un escenario mecánico dado y trazar el diagrama correcto de cuerpo libre [Familiarizarse] **II:** Implementar un algoritmo de integración numérica como el método de Runge-Kutta para calcular la trayectoria de una partícula bajo fuerzas variables [Usar] **III:** Construir e interpretar diagramas del espacio de fases para caracterizar la estabilidad y periodicidad de un sistema mecánico simulado [Evaluar]

Bibliografía: [Serway and Jewett, 2018, Young and Freedman, 2015]

5.2 Trabajo, Energía y Leyes de Conservación, 12 hr(s)

Competencias: AG-C07

Tópicos: **I:** Teorema trabajo-energía que relaciona el trabajo neto realizado con el cambio en la energía cinética **II:** Funciones de energía potencial y la relación fuerza-gradiente para campos de fuerza conservativos **III:** Conservación de la energía mecánica total y condiciones para su aplicabilidad **IV:** Diagramas de energía para identificar puntos de equilibrio, puntos de retorno y estabilidad **V:** Modelos de colisiones elásticas e inelásticas: intercambio de momento y energía en implementaciones de motores de física

Objetivos de Aprendizaje **I:** Explicar la distinción entre fuerzas conservativas y no conservativas y las condiciones bajo las cuales se conserva la energía mecánica [Familiarizarse] **II:** Aplicar métodos

de conservación de energía para calcular velocidades y posiciones en problemas de movimiento unidimensional comunes en implementaciones de motores de física [Usar] **III:** Analizar eventos de colisión elástica e inelástica, calculando velocidades post-colisión y cuantificando la disipación de energía [Evaluar]

Bibliografía: [Serway and Jewett, 2018, Tipler and Mosca, 2007]

5.3 Sistemas de Partículas y Momento Lineal, 8 hr(s)

Competencias: AG-C07

Tópicos: **I:** Definición de centro de masa: $\vec{R}_{CM} = \frac{1}{M}\sum m_i \vec{r}_i$ **II:** Momento lineal de una partícula y de un sistema: $\vec{p} = m\vec{v}$ y $\vec{P} = M\vec{V}_{CM}$ **III:** Segunda ley de Newton para sistemas: $\vec{F}_{ext} = \frac{d\vec{P}}{dt}$ **IV:** Conservación del momento lineal para sistemas aislados **V:** Distinción entre fuerzas internas y externas en un sistema **VI:** Colisiones de dos cuerpos, incluyendo casos elásticos, inelásticos y completamente inelásticos **VII:** Coeficiente de restitución y su significado físico en las colisiones **VIII:** Sistemas con masa variable, como la ecuación del cohete: $M\frac{dV}{dt} = v_{ex}\frac{dM}{dt}$ **IX:** Sistemas de muchas partículas y su dinámica del centro de masa **X:** Distribuciones continuas de masa y el cálculo de su centro de masa mediante integración

Objetivos de Aprendizaje **I:** Localizar el centro de masa para un conjunto discreto de partículas en Una, Dos y Tres dimensiones [Usar] **II:** Determinar el momento lineal total de un sistema a partir del movimiento de su centro de masa [Usar] **III:** Aplicar la conservación del momento lineal para resolver problemas de colisión en Una y Dos dimensiones [Usar] **IV:** Clasificar colisiones como elásticas, inelásticas o completamente inelásticas basándose en la conservación de la energía cinética [Familiarizarse] **V:** Calcular el coeficiente de restitución a partir de las velocidades antes y después de la colisión [Usar] **VI:** Resolver la ecuación del cohete para determinar la velocidad en función de la masa expulsada [Usar] **VII:** Analizar cómo las fuerzas internas afectan a las partículas individuales pero no al momento total de un sistema [Evaluar] **VIII:** Integrar para hallar el centro de masa de cuerpos continuos simétricos como varillas, discos y esferas [Usar]

Bibliografía: [Young and Freedman, 2015, Tipler and Mosca, 2007]

5.4 Movimiento Rotacional y Momento Angular, 12 hr(s)

Competencias: AG-C07

Tópicos: **I:** Cinemática rotacional incluyendo desplazamiento angular, velocidad y aceleración (θ, ω, α) **II:** Definición de torque: $\vec{\tau} = \vec{r} \times \vec{F}$ y el cálculo de su magnitud **III:** Momento angular de una partícula: $\vec{L} = \vec{r} \times \vec{p}$ **IV:** Forma rotacional de la segunda ley de Newton: $\vec{\tau}_{net} = \frac{d\vec{L}}{dt}$ **V:** Momento de inercia $I = \sum m_i r_i^2$ y el teorema de los ejes paralelos **VI:** Energía cinética rotacional: $K_{rot} = \frac{1}{2}I\omega^2$ **VII:** Conservación del momento angular para sistemas aislados **VIII:** Movimiento giroscópico y el fenómeno de la precesión **IX:** Dinámica de rodadura sin deslizamiento: $v_{CM} = R\omega$ **X:** El efecto Coriolis como manifestación de la dinámica rotacional en marcos no inerciales

Objetivos de Aprendizaje **I:** Relacionar las magnitudes cinemáticas lineales y angulares para el movimiento circular [Usar] **II:** Calcular el torque respecto a un eje dado para una fuerza aplicada en un punto específico [Usar] **III:** Determinar el momento de inercia para geometrías simples como una varilla, un aro, un disco o una esfera [Usar] **IV:** Aplicar la conservación del momento angular para analizar cambios en la velocidad de rotación cuando cambia el momento de inercia [Evaluar] **V:** Calcular la energía cinética rotacional de un cuerpo rígido en rotación [Usar] **VI:** Analizar el movimiento de objetos que ruedan sin deslizar por un plano inclinado [Evaluar] **VII:** Predecir la dirección de la precesión para una peonza o un giroscopio en rotación [Evaluar] **VIII:** Explicar el efecto Coriolis en términos de marcos de referencia rotatorios y su efecto sobre los proyectiles [Familiarizarse]

Bibliografía: [Serway and Jewett, 2018, Kleppner and Kolenkow, 2013]

5.5 Elasticidad, 2 hr(s)

Competencias: AG-C07

Tópicos: **I:** Definición de esfuerzo (fuerza/área) y deformación unitaria ($\Delta L/L$) **II:** Módulo de Young y ley de Hooke para deformación longitudinal: $\sigma = E\epsilon$ **III:** Módulo de corte (rigidez) y deformación por cizalladura **IV:** Módulo de volumen (compresibilidad) y deformación volumétrica **V:** Límite elástico, región plástica y punto de ruptura en el diagrama esfuerzo-deformación **VI:** Energía elástica almacenada en un material deformado

Objetivos de Aprendizaje **I:** Aplicar la ley de Hooke escalar para calcular la deformación de una barra bajo carga axial [Usar] **II:** Calcular el módulo de Young, de corte o de volumen a partir de datos experimentales esfuerzo-deformación [Usar] **III:** Identificar la región elástica, el límite elástico y el punto de fractura en una curva esfuerzo-deformación [Familiarizarse] **IV:** Calcular la energía potencial elástica almacenada en un resorte o barra deformada [Usar]

Bibliografía: [Serway and Jewett, 2018, Young and Freedman, 2015]

5.6 Fluidos: Hidrostática, Hidrodinámica y Viscosidad, 10 hr(s)

Competencias: AG-C07

Tópicos: **I:** Densidad, presión y su variación con la profundidad en un fluido en reposo: $p = p_0 + \rho gh$ **II:** Principio de Pascal y sus aplicaciones (prensa hidráulica) **III:** Principio de Arquímedes, empuje y flotación **IV:** Medición de presión con manómetros y barómetros **V:** Ecuación de continuidad para flujo incompresible: $A_1 v_1 = A_2 v_2$ **VI:** Ecuación de Bernoulli y conservación de energía en flujo ideal **VII:** Aplicaciones: tubo de Venturi, teorema de Torricelli, tubo de Pitot **VIII:** Viscosidad y ley de Poiseuille para flujo laminar en tuberías **IX:** Número de Reynolds y transición a flujo turbulento **X:** Ley de Stokes para la fuerza de arrastre sobre una esfera en un fluido viscoso

Objetivos de Aprendizaje **I:** Calcular la presión en un punto de un fluido en reposo dado la profundidad y densidad [Usar] **II:** Aplicar el principio de Arquímedes para determinar si un objeto flota, se hunde o permanece en equilibrio [Usar] **III:** Aplicar las ecuaciones de continuidad y de Bernoulli para resolver problemas de flujo en tuberías de sección variable [Usar] **IV:** Interpretar lecturas de manómetros y barómetros para determinar presiones absolutas y manométricas [Evaluar] **V:** Aplicar la ley de Poiseuille para calcular el caudal de un fluido viscoso en un tubo cilíndrico [Usar] **VI:** Estimar el número de Reynolds de un flujo y predecir si es laminar o turbulento [Evaluar]

Bibliografía: [Serway and Jewett, 2018, Young and Freedman, 2015]

5.7 Vibraciones y Ondas, 14 hr(s)

Competencias: AG-C07

Tópicos: **I:** Movimiento armónico simple (MAS): ecuación de movimiento, posición, velocidad y aceleración **II:** Sistemas oscilantes: resorte-masa y péndulo simple/físico **III:** Energía en el movimiento armónico simple **IV:** Oscilaciones amortiguadas y forzadas; introducción a la resonancia **V:** Ondas transversales y longitudinales; velocidad de propagación **VI:** Ecuación de onda y velocidad de una onda en una cuerda tensa: $v = \sqrt{T/\mu}$ **VII:** Principio de superposición y ondas estacionarias **VIII:** Ondas sonoras, intensidad y nivel de intensidad sonora (decibeles) **IX:** Efecto Doppler para ondas sonoras **X:** Pulsaciones (beats) e interferencia de ondas de frecuencias cercanas

Objetivos de Aprendizaje **I:** Deducir la ecuación de movimiento y el período de un sistema resorte-masa y de un péndulo simple [Usar] **II:** Calcular la energía cinética, potencial y total de un oscilador armónico simple en cualquier instante [Usar] **III:** Analizar la respuesta en amplitud de un oscilador forzado cerca de la frecuencia de resonancia [Evaluar] **IV:** Calcular la velocidad de propagación de una onda en una cuerda a partir de la tensión y la densidad lineal [Usar]

V: Determinar las frecuencias y longitudes de onda de los modos normales de una onda estacionaria en una cuerda fija [Usar] **VI:** Calcular el nivel de intensidad sonora en decibeles a partir de la intensidad de una onda sonora [Usar] **VII:** Aplicar la fórmula del efecto Doppler para fuente y/o observador en movimiento [Usar]

Bibliografía: [Serway and Jewett, 2018, Young and Freedman, 2015]

5.8 Fenómeno de Transporte, 4 hr(s)

Competencias: AG-C07

Tópicos: I: Conducción de calor y la ley de Fourier: $\vec{q} = -k\nabla T$ **II:** Conductividad térmica de materiales y resistencia térmica en paredes compuestas **III:** Difusión molecular y la primera ley de Fick: $\vec{J} = -D\nabla C$ **IV:** Analogía entre el transporte de calor, masa y momento (números de Prandtl y Schmidt)

Objetivos de Aprendizaje I: Aplicar la ley de Fourier para calcular el flujo de calor a través de una pared plana o compuesta [Usar] **II:** Aplicar la primera ley de Fick para calcular el flujo difusivo de una especie en un gradiente de concentración [Usar] **III:** Identificar analogías matemáticas entre los procesos de transporte de calor, masa y momento [Familiarizarse]

Bibliografía: [Serway and Jewett, 2018, Tipler and Mosca, 2007]

6. Bibliografía

[Kleppner and Kolenkow, 2013] Kleppner, D. and Kolenkow, R. J. (2013). *An Introduction to Mechanics*. Cambridge University Press, 2nd edition.

[Serway and Jewett, 2018] Serway, R. A. and Jewett, J. W. (2018). *Physics for Scientists and Engineers with Modern Physics*. Cengage Learning, 10th edition.

[Tipler and Mosca, 2007] Tipler, P. A. and Mosca, G. (2007). *Physics for Scientists and Engineers*. W. H. Freeman and Company, 6th edition.

[Young and Freedman, 2015] Young, H. D. and Freedman, R. A. (2015). *Sears and Zemansky's University Physics with Modern Physics*. Pearson Education, 14th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BEI102 Inglés II (Obligatorio)

1. Información general

- **Semestre**: 2do Sem. **Créditos**: 2
- **Horas del curso**: **Práctica**: 4 horas;
- **Prerrequisitos**:
  - BEI101 Inglés I (1er Sem-Pág 272)

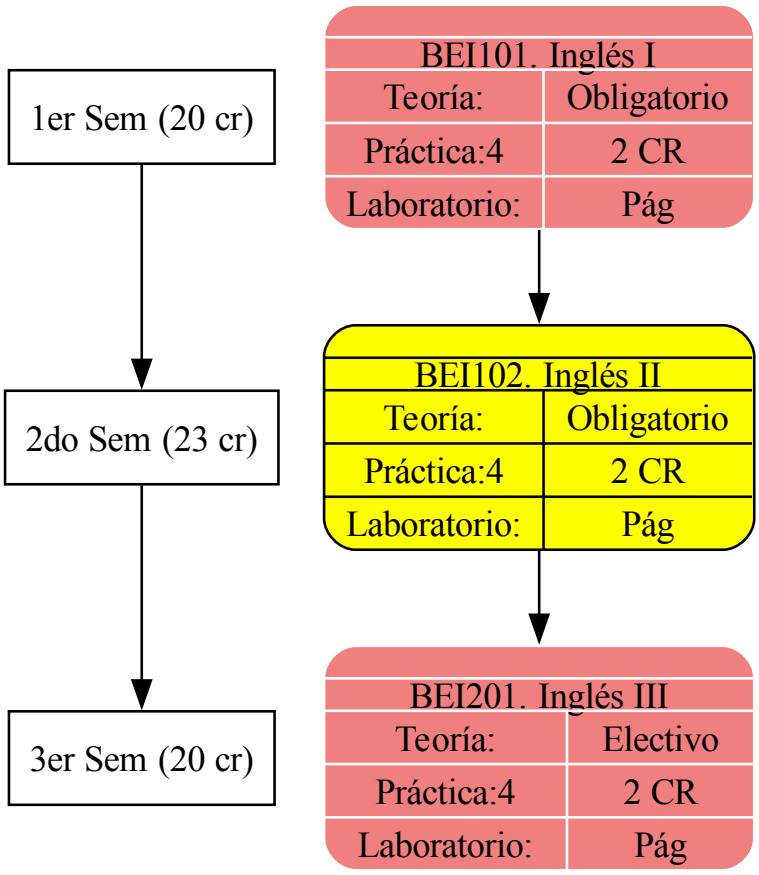

Figura 5.11: Mapa de Conexión. BEI102 Inglés II

2. Fundamentación

Sobre la base de las habilidades fundamentales del inglés, este curso desarrolla la capacidad del estudiante para comunicarse en contextos técnicos y científicos con mayor fluidez. Las estructuras gramaticales se practican mediante la lectura y producción de descripciones de sistemas tecnológicos, fenómenos científicos y procesos de ingeniería. Los estudiantes adquieren vocabulario relevante para la computación, los datos y las ciencias físicas.

3. Objetivos del curso.

- Desarrollar fluidez en inglés aplicado a la comunicación técnica y científica.
- Ampliar el vocabulario en computación, datos y dominios de ingeniería.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Familiarizarse)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Usar)

5. Contenido

5.1 Avances y Descubrimientos, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Repaso de Pasado Simple: Afirmativo y Negativo en contextos científicos. **II:** Expresiones de tiempo con 'ago': Cuándo se inventaron o lanzaron tecnologías. **III:** Conjunciones: and, so, but, because — estructuración de explicaciones técnicas. **IV:** Vocabulario: Hitos científicos, inventos y descubrimientos tecnológicos.

Objetivos de Aprendizaje I: Al finalizar la unidad, el estudiante narra avances tecnológicos pasados y conecta relaciones causa-efecto utilizando el pasado simple y conjunciones (e.g., "Scientists discovered X, so engineers developed Y").

Bibliografía: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.2 Datos y Variables, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Sustantivos contables e incontables en computación: files, data, code, memory. **II:** Uso de some, any, a lot of para cuantificar recursos técnicos. **III:** Expresiones de cantidad: a little memory, a few errors, too many processes. **IV:** Vocabulario: Tipos de datos, variables, constantes y unidades de almacenamiento.

Objetivos de Aprendizaje I: El estudiante utiliza correctamente estructuras de sustantivos contables e incontables para describir datos, recursos y estados del sistema en contextos técnicos.

Bibliografía: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.3 El Futuro de la Tecnología, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Planes futuros: Verb + to + infinitive en la planificación de proyectos. **II:** Uso de 'Will' vs 'Going to' para predicciones y lanzamientos programados. **III:** Verbos auxiliares para pronosticar: tendencias de IA, computación en la nube, computación cuántica. **IV:** Vocabulario: Tecnologías emergentes y trayectorias profesionales en computación.

Objetivos de Aprendizaje I: El estudiante expresa intenciones tecnológicas y planes de proyecto, además de realizar predicciones basadas en evidencia sobre tendencias en computación e ingeniería.

Bibliografía: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.4 Comparando Tecnologías, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Adjetivos comparativos y superlativos aplicados a hardware y software. **II:** Comparación de lenguajes de programación, sistemas operativos y algoritmos. **III:** Sinónimos y antónimos en vocabulario técnico. **IV:** Lectura: Artículos cortos de comparación tecnológica e informes de benchmarks.

Objetivos de Aprendizaje I: El estudiante realiza comparaciones técnicas detalladas entre sistemas, lenguajes y herramientas, utilizando estructuras comparativas y superlativas con precisión.

Bibliografía: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.5 Procesos en Ejecución, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Presente Continuo para describir procesos técnicos activos y en curso. **II:** Uso actual: "The server is processing requests." **III:** Uso futuro con Presente Continuo: despliegues y experimentos programados. **IV:** Redacción: Actualizaciones de estado e informes de avance en proyectos técnicos.

Objetivos de Aprendizaje I: El estudiante describe procesos computacionales activos, experimentos en curso y actividades técnicas programadas utilizando la forma del presente continuo.

Bibliografía: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.6 Hitos del Software, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Introducción al Presente Perfecto para describir logros con relevancia actual. **II:** Palabras clave: ever, never, yet, just — aplicadas a versiones de software e investigación. **III:** Vocabulario: Historial de versiones, parches, actualizaciones, artículos publicados. **IV:** Lectura: Noticias tecnológicas y changelogs de software.

Objetivos de Aprendizaje I: El estudiante comunica logros técnicos e hitos de software con relevancia presente, sin especificar un tiempo exacto, utilizando el presente perfecto.

Bibliografía: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.7 Desafíos Técnicos, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Adjetivos terminados en -ed e -ing para describir experiencias y procesos técnicos. **II:** Verbos modales de obligación: must, have to — aplicados a requisitos de ingeniería. **III:** Vocabulario: Desafíos de ingeniería, restricciones, trade-offs y estándares. **IV:** Redacción: Planteamiento corto de un problema técnico.

Objetivos de Aprendizaje I: El estudiante describe los aspectos técnicos de los desafíos de ingeniería y expresa obligaciones y restricciones utilizando vocabulario especializado.

Bibliografía: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

5.8 Inglés para Tecnología II, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Terminología técnica: Dispositivos de entrada/salida, almacenamiento y periféricos. **II:** Lectura y seguimiento de instrucciones de instalación y configuración de software. **III:** Comunicación en soporte técnico: Descripción de síntomas y soluciones. **IV:** Presentación: Describir un sistema computacional a una audiencia no especializada.

Objetivos de Aprendizaje I: Al finalizar el curso, el estudiante interpreta instrucciones técnicas, describe componentes de hardware y comunica conceptos básicos de computación a audiencias diversas en inglés.

Bibliografía: [Soars and John, 2002, Cambridge, 2006, MacGrew, 1999]

6. Bibliografía

[Cambridge, 2006] Cambridge (2006). *Diccionario Inglés-Espanol Cambridge*. Editorial Oxford.

[MacGrew, 1999] MacGrew, J. (1999). *Focus on Grammar Basic*. Editorial Oxford.

[Soars and John, 2002] Soars, L. and John (2002). *American Headway N 1 Student Book*. Editorial Oxford.

Tercer Semestre

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS113 Programación Orientada a Objetos II (Obligatorio)

1. Información general

- **Semestre**: 3er Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS112 Programación Orientada a Objetos I (2do Sem-Pág 287)

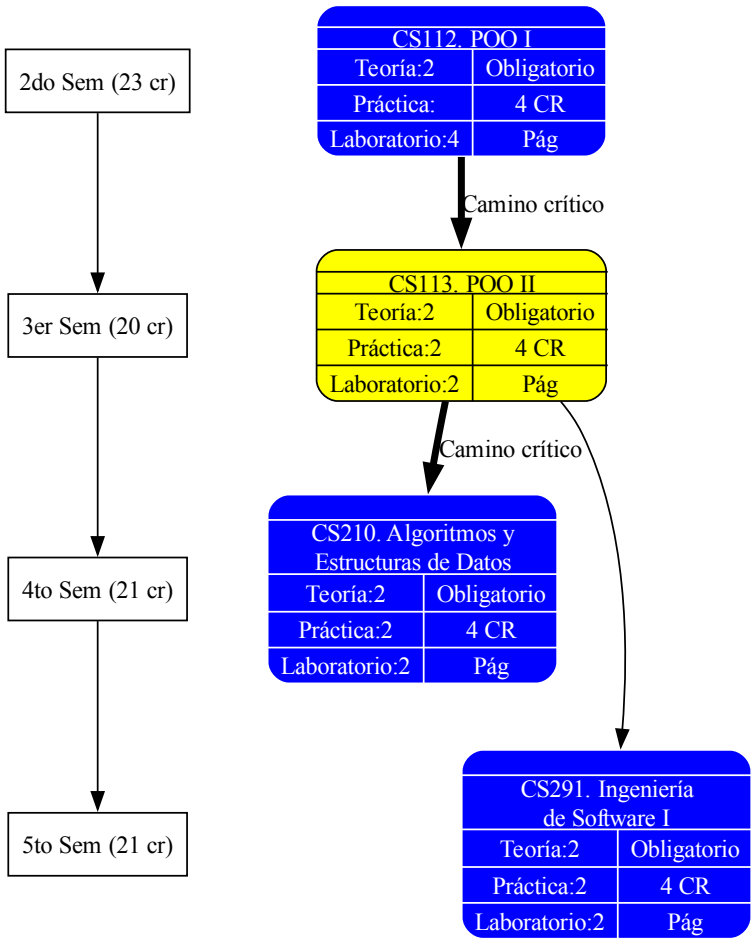

Figura 5.12: Mapa de Conexión. CS113 Programación Orientada a Objetos II

2. Fundamentación

Este es el tercer curso en la secuencia de cursos introductorios en ciencia de la computación. Este curso explora características avanzadas de C++17/C++20 centrándose en el desarrollo de sistemas de alto rendimiento. Los temas clave incluyen metaprogramación con plantillas, semántica de movimiento, reenvío perfecto, optimización RAII, patrones de herencia múltiple/virtual, concurrencia con

std::thread y async/await, programación sin bloqueo, paradigmas modernos de C++ como CRTP y plantillas de expresión, e interfaz de C++ con otros lenguajes. El curso prepara a los estudiantes para motores de juego, computación de alto rendimiento y desarrollo embebido donde C++ es dominante.

3. Objetivos del curso.

- Dominar características avanzadas de C++17/C++20, incluyendo plantillas y metaprogramación.
- Desarrollar habilidades en programación concurrente y de sistemas.
- Comprender técnicas de optimización de rendimiento.
- Prepararse para aplicaciones de computación de alto rendimiento.
- Dominar la metaprogramación con plantillas y el cómputo en tiempo de compilación.

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Evaluar)

5. Contenido

5.1 Programación Orientada a Objetos: Encapsulación, Subtipado y Reflexión, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Clases de colecciones, iteradores y otros componentes comunes de biblioteca. **II:** Contenedores asociativos (std::set, std::map, variantes no ordenadas). **III:** Adaptadores de contenedor (stack, queue, priority_queue). **IV:** Algoritmos avanzados de STL. **V:** Funtores y predicados. **VI:** Categorías de iteradores y rasgos.

Objetivos de Aprendizaje I: Usar clases de colecciones e iteradores efectivamente para resolver un problema [Usar] **II:** Definir y usar iteradores y otras operaciones en agregados, incluyendo operaciones que toman funciones como argumentos, en múltiples lenguajes de programación, seleccionando los idiomas más naturales para cada lenguaje [Evaluar] **III:** Comparar y contrastar los beneficios y costos/impacto de usar herencia (subclases) y composición (específicamente, cómo basar la composición en funciones de orden superior) [Usar] **IV:** Implementar asignadores personalizados para contenedores STL [Usar].

Bibliografía: [Stroustrup, 2013, Josuttis, 2019, Vandevoorde et al., 2017]

5.2 Sistemas de Tipos: Polimorfismo y Tipado Complementario, 12 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Tipos genéricos (polimorfismo paramétrico): *a*) Definición y ventajas del polimorfismo: paramétrico, de subtipado (subtyping), de sobrecarga (overloading) y de coerción *b*) Comparación de tipos monomórficos y polimórficos *c*) Comparación con polimorfismo ad-hoc (sobrecarga) y polimorfismo de subtipos *d*) Parámetros genéricos y tipado *e*) Uso de bibliotecas genéricas como colecciones *f*) Comparación con polimorfismo ad hoc (sobrecarga) y polimorfismo de subtipos *g*) Polimorfismo prescriptivo vs descriptivo *h*) Modelos de implementación de tipos polimórficos *i*) Subtipado **II:** SFINAE (Fallo de Sustitución No es un Error): fundamento histórico de la programación genérica restringida en C++, base de bibliotecas anteriores a C++20. **III:** Concepts (C++20): sintaxis declarativa para restringir parámetros de plantilla, sucesor moderno de SFINAE con mejores diagnósticos de error. **IV:** if constexpr (C++17): ramificación en tiempo de compilación, reemplazo de gran parte del despacho por etiquetas (tag dispatch) antes resuelto con SFINAE. **V:** Reenvío perfecto y referencias universales. **VI:** Plantillas variádicas. **VII:** Cómputo en tiempo de compilación con constexpr y consteval.

Objetivos de Aprendizaje I: Definir y usar piezas de programa (como funciones, clases, métodos) que usen tipos genéricos, incluyendo para colecciones [Evaluar] **II:** Discutir las diferencias entre genéricos, subtipado y sobrecarga [Evaluar] **III:** Aplicar Concepts para restringir parámetros de plantilla y mejorar diagnósticos de error [Usar]. **IV:** Distinguir cuándo usar Concepts, if constexpr o SFINAE heredado al mantener código C++ existente [Evaluar]. **V:** Implementar reenvío perfecto para paso eficiente de argumentos [Usar].

Bibliografía: [Stroustrup, 2013, Vandevoorde et al., 2017, Stroustrup, 2022]

5.3 Gestión de Memoria, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: I: Asignación y acceso de bajo nivel a estructuras de datos de alto nivel como tipos de datos básicos, arreglos n-dimensionales, vectores, registros y objetos **II:** Retorno de procedimiento como mecanismo de desasignación automática para elementos de datos locales en la pila **III:** Gestión manual de memoria: asignar, desasignar y reutilizar memoria del montón **IV:** Gestión automática de memoria: recolección de basura (garbage collection) como técnica automatizada usando la noción de alcanzabilidad (reachability) **V:** Lvalues y rvalues en C++11/14/17/20. **VI:** Constructores de movimiento y operadores de asignación de movimiento. **VII:** Referencias a valor-r y colapso de referencias. **VIII:** Patrón RAII (Adquisición de Recursos es Inicialización). **IX:** Personalización de punteros inteligentes. **X:** Detección y prevención de fugas de memoria.

Objetivos de Aprendizaje I: Explicar por qué ocurren las fugas de memoria (memory leaks) y los problemas de punteros colgantes (dangling pointers), y qué puede hacer un programador para evitarlos/corregirlos [Usar] **II:** Explicar cómo las implementaciones de lenguajes de programación típicamente organizan la memoria en secciones de datos globales, texto, montón y pila, y cómo características como recursión y gestión de memoria se mapean a este modelo de memoria [Evaluar] **III:** Explicar cómo se ejecuta una construcción central del lenguaje, como abstracciones de datos y abstracciones de control [Evaluar] **IV:** Implementar semántica de movimiento para optimización de recursos [Usar]. **V:** Diseñar clases RAII para gestión de recursos segura ante excepciones [Usar].

Bibliografía: [Meyers, 2014, Stroustrup, 2013]

5.4 Computación Paralela y Distribuida, 12 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: I: Seguridad (safety) y vivacidad (liveness): *a*) Condiciones de carrera (race conditions) *b*) Dependencias/precondiciones *c*) Modelos de fallos (fault models) *d*) Terminación **II:** Modelos de programación: *a*) Modelo de actores (*actor models*) *b*) Modelos procedurales y reactivos *c*) Modelos de programación síncrona/asíncrona *d*) Paralelismo de datos (*data parallelism*) **III:** Comunicación y coordinación: *a*) Mutexes *b*) Paso de mensajes (*message-passing*) *c*) Memoria compartida (*shared memory*) *d*) Cobegin-coend *e*) Monitores *f*) Canales (*channels*) *g*) Hilos (*threads*) *h*) Guardas (*guards*) **IV:** std::thread y gestión de hilos. **V:** std::jthread (C++20): unión automática (RAII) y cancelación cooperativa vía stop_token. **VI:** Mutexes, candados y variables de condición. **VII:** Mecanismos de sincronización de C++20: std::latch, std::barrier, std::counting_semaphore. **VIII:** Operaciones atómicas y programación sin bloqueo. **IX:** std::async y futures. **X:** std::execution y el modelo de senders-receivers (C++26, experimental): panorama hacia dónde evoluciona la concurrencia estándar en C++, presentado a nivel conceptual, sin evaluación en laboratorio dada su disponibilidad limitada (parcial en GCC). **XI:** Patrones de diseño seguros para hilos. **XII:** Carreras de datos y prevención de interbloqueos.

Objetivos de Aprendizaje I: Implementar programas concurrentes correctos usando múltiples modelos de programación, como memoria compartida, actores, futuros, construcciones de sincronización y primitivas de paralelismo de datos [Usar] **II:** Usar construcciones de sincronización como métodos

monitorizados/sincronizados en un programa simple [Usar] **III:** Usar un modelo de paso de mensajes para analizar un protocolo de comunicación [Evaluar] **IV:** Implementar estructuras de datos sin bloqueo usando operaciones atómicas [Usar]. **V:** Diferenciar std::thread de std::jthread y justificar cuándo usar cada uno [Usar]. **VI:** Analizar la seguridad en hilos de algoritmos concurrentes [Evaluar].

Bibliografía: [Williams, 2019, Stroustrup, 2013]

5.5 Construcciones de Programación Avanzadas, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Abstracciones orientadas a objetos: herencia múltiple, mixins, rasgos (traits), multimétodos (multimethods) **II:** Metaprogramación: macros, programación generativa, desarrollo basado en modelos (model-based development) **III:** Herencia múltiple y herencia virtual. **IV:** Patrón CRTP (Patrón de Plantilla Curiosamente Recursivo). **V:** deducing this (C++23): simplifica CRTP al eliminar la necesidad de static_cast<Derived&>(*this) para polimorfismo estático. **VI:** Plantillas de expresión. **VII:** Rasgos de tipos e introspección en tiempo de compilación. **VIII:** Diseño basado en políticas. **IX:** Diseño de interfaz para bibliotecas.

Objetivos de Aprendizaje **I:** Usar varias construcciones e idiomas de programación avanzados correctamente [Usar] **II:** Discutir cómo varias construcciones de programación avanzadas tienen como objetivo mejorar la estructura del programa, la calidad del software y la productividad del programador [Evaluar] **III:** Discutir cómo varias construcciones de programación avanzadas interactúan con la definición e implementación de otras características del lenguaje [Evaluar] **IV:** Implementar CRTP para polimorfismo estático [Usar]. **V:** Diseñar interfaces de biblioteca extensibles usando diseño basado en políticas [Usar].

Bibliografía: [Stroustrup, 2013, Gregor, 2017]

5.6 Modelo de Ejecución y Memoria de Sistemas, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Representación en tiempo de ejecución de abstracciones de datos como variables, arreglos, vectores, registros, elementos de datos basados en punteros como listas enlazadas y árboles, y objetos **II:** Estructuras de datos para traducción, ejecución, traducción y movilidad de código como pila, montón (heap), alias (compartición usando punteros), secuencia indexada y cadena **III:** Implementación del despacho dinámico introducido en POO I: tabla de funciones virtuales (vtable), puntero vtable (vptr), y su costo en memoria y rendimiento frente al despacho estático. **IV:** Interfaz de C++ con otros lenguajes (FFI). **V:** Llamadas al sistema e interacción con el SO. **VI:** E/S mapeada en memoria. **VII:** Perfilado de rendimiento y evaluación comparativa. **VIII:** Técnicas de optimización para HPC.

Objetivos de Aprendizaje **I:** Explicar cómo se ejecuta una construcción central del lenguaje, como abstracciones de datos y abstracciones de control [Usar] **II:** Explicar cómo las implementaciones de lenguajes de programación típicamente organizan la memoria en secciones de datos globales, texto, montón y pila, y cómo características como recursión y gestión de memoria se mapean a este modelo de memoria [Evaluar] **III:** Explicar por qué ocurren las fugas de memoria (memory leaks) y los problemas de punteros colgantes (dangling pointers), y qué puede hacer un programador para evitarlos/corregirlos [Usar] **IV:** Perfilar y optimizar programas C++ para rendimiento [Usar]. **V:** Explicar cómo el compilador implementa el polimorfismo en tiempo de ejecución mediante vtables [Evaluar]. **VI:** Interfazar código C++ con bibliotecas C [Usar].

Bibliografía: [Stroustrup, 2013, Williams, 2019]

5.7 Construcciones de Programación Avanzadas, 6 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Abstracciones orientadas a objetos: herencia múltiple, mixins, rasgos (traits), multimétodos (multimethods) **II:** Patrones creacionales, estructurales y de comportamiento. **III:** Patrones singleton seguros para hilos. **IV:** Modismos de diseño modernos en C++. **V:** Inyección de dependencias e inversión de control.

Objetivos de Aprendizaje **I:** Usar varias construcciones e idiomas de programación avanzados correctamente [Usar] **II:** Discutir cómo varias construcciones de programación avanzadas tienen como objetivo mejorar la estructura del programa, la calidad del software y la productividad del programador [Evaluar] **III:** Discutir cómo varias construcciones de programación avanzadas interactúan con la definición e implementación de otras características del lenguaje [Familiarizarse] **IV:** Implementar patrones de diseño seguros para hilos [Usar]. **V:** Evaluar compensaciones arquitectónicas en sistemas C++ [Evaluar].

Bibliografía: [Gamma et al., 1994, Stroustrup, 2013]

5.8 Construcciones de Programación Avanzadas, 6 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Abstracciones de control: manejo de excepciones, continuaciones (continuations), mónadas (monads). **II:** Optimizaciones del compilador y banderas. **III:** Metodologías de evaluación comparativa. **IV:** Estructuras de datos amigables con la caché. **V:** Predicción de bifurcación y alineación. **VI:** Reducción del tiempo de compilación. **VII:** Integración continua para pruebas de rendimiento.

Objetivos de Aprendizaje **I:** Usar varias construcciones e idiomas de programación avanzados correctamente [Usar] **II:** Discutir cómo varias construcciones de programación avanzadas tienen como objetivo mejorar la estructura del programa, la calidad del software y la productividad del programador [Evaluar] **III:** Discutir cómo varias construcciones de programación avanzadas interactúan con la definición e implementación de otras características del lenguaje [Familiarizarse] **IV:** Aplicar técnicas de optimización para mejorar el rendimiento del programa [Usar]. **V:** Interpretar resultados de evaluación comparativa para guiar decisiones de optimización [Usar].

Bibliografía: [Meyers, 2014, Stroustrup, 2013]

5.9 Control de Versiones y CI/CD, 4 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Automatización de procesos de software: *a*) Sistemas de construcción (*build systems*): el valor de builds rápidos, herméticos y reproducibles, comparar/contrastar enfoques para construir un proyecto. *b*) Integración Continua (CI): el uso de automatización y pruebas automatizadas para hacer una validación preliminar de que la revisión actual del tronco se construye y pasa las pruebas (básicas). *c*) Despliegue Continuo (CD): el uso de automatización para liberar de forma automática cada cambio que supera las pruebas hacia el entorno de producción, garantizando entregas frecuentes y confiables. *d*) Gestión de dependencias: actualización de dependencias externas/aguas arriba, gestión de paquetes, SemVer. **II:** Sistemas de construcción para C++: CMake, el estándar de facto de la industria. **III:** Alternativas modernas: xmake, Meson y Bazel – comparación de enfoques declarativos frente a basados en Lua/Python. **IV:** Organización de proyectos multi-archivo: unidades de traducción, cabeceras, bibliotecas estáticas y dinámicas. **V:** Gestión de dependencias en C++ (vcpkg, Conan).

Objetivos de Aprendizaje **I:** Comprender el uso de sistemas de CI/CD como una fuente de verdad para el estado del código compartido del equipo (éxito en la construcción y pruebas) [Familiarizarse] **II:** Configurar un proyecto C++ multi-archivo usando un sistema de construcción declarativo

[Usar]. **III:** Comparar CMake con al menos una alternativa moderna (xmake, Meson o Bazel) en velocidad de compilación y reproducibilidad [Evaluar].

Bibliografía: [Stroustrup, 2013, Stroustrup, 2022]

6. Bibliografía

[Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, Reading, MA, 1st edition.

[Gregor, 2017] Gregor, D. (2017). *Advanced C++: Master the Technique of Template Metaprogramming*. Addison-Wesley Professional, Upper Saddle River, NJ, 1st edition.

[Josuttis, 2019] Josuttis, N. M. (2019). *C++17: The Complete Guide*. Pearson Education, Indianapolis, IN, 1st edition.

[Meyers, 2014] Meyers, S. (2014). *Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14*. O'Reilly Media, Sebastopol, CA, 1st edition.

[Stroustrup, 2013] Stroustrup, B. (2013). *The C++ Programming Language*. Addison-Wesley Professional, Upper Saddle River, NJ, 4th edition.

[Stroustrup, 2022] Stroustrup, B. (2022). *A Tour of C++*. Addison-Wesley Professional, Boston, MA, 3rd edition.

[Vandevoorde et al., 2017] Vandevoorde, D., Josuttis, N. M., and Gregor, D. (2017). *C++ Templates: The Complete Guide*. Addison-Wesley Professional, Upper Saddle River, NJ, 2nd edition.

[Williams, 2019] Williams, A. (2019). *C++ Concurrency in Action*. Manning Publications, Shelter Island, NY, 2nd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS2B1 Desarrollo Basado en Plataformas (Obligatorio)

1. Información general

- **Semestre**: 3er Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - CS112 Programación Orientada a Objetos I (2do Sem-Pág 287)

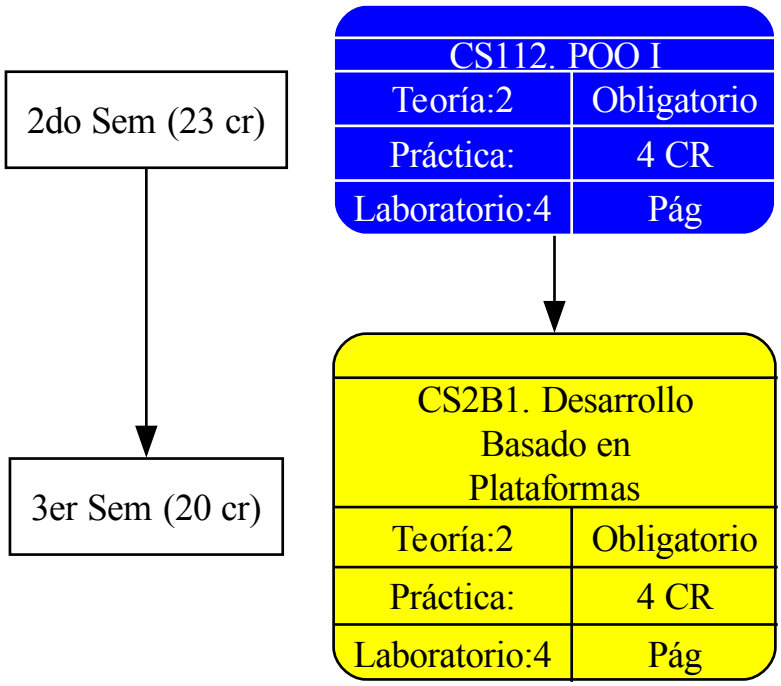

Figura 5.13: Mapa de Conexión. CS2B1 Desarrollo Basado en Plataformas

2. Fundamentación

El desarrollo basado en plataformas se enfoca en el diseño y la implementación de aplicaciones dentro de entornos informáticos específicos. Este curso utiliza el marco de Proceso y Diseño de Software (SPD) para explorar las restricciones y capacidades únicas de las plataformas Web, Móviles e Interactivas. Los estudiantes aprenderán a aplicar preocupaciones arquitectónicas compartidas y APIs específicas de plataforma para construir soluciones de software robustas, escalables y centradas en el usuario.

3. Objetivos del curso.

- Comprender los aspectos compartidos y las restricciones únicas de las plataformas modernas.
- Desarrollar aplicaciones profesionales para entornos Web y Móviles.
- Aplicar principios de computación interactiva para mejorar la experiencia del usuario.
- Diseñar arquitecturas de software que aprovechen los servicios específicos de la plataforma.

4. Competencias Generales (*Outcomes*)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

5. Contenido

5.1 Aspectos Comunes: Plataformas, APIs y Restricciones, 5 hr(s)

Competencias: AG-C04,AG-C09

Tópicos: **I:** Visión general de las plataformas de desarrollo (es decir, web, móvil, juegos, robótica, embebida e interactiva). *a*) Entrada/sensores/dispositivos de control/dispositivos hápticos *b*) Limitaciones de recursos 1) Computacional 2) Almacenamiento de datos 3) Memoria 4) Comunicación *c*) Requisitos - seguridad, disponibilidad de tiempo de actividad, tolerancia a fallos Conceptos de Confiabilidad, Ingeniería de Confiabilidad, Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad *d*) Salida/actuadores/dispositivos hápticos **II:** Programación mediante Interfaces de Programación de Aplicaciones (API) específicas de la plataforma vs construcción de aplicaciones tradicional **III:** Programación bajo restricciones y requisitos de la plataforma (por ejemplo, herramientas de desarrollo disponibles, desarrollo, consideraciones de seguridad) Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad

Objetivos de Aprendizaje **I:** Enumerar las restricciones de la programación móvil [Listar/Enumerar]

Bibliografía: [Fielding, 2000, Gasca, 2021]

5.2 Aspectos Comunes: Lenguajes de Plataforma y Patrones Web, 3 hr(s)

Competencias: AG-C04,AG-C09

Tópicos: **I:** Visión general de los Lenguajes de plataforma (por ejemplo, Python, Swift, Lua, Kotlin) **II:** Técnicas para aprender y dominar un lenguaje de programación específico de la plataforma

Objetivos de Aprendizaje **I:** Enumerar las características de los lenguajes de scripting [Listar/Enumerar] **II:** Describir el modelo de tres capas (three-tier) de la programación web [Describir] **III:** Describir cómo se mantiene el estado en la programación web [Describir]

Bibliografía: [Fielding, 2000, Gasca, 2021]

5.3 Frontend, Frameworks y Estándares Web, 12 hr(s)

Competencias: AG-C04,AG-C09

Tópicos: **I:** Lenguajes de programación web (por ejemplo, HTML5, JavaScript, Typescript, CSS) **II:** Plataformas, frameworks o meta-frameworks web *a*) Servicios en la nube *b*) API, Componentes Web (Web Components) **III:** Software como Servicio (SaaS) **IV:** Estándares web como el modelo de objetos del documento (DOM), accesibilidad Accesibilidad y Diseño Inclusivo **V:** Consideraciones de Seguridad y Privacidad Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad

Objetivos de Aprendizaje **I:** Comparar y contrastar la programación web con la programación de propósito general [Comparar] **II:** Describir las diferencias entre Software-como-Servicio (SaaS) y productos de software tradicionales [Describir] **III:** Discutir cómo los estándares web impactan el desarrollo de software [Debatir] **IV:** Evaluar una aplicación web existente contra los estándares web actuales [Evaluar]

Bibliografía: [Flanagan, 2020, Duckett, 2014, Fielding, 2000]

5.4 Plataformas Móviles: Fundamentos del Desarrollo, 6 hr(s)

Competencias: AG-C04,AG-C09

Tópicos: **I:** Desarrollo con *a*) Lenguajes de programación móvil *b*) Entornos de programación móvil **II:** Restricciones de la plataforma móvil *a*) Diseño de interfaz de usuario Comprender al Usuario: Objetivos individuales e interacciones con otros *b*) Seguridad **III:** Acceso *a*) Acceso a datos a través de APIs Construcción de Consultas *b*) Diseño de puntos finales de API (endpoints) para aplicaciones móviles: trampas y consideraciones de diseño *c*) Interfaces de red y web Modelado de Datos, Fundamentos de Redes y Comunicaciones

Objetivos de Aprendizaje **I:** Comparar la programación móvil con la programación de propósito general [Comparar] **II:** Evaluar los pros y los contras del desarrollo de aplicaciones móviles nativas y multiplataforma [Evaluar]

Bibliografía: [Deitel et al., 2017, Hellman, 2013]

5.5 Plataformas Móviles: Arquitectura y Restricciones, 3 hr(s)

Competencias: AG-C04,AG-C09

Tópicos: **I:** Desarrollo *a*) Desarrollo nativo (native) versus multiplataforma (cross-platform) *b*) Patrones de diseño/arquitectura de software para aplicaciones móviles Diseño de API y Modelado, Principios y Arquitectura de Software, Calidad y Evaluación del Diseño **II:** Restricciones de la plataforma móvil *a*) Diseño de interfaz de usuario responsivo (responsive) Accesibilidad y Diseño Inclusivo *b*) Heterogeneidad y movilidad de los dispositivos *c*) Diferencias en las experiencias de usuario (por ejemplo, entre aplicaciones móviles y basadas en web) *d*) Compensación (tradeoff) entre energía y rendimiento **III:** Especificación y pruebas Prácticas de Desarrollo de Software, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas

Objetivos de Aprendizaje **I:** Desarrollar una aplicación móvil consciente de la ubicación con integración de API de datos [Crear] **II:** Construir una aplicación móvil impulsada por sensores capaz de registrar datos en un servidor remoto [Crear] **III:** Crear una aplicación de comunicación que incorpore telefonía y mensajería instantánea [Crear]

Bibliografía: [Deitel et al., 2017, Hellman, 2013]

5.6 Plataformas Móviles: Capacidades y Computación Asíncrona, 3 hr(s)

Competencias: AG-C04,AG-C09

Tópicos: **I:** Capacidades (*affordances*) de la computación móvil *a*) Aplicaciones conscientes de la ubicación (*location-aware*) *b*) Computación impulsada por sensores (por ejemplo, giroscopio, acelerómetro, datos de salud de un reloj) *c*) Telefonía y mensajería instantánea *d*) Realidad aumentada Inmersión **II:** Computación asíncrona Algoritmos *a*) Diferencia con la programación síncrona tradicional *b*) Manejo del éxito mediante devoluciones de llamada (*callbacks*) *c*) Manejo de errores de forma asíncrona *d*) Pruebas de código asíncrono y problemas típicos en las pruebas

Objetivos de Aprendizaje **I:** Desarrollar una aplicación móvil consciente de la ubicación con integración de API de datos [Crear] **II:** Construir una aplicación móvil impulsada por sensores capaz de registrar datos en un servidor remoto [Crear] **III:** Crear una aplicación de comunicación que incorpore telefonía y mensajería instantánea [Crear]

Bibliografía: [Deitel et al., 2017, Hellman, 2013]

5.7 SPD-SEP/Interactiva, 10 hr(s)

Competencias: AG-C04,AG-C09

Tópicos: **I:** Pautas éticas al usar modelos de IA para ayudar en el periodismo y la creación de contenido **II:** Rendición de cuentas (*accountability*) por las salidas generadas por IA **III:** Comportamiento entre programadores de prompts (*prompt programmers*) y desarrolladores de IA **IV:** Confianza con el público al usar modelos de IA

Objetivos de Aprendizaje **I:** Indicar un marco para la rendición de cuentas (*accountability*) en el despliegue de modelos de IA, incluyendo documentación clara y atribución [Evaluar] **II:** Discutir códigos de conducta éticos y estándares profesionales relevantes para la programación de prompts (*prompt programming*) y el desarrollo de IA [Debatir] **III:** Crear planes y materiales de comunicación para educar al público sobre las capacidades, limitaciones y salvaguardas éticas de la IA [Crear]

Bibliografía: [Duckett, 2014, Deitel et al., 2017]

5.8 Plataformas Interactivas de Datos y Análisis, 6 hr(s)

Competencias: AG-C04,AG-C09

Tópicos: **I:** Plataformas de Cuadernos Computacionales (por ejemplo, Cuadernos Jupyter; Google Colab; R; SPSS; Observable) **II:** Plataformas de Análisis de Datos y SQL en la Nube (por ejemplo, BigQuery) Construcción de Consultas *a*) Apache Spark *b*) Visualizaciones de Datos Visualización **III:** Presentaciones Interactivas Respaldadas por Datos **IV:** Herramientas de Diseño con Bucles de Retroalimentación de Baja Latencia (*Low-Latency Feedback Loops*) *a*) Herramientas de renderizado *b*) Herramientas de diseño gráfico

Objetivos de Aprendizaje **I:** Analizar grandes conjuntos de datos de forma interactiva [Analizar] **II:** Crear una pista de acompañamiento para una interpretación musical, como la codificación en vivo (*live coding*) [Crear] **III:** Crear cuadernos computacionales convincentes que construyan una narrativa para un objetivo/historia periodística dada [Crear] **IV:** Implementar código interactivo que use un conjunto de datos y genere gráficos exploratorios [Implementar] **V:** Contextualizar los atributos de diferentes estilos de análisis de datos, como interactivo versus tubería (*pipeline*) de ingeniería [Contextualizar] **VI:** Escribir un programa usando una plataforma de computación de cuadernos (por ejemplo, búsqueda, ordenación o manipulación de grafos) [Escribir]

Bibliografía: [Flanagan, 2020]

6. Bibliografía

[Deitel et al., 2017] Deitel, P., Deitel, H., and Deitel, A. (2017). *Android How to Program*. Pearson, 3rd edition.

[Duckett, 2014] Duckett, J. (2014). *JavaScript and JQuery: Interactive Front-End Web Development*. Wiley.

[Fielding, 2000] Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine.

[Flanagan, 2020] Flanagan, D. (2020). *JavaScript: The Definitive Guide*. O'Reilly Media, 7th edition.

[Gasca, 2021] Gasca, J. (2021). *Web Security for Developers*. No Starch Press.

[Hellman, 2013] Hellman, E. (2013). *Android Programming: Pushing the Limits*. Wiley.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI161 IA Aplicada (Obligatorio)

1. Información general

- **Semestre**: 3er Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - BIC101 Introducción a la Computación (1er Sem-Pág 257)

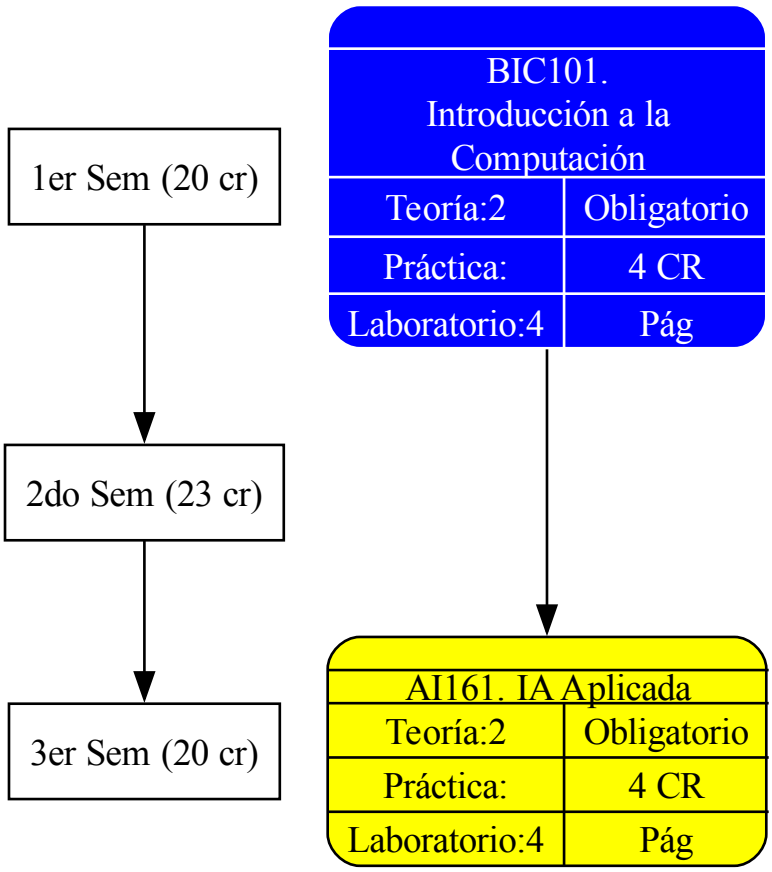

Figura 5.14: Mapa de Conexión. AI161 IA Aplicada

2. Fundamentación

Este curso proporciona una introducción práctica a la Inteligencia Artificial (IA) para estudiantes de todas las disciplinas científicas y de ingeniería. Centrado en desarrollar alfabetización en IA y habilidades prácticas, cubre conceptos fundamentales, herramientas modernas de IA (incluyendo plataformas occidentales y chinas), y uso responsable. Los estudiantes aprenderán a interactuar efectivamente con diversos sistemas de IA, escribir prompts de calidad y aplicar soluciones de IA a problemas en diversos dominios, comprendiendo las implicaciones éticas y contextos culturales del despliegue de IA.

3. Objetivos del curso.

- Desarrollar alfabetización integral en IA comprendiendo conceptos fundamentales, capacidades y limitaciones de los sistemas modernos de IA en diferentes plataformas y culturas.
- Dominar técnicas efectivas de ingeniería de prompts y patrones de interacción con diversas herramientas de IA (Occidentales: ChatGPT, Claude, Gemini; Chinas: DeepSeek, Kimi, ERNIE).
- Aplicar diversas herramientas de IA para resolver problemas prácticos en contextos científicos, de ingeniería y académicos, manteniendo evaluación crítica de los resultados entre plataformas.
- Comprender consideraciones éticas, sesgos, contextos culturales y uso responsable de tecnologías de IA en entornos profesionales y académicos globales.
- Desarrollar flujos de trabajo que integren múltiples herramientas de IA para mejorar la productividad y capacidades de resolución de problemas, comprendiendo fortalezas y especializaciones regionales.

4. Competencias Generales (*Outcomes*)

AG-C01) El Profesional y el Mundo: Analiza y evalúa el impacto de las soluciones a problemas complejos de computación en el desarrollo sostenible de la sociedad. (Familiarizarse)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Usar)

5. Contenido

5.1 Fundamentos de IA y Panorama Global, 16 hr(s)

Competencias: AG-C01,AG-C11

Tópicos: I: ¿Qué es la IA? Definiciones, historia y panorama global actual. **II:** Tipos de sistemas de IA: chatbots, generadores de imágenes, asistentes de investigación entre regiones. **III:** Ecosistemas occidentales de IA: OpenAI, Anthropic, Google, Microsoft. **IV:** Ecosistemas chinos de IA: DeepSeek, Kimi, ERNIE, Zhipu AI, Baidu. **V:** Capacidades y limitaciones de la IA: análisis comparativo entre plataformas. **VI:** Alfabetización digital en la era de la IA: pensamiento crítico sobre diversos resultados de IA.

Objetivos de Aprendizaje I: Aplicar conocimientos fundamentales de IA para identificar tipos de sistemas y sus capacidades [Usar]. **II:** Analizar sistemas inteligentes globales y sus características distintivas [Evaluar]. **III:** Evaluar críticamente los resultados de IA en diferentes contextos culturales [Evaluar].

Bibliografía: [Ng, 2019, Mollick and Mollick, 2024, UNESCO, 2023, Institute, 2024]

5.2 Interacción Efectiva con IA e Ingeniería de Prompts Multiplataforma, 20 hr(s)

Competencias: AG-C01,AG-C11

Tópicos: I: Principios de escritura efectiva de prompts: claridad, contexto, restricciones entre plataformas. **II:** Patrones de prompts: persona, plantilla, cadena de pensamiento, few-shot para diferentes sistemas de IA. **III:** Herramientas occidentales: técnicas específicas para ChatGPT, Claude, Gemini, Copilot. **IV:** Herramientas chinas: características específicas y mejores prácticas para DeepSeek, Kimi, ERNIE. **V:** Estrategias multiplataforma: aprovechar diferentes fortalezas de IA para tareas complejas. **VI:** Refinamiento iterativo: cómo mejorar prompts basándose en resultados de diversos sistemas. **VII:** Sesiones prácticas: talleres comparativos de escritura de prompts entre plataformas.

Objetivos de Aprendizaje I: Diseñar soluciones de prompts efectivas para diversos sistemas de IA [Evaluar]. **II:** Desarrollar flujos de trabajo multiplataforma que integren diferentes herramientas de IA [Usar]. **III:** Implementar estrategias de refinamiento iterativo para optimizar resultados de IA [Usar].

Bibliografía: [White et al., 2023, Prompt Engineering Institute, 2024, Microsoft, 2024, DeepSeek, 2024]

5.3 Aplicaciones de IA entre Disciplinas y Plataformas, 16 hr(s)

Competencias: AG-C01,AG-C11

Tópicos: **I:** IA para investigación: revisión de literatura, análisis de datos usando múltiples asistentes de IA. **II:** IA para escritura: documentos académicos, informes con verificación multiplataforma. **III:** IA para resolución de problemas: cálculos científicos, diseño de ingeniería con herramientas especializadas. **IV:** IA para creatividad: lluvia de ideas, desarrollo de conceptos entre contextos culturales. **V:** Fortalezas específicas por plataforma: cuándo usar herramientas de IA occidentales vs. chinas. **VI:** Talleres disciplinares: aplicaciones personalizadas usando diversos ecosistemas de IA. **VII:** Estudios de caso: aplicaciones en el mundo real en investigación científica entre regiones.

Objetivos de Aprendizaje **I:** Aplicar fundamentos de IA para resolver problemas específicos de cada disciplina [Usar]. **II:** Diseñar soluciones que integren múltiples sistemas inteligentes para tareas complejas [Evaluar]. **III:** Desarrollar aplicaciones prácticas usando herramientas de IA occidentales y chinas [Usar]. **IV:** Implementar flujos de trabajo que aprovechen fortalezas específicas de cada plataforma [Evaluar].

Bibliografía: [Mollick, 2023, Cloud, 2024b, Cloud, 2024a, OpenAI, 2024]

5.4 Ética Global en IA y Uso Responsable, 12 hr(s)

Competencias: AG-C01,AG-C11

Tópicos: **I:** Comprensión de sesgos en IA: perspectivas culturales occidentales y orientales. **II:** Marcos éticos: comparando la Ley de IA de la UE, regulaciones chinas y estándares globales. **III:** Integridad académica: citación apropiada y uso de IA en trabajos académicos entre plataformas. **IV:** Privacidad y seguridad de datos: diferencias regionales en el manejo de datos de IA. **V:** Sensibilidad cultural: navegando resultados de IA en contextos globales. **VI:** Impacto ambiental: consideraciones de sostenibilidad de diferentes sistemas de IA. **VII:** Desarrollo de pautas personales para el uso ético de IA en entornos internacionales. **VIII:** Estudios de caso: dilemas éticos en el despliegue de IA entre regiones.

Objetivos de Aprendizaje **I:** Demostrar responsabilidad profesional en el uso de tecnologías de IA [Evaluar]. **II:** Aplicar principios éticos profesionales en contextos globales de IA [Usar]. **III:** Desarrollar pautas éticas para mitigar sesgos en sistemas de IA [Evaluar]. **IV:** Evaluar implicaciones éticas del despliegue de IA en diferentes culturas [Usar].

Bibliografía: [UNESCO, 2023, European Union, 2024, Bender et al., 2021, Chinese Academy of Cyberspace Studies, 2024, Tencent AI Lab, 2023]

6. Bibliografía

[Bender et al., 2021] Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21)*, pages 610–623. Association for Computing Machinery. Foundational paper on AI ethics with ongoing relevance.

[Chinese Academy of Cyberspace Studies, 2024] Chinese Academy of Cyberspace Studies (2024). Ai governance and ethics in china. Technical report, Cyberspace Administration of China. Official Chinese perspective on AI governance and ethical standards.

[Cloud, 2024a] Cloud, A. (2024a). AI Ethics and best practices in the chinese context. Chinese perspective on AI ethics and implementation guidelines.

[Cloud, 2024b] Cloud, G. (2024b). Introduction to responsible ai. Google's updated framework for responsible AI development and use.

[DeepSeek, 2024] DeepSeek (2024). Deepseek model documentation and best practices. Official documentation for DeepSeek AI models and usage guidelines.

[European Union, 2024] European Union (2024). AI Act: Regulatory framework. Technical report, European Commission. Comprehensive AI regulation framework effective in 2024.

[Institute, 2024] Institute, T. R. (2024). Ai development in china: Current status and future trends. Technical report, Tencent Research Institute. Comprehensive analysis of Chinese AI ecosystem development.

[Microsoft, 2024] Microsoft (2024). Prompt crafting for ai systems. Official Microsoft prompt engineering guidance updated for 2024.

[Mollick, 2023] Mollick, E. (2023). Chatgpt and how ai disrupts industries. *Harvard Business Review*, 101(5):42–51. Analysis of AI's practical impact across global sectors.

[Mollick and Mollick, 2024] Mollick, E. and Mollick, L. (2024). *Co-Intelligence: Living and Working with AI*. Portfolio. Practical guide for human-AI collaboration.

[Ng, 2019] Ng, A. (2019). AI for everyone. Technical report, DeepLearning.AI. Continuously updated online course, available on Coursera.

[OpenAI, 2024] OpenAI (2024). Best practices for prompt engineering. Updated official prompt engineering guidelines from OpenAI.

[Prompt Engineering Institute, 2024] Prompt Engineering Institute (2024). Prompt engineering guide. `https://www.promptingguide.ai/`. Recurso integral de ingeniería de prompts actualizado continuamente. Accedido: 15 de enero de 2024.

[Tencent AI Lab, 2023] Tencent AI Lab (2023). Responsible ai practices in the chinese technology industry. Technical report, Tencent AI Lab. Industry perspective on AI ethics from a leading Chinese technology company.

[UNESCO, 2023] UNESCO (2023). AI and education: Guidance for policy-makers. Technical report, UNESCO. Latest guidance on AI literacy and ethical implementation in education.

[White et al., 2023] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., and Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt. Peer-reviewed research on prompt engineering patterns.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BMA104 Cálculo Diferencial e Integral Avanzado (Obligatorio)

1. Información general

- **Semestre**: 3er Sem. **Créditos**: 5
- **Horas del curso**: **Teoría**: 4 horas; **Práctica**: 2 horas;
- **Prerrequisitos**:
  - BMA102 Cálculo Diferencial (1er Sem-Pág 264)

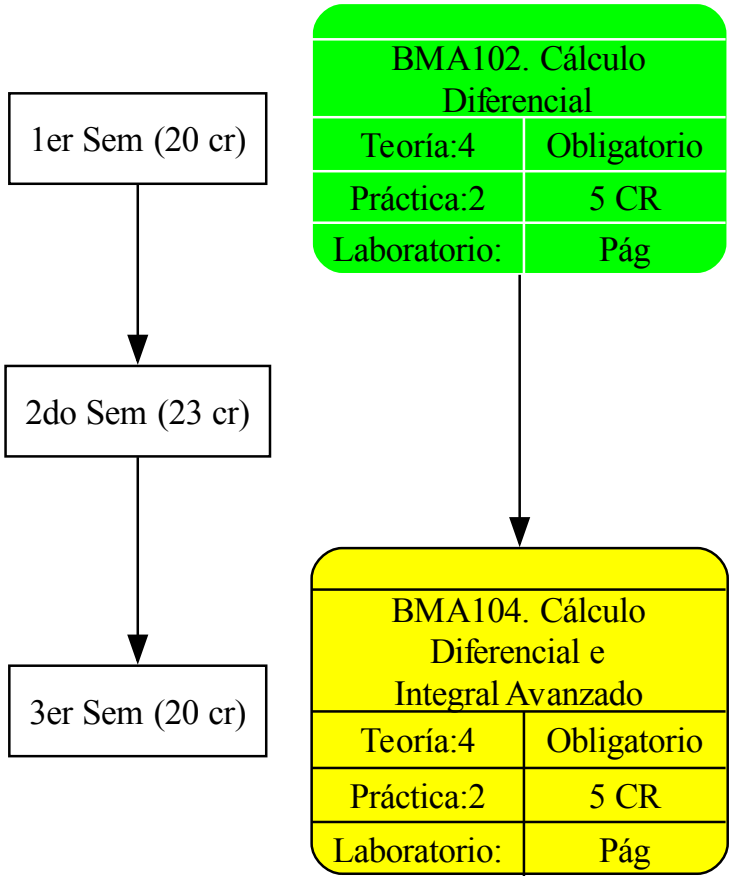

Figura 5.15: Mapa de Conexión. BMA104 Cálculo Diferencial e Integral Avanzado

2. Fundamentación

El cálculo diferencial e integral avanzado es fundamental para la comprensión rigurosa de los fenómenos multidimensionales en ingeniería y ciencias de la computación. Este curso aborda el estudio de funciones vectoriales, el cálculo multivariable, las transformaciones diferenciables, las integrales múltiples y el cálculo integral en variedades, incluyendo los grandes teoremas de Green, Stokes y Gauss. Estas herramientas son esenciales para modelar sistemas complejos, optimizar funciones multivariables y resolver problemas en gráficos por computadora, visión artificial y simulaciones físicas.

3. Objetivos del curso.

- Estudiar las funciones vectoriales de variable real y la geometría diferencial de curvas en el espacio.
- Aplicar el cálculo diferencial a funciones de varias variables: derivadas parciales, gradiente, plano tangente, optimización y Teorema de Taylor.
- Analizar transformaciones diferenciables entre espacios vectoriales mediante la matriz Jacobiana y los teoremas de función inversa e implícita.
- Evaluar integrales múltiples en distintos sistemas de coordenadas y calcular áreas, volúmenes y centros de masa.
- Calcular integrales de línea y aplicar el Teorema de Green y el concepto de campos conservativos.
- Calcular integrales de superficie y aplicar los Teoremas de Stokes y de la Divergencia.

4. Competencias Generales (*Outcomes*)

AG-C07) Conocimientos de Computación: Aplica conocimientos de matemáticas, ciencias y computación. (Evaluar)

5. Contenido

5.1 Funciones Vectoriales de una Variable Real, 14 hr(s)

Competencias: AG-C07

Tópicos: **I:** Definición, dominio y rango de funciones vectoriales de una variable real **II:** Límites, continuidad y derivadas de funciones vectoriales **III:** Longitud de arco y parametrización natural de una curva **IV:** Vectores tangente, normal y binormal; triedro móvil de Frenet **V:** Curvatura, torsión y fórmulas de Frenet-Serret

Objetivos de Aprendizaje **I:** Identificar límites, continuidad y derivadas de funciones vectoriales de una variable real [Familiarizarse] **II:** Calcular vectores tangente, normal y binormal y reparametrizar una curva por longitud de arco [Usar] **III:** Calcular curvatura y torsión usando las fórmulas de Frenet-Serret [Evaluar]

Bibliografía: [Apostol, 1997, Marsden and Tromba, 2011]

5.2 Cálculo Diferencial Multivariable, 18 hr(s)

Competencias: AG-C07

Tópicos: **I:** Límites y continuidad de funciones de varias variables en $\mathbb{R}^n$ **II:** Derivadas parciales, derivadas direccionales y el vector gradiente **III:** Plano tangente, diferencial total y aproximación lineal **IV:** Extremos de funciones multivariables y multiplicadores de Lagrange **V:** Teorema de Taylor para funciones de varias variables

Objetivos de Aprendizaje **I:** Evaluar límites y determinar la continuidad de funciones de varias variables [Familiarizarse] **II:** Calcular gradientes, derivadas direccionales y planos tangentes para funciones multivariables [Usar] **III:** Aplicar los multiplicadores de Lagrange para resolver problemas de optimización con restricciones en computación y ciencia de datos [Usar] **IV:** Expandir funciones de varias variables usando el Teorema de Taylor y estimar cotas de error [Evaluar]

Bibliografía: [Apostol, 1997, Marsden and Tromba, 2011]

5.3 Transformaciones Multivariables, 12 hr(s)

Competencias: AG-C07

Tópicos: **I:** Transformaciones de $\mathbb{R}^n$ a $\mathbb{R}^m$ y su diferenciabilidad **II:** Matriz jacobiana y determinante jacobiano **III:** Cambio de variable en coordenadas polares, cilíndricas y esféricas **IV:** Teorema de la función inversa **V:** Teorema de la función implícita en contextos multivariables

Objetivos de Aprendizaje I: Describir transformaciones entre sistemas de coordenadas usando la matriz jacobiana [Familiarizarse] **II:** Calcular el jacobiano y aplicar cambio de variable para evaluar integrales múltiples [Usar] **III:** Aplicar los teoremas de la función inversa y de la función implícita para analizar invertibilidad local [Evaluar]

Bibliografía: [Apostol, 1997, Marsden and Tromba, 2011]

5.4 Integrales Múltiples, 18 hr(s)

Competencias: AG-C07

Tópicos: I: Integrales dobles sobre rectángulos y regiones generales; integrales iteradas (teorema de Fubini) **II:** Integrales dobles en coordenadas polares y fórmula de cambio de variable **III:** Integrales triples en coordenadas cartesianas y aplicaciones a volumen y masa **IV:** Integrales triples en coordenadas cilíndricas y esféricas **V:** Aplicaciones: área, volumen, centros de masa y momentos de inercia

Objetivos de Aprendizaje I: Evaluar integrales dobles y triples sobre regiones generales usando integrales iteradas [Usar] **II:** Aplicar cambios de coordenadas en sistemas polares, cilíndricos y esféricos para simplificar integrales múltiples [Usar] **III:** Calcular áreas, volúmenes y centros de masa usando integrales múltiples [Evaluar]

Bibliografía: [Apostol, 1997, Marsden and Tromba, 2011]

5.5 Integrales de Línea y Campos Vectoriales, 10 hr(s)

Competencias: AG-C07

Tópicos: I: Campos vectoriales: definición, visualización, divergencia y rotacional **II:** Integrales de línea de funciones escalares: longitud de arco y masa **III:** Integrales de línea de campos vectoriales: trabajo y circulación **IV:** Campos conservativos, funciones potencial e independencia de la trayectoria **V:** Teorema de Green y sus aplicaciones en el plano

Objetivos de Aprendizaje I: Calcular integrales de línea escalares y vectoriales sobre curvas orientadas [Usar] **II:** Determinar si un campo vectorial es conservativo y construir su función potencial [Usar] **III:** Aplicar el Teorema de Green para evaluar integrales de línea mediante integrales dobles [Evaluar]

Bibliografía: [Apostol, 1997, Marsden and Tromba, 2011]

5.6 Integrales de Superficie y Teoremas Fundamentales, 12 hr(s)

Competencias: AG-C07

Tópicos: I: Superficies paramétricas, plano tangente y área de superficie **II:** Integrales de superficie de funciones escalares **III:** Integrales de flujo: integrales de superficie de campos vectoriales **IV:** Teorema de Stokes: relación entre integrales de línea e integrales de superficie **V:** Teorema de la Divergencia (Gauss): relación entre flujo e integral de volumen

Objetivos de Aprendizaje I: Parametrizar superficies y evaluar integrales de superficie escalares y de flujo [Usar] **II:** Aplicar los Teoremas de Stokes y de la Divergencia para reducir integrales complejas [Evaluar] **III:** Relacionar propiedades diferenciales locales (rotacional, divergencia) con el comportamiento global mediante teoremas integrales [Evaluar]

Bibliografía: [Apostol, 1997, Marsden and Tromba, 2011]

6. Bibliografía

[Apostol, 1997] Apostol, T. M. (1997). *Calculus, Vol. II: Multi-Variable Calculus and Linear Algebra with Applications*. John Wiley & Sons, 2nd edition.

[Marsden and Tromba, 2011] Marsden, J. E. and Tromba, A. (2011). *Vector Calculus*. W. H. Freeman and Company, 6th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

STA251 Estadística y Probabilidades (Obligatorio)

1. Información general

- **Semestre**: 3er Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas;
- **Prerrequisitos**:
  - BMA102 Cálculo Diferencial (1er Sem-Pág 264)

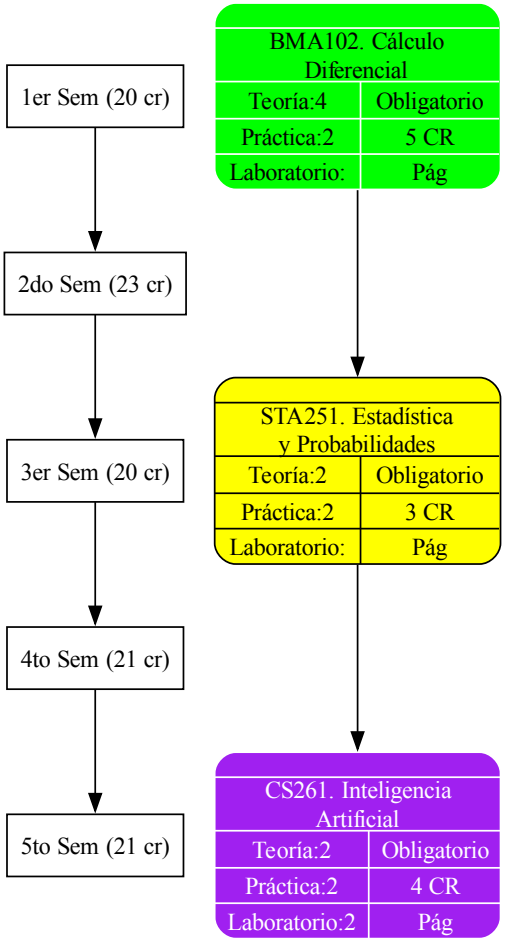

Figura 5.16: Mapa de Conexión. STA251 Estadística y Probabilidades

2. Fundamentación

La probabilidad y la estadística proporcionan la base matemática para razonar bajo incertidumbre, analizar datos y construir modelos predictivos — habilidades esenciales tanto en computación, inteligencia artificial, matemática computacional como en ingeniería. Este curso desarrolla una comprensión rigurosa de la teoría de la probabilidad y su uso para describir fenómenos aleatorios, junto con los fundamentos de la estadística descriptiva, la inferencia estadística (estimación, pruebas de hipótesis y razonamiento bayesiano) y el modelado de regresión, de modo que los estudiantes puedan resumir y

analizar datos, cuantificar la incertidumbre y extraer conclusiones cuantitativas sólidas en su propio campo de aplicación.

3. Objetivos del curso.

- Comprender los fundamentos de la teoría de la probabilidad y la estadística descriptiva.
- Aplicar distribuciones de probabilidad y métodos de inferencia estadística para analizar datos y cuantificar la incertidumbre.
- Desarrollar habilidades para modelar fenómenos aleatorios y construir modelos estadísticos predictivos como la regresión lineal.

4. Competencias Generales (*Outcomes*)

AG-C07) Conocimientos de Computación: Aplica conocimientos de matemáticas, ciencias y computación. (Familiarizarse)

5. Contenido

5.1 Resúmenes Numéricos: Posición, Dispersión y Forma, 4 hr(s)

Competencias: AG-C07

Tópicos: **I:** Medidas de posición: media, mediana, moda, media recortada y media ponderada **II:** Medidas de dispersión: varianza, desviación estándar, RIQ, MAD y rango **III:** Medidas de forma: asimetría y curtosis; cuantiles y percentiles **IV:** Estadísticas de resumen robustas: resistencia a valores atípicos y distribuciones de colas pesadas

Objetivos de Aprendizaje **I:** Calcular e interpretar medidas de posición, dispersión y forma para un conjunto de datos univariado [Usar] **II:** Seleccionar estadísticas de resumen apropiadas según las propiedades distribucionales de los datos [Usar] **III:** Comparar estadísticas de resumen clásicas y robustas y justificar la elección para un conjunto de datos dado [Evaluar]

Bibliografía: [Devore, 2016]

5.2 Espacios Muestrales y Eventos, 4 hr(s)

Competencias: AG-C07

Tópicos: **I:** Experimentos aleatorios, resultados y espacios muestrales: finitos, contables e incontables **II:** Operaciones de teoría de conjuntos sobre eventos: unión, intersección, complemento y leyes de De Morgan **III:** Sigma-álgebras y espacios medibles como fundamento de la probabilidad rigurosa **IV:** Conjuntos de Borel y la sigma-álgebra de Borel sobre la recta real

Objetivos de Aprendizaje **I:** Definir espacios muestrales, eventos y sigma-álgebras para un experimento aleatorio dado [Familiarizarse] **II:** Aplicar operaciones de teoría de conjuntos para calcular probabilidades de eventos compuestos [Usar] **III:** Construir sigma-álgebras apropiadas para espacios medibles que surgen en problemas estadísticos [Evaluar]

Bibliografía: [Ross, 2014, Devore, 2016]

5.3 Axiomas y Propiedades de la Probabilidad, 4 hr(s)

Competencias: AG-C07

Tópicos: **I:** Los tres axiomas de probabilidad de Kolmogorov y sus consecuencias **II:** Propiedades derivadas: regla del complemento, regla de la adición, monotonicidad y continuidad de la probabilidad **III:** Interpretaciones clásica, frecuentista y subjetiva (bayesiana) de la probabilidad **IV:** Métodos de conteo: permutaciones, combinaciones y el principio de multiplicación

Objetivos de Aprendizaje I: Enunciar los axiomas de Kolmogorov y derivar propiedades fundamentales de la probabilidad [Familiarizarse] **II:** Calcular probabilidades de eventos utilizando la regla de la adición y la regla del complemento [Usar] **III:** Aplicar métodos de conteo para calcular probabilidades en espacios muestrales finitos [Usar]

Bibliografía: [Ross, 2014, Devore, 2016]

5.4 Probabilidad Condicional e Independencia, 4 hr(s)

Competencias: AG-C07

Tópicos: I: Definición e interpretación de la probabilidad condicional P(A|B) y sus propiedades **II:** Teorema de Bayes y ley de la probabilidad total: derivación y aplicaciones **III:** Independencia de eventos: independencia por pares e independencia mutua **IV:** Independencia condicional y su papel en los modelos gráficos probabilísticos

Objetivos de Aprendizaje I: Explicar la probabilidad condicional y la independencia y su relación [Familiarizarse] **II:** Aplicar el teorema de Bayes y la ley de la probabilidad total para resolver problemas de probabilidad [Usar] **III:** Analizar si los eventos son independientes o condicionalmente independientes en un modelo de probabilidad dado [Evaluar]

Bibliografía: [Ross, 2014, Devore, 2016]

5.5 Variables Aleatorias Discretas y Distribuciones, 4 hr(s)

Competencias: AG-C07

Tópicos: I: Funciones de masa de probabilidad (FMP) y funciones de distribución acumulada (FDA) para variables aleatorias discretas **II:** Distribuciones discretas clave: Bernoulli, Binomial, Geométrica, Binomial Negativa, Poisson, Hipergeométrica **III:** Propiedades de las distribuciones discretas: media, varianza, asimetría y curtosis **IV:** Aproximación de Poisson a la Binomial y argumentos de distribución límite

Objetivos de Aprendizaje I: Identificar y describir las principales distribuciones de probabilidad discretas y sus parámetros [Familiarizarse] **II:** Calcular probabilidades, medias y varianzas utilizando las FMP de distribuciones discretas estándar [Usar] **III:** Seleccionar una distribución discreta apropiada para modelar un fenómeno aleatorio dado y justificar la elección [Evaluar]

Bibliografía: [Ross, 2014, Devore, 2016]

5.6 Variables Aleatorias Continuas y Distribuciones, 4 hr(s)

Competencias: AG-C07

Tópicos: I: Funciones de densidad de probabilidad (FDP) y FDA para variables aleatorias continuas **II:** Distribuciones continuas clave: Uniforme, Normal, Exponencial, Gamma, Beta, Chi-cuadrado, t, F **III:** La distribución Normal: propiedades, normal estándar y su papel central en la inferencia estadística **IV:** La familia exponencial de distribuciones: forma canónica y estadísticos suficientes

Objetivos de Aprendizaje I: Describir las propiedades y parámetros de las principales distribuciones de probabilidad continuas [Familiarizarse] **II:** Calcular probabilidades y cuantiles para distribuciones continuas estándar [Usar] **III:** Identificar distribuciones pertenecientes a la familia exponencial y derivar sus estadísticos suficientes [Evaluar]

Bibliografía: [Ross, 2014, Devore, 2016]

5.7 Distribuciones Conjuntas, Marginales y Condicionales, 4 hr(s)

Competencias: AG-C07

Tópicos: **I:** FMP y FDP conjuntas para dos o más variables aleatorias **II:** Distribuciones marginales y condicionales derivadas de distribuciones conjuntas **III:** Covarianza, correlación y sus propiedades; la independencia implica covarianza cero (el recíproco no siempre es cierto) **IV:** La distribución normal bivariada y sus distribuciones condicionales

Objetivos de Aprendizaje **I:** Derivar distribuciones marginales y condicionales a partir de una distribución conjunta dada [Usar] **II:** Calcular la covarianza y correlación entre dos variables aleatorias [Usar] **III:** Analizar la estructura condicional de la distribución normal bivariada [Evaluar]

Bibliografía: [Ross, 2014, Devore, 2016]

5.8 Teoremas Límite: LGN y TCL, 4 hr(s)

Competencias: AG-C07

Tópicos: **I:** Ley débil de los grandes números: convergencia en probabilidad y sus implicaciones para la estimación **II:** Teorema central del límite: enunciado, condiciones y papel en la justificación de aproximaciones normales **III:** Aplicaciones del TCL: aproximación normal a la binomial y distribuciones muestrales **IV:** Ley fuerte de los grandes números: convergencia casi segura **V:** El método delta: distribución asintótica de funciones suaves de estimadores

Objetivos de Aprendizaje **I:** Enunciar la LGN y el TCL y explicar las condiciones bajo las cuales se cumplen [Familiarizarse] **II:** Aplicar el TCL para derivar aproximaciones normales y distribuciones muestrales [Usar] **III:** Aplicar el método delta para derivar distribuciones asintóticas de estimadores no lineales [Evaluar]

Bibliografía: [Ross, 2014, Devore, 2016]

5.9 Estimación Puntual, 4 hr(s)

Competencias: AG-C07

Tópicos: **I:** Propiedades deseables: insesgamiento, consistencia, eficiencia y suficiencia **II:** Estimación por método de momentos: derivación y propiedades asintóticas **III:** Estimación de máxima verosimilitud: derivación, propiedades y principio de invarianza **IV:** Cota inferior de Cramér-Rao y el concepto de información de Fisher

Objetivos de Aprendizaje **I:** Describir las propiedades clave de los estimadores y explicar las concesiones entre ellas [Familiarizarse] **II:** Derivar estimadores por método de momentos y de máxima verosimilitud para distribuciones estándar [Usar] **III:** Evaluar la eficiencia de un estimador usando la cota inferior de Cramér-Rao [Evaluar]

Bibliografía: [Devore, 2016]

5.10 Estimación por Intervalos e Intervalos de Confianza, 4 hr(s)

Competencias: AG-C07

Tópicos: **I:** Construcción de intervalos de confianza mediante cantidades pivote y distribuciones exactas **II:** Interpretación correcta de los intervalos de confianza: probabilidad de cobertura y errores conceptuales comunes **III:** Intervalos de confianza para muestras grandes basados en normalidad asintótica **IV:** Dualidad entre intervalos de confianza y pruebas de hipótesis

Objetivos de Aprendizaje **I:** Construir intervalos de confianza para medias, proporciones y varianzas usando métodos exactos y de gran muestra [Usar] **II:** Interpretar correctamente los intervalos de confianza e identificar interpretaciones erróneas comunes [Usar] **III:** Explicar la dualidad entre intervalos de confianza y pruebas de hipótesis y utilizarla para extraer inferencias [Evaluar]

Bibliografía: [Devore, 2016]

5.11 Fundamentos de las Pruebas de Hipótesis, 4 hr(s)

Competencias: AG-C07

Tópicos: **I:** Marco de Neyman-Pearson: hipótesis nula y alternativa, regiones de rechazo, errores tipo I y II **II:** Valores p: definición, interpretación correcta y usos incorrectos comunes; potencia y tamaño de muestra **III:** Pruebas paramétricas comunes: prueba z, prueba t, prueba F y prueba ji-cuadrada **IV:** Lema de Neyman-Pearson y pruebas uniformemente más poderosas

Objetivos de Aprendizaje **I:** Explicar la lógica de las pruebas de hipótesis, los errores tipo I y II y el significado de un valor p [Familiarizarse] **II:** Aplicar pruebas paramétricas estándar e interpretar los resultados en contexto [Usar] **III:** Derivar la prueba más poderosa para una hipótesis simple usando el lema de Neyman-Pearson [Evaluar]

Bibliografía: [Devore, 2016]

5.12 Inferencia Bayesiana, 4 hr(s)

Competencias: AG-C07

Tópicos: **I:** Marco bayesiano: a priori, verosimilitud, a posteriori y el papel del teorema de Bayes **II:** Distribuciones a priori: informativas, no informativas y conjugadas **III:** Inferencia a posteriori: estimaciones puntuales, intervalos de credibilidad y distribuciones predictivas a posteriori **IV:** Comparación de modelos bayesianos: factores de Bayes y probabilidades a posteriori de modelos

Objetivos de Aprendizaje **I:** Explicar el enfoque bayesiano de la inferencia y contrastarlo con el enfoque frecuentista [Familiarizarse] **II:** Derivar la distribución a posteriori para modelos conjugados estándar [Usar] **III:** Aplicar factores de Bayes para comparar modelos estadísticos e interpretar la evidencia [Evaluar]

Bibliografía: [Ross, 2014, Devore, 2016]

5.13 Regresión Lineal Simple, 4 hr(s)

Competencias: AG-C07

Tópicos: **I:** Estimación por mínimos cuadrados ordinarios (MCO): derivación de coeficientes y sus propiedades **II:** Inferencia sobre coeficientes de regresión: pruebas t, intervalos de confianza y valores p **III:** Coeficiente de determinación R-cuadrado y descomposición de la varianza mediante análisis de varianza **IV:** Teorema de Gauss-Markov: propiedad MELI del MCO bajo supuestos clásicos

Objetivos de Aprendizaje **I:** Ajustar un modelo de regresión lineal simple e interpretar los coeficientes estimados [Usar] **II:** Realizar pruebas de hipótesis y construir intervalos de confianza para parámetros de regresión [Usar] **III:** Verificar los supuestos de Gauss-Markov y evaluar las consecuencias cuando se violan [Evaluar]

Bibliografía: [Devore, 2016]

6. Bibliografía

[Devore, 2016] Devore, J. L. (2016). *Probability and Statistics for Engineering and the Sciences*. Cengage Learning.

[Ross, 2014] Ross, S. M. (2014). *A First Course in Probability*. Pearson.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BEI201 Inglés III (Electivo)

1. Información general

- **Semestre**: 3er Sem. **Créditos**: 2
- **Horas del curso**: **Práctica**: 4 horas;
- **Prerrequisitos**:
  - BEI102 Inglés II (2do Sem-Pág 302)

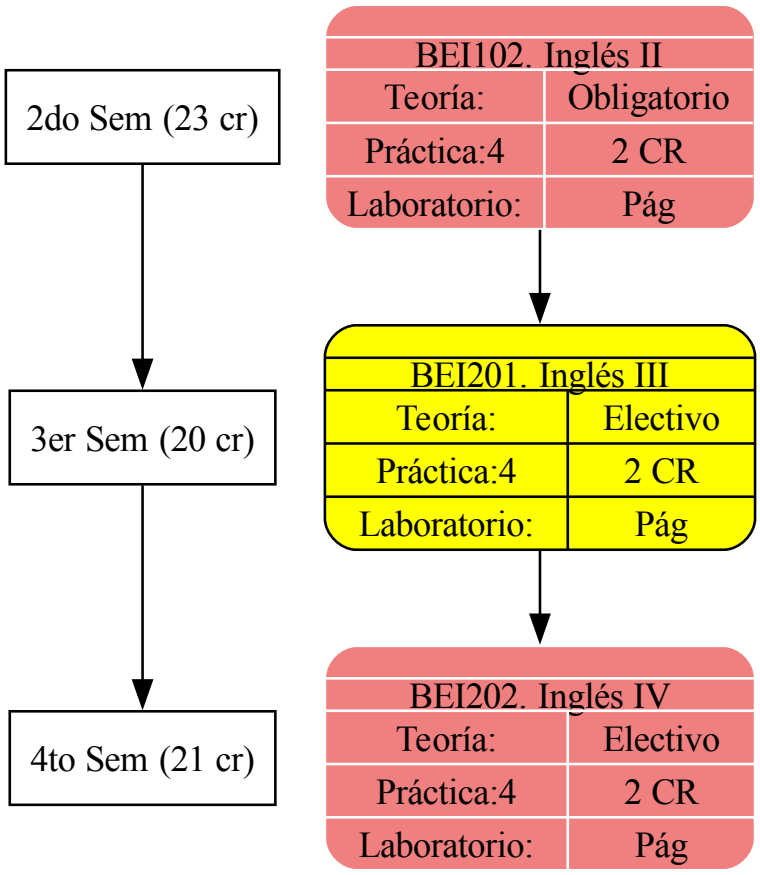

Figura 5.17: Mapa de Conexión. BEI201 Inglés III

2. Fundamentación

En este nivel, los estudiantes utilizan el inglés como medio principal para acceder y producir contenido técnico en computación, ingeniería y ciencias. Las estructuras gramaticales se aplican a la lectura de artículos científicos, la redacción de informes técnicos, la descripción de procesos de ingeniería y la participación en discusiones profesionales. El curso conecta la fluidez cotidiana en inglés con las exigencias de la comunicación técnica especializada.

3. Objetivos del curso.

- Capacitar a los estudiantes para leer, comprender y producir comunicación técnica en inglés.
- Proporcionar herramientas lingüísticas para conectar ideas en la escritura descriptiva y analítica sobre computación e ingeniería.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Familiarizarse)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Usar)

5. Contenido

5.1 Comunicación Técnica, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Repaso de Tiempos: Presente, Pasado y Futuro en documentación técnica. **II:** Oraciones interrogativas con Wh- questions complejas para el análisis de sistemas. **III:** Palabras con más de un significado en contextos técnicos (Polisemia). **IV:** Expresiones para describir actividades profesionales e intereses de investigación.

Objetivos de Aprendizaje **I:** Al finalizar la unidad, el estudiante utiliza diversos tiempos verbales para intercambiar información técnica detallada y formular preguntas precisas sobre sistemas computacionales y procesos de ingeniería.

Bibliografía: [**?** line:46 , **?** line:46 , **?** line:46]

5.2 Comparando Arquitecturas, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Uso de ”What’s it like?”: Descripción de características de sistemas e interfaces. **II:** Adjetivos comparativos y superlativos avanzados en análisis de benchmarks. **III:** Colocaciones (collocations) en computación: high-performance, low-level, fault-tolerant. **IV:** Lectura: Informes de comparación de arquitecturas (RISC vs. CISC, SQL vs. NoSQL).

Objetivos de Aprendizaje **I:** El estudiante analiza y describe diferencias entre arquitecturas, marcos de trabajo y sistemas computacionales utilizando estructuras comparativas variadas con precisión técnica.

Bibliografía: [**?** line:59 , **?** line:59 , **?** line:59]

5.3 Antes y Ahora en la Computación, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Presente Perfecto vs. Pasado Simple: Contraste entre eventos históricos y estado actual. **II:** Expresiones temporales: for, since, ever, never aplicadas a la evolución tecnológica. **III:** Adverbios de tiempo y modo en la narrativa científica. **IV:** Vocabulario: Generaciones de tecnología, sistemas heredados (legacy), plataformas actuales.

Objetivos de Aprendizaje **I:** El estudiante diferencia y aplica correctamente el presente perfecto y el pasado simple para narrar la evolución de las tecnologías computacionales y contrastar logros históricos con el estado actual del campo.

Bibliografía: [**?** line:72 , **?** line:72 , **?** line:72]

5.4 Estándares y Protocolos de Ingeniería, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Modales de obligación y permiso: Have to, Must, Can en normas profesionales. **II:** Recomendaciones técnicas: Should, Ought to — revisiones de código y diseño. **III:** Vocabulario: Estándares IEEE/ACM, convenciones de codificación, políticas de seguridad. **IV:** Redacción formal: Memorandos técnicos y recomendaciones de ingeniería.

Objetivos de Aprendizaje I: El estudiante expresa normas de ingeniería, estándares y recomendaciones profesionales utilizando verbos modales apropiados en escritura técnica formal y discusión entre pares.

Bibliografía: [? line:85, ? line:85, ? line:85]

5.5 ¿Cómo se construye el Software?, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Voz Pasiva: Presente y Pasado Simple para describir procesos de ingeniería. **II:** Vocabulario: Fabricación, despliegue, integración y pruebas. **III:** Expresiones para dar opiniones técnicas y discutir trade-offs de diseño. **IV:** Redacción: Descripción de proceso para un sistema de software o hardware.

Objetivos de Aprendizaje I: El estudiante utiliza la voz pasiva para describir cómo se diseñan, construyen, prueban y despliegan los sistemas, participando en discusiones técnicas sobre procesos de ingeniería.

Bibliografía: [? line:98, ? line:98, ? line:98]

5.6 Depuración y Análisis, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Pasado Continuo vs. Pasado Simple en reportes de incidentes y fallos. **II:** Conectores de narrativa y secuencia: while, when, suddenly, as a result. **III:** Vocabulario: Bugs, excepciones, crashes, logs, análisis de causa raíz. **IV:** Redacción: Informe de incidente de un fallo de software o de red.

Objetivos de Aprendizaje I: El estudiante narra sesiones de depuración, fallos de sistemas e incidentes experimentales utilizando tiempos narrativos y conectores de secuencia técnicos con fluidez.

Bibliografía: [? line:111, ? line:111, ? line:111]

5.7 Investigación e Innovación, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Formas de Futuro: Will, Going to, Presente Continuo para proyección de investigación. **II:** Expresiones de probabilidad y certeza en contextos científicos. **III:** Vocabulario: Ciclos de innovación, propuestas de investigación, prospectiva tecnológica. **IV:** Presentaciones orales: .[E1] futuro de la computación en mi campo".

Objetivos de Aprendizaje I: El estudiante proyecta metas de investigación y discute direcciones futuras en computación e ingeniería utilizando diversas estructuras de futuro con grados apropiados de certeza.

Bibliografía: [? line:124, ? line:124, ? line:124]

5.8 Inglés Técnico III, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Terminología de Programación y Desarrollo Web: funciones, APIs, frameworks. **II:** Descripciones de Interfaz de Usuario (UI/UX) y conceptos de usabilidad. **III:** Vocabulario básico de gestión de proyectos: sprints, milestones, deliverables. **IV:** Redacción: Especificación técnica simple o documento README.

Objetivos de Aprendizaje I: Al finalizar el curso, el estudiante utiliza vocabulario especializado de desarrollo de software para describir componentes lógicos y funcionales, y produce documentación técnica básica en inglés.

Bibliografía: [**?** line:137 , **?** line:137 , **?** line:137]

6. Bibliografía

line:142 line:142 line:142

Cuarto Semestre

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS210 Algoritmos y Estructuras de Datos (Obligatorio)

1. Información general

- **Semestre**: 4to Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS113 Programación Orientada a Objetos II (3er Sem-Pág 306)

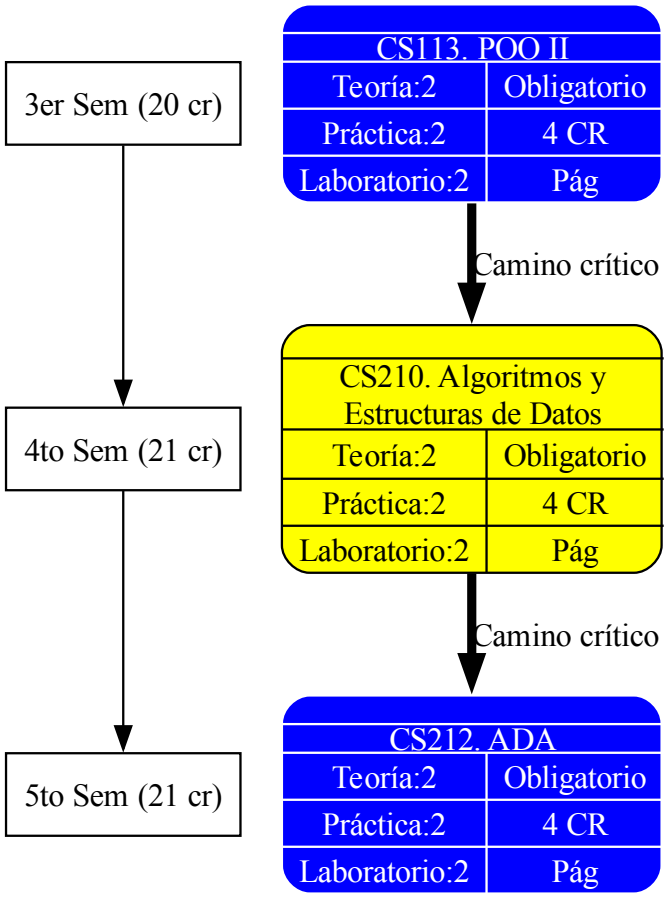

Figura 5.18: Mapa de Conexión. CS210 Algoritmos y Estructuras de Datos

2. Fundamentación

Este curso proporciona un estudio exhaustivo de las estructuras de datos y algoritmos fundamentales con énfasis en su implementación utilizando características avanzadas de C++. Basándose en conocimientos previos de C++, los estudiantes implementarán estructuras de datos genéricas usando plantillas y rasgos, asegurando flexibilidad de tipo y rendimiento. El curso cubre estructuras lineales

(vectores, varias listas enlazadas), estructuras de árbol (binario, AVL, B-trees), montículos, tablas hash, tries y estructuras de datos concurrentes. Se pone especial atención en técnicas modernas de C++ incluyendo plantillas variádicas para operaciones genéricas y programación concurrente para estructuras de datos seguras en hilos.

3. Objetivos del curso.

- Implementar estructuras de datos genéricas usando plantillas y rasgos.
- Dominar estructuras de árbol avanzadas y árboles balanceados.
- Comprender e implementar estructuras de datos concurrentes.
- Aplicar plantillas variádicas para operaciones genéricas en contenedores.
- Analizar la complejidad algorítmica de todas las estructuras implementadas.
- Diseñar estructuras de datos eficientes para restricciones de problemas específicas.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

5. Contenido

5.1 Fundamentos avanzados para estructuras de datos, 8 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: **I:** Revisión de metaprogramación con plantillas y rasgos avanzados. **II:** Rasgos de tipos y SFINAE para compilación condicional. **III:** Plantillas variádicas y paquetes de parámetros. **IV:** Reenvío perfecto y referencias universales. **V:** Expresiones lambda con capturas genéricas. **VI:** Diseño basado en políticas y CRTP (Patrón de Plantilla Curiosamente Recursivo). **VII:** Restricciones basadas en conceptos (conceptos C++20 si son aplicables).

Objetivos de Aprendizaje **I:** Diseñar rasgos de tipos para personalizar el comportamiento de estructuras de datos [Usar]. **II:** Implementar plantillas variádicas para operaciones genéricas de contenedores [Usar]. **III:** Aplicar diseño basado en políticas para crear estructuras de datos configurables [Evaluar]. **IV:** Usar reenvío perfecto para implementar constructores eficientes [Usar].

Bibliografía: [Stroustrup, 2013, Vandevoorde et al., 2017]

5.2 Estructuras de Datos Fundamentales, 8 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: **I:** Tipo de Dato Abstracto (ADT) y operaciones sobre un ADT: *a*) Operaciones de diccionario (insertar, eliminar, encontrar) **II:** Arreglos: *a*) Numéricos vs no numéricos, cadenas de caracteres *b*) Unidimensionales (vector) vs multidimensionales (matriz) **III:** Registros/Estructuras/Tuplas y Objetos **IV:** Listas enlazadas (por razones históricas): *a*) Simples vs Dobles y Lineales vs Circulares **V:** Pilas **VI:** Colas y deques: *a*) Cola de prioridad basada en montículo **VII:** Tablas/mapas hash: *a*) Resolución de colisiones y complejidad (por ejemplo, sondeo, encadenamiento, rehash) **VIII:** árboles: *a*) Binarios, n-arios y árboles de búsqueda *b*) Balanceados (por ejemplo, AVL, Rojo-Negro, Montículo) **IX:** Conjuntos

Objetivos de Aprendizaje **I:** Para cada ADT/Estructura de Datos en esta unidad: *a*) Explicar su definición, propiedades, representación(es) y operaciones de ADT asociadas. *b*) Explicar paso a paso cómo las operaciones de ADT asociadas con la estructura de datos la transforman. [Usar] **II:** Para cada algoritmo en esta unidad explicar paso a paso cómo opera el algoritmo [Usar] **III:** Implementar vector con política de asignación personalizable usando plantillas [Usar]. **IV:** Diseñar nodos de lista enlazada genéricos con gestión de memoria basada en rasgos [Usar].

V: Comparar complejidad tiempo/espacio de diferentes estructuras lineales [Evaluar]. **VI:** Aplicar plantillas variádicas para inicializar contenedores con múltiples elementos [Usar].

Bibliografía: [Cormen et al., 2009, Stroustrup, 2013]

5.3 Estructuras de Datos Fundamentales, 8 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: árboles: *a*) Binarios, n-arios y árboles de búsqueda *b*) Balanceados (por ejemplo, AVL, Rojo-Negro, Montículo) **II:** Implementación de plantilla de árbol binario con soporte de iteradores. **III:** Árbol de búsqueda binaria con operadores de comparación basados en rasgos. **IV:** Algoritmos de recorrido de árbol (iterativo y recursivo). **V:** Árboles de expresión y sus aplicaciones. **VI:** Estructuras de nodo basadas en plantilla con carga útil configurable.

Objetivos de Aprendizaje I: Dados los requisitos para un problema, desarrollar múltiples soluciones usando varias estructuras de datos y algoritmos. Posteriormente, evaluar la idoneidad, fortalezas y debilidades seleccionando un enfoque que satisfaga mejor los requisitos [Evaluar] **II:** Explicar factores más allá de la eficiencia computacional que influyen en la elección de algoritmos, como el tiempo de programación, la mantenibilidad y el uso de patrones específicos de la aplicación en los datos de entrada [Familiarizarse] **III:** Implementar árbol binario genérico con orden de recorrido configurable [Usar]. **IV:** Diseñar evaluador de árbol de expresión usando patrón visitante [Usar]. **V:** Analizar complejidades de operaciones de árbol para diferentes estrategias de balanceo [Evaluar].

Bibliografía: [Cormen et al., 2009, Knuth, 1997]

5.4 Balanced and Multi-way Trees, 8 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Implementación de árbol AVL con rasgo de balanceo basado en plantillas. **II:** Propiedades y operaciones de árbol rojo-negro. **III:** Implementación de plantilla de B-Tree para almacenamiento basado en disco. **IV:** Estructuras de árbol 2-3 y árbol 2-3-4. **V:** Implementación de trie digital (árbol de prefijos). **VI:** Árboles de sufijos y sus aplicaciones en procesamiento de cadenas. **VII:** Estrategias de división/fusión de nodos para B-trees.

Objetivos de Aprendizaje I: Implementar árboles auto-balanceados usando diseño basado en políticas [Usar]. **II:** Diseñar B-tree para aplicaciones de índice de base de datos [Usar]. **III:** Evaluar compensaciones entre diferentes estructuras de árbol balanceado [Evaluar]. **IV:** Implementar trie para coincidencia eficiente de prefijos de cadenas [Usar].

Bibliografía: [Cormen et al., 2009, Knuth, 1997, Sedgewick and Wayne, 2011]

5.5 Estructuras de Datos Fundamentales, 8 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: árboles: *a*) Binarios, n-arios y árboles de búsqueda *b*) Balanceados (por ejemplo, AVL, Rojo-Negro, Montículo) **II:** Implementación de montículo binario usando representación de arreglo. **III:** Variantes de montículo mínimo y montículo máximo. **IV:** Algoritmos heapify y análisis de complejidad. **V:** Implementación de ADT de cola de prioridad. **VI:** Visión general de montículo binomial y montículo de Fibonacci. **VII:** Aplicaciones de montículos: algoritmo de Dijkstra, ordenamiento por montículo.

Objetivos de Aprendizaje I: Explicar la propiedad de montículo y el uso de montículos como una implementación de una cola de prioridad [Familiarizarse] **II:** Implementar montículo genérico con rasgo de comparación configurable [Usar]. **III:** Diseñar cola de prioridad que soporte múltiples estrategias de actualización de prioridad [Usar]. **IV:** Analizar complejidad de operaciones de montículo para diferentes tipos de montículo [Evaluar]. **V:** Aplicar montículo en implementaciones de algoritmos de grafos [Usar].

Bibliografía: [Cormen et al., 2009, Sedgewick and Wayne, 2011]

5.6 Estructuras de Datos Fundamentales, 8 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Tablas/mapas hash: *a*) Resolución de colisiones y complejidad (por ejemplo, sondeo, encadenamiento, rehash) **II:** Diseño de función hash y propiedades. **III:** Implementación de encadenamiento separado con listas enlazadas. **IV:** Direccionamiento abierto: sondeo lineal, sondeo cuadrático, doble hashing. **V:** Hashing perfecto y hashing perfecto mínimo. **VI:** Hashing cuco y hashing de rayuela. **VII:** Estrategias de redimensionamiento de tabla hash y gestión de factor de carga.

Objetivos de Aprendizaje I: Explicar cómo se maneja la evitación de colisiones y la resolución de colisiones en tablas hash [Familiarizarse] **II:** Diseñar tabla hash genérica con resolución de colisiones configurable [Usar]. **III:** Implementar funciones hash personalizadas para tipos definidos por el usuario [Usar]. **IV:** Evaluar rendimiento de diferentes esquemas de hashing [Evaluar]. **V:** Aplicar tablas hash en la implementación de tabla de símbolos del compilador [Usar].

Bibliografía: [Cormen et al., 2009, Knuth, 1997]

5.7 Concurrent Data Structures, 8 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Principios de diseño de estructuras de datos seguras para hilos. **II:** Listas enlazadas concurrentes basadas en bloqueos. **III:** Algoritmos sin bloqueo y libres de espera. **IV:** Tablas hash concurrentes con bloqueo de grano fino. **V:** Operaciones atómicas y ordenamiento de memoria. **VI:** Colas concurrentes (limitadas e ilimitadas). **VII:** Pilas concurrentes y técnicas de eliminación.

Objetivos de Aprendizaje I: Diseñar estructuras de datos seguras para hilos usando mutex y guardas de bloqueo [Usar]. **II:** Implementar lista enlazada sin bloqueo usando operaciones atómicas [Usar]. **III:** Analizar rendimiento de estructuras de datos concurrentes vs secuenciales [Evaluar]. **IV:** Aplicar colas concurrentes en patrones productor-consumidor [Usar].

Bibliografía: [Herlihy and Shavit, 2012, Williams, 2019]

5.8 Specialized Structures and Applications, 8 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Estructuras de datos persistentes. **II:** Unión-búsqueda de conjuntos disjuntos (union-find) con compresión de trayectoria. **III:** Filtros de Bloom y sus propiedades probabilísticas. **IV:** Listas por niveles para búsqueda balanceada probabilística. **V:** Árboles de segmentos y árboles de Fenwick para consultas de rango. **VI:** Estructuras de datos espaciales: árboles k-d, quadtrees. **VII:** Estructuras de datos inconscientes de la caché.

Objetivos de Aprendizaje I: Implementar estructuras de datos persistentes con compartición estructural [Usar]. **II:** Diseñar union-find con unión por rango y compresión de trayectoria [Usar]. **III:** Aplicar estructuras especializadas para resolver problemas específicos de dominio [Evaluar]. **IV:** Evaluar compensaciones espacio-tiempo en estructuras de datos probabilísticas [Evaluar].

Bibliografía: [Okasaki, 1999, Sedgewick and Wayne, 2011]

6. Bibliografía

[Cormen et al., 2009] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, 3rd edition.

[Herlihy and Shavit, 2012] Herlihy, M. and Shavit, N. (2012). *The Art of Multiprocessor Programming*. Morgan Kaufmann, revised 1st edition.

[Knuth, 1997] Knuth, D. E. (1997). *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley, 3rd edition.

[Okasaki, 1999] Okasaki, C. (1999). *Purely Functional Data Structures*. Cambridge University Press.

[Sedgewick and Wayne, 2011] Sedgewick, R. and Wayne, K. (2011). *Algorithms*. Addison-Wesley, 4th edition.

[Stroustrup, 2013] Stroustrup, B. (2013). *The C++ Programming Language*. Addison-Wesley, 4th edition.

[Vandevoorde et al., 2017] Vandevoorde, D., Josuttis, N. M., and Gregor, D. (2017). *C++ Templates: The Complete Guide*. Addison-Wesley, 2nd edition.

[Williams, 2019] Williams, A. (2019). *C++ Concurrency in Action*. Manning Publications, 2nd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS211 Teoría de la Computación (Obligatorio)

1. Información general

- **Semestre**: 4to Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS1D1 Estructuras Discretas (2do Sem-Pág 292)

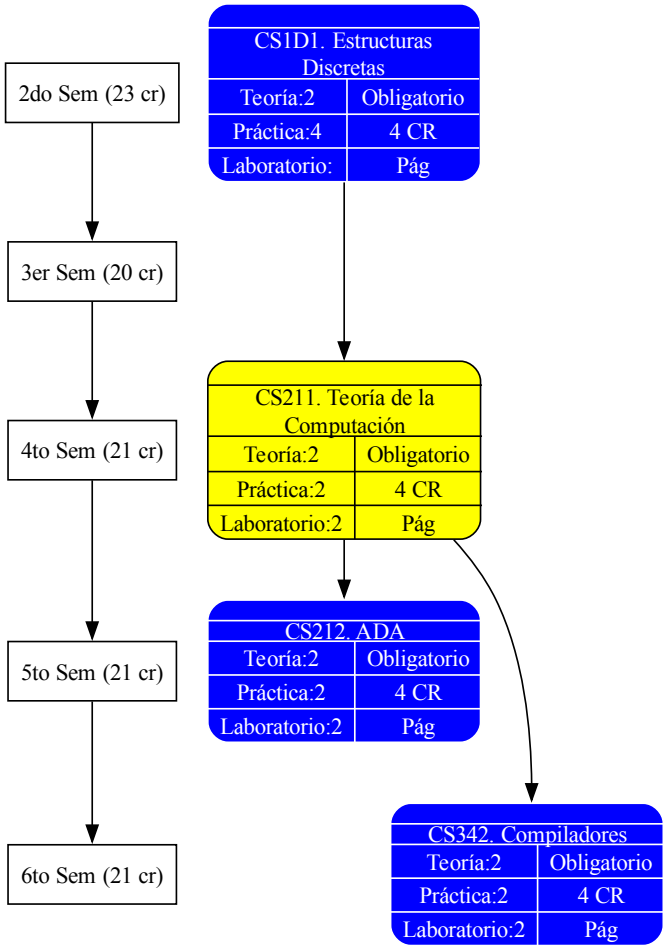

Figura 5.19: Mapa de Conexión. CS211 Teoría de la Computación

2. Fundamentación

La Teoría de la Computación proporciona los fundamentos matemáticos para entender qué puede ser computado y con qué eficiencia. Introduce lenguajes formales, autómatas y los límites de la resolubilidad algorítmica. Este conocimiento es fundamental para comprender la construcción de compiladores, la verificación formal y la complejidad inherente de los problemas computacionales.

3. Objetivos del curso.

- Dominar los conceptos de autómatas finitos y expresiones regulares.

- Analizar gramáticas libres de contexto y autómatas de pila.
- Comprender la Máquina de Turing Universal como modelo de computación.
- Comprender los límites de la computabilidad y el Problema de la Parada.
- Diferenciar entre clases de complejidad como P y NP.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Autómatas Finitos y Lenguajes Regulares, 18 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Autómatas formales: *a*) Estado Finito *b*) Pila *c*) Linealmente Acotado *d*) Máquina de Turing **II:** Lenguajes formales, gramáticas y Jerarquía de Chomsky: *a*) Regulares (Tipo-3): 1) Expresiones Regulares *b*) Libres de Contexto (Tipo-2) *c*) Sensibles al Contexto (Tipo-1) *d*) Recursivamente Enumerables (Tipo-0) **III:** Autómatas deterministas y no deterministas **IV:** Demostraciones del Lema de Bombeo: *a*) Demostración de la limitación de Estado Finito/Lenguaje Regular *b*) Limitación de Autómatas de Pila/Lenguaje Libre de Contexto

Objetivos de Aprendizaje I: Para cada autómata formal en esta unidad: *a*) Explicar su definición comparando sus características con los otros autómatas de esta unidad *b*) Usando un ejemplo, explicar paso a paso cómo el autómata opera sobre la entrada incluyendo si acepta la entrada asociada *c*) Explicar un ejemplo de entradas que pueden y no pueden ser aceptadas por el autómata. [Explicar] **II:** Dado un problema, desarrollar un autómata apropiado que aborde el problema [Crear] **III:** Desarrollar una expresión regular para un lenguaje regular dado expresado en lenguaje natural [Crear] **IV:** Convertir entre notaciones equivalentemente poderosas para un lenguaje, incluyendo entre DFAs, NFAs y expresiones regulares, y entre PDAs y CFGs [Aplicar] **V:** Aplicar lemas de bombeo, o medios alternativos, para demostrar las limitaciones de los autómatas de Estado Finito y Pila [Aplicar]

Bibliografía: [Sipser, 2012, Hopcroft et al., 2013]

5.2 Lenguajes Libres de Contexto y PDA, 16 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Autómatas formales: *a*) Estado Finito *b*) Pila *c*) Linealmente Acotado *d*) Máquina de Turing **II:** Lenguajes formales, gramáticas y Jerarquía de Chomsky: *a*) Regulares (Tipo-3): 1) Expresiones Regulares *b*) Libres de Contexto (Tipo-2) *c*) Sensibles al Contexto (Tipo-1) *d*) Recursivamente Enumerables (Tipo-0) **III:** Formas Normales: Chomsky y Greibach **IV:** Demostraciones del Lema de Bombeo: *a*) Demostración de la limitación de Estado Finito/Lenguaje Regular *b*) Limitación de Autómatas de Pila/Lenguaje Libre de Contexto **V:** Relaciones entre autómatas formales, lenguajes y gramáticas

Objetivos de Aprendizaje I: Para cada autómata formal en esta unidad: *a*) Explicar su definición comparando sus características con los otros autómatas de esta unidad *b*) Usando un ejemplo, explicar paso a paso cómo el autómata opera sobre la entrada incluyendo si acepta la entrada asociada *c*) Explicar un ejemplo de entradas que pueden y no pueden ser aceptadas por el autómata. [Explicar] **II:** Dado un problema, desarrollar un autómata apropiado que aborde el problema [Crear] **III:** Convertir entre notaciones equivalentemente poderosas para un lenguaje, incluyendo entre DFAs, NFAs y expresiones regulares, y entre PDAs y CFGs [Aplicar] **IV:** Aplicar lemas de bombeo, o medios alternativos, para demostrar las limitaciones de los autómatas

de Estado Finito y Pila [Aplicar] **V:** Construir GLCs para descripciones de lenguajes formales [Evaluar]

Bibliografía: [Sipser, 2012, Hopcroft et al., 2013]

5.3 Máquinas de Turing y Computabilidad, 16 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Autómatas formales: *a*) Estado Finito *b*) Pila *c*) Linealmente Acotado *d*) Máquina de Turing **II:** Lenguajes formales, gramáticas y Jerarquía de Chomsky: *a*) Regulares (Tipo-3): 1) Expresiones Regulares *b*) Libres de Contexto (Tipo-2) *c*) Sensibles al Contexto (Tipo-1) *d*) Recursivamente Enumerables (Tipo-0) **III:** Decidabilidad, (in)computabilidad y parada **IV:** La tesis de Church-Turing **V:** Reducibilidad y reducciones **VI:** Modelos equivalentes de computación algorítmica: *a*) Máquinas de Turing y Variaciones (por ejemplo, multicinta, no deterministas) *b*) Cálculo Lambda *c*) Funciones Mu-Recursivas

Objetivos de Aprendizaje I: Para cada autómata formal en esta unidad: *a*) Explicar su definición comparando sus características con los otros autómatas de esta unidad *b*) Usando un ejemplo, explicar paso a paso cómo el autómata opera sobre la entrada incluyendo si acepta la entrada asociada *c*) Explicar un ejemplo de entradas que pueden y no pueden ser aceptadas por el autómata. [Explicar] **II:** Explicar una Máquina de Turing universal y su operación [Explicar] **III:** Presentar a una audiencia de compañeros de trabajo y gerentes la imposibilidad de proporcionarles un programa que verifique todos los otros programas, incluyendo algunos aparentemente simples, en busca de bucles infinitos incluyendo una explicación del problema de la parada, por qué no tiene solución algorítmica y su importancia para la computación algorítmica del mundo real [Explicar] **IV:** Explicar la Tesis de Church-Turing y su importancia para la computación algorítmica [Explicar] **V:** Aplicar aritmetización y diagonalización para demostrar que el Problema de la Parada para Máquinas de Turing es Indecidible [Aplicar] **VI:** Dado un lenguaje indecidible conocido, aplicar una reducción por mapeo o historia computacional para demostrar que otro lenguaje es indecidible [Aplicar]

Bibliografía: [Sipser, 2012, Kozen, 2006]

5.4 Jerarquía de Chomsky y Modelos Equivalentes, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Lenguajes formales, gramáticas y Jerarquía de Chomsky: *a*) Regulares (Tipo-3): 1) Expresiones Regulares *b*) Libres de Contexto (Tipo-2) *c*) Sensibles al Contexto (Tipo-1) *d*) Recursivamente Enumerables (Tipo-0) **II:** Relaciones entre autómatas formales, lenguajes y gramáticas **III:** Modelos equivalentes de computación algorítmica: *a*) Máquinas de Turing y Variaciones (por ejemplo, multicinta, no deterministas) *b*) Cálculo Lambda *c*) Funciones Mu-Recursivas **IV:** Lenguajes Sensibles al Contexto (Tipo-1) y aplicaciones

Objetivos de Aprendizaje I: Para cada modelo formal en esta unidad: *a*) Explicar su definición comparando sus características con las otras en esta unidad *b*) Explicar ejemplos de entradas que son y no pueden ser aceptadas por el lenguaje/gramática. [Explicar] **II:** Explicar las funciones Recursivas Primitivas y Generales (cero, sucesor, selección, recursión primitiva, composición y Mu), su importancia e implementaciones en Máquina de Turing [Explicar] **III:** Explicar cómo se realiza la computación en el Cálculo Lambda (por ejemplo, conversión alfa y reducción beta) [Explicar] **IV:** Explicar el teorema de Rice y su importancia [Explicar]

Bibliografía: [Sipser, 2012, Kozen, 2006]

5.5 Marco de Análisis de Complejidad, 3 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Marco de Análisis de Complejidad: *a*) Rendimiento de un algoritmo en el mejor caso, caso promedio y peor caso *b*) Mediciones empíricas y relativas (Orden de Crecimiento) *c*) Tamaño de entrada y operaciones primitivas *d*) Eficiencia de tiempo y espacio

Objetivos de Aprendizaje I: Preparar una presentación que explique a estudiantes de primer año los conceptos básicos de complejidad algorítmica incluyendo comportamiento de algoritmo en mejor caso, caso promedio y peor caso, notaciones Big-O, Omega y Theta, clases de complejidad, compromisos tiempo-espacio, medición empírica e impacto en problemas prácticos [Explicar] **II:** Evaluar informalmente la clase de complejidad fundamental de algoritmos simples [Evaluar]

Bibliografía: [Sipser, 2012, Cormen et al., 2022]

5.6 Notación Asintótica y Clases de Complejidad, 4 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Análisis de complejidad asintótica (cotas promedio y peor caso): *a*) Notaciones formales Big-O, Big-Omega y Big-Theta *b*) Clases de Complejidad Fundamentales y Ejemplos/Problemas Representativos: 1) $O(1)$ Constante (por ejemplo, acceso a arreglo) 2) $O(\log_2 n)$ Logarítmica (por ejemplo, búsqueda binaria) 3) $O(n)$ Lineal (por ejemplo, búsqueda lineal) 4) $O(n \log_2 n)$ Log Lineal (por ejemplo, mergesort) 5) $O(n^2)$ Cuadrática (por ejemplo, ordenamiento por selección) 6) $O(n^c)$ Polinomial (por ejemplo, $O(n^3)$ eliminación gaussiana) 7) $O(2^n)$ Exponencial (por ejemplo, Mochila, Satisfactibilidad (SAT), Viajante de Comercio (TSP), todos los subconjuntos) 8) $O(n!)$ Factorial (por ejemplo, circuito hamiltoniano, todas las permutaciones)

Objetivos de Aprendizaje I: Usando ejemplos, explicar cada una de las clases de complejidad fundamentales en esta unidad [Explicar] **II:** Explicar a una audiencia no técnica la importancia de los algoritmos tratables versus intratables usando una explicación intuitiva de la complejidad Big-O [Explicar]

Bibliografía: [Sipser, 2012, Cormen et al., 2022]

5.7 Análisis de Complejidad: Recursión, Amortización y Cotas Ajustadas, 7 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Notaciones Little-o, Little-Omega y Little Theta **II:** Análisis recursivo formal

Objetivos de Aprendizaje I: Usar relaciones de recurrencia para evaluar la complejidad de tiempo de algoritmos definidos recursivamente [Aplicar] **II:** Aplicar relaciones de recurrencia elementales usando una forma del Teorema Maestro [Aplicar]

Bibliografía: [Sipser, 2012, Cormen et al., 2022]

5.8 Teoría de la Complejidad Computacional, 6 hr(s)

Competencias:

Tópicos: I: Tractabilidad e intractabilidad: *a*) Clases de Complejidad P, NP y NP-Completo *b*) Problemas NP-Completos (por ejemplo, SAT, Mochila, TSP) *c*) Reducciones **II:** Modelos de complejidad basados en Máquinas de Turing: *a*) Complejidad de tiempo: 1) Clases P, NP, NP-C y EXP 2) Teorema de Cook-Levin *b*) Complejidad de espacio: 1) NSpace y PSpace 2) Teorema de Savitch

Objetivos de Aprendizaje I: Explicar la importancia de la NP-Completitud [Explicar] **II:** Explicar cómo NP-Duro es una cota inferior y NP es una cota superior para la NP-Completitud [Explicar] **III:** Explicar ejemplos de problemas NP-completos [Explicar] **IV:** Explicar el Teorema de Cook-Levin y la NP-Completitud de SAT [Explicar] **V:** Explicar las clases P y NP [Explicar] **VI:** Demostrar que un problema es NP-Completo reduciendo un problema NP-C clásico conocido a él (por ejemplo, 3SAT y Clique) [Crear] **VII:** Explicar la clase P-espacio y su relación con la clase EXP [Explicar]

Bibliografía: [Sipser, 2012, Cormen et al., 2022]

6. Bibliografía

[Cormen et al., 2022] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2022). *Introduction to Algorithms*. MIT Press, 4th edition.

[Hopcroft et al., 2013] Hopcroft, J. E., Motwani, R., and Ullman, J. D. (2013). *Introduction to Automata Theory, Languages, and Computation*. Pearson, 3rd edition.

[Kozen, 2006] Kozen, D. C. (2006). *Theory of Computation*. Springer.

[Sipser, 2012] Sipser, M. (2012). *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS221 Arquitectura de Computadores (Obligatorio)

1. Información general

- **Semestre**: 4to Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - CS1D1 Estructuras Discretas (2do Sem-Pág 292)

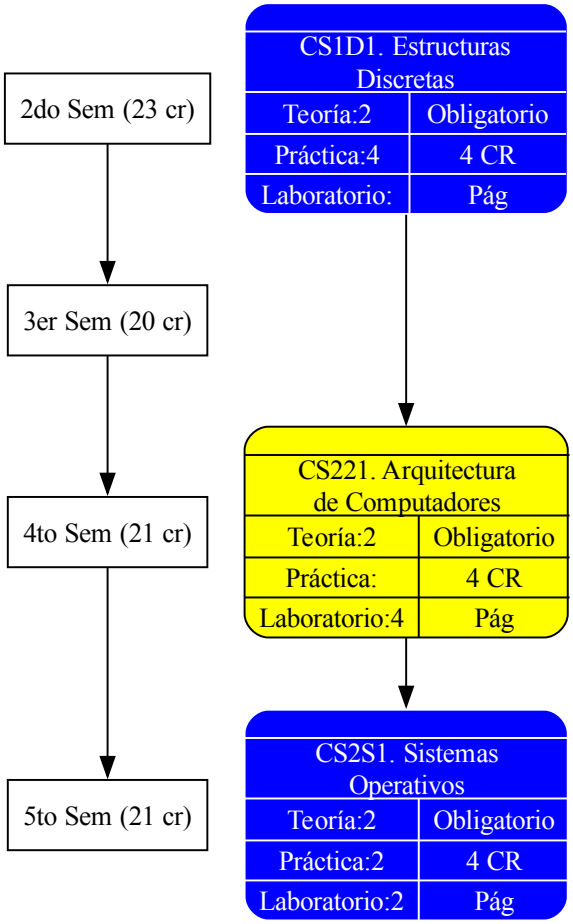

Figura 5.20: Mapa de Conexión. CS221 Arquitectura de Computadores

2. Fundamentación

Una comprensión profunda de la arquitectura de computadoras es esencial para desarrollar software de alto rendimiento y entender la interfaz hardware-software. Este curso explora cómo las computadoras ejecutan instrucciones, gestionan memoria y se comunican con dispositivos externos. Al estudiar la organización interna de los procesadores y la jerarquía de los sistemas de memoria, los estudiantes adquieren la capacidad de optimizar código para arquitecturas de hardware específicas y comprender las compensaciones en el diseño de sistemas modernos.

3. Objetivos del curso.

- Comprender los principios de la lógica digital y la representación de datos.
- Comprender la organización de la Unidad Central de Procesamiento (CPU).
- Dominar los conceptos básicos del lenguaje ensamblador y la arquitectura del conjunto de instrucciones.
- Analizar la jerarquía de memoria, incluyendo memoria caché y memoria virtual.
- Evaluar mejoras de rendimiento mediante segmentación y paralelismo.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

5. Contenido

5.1 Organización de Máquina a Nivel de Ensamblador, 20 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Arquitectura de máquina von Neumann **II:** Unidad de control: captación, decodificación y ejecución de instrucciones **III:** Introducción a SIMD vs MIMD y la taxonomía de Flynn **IV:** Organización de multiprocesadores/multinúcleo de memoria compartida **V:** Arquitectura del conjunto de instrucciones (ISA) (por ejemplo, x86, ARM y RISC-V): *a*) Conjuntos de instrucciones de ancho fijo vs variable *b*) Formatos de instrucción *c*) Manipulación de datos, control, E/S *d*) Modos de direccionamiento *e*) Programación en lenguaje de máquina *f*) Programación en lenguaje ensamblador **VI:** Mecanismos de llamada y retorno de subrutinas **VII:** E/S e interrupciones **VIII:** Segmentos de montículo, estáticos, pila y código

Objetivos de Aprendizaje **I:** Discutir cómo las unidades funcionales clásicas de von Neumann se implementan en sistemas embebidos, particularmente en memoria dentro y fuera del chip [Debatir] **II:** Describir cómo se ejecutan las instrucciones en una máquina von Neumann clásica, con extensiones para hilos, sincronización de multiprocesadores y ejecución SIMD [Describir] **III:** Evaluar un diagrama de ejemplo con paralelismo a nivel de instrucción y riesgos para describir cómo se manejan en las canalizaciones de procesadores típicos [Evaluar (valorar)] **IV:** Discutir cómo se representan las instrucciones a nivel de máquina y en el contexto de un ensamblador simbólico [Debatir] **V:** Mapear un ejemplo de patrones de lenguaje de alto nivel a notaciones de lenguaje ensamblador/máquina [Mapear/Mapa conceptual] **VI:** Contrastar diferentes formatos de instrucción considerando aspectos como direcciones por instrucción y formatos de longitud variable vs longitud fija [Contrastar] **VII:** Analizar un diagrama de subrutina para comentar cómo se manejan las llamadas de subrutina a nivel de ensamblador [Analizar] **VIII:** Describir conceptos básicos de interrupciones y operaciones de E/S [Describir] **IX:** Escribir un programa simple en lenguaje ensamblador para procesamiento y manipulación de cadenas/arreglos [Escribir]

Bibliografía: [Patterson and Hennessy, 2020, Harris and Harris, 2012]

5.2 Jerarquía de Memoria, 18 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Jerarquía de memoria: la importancia de la localidad temporal y espacial **II:** Organización y operaciones de memoria principal **III:** Memoria persistente (por ejemplo, SSD, discos estándar) **IV:** Latencia, tiempo de ciclo, ancho de banda e intercalado **V:** Memorias caché: *a*) Mapeo de direcciones *b*) Tamaño de bloque *c*) Política de reemplazo y almacenamiento *d*) Prebúsqueda **VI:** Coherencia de caché multiprocesador **VII:** Memoria virtual (soporte de hardware) **VIII:** Manejo de fallas y confiabilidad **IX:** Confiabilidad: *a*) Codificación de error *b*) Compresión de datos *c*) Integridad de datos **X:** Procesamiento en Memoria (PIM)

Objetivos de Aprendizaje I: Usando un diagrama del sistema de memoria, identificar los principales tipos de tecnología de memoria (por ejemplo, SRAM, DRAM) y su costo y rendimiento relativos [Analizar] **II:** Medir el efecto de la latencia de memoria en el tiempo de ejecución [Evaluar] **III:** Enumerar las funciones de un sistema con gestión de memoria virtual [Enumerar] **IV:** Calcular el tiempo promedio de acceso a memoria bajo varias configuraciones de caché y memoria y mezclas de referencias de instrucciones y datos [Computar/Calcular]

Bibliografía: [Patterson and Hennessy, 2020, Stallings, 2015]

5.3 Interconexión y Comunicación, 8 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Fundamentos de E/S: *a*) Protocolo de enlace y almacenamiento en búfer *b*) E/S programada *c*) E/S basada en interrupciones **II:** Estructuras de interrupción: vectorizadas y priorizadas, acuse de recibo de interrupción **III:** Dispositivos de E/S (por ejemplo, ratón, teclado, pantalla, cámara, sensores, actuadores) **IV:** Almacenamiento externo, organización física y unidades **V:** Fundamentos de buses: *a*) Protocolos de bus *b*) Arbitraje *c*) Acceso directo a memoria (DMA)

Objetivos de Aprendizaje I: Analizar un diagrama de control de interrupciones para comentar cómo se usan las interrupciones para implementar control de E/S y transferencias de datos [Analizar] **II:** Enumerar varios tipos de buses en un sistema informático [Enumerar] **III:** Enumerar las ventajas de los discos magnéticos y contrastarlas con las de los discos de estado sólido [Listar/Enumerar]

Bibliografía: [Patterson and Hennessy, 2020, Tanenbaum and Austin, 2012]

5.4 Organización Funcional, 8 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Implementación de rutas de datos simples, incluyendo segmentación de instrucciones, detección de riesgos y resolución (por ejemplo, paradas, reenvío) **II:** Unidad de control: *a*) Implementación cableada *b*) Realización microprogramada **III:** Segmentación de instrucciones **IV:** Introducción al paralelismo a nivel de instrucción (ILP)

Objetivos de Aprendizaje I: Comparar implementaciones alternativas de rutas de datos en arquitecturas de computadoras modernas [Comparar] **II:** Producir un conjunto de señales de control para sumar dos enteros usando implementaciones cableadas y microprogramadas [Producir] **III:** Discutir el paralelismo a nivel de instrucción usando segmentación y los riesgos significativos que pueden ocurrir [Debatir] **IV:** Diseñar un procesador completo, incluyendo ruta de datos y control [Diseñar] **V:** Calcular los ciclos promedio por instrucción para una implementación dada de procesador y sistema de memoria [Computar/Calcular]

Bibliografía: [Harris and Harris, 2012, Patterson and Hennessy, 2020]

5.5 Arquitecturas Heterogéneas, 10 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Arquitecturas SIMD y MIMD (por ejemplo, GPUs de Propósito General, TPUs y NPUs) **II:** Sistemas de memoria heterogéneos: *a*) Memoria compartida versus memoria distribuida *b*) Memoria volátil vs no volátil *c*) Protocolos de coherencia **III:** Arquitecturas Específicas de Dominio (DSAs): *a*) Acelerador de Aprendizaje Automático *b*) Computación en red *c*) Sistemas embebidos para aplicaciones emergentes *d*) Computación neuromórfica *e*) Dispositivos de computación perimetral **IV:** Soluciones de empaquetado e integración como 3DIC y chiplets **V:** Aprendizaje automático en diseño de arquitectura: *a*) Algoritmos de IA para análisis de carga de trabajo *b*) Optimización de configuraciones de arquitectura para rendimiento y eficiencia

energética **VI:** Diseño Orientado a Datos (DOD): organización de datos en memoria (arreglo de estructuras vs. estructura de arreglos) para maximizar localidad de caché y aprovechar vectorización SIMD; contraste y complementariedad con el paradigma orientado a objetos.

Objetivos de Aprendizaje I: Analizar un diagrama de sistema con arquitecturas paralelas alternativas, por ejemplo, SIMD y MIMD, e identificar las diferencias clave [Evaluar] **II:** Discutir qué problemas de gestión de memoria se encuentran en multiprocesadores que no están presentes en uniprocesadores y cómo estos problemas podrían resolverse [Evaluar] **III:** Indicar las diferencias entre placa base de memoria, interconexión de memoria del procesador y memoria remota a través de redes, sus implicaciones para la latencia de acceso y su impacto en el rendimiento del programa [Familiarizarse] **IV:** Discutir cómo determinarías cuándo usar un acelerador específico de dominio en lugar de una CPU de propósito general [Evaluar] **V:** Enumerar diferencias clave en los principios de diseño arquitectónico entre una unidad de procesamiento basada en vector y basada en escalar [Familiarizarse] **VI:** Enumerar las ventajas y desventajas de una arquitectura PIM [Familiarizarse] **VII:** Reestructurar una colección de objetos (AoS) a un diseño DOD (SoA) y medir el impacto en rendimiento [Usar].

Bibliografía: [Patterson and Hennessy, 2020, Stallings, 2015]

5.6 Arquitecturas de Procesador Seguras, 8 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Principios de Hardware Seguro: *a*) Análisis de Riesgos de Seguridad, Protección de Activos y Modelo de Amenazas *b*) Aceleración Criptográfica con Hardware *c*) Soporte para virtualización (por ejemplo, aislamiento de SO) **II:** Raíces de confianza en hardware, Funciones Físicamente No Clonables (PUF) **III:** Generadores de Números Aleatorios por Hardware **IV:** Extensiones de protección de memoria: *a*) Verificación de límites de puntero en tiempo de ejecución (por ejemplo, desbordamiento de búfer) *b*) Protección a nivel microarquitectónico *c*) Protección a nivel de ISA **V:** Entorno de Ejecución Confiable (TEE): *a*) Protecciones de Base de Computadora Confiable *b*) Protección de máquinas virtuales *c*) Protección de contenedores *d*) Módulos de software confiables (Enclaves) **VI:** Cifrado homomórfico para procesamiento de datos que preserva la privacidad

Objetivos de Aprendizaje I: Discutir principios de hardware seguro, explorando un marco para análisis de riesgos y protección de activos [Debatir] **II:** Resumir cómo las Funciones Físicamente No Clonables (PUF) pueden ser un identificador único de dispositivo en aplicaciones de seguridad [Resumir] **III:** Distinguir un generador de números aleatorios con soporte de hardware dedicado de generadores sin hardware dedicado a generar entropía [Distinguir] **IV:** Enumerar las ventajas y desventajas de la protección de memoria a nivel de ISA [Listar/Enumerar] **V:** Describir problemas de diseño clave de un entorno de ejecución confiable (TEE) para soportar máquinas virtuales [Describir]

Bibliografía: [Patterson and Hennessy, 2020, Anderson, 2020]

6. Bibliografía

[Anderson, 2020] Anderson, R. J. (2020). *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 3rd edition.

[Harris and Harris, 2012] Harris, D. and Harris, S. (2012). *Digital Design and Computer Architecture*. Morgan Kaufmann, 2nd edition.

[Patterson and Hennessy, 2020] Patterson, D. A. and Hennessy, J. L. (2020). *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*. Morgan Kaufmann, 2nd edition.

[Stallings, 2015] Stallings, W. (2015). *Computer Organization and Architecture: Designing for Performance*. Pearson, 10th edition.

[Tanenbaum and Austin, 2012] Tanenbaum, A. S. and Austin, T. (2012). *Structured Computer Organization*. Pearson, 6th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS271 Bases de Datos I (Obligatorio)

1. Información general

- **Semestre**: 4to Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - CS112 Programación Orientada a Objetos I (2do Sem-Pág 287)
  - CS1D1 Estructuras Discretas (2do Sem-Pág 292)

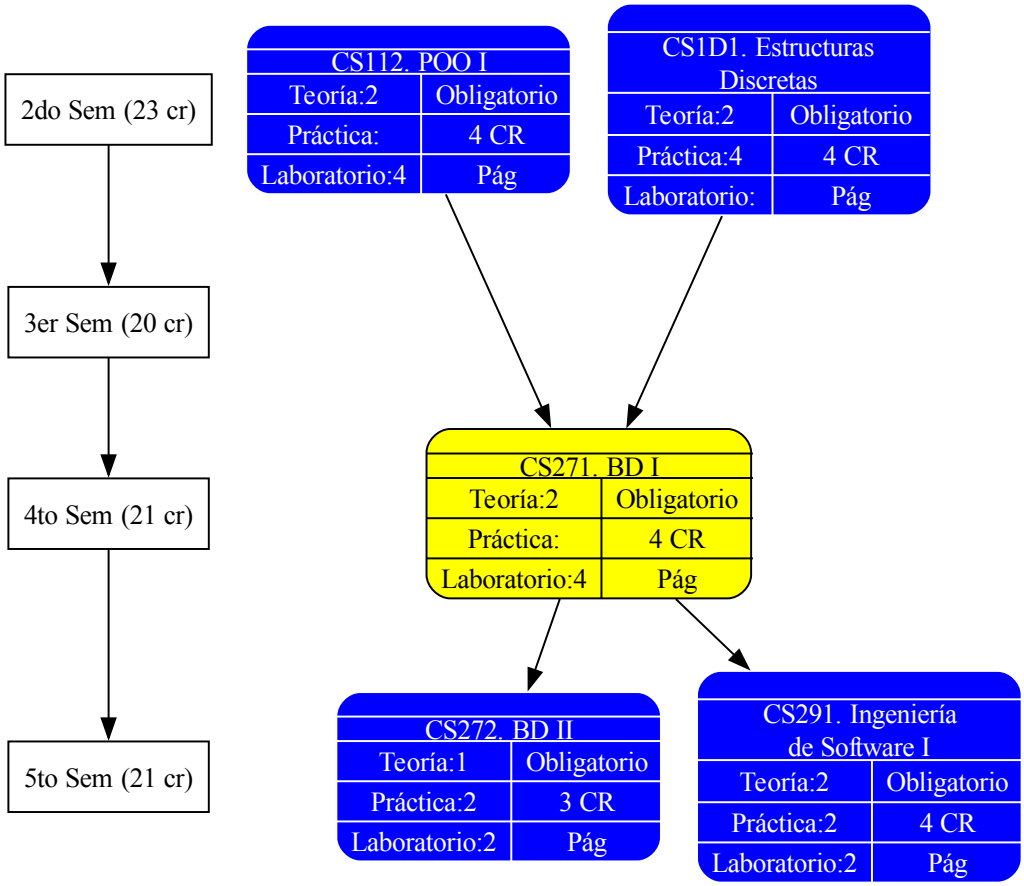

Figura 5.21: Mapa de Conexión. CS271 Bases de Datos I

2. Fundamentación

Este curso introduce conceptos y técnicas fundamentales en gestión de datos, centrándose en sistemas de bases de datos relacionales. Los estudiantes aprenderán modelado de datos usando diagramas entidad-relación, principios de diseño de bases de datos relacionales, teoría de normalización y Lenguaje de Consulta Estructurado (SQL). El curso cubre componentes centrales de sistemas de bases de datos, gestión de transacciones y procesamiento básico de consultas. Se pone énfasis en habilidades prácticas para diseñar, implementar y consultar bases de datos, así como en comprender el rol de los datos en las organizaciones y las consideraciones éticas asociadas en la gestión de datos.

3. Objetivos del curso.

- Diseñar e implementar bases de datos relacionales normalizadas.
- Formular consultas complejas usando SQL para recuperación y manipulación de datos.
- Aplicar técnicas de modelado de datos usando diagramas entidad-relación.
- Comprender la arquitectura de sistemas de bases de datos y gestión de transacciones.
- Reconocer consideraciones éticas y de seguridad en la gestión de datos.
- Utilizar herramientas y técnicas básicas de administración de bases de datos.

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Conceptos Fundamentales de Sistemas de Bases de Datos, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Propósito y ventajas de los sistemas de bases de datos **II:** Componentes de los sistemas de bases de datos **III:** Diseño de funciones fundamentales de DBMS (por ejemplo, mecanismos de consulta, gestión de transacciones, gestión de búfer, métodos de acceso) **IV:** Arquitectura de base de datos, independencia de datos y abstracción de datos **V:** Gestión de transacciones **VI:** Normalización **VII:** Enfoques para gestionar grandes volúmenes de datos (por ejemplo, sistemas de bases de datos NoSQL, uso de MapReduce) **VIII:** Cómo soportar aplicaciones solo de CRUD **IX:** Bases de datos distribuidas/sistemas basados en la nube **X:** Datos estructurados, semiestructurados y no estructurados **XI:** Uso de un lenguaje de consulta declarativo **XII:** Sistemas que soportan contenido estructurado y/o de flujo

Objetivos de Aprendizaje **I:** Identificar al menos cuatro ventajas que proporciona el uso de un sistema de base de datos [Familiarizarse] **II:** Enumerar los componentes de un sistema de base de datos (relacional) [Familiarizarse] **III:** Explicar las ventajas de los sistemas de bases de datos sobre los sistemas de archivos [Usar]. **IV:** Describir la arquitectura de tres esquemas y la independencia de datos [Familiarizarse]. **V:** Identificar diferentes tipos de usuarios de bases de datos y sus roles [Familiarizarse].

Bibliografía: [Ramakrishnan and Gehrke, 2002, Silberschatz et al., 2019]

5.2 Modelado de Datos, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Modelos conceptuales (por ejemplo, entidad-relación, diagramas UML) **II:** Conjuntos de entidades, conjuntos de relaciones y atributos. **III:** Restricciones de cardinalidad: uno-a-uno, uno-a-muchos, muchos-a-muchos. **IV:** Conjuntos de entidades débiles y relaciones identificadoras. **V:** Características extendidas E-R: especialización, generalización, agregación. **VI:** Diagramas de clases UML para modelado de datos.

Objetivos de Aprendizaje **I:** Describir los componentes del modelo E-R (o algún otro modelo no relacional) [Familiarizarse] **II:** Modelar un entorno dado usando un modelo de datos conceptual [Usar] **III:** Diseñar diagramas E-R para escenarios del mundo real [Usar]. **IV:** Convertir diagramas E-R a esquemas relacionales [Usar]. **V:** Aplicar características E-R extendidas para modelado complejo [Evaluar].

Bibliografía: [Ramakrishnan and Gehrke, 2002, Elmasri and Navathe, 2015]

5.3 Bases de Datos Relacionales, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Integridad de entidad y referencial: Clave candidata, superclaves **II:** Diseño de base de datos relacional **III:** Operaciones de álgebra relacional: selección, proyección, join, unión, etc. **IV:** Restricciones de integridad: dominio, clave, entidad, referencial. **V:** Cálculo relacional: cálculo de tuplas y de dominio. **VI:** Mapeo de diagramas E-R a esquemas relacionales.

Objetivos de Aprendizaje **I:** Describir las características definitorias detrás del modelo de datos relacional [Familiarizarse] **II:** Comentar sobre la diferencia entre una clave foránea y una superclave [Usar] **III:** Expresar consultas usando álgebra relacional [Usar]. **IV:** Diseñar esquemas relacionales con restricciones apropiadas [Usar]. **V:** Mapear diagramas E-R complejos a relaciones normalizadas [Evaluar].

Bibliografía: [Ramakrishnan and Gehrke, 2002, Elmasri and Navathe, 2015]

5.4 Construcción de Consultas, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Formación de Consultas SQL: *a*) Ejecución interactiva de SQL *b*) Ejecución programática de una consulta SQL **II:** Lenguaje de definición de datos: CREATE, ALTER, DROP. **III:** Tipos de datos y restricciones de dominio. **IV:** Creación y gestión de índices. **V:** Definición de vistas y vistas materializadas. **VI:** Diseño e implementación de esquemas de base de datos.

Objetivos de Aprendizaje **I:** Componer consultas SQL que incorporen selección, proyección, unión, intersección, diferencia de conjuntos y división de conjuntos [Usar] **II:** Definir, en SQL, un esquema de relación, incluyendo todas las restricciones de integridad y activadores de eliminación/actualización [Usar] **III:** Crear y modificar esquemas de base de datos usando SQL DDL [Usar]. **IV:** Implementar restricciones de integridad a nivel de esquema [Usar]. **V:** Diseñar y crear índices apropiados para rendimiento [Evaluar].

Bibliografía: [Ramakrishnan and Gehrke, 2002, Beaulieu, 2009]

5.5 Construcción de Consultas, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Consultas SQL complejas: subconsultas anidadas, subconsultas correlacionadas. **II:** Funciones de agregación y operaciones GROUP BY. **III:** Operaciones de conjunto: UNION, INTERSECT, EXCEPT. **IV:** Operaciones JOIN: inner, outer, natural, cross joins. **V:** Manipulación de datos: INSERT, UPDATE, DELETE. **VI:** Control de transacciones: COMMIT, ROLLBACK, SAVEPOINT.

Objetivos de Aprendizaje **I:** Determinar cuándo una consulta SQL anidada está correlacionada o no [Usar] **II:** Componer una consulta SQL para actualizar una tupla en una relación [Usar] **III:** Formular consultas complejas con múltiples joins y subconsultas [Usar]. **IV:** Aplicar agregación para análisis de datos y generación de informes [Usar]. **V:** Implementar operaciones de modificación de datos con control de transacciones [Evaluar].

Bibliografía: [Ramakrishnan and Gehrke, 2002, Beaulieu, 2009]

5.6 Bases de Datos Relacionales, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Introducción a la teoría de dependencias funcionales **II:** Teoría de Normalización: *a*) Descomposición de un esquema; propiedades de unión sin pérdida y preservación de dependencias de una descomposición *b*) Formas normales (BCNF) *c*) Desnormalización (para eficiencia)

III: Dependencias funcionales y reglas de inferencia. **IV:** Formas normales: 1FN, 2FN, 3FN, FNBC. **V:** Algoritmos de descomposición: sin pérdida de join y que preserva dependencias. **VI:** Consideraciones prácticas en normalización.

Objetivos de Aprendizaje **I:** Describir cómo la teoría de dependencias funcionales generaliza la noción de clave [Familiarizarse] **II:** Detectar qué forma normal produce una descomposición dada [Usar] **III:** Identificar dependencias funcionales en esquemas relacionales [Usar]. **IV:** Normalizar relaciones a FNBC [Evaluar]. **V:** Evaluar descomposiciones para sin pérdida de join y preservación de dependencias [Evaluar].

Bibliografía: [Ramakrishnan and Gehrke, 2002, Elmasri and Navathe, 2015]

5.7 Internos de DBMS, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Gestión de Transacciones: *a*) Niveles de Aislamiento *b*) ACID *c*) Serializabilidad *d*) Transacciones Distribuidas **II:** Control de Concurrencia: *a*) Bloqueo en 2 Fases *b*) Estrategias de manejo de bloqueos mutuos *c*) Modelos de consistencia basados en quórum **III:** Propiedades ACID: Atomicidad, Consistencia, Aislamiento, Durabilidad. **IV:** Serializabilidad y serializabilidad por conflicto. **V:** Control de concurrencia basado en bloqueos: bloqueo en dos fases. **VI:** Detección y prevención de interbloqueos. **VII:** Niveles de aislamiento y sus compensaciones.

Objetivos de Aprendizaje **I:** Describir las cuatro propiedades para un gestor de transacciones correcto [Familiarizarse] **II:** Esbozar el principio de serializabilidad [Familiarizarse] **III:** Explicar las propiedades ACID y su importancia [Usar]. **IV:** Analizar programaciones de transacciones para serializabilidad [Evaluar]. **V:** Aplicar protocolos de bloqueo para control de concurrencia [Usar].

Bibliografía: [Ramakrishnan and Gehrke, 2002, Bernstein et al., 1987]

5.8 Seguridad y Privacidad de Datos, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Diferencias entre seguridad de datos y privacidad de datos **II:** Información de identificación personal (PII) y su protección **III:** Consideraciones éticas para asegurar la seguridad y privacidad de los datos **IV:** Seguridad de bases de datos: autenticación, autorización, encriptación. **V:** Ataques de inyección SQL y prevención. **VI:** Introducción a bases de datos NoSQL y big data. **VII:** Gobernanza de datos y cumplimiento normativo.

Objetivos de Aprendizaje **I:** Describir las diferencias en los objetivos para seguridad de datos y privacidad de datos [Familiarizarse] **II:** Identificar y mitigar riesgos asociados con diferentes enfoques para proteger datos [Usar] **III:** Implementar medidas básicas de seguridad de bases de datos [Usar]. **IV:** Prevenir vulnerabilidades de inyección SQL [Evaluar]. **V:** Discutir consideraciones éticas en la recolección y uso de datos [Evaluar].

Bibliografía: [Ramakrishnan and Gehrke, 2002, Elmasri and Navathe, 2015]

6. Bibliografía

[Beaulieu, 2009] Beaulieu, A. (2009). *Learning SQL*. O'Reilly Media, 2nd edition.

[Bernstein et al., 1987] Bernstein, P. A., Hadzilacos, V., and Goodman, N. (1987). *Concurrency Control and Recovery in Database Systems*. Addison-Wesley.

[Elmasri and Navathe, 2015] Elmasri, R. and Navathe, S. B. (2015). *Fundamentals of Database Systems*. Pearson, 7th edition.

[Ramakrishnan and Gehrke, 2002] Ramakrishnan, R. and Gehrke, J. (2002). *Database Management Systems*. McGraw-Hill, 3rd edition.

[Silberschatz et al., 2019] Silberschatz, A., Korth, H. F., and Sudarshan, S. (2019). *Database System Concepts*. McGraw-Hill, 7th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS401 Metodología de la Investigación (Obligatorio)

1. Información general

- **Semestre**: 4to Sem. **Créditos**: 2
- **Horas del curso**: **Teoría**: 1 horas; **Práctica**: 2 horas;
- **Prerrequisitos**:
  - CS100 Introducción a la Ciencia de la Computación (2do Sem-Pág 276)

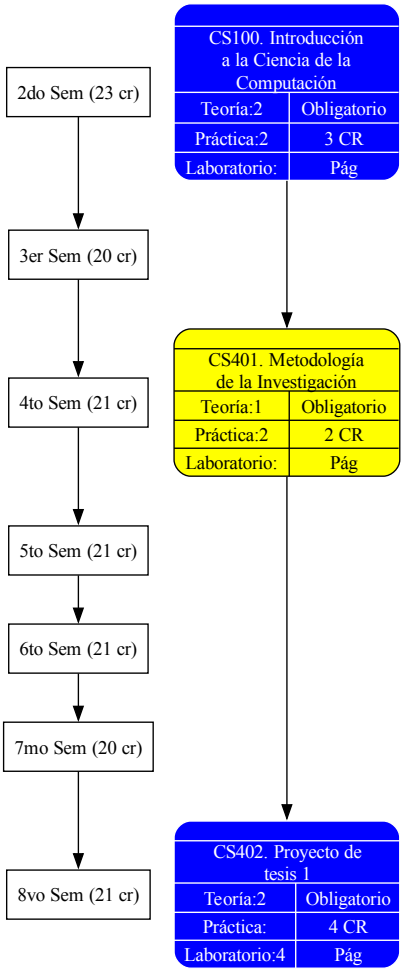

Figura 5.22: Mapa de Conexión. CS401 Metodología de la Investigación

2. Fundamentación

Este curso tiene por objetivo que el alumno aprenda a realizar una investigación de carácter científico en el área de computación. Los docentes del curso determinarán un área de estudio para cada alumno y se les hará entrega de bibliografía para analizar. A partir de la misma y de fuentes adicionales investigadas por el alumno, el alumno deberá construir un artículo del tipo survey sobre el tema asignado.

3. Objetivos del curso.

- Aprender a iniciar una investigación científica en el área de computación con rigor metodológico.
- Conocer las principales fuentes bibliográficas académicas: IEEE Xplore, ACM Digital Library, Google Scholar y arXiv.
- Analizar críticamente propuestas existentes sobre un tópico y relacionarlas en una revisión bibliográfica.
- Redactar documentos técnicos en computación usando LaTeXcon plantillas de conferencias IEEE y ACM.
- Reproducir resultados existentes en un tópico mediante implementación y experimentación.
- Construir un artículo de tipo survey con taxonomía, síntesis y comparación de enfoques existentes.
- Los entregables de este curso son:

  Avance parcial: Dominio del tema del artículo y bibliografía preliminar en formato de artículo LaTeX.

  Final: Artículo del tipo survey concluido, conteniendo opcionalmente los resultados experimentales de la(s) técnica(s) estudiada(s).

4. Competencias Generales (*Outcomes*)

AG-C02) Ética: Aplica principios éticos y se compromete con la ética profesional y las normas de la práctica profesional de la computación. (Evaluar)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Evaluar)

AG-C05) Gestión de Proyectos: Aplica los principios de gestión en computación para gestionar proyectos. (Evaluar)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Evaluar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Evaluar)

5. Contenido

5.1 Fundamentos de Investigación Científica en Computación, 8 hr(s)

Competencias: AG-C05,AG-C08

Tópicos: I: Tipos de investigación en computación: empírica, teórica, constructiva y evaluativa. **II:** El método científico aplicado a la investigación en computación. **III:** Principales bases de datos académicas: IEEE Xplore, ACM Digital Library, Google Scholar, Semantic Scholar y arXiv. **IV:** Estrategias de búsqueda bibliográfica: operadores booleanos, seguimiento de citas y búsqueda sistemática. **V:** Gestores de referencias: Zotero, Mendeley e integración con BibTeX.

Objetivos de Aprendizaje I: Identificar los tipos de investigación en computación y sus diferencias metodológicas. [Familiarizarse] **II:** Utilizar las principales bases de datos académicas para encontrar literatura relevante en un tópico dado. [Usar] **III:** Organizar referencias bibliográficas usando gestores de referencias y formato BibTeX. [Usar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.2 Lectura Crítica y Análisis de Literatura Científica, 8 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: **I:** Estructura de un artículo científico: resumen, introducción, trabajos relacionados, metodología, resultados y conclusiones. **II:** Técnicas para la lectura eficiente de artículos científicos (lectura en tres pasadas). **III:** Evaluación crítica: identificación de contribuciones, limitaciones y problemas abiertos. **IV:** Construcción de una matriz de lectura y tablas comparativas de trabajos relacionados. **V:** Clasificación y diseño de taxonomías de enfoques existentes sobre un tópico.

Objetivos de Aprendizaje **I:** Leer y evaluar críticamente artículos científicos identificando sus contribuciones y limitaciones. [Usar] **II:** Construir una matriz de lectura organizada que sintetice múltiples trabajos sobre un tópico dado. [Usar] **III:** Diseñar una taxonomía preliminar de enfoques para un área de investigación asignada. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008, CiteSeer.IST, 2008]

5.3 Ética en Investigación e Integridad Académica, 6 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: **I:** Integridad académica: plagio, autoplagio, fabricación y falsificación de datos. **II:** Ética en investigación en computación: privacidad, sesgo algorítmico y estudios con sujetos humanos. **III:** Acceso abierto, licencias (Creative Commons, GPL) y propiedad intelectual en software. **IV:** Buenas prácticas de citación y atribución de autoría en trabajos colaborativos. **V:** El proceso de revisión por pares: rol del revisor y responsabilidades del autor.

Objetivos de Aprendizaje **I:** Distinguir las formas de mala conducta académica y aplicar prácticas de citación correctas. [Usar] **II:** Analizar dilemas éticos en investigación computacional aplicando principios de integridad. [Evaluar] **III:** Explicar los derechos de propiedad intelectual y las licencias relevantes en computación. [Familiarizarse]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.4 Redacción Técnica y Herramientas de Escritura Científica, 10 hr(s)

Competencias: AG-C04,AG-C09

Tópicos: **I:** Estructura de LaTeX: preámbulo, secciones, ecuaciones, figuras, tablas y referencias cruzadas. **II:** Plantillas de conferencias y revistas: IEEE y ACM. **III:** Estilo de escritura académica en computación: claridad, precisión y voz activa. **IV:** Redacción de cada sección del artículo: resumen, introducción, trabajos relacionados, metodología, resultados y conclusiones. **V:** Integración Continua de documentos: control de versiones de documentos LaTeXcon Git. **VI:** Gestión de bibliografía con BibTeX y JabRef.

Objetivos de Aprendizaje **I:** Redactar un artículo técnico completo en LaTeXusando plantillas IEEE o ACM. [Usar] **II:** Aplicar las convenciones de escritura académica propias de la comunidad de computación. [Usar] **III:** Evaluar la calidad de redacción de un artículo propio o ajeno según criterios de claridad y estructura. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.5 Revisión Sistemática de Literatura y Artículos de Survey, 10 hr(s)

Competencias: AG-C04,AG-C08,AG-C09

Tópicos: **I:** Tipos de revisiones: narrativa, sistemática, estudio de mapeo y meta-análisis. **II:** Protocolo de revisión sistemática de literatura (SLR): preguntas de investigación, estrategia de búsqueda y criterios de selección. **III:** Pautas PRISMA: diagrama de flujo de selección de estudios primarios. **IV:** Técnicas de síntesis: narrativa, votación y análisis temático. **V:** Diseño de taxonomías y esquemas de clasificación para un survey. **VI:** Estructura y redacción de un artículo del tipo survey.

Objetivos de Aprendizaje I: Diseñar y ejecutar un protocolo de revisión sistemática de literatura para un tópico de computación. [Usar] **II:** Aplicar las pautas PRISMA para documentar y reportar el proceso de selección de estudios. [Usar] **III:** Redactar un artículo de tipo survey con taxonomía, síntesis crítica y comparación de enfoques. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008, CiteSeer.IST, 2008]

5.6 Presentación y Difusión de Resultados de Investigación, 6 hr(s)

Competencias: AG-C02,AG-C04,AG-C05

Tópicos: I: Técnicas de presentación oral para audiencias científicas: estructura, manejo del tiempo y sesión de preguntas. **II:** Diseño de pósters para conferencias y simposios de investigación. **III:** Respuesta a revisores: cómo abordar comentarios de forma constructiva y argumentada. **IV:** Planificación de un proyecto de investigación: cronograma, hitos y entregables. **V:** Prácticas de colaboración y coautoría en investigación científica.

Objetivos de Aprendizaje I: Preparar y exponer una presentación oral de resultados de investigación ante una audiencia académica. [Usar] **II:** Elaborar un plan de trabajo para un proyecto de investigación con cronograma e hitos. [Usar] **III:** Responder constructivamente a comentarios de revisores en el proceso de publicación. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

6. Bibliografía

[ACM, 2008] ACM (2008). *Digital Library.* Association for Computing Machinery. http://portal.acm.org/dl.cfm.

[CiteSeer.IST, 2008] CiteSeer.IST (2008). *Scientific Literature Digital Library.* College of Information Sciences and Technology, Penn State University. http://citeseer.ist.psu.edu.

[IEEE-Computer Society, 2008] IEEE-Computer Society (2008). *Digital Library.* IEEE-Computer Society. http://www.computer.org/publications/dlib.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

MA202 Métodos Numéricos (Obligatorio)

1. Información general

- **Semestre**: 4to Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas;
- **Prerrequisitos**:
  - BMA103 Cálculo Integral (2do Sem-Pág 295)

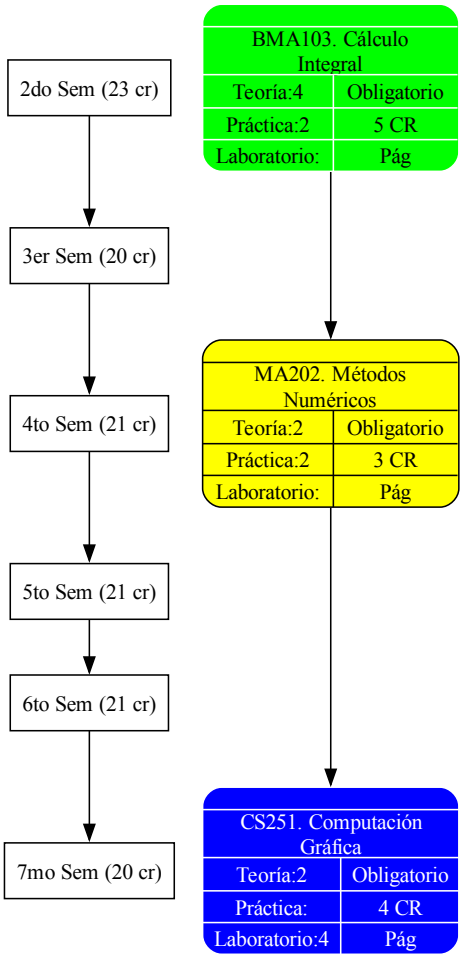

Figura 5.23: Mapa de Conexión. MA202 Métodos Numéricos

2. Fundamentación

Los métodos numéricos son una herramienta fundamental compartida por la informática y las disciplinas de ingeniería para aproximar soluciones a problemas matemáticos que no pueden resolverse analíticamente. Este curso proporciona una introducción a los métodos numéricos más comunes, incluyendo el análisis de errores, la solución de ecuaciones no lineales, la interpolación, la integración numérica y la solución de ecuaciones diferenciales, en contextos científicos y de ingeniería.

3. Objetivos del curso.

- Comprender la importancia de los métodos numéricos en la resolución de problemas científicos y de ingeniería.
- Aplicar diferentes métodos numéricos para aproximar soluciones a problemas matemáticos.
- Analizar la precisión y eficiencia de los métodos numéricos utilizados.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Usar)

5. Contenido

5.1 Análisis de Errores y Aritmética de Punto Flotante, 6 hr(s)

Competencias: AG-C08,AG-C11

Tópicos: **I:** Representación en punto flotante IEEE 754: épsilon de máquina, desbordamiento, subdesbordamiento y modos de redondeo **II:** Errores de redondeo, cancelación catastrófica y pérdida de significado **III:** Análisis de error directo e inverso; número de condición de un problema **IV:** Estabilidad numérica: algoritmos estables vs. inestables; análisis de error inverso de Wilkinson **V:** Propagación de errores en operaciones aritméticas y evaluación de funciones

Objetivos de Aprendizaje **I:** Explicar el estándar IEEE 754 de punto flotante e identificar las fuentes de errores de redondeo y cancelación [Familiarizarse] **II:** Estimar el número de condición de un problema y predecir la precisión de un resultado numérico [Usar] **III:** Analizar un algoritmo para determinar su estabilidad numérica usando cotas de error directo e inverso [Evaluar]

Bibliografía: [Chapra and Canale, 2015, Burden and Faires, 2010]

5.2 Búsqueda de Raíces de Ecuaciones No Lineales, 10 hr(s)

Competencias: AG-C08,AG-C11

Tópicos: **I:** Método de bisección: acotamiento del intervalo, criterio de parada y cota del error **II:** Método de Newton-Raphson: derivación, convergencia cuadrática y casos de falla **III:** Método de la secante como alternativa al método de Newton sin derivada explícita **IV:** Orden de convergencia y comparación de la eficiencia entre métodos iterativos **V:** Extensión del método de Newton a sistemas de ecuaciones no lineales mediante la matriz Jacobiana, base de solucionadores en aprendizaje automático

Objetivos de Aprendizaje **I:** Comparar las garantías de convergencia de los métodos de bisección, Newton-Raphson y secante [Familiarizarse] **II:** Implementar el método de Newton-Raphson para aproximar raíces de una función no lineal [Usar] **III:** Analizar el orden de convergencia de un método iterativo y justificar su elección según el costo computacional [Evaluar]

Bibliografía: [Chapra and Canale, 2015, Burden and Faires, 2010]

5.3 Teoría de Aproximación e Interpolación, 10 hr(s)

Competencias: AG-C08,AG-C11

Tópicos: **I:** Interpolación polinomial: Lagrange, diferencias divididas de Newton y fenómeno de Runge **II:** Interpolación por splines: splines cúbicos, B-splines y métodos de funciones polinómicas a trozos **III:** Mejor aproximación en espacios normados: Chebyshev (minimax) y mínimos cuadrados **IV:** Polinomios de Chebyshev: propiedades, ortogonalidad y convergencia espectral **V:** Aproximación trigonométrica y transformada rápida de Fourier (FFT)

Objetivos de Aprendizaje **I:** Explicar el fenómeno de Runge y justificar la elección de nodos de Chebyshev para mitigarlo [Familiarizarse] **II:** Construir interpolantes por splines cúbicos y de Chebyshev para datos dados y estimar el error de interpolación [Usar] **III:** Aplicar la FFT para calcular eficientemente aproximaciones trigonométricas de una función muestreada [Evaluar]

Bibliografía: [Chapra and Canale, 2015, Burden and Faires, 2010]

5.4 Integración Numérica y Cuadratura, 10 hr(s)

Competencias: AG-C08,AG-C11

Tópicos: **I:** Reglas de Newton-Cotes: trapezoidal, Simpson y reglas compuestas; análisis de error **II:** Cuadratura adaptativa y control automático del error **III:** Cuadratura gaussiana: nodos y pesos óptimos, conexión con polinomios ortogonales **IV:** Tratamiento numérico de integrales impropias y singulares **V:** Integración de Monte Carlo y métodos cuasi-Monte Carlo para integrales de alta dimensión

Objetivos de Aprendizaje **I:** Derivar la fórmula de error para la regla compuesta de Simpson e identificar su orden de precisión [Familiarizarse] **II:** Seleccionar y aplicar una regla de cuadratura apropiada (Gaussiana, adaptativa) según la regularidad del integrando [Usar] **III:** Aplicar integración de Monte Carlo para estimar integrales de alta dimensión y cuantificar el error estadístico [Evaluar]

Bibliografía: [Chapra and Canale, 2015, Burden and Faires, 2010]

5.5 Métodos Numéricos para Ecuaciones Diferenciales, 12 hr(s)

Competencias: AG-C08,AG-C11

Tópicos: **I:** Métodos de Runge-Kutta: Euler, RK4 y métodos embebidos para control de errores **II:** Métodos de pasos múltiples: Adams-Bashforth, Adams-Moulton y métodos BDF **III:** Análisis de estabilidad: cero-estabilidad, regiones de estabilidad absoluta y EDOs rígidas **IV:** Métodos de diferencias finitas para EDPs parabólicas y elípticas: estabilidad y convergencia

Objetivos de Aprendizaje **I:** Comparar solucionadores de EDOs explícitos e implícitos y explicar su estabilidad relativa para problemas rígidos [Familiarizarse] **II:** Implementar un método de Runge-Kutta con control adaptativo del paso y aplicarlo a un sistema de EDOs [Usar] **III:** Analizar la estabilidad y convergencia de un esquema de diferencias finitas para una EDP parabólica [Evaluar]

Bibliografía: [Chapra and Canale, 2015, Burden and Faires, 2010]

5.6 Álgebra Lineal Numérica, 12 hr(s)

Competencias: AG-C08,AG-C11

Tópicos: **I:** Eliminación gaussiana con pivoteo parcial, factorización LU y análisis de complejidad **II:** Factorización QR vía Householder y Gram-Schmidt; problemas de mínimos cuadrados **III:** Algoritmos de valores propios: iteración de la potencia, algoritmo QR y el método de Lanczos **IV:** Descomposición en valores singulares: cálculo, truncamiento y aplicaciones (ACP,

pseudoinversa) **V:** Métodos iterativos para sistemas dispersos grandes: GC, GMRES y precondicionamiento

Objetivos de Aprendizaje I: Comparar solucionadores directos e iterativos y seleccionar el método adecuado para una estructura matricial dada [Familiarizarse] **II:** Aplicar la factorización QR y la SVD para resolver problemas de mínimos cuadrados y calcular aproximaciones de bajo rango [Usar] **III:** Analizar la estabilidad numérica y el costo computacional de los algoritmos de valores propios [Evaluar]

Bibliografía: [Burden and Faires, 2010, Trefethen and III, 1997]

5.7 Método de Elementos Finitos, 8 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Formulación débil (variacional) de problemas de valor en la frontera y espacios de Sobolev **II:** Método de Galerkin: espacios de aproximación de dimensión finita y matriz de rigidez **III:** Espacios de elementos finitos triangulares y cuadriláteros: elementos de Lagrange y conformidad **IV:** Estimaciones de error a priori: lema de Céa y cotas de error de interpolación **V:** Estimaciones de error a posteriori y refinamiento adaptativo de malla

Objetivos de Aprendizaje I: Derivar la formulación débil de un PVE elíptico y mostrar su equivalencia con la forma fuerte [Familiarizarse] **II:** Ensamblar la matriz de rigidez global y el vector de carga para una discretización lineal por elementos finitos [Usar] **III:** Estimar el error en H^1 de una solución por elementos finitos usando el lema de Céa y la teoría de interpolación [Evaluar]

Bibliografía: [Reddy, 2019, Burden and Faires, 2010]

5.8 Algoritmos de Optimización, 4 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Descenso de gradiente y métodos de búsqueda lineal: condiciones de Armijo-Wolfe y tasas de convergencia **II:** Método de Newton y métodos cuasi-Newton (BFGS, L-BFGS) **III:** Optimización con restricciones: condiciones KKT, métodos de penalización y programación cuadrática secuencial **IV:** Algoritmos de optimización convexa: métodos de punto interior y ADMM **V:** Descenso de gradiente estocástico (SGD), reducción de varianza y el optimizador Adam

Objetivos de Aprendizaje I: Explicar las garantías de convergencia del descenso de gradiente para funciones convexas suaves [Familiarizarse] **II:** Aplicar los métodos de Newton y cuasi-Newton a problemas de optimización sin restricciones [Usar] **III:** Formular un problema de optimización con restricciones como un sistema KKT y aplicar un método de punto interior [Evaluar]

Bibliografía: [Nocedal and Wright, 2006]

5.9 Computación Paralela y de Alto Rendimiento, 4 hr(s)

Competencias: AG-C11

Tópicos: I: Arquitecturas paralelas: memoria compartida (OpenMP), memoria distribuida (MPI) y GPU (CUDA) **II:** Modelos de rendimiento: ley de Amdahl, modelo de línea techada y complejidad de comunicación **III:** Algoritmos paralelos para álgebra lineal densa y dispersa (ScaLAPACK, PETSc) **IV:** Descomposición de dominio y solucionadores paralelos de EDPs **V:** Diferenciación automática (modo directo y reverso) y su rol en ML científico

Objetivos de Aprendizaje I: Explicar la ley de Amdahl e identificar los cuellos de botella que limitan la aceleración paralela en un algoritmo dado [Familiarizarse] **II:** Implementar una rutina paralela de álgebra lineal usando MPI o OpenMP y medir la eficiencia paralela [Usar] **III:** Aplicar diferenciación automática para calcular gradientes de funciones científicas complejas con fines de optimización [Evaluar]

Bibliografía: [Pacheco and Malensek, 2022]

5.10 Teoría de Grafos, 4 hr(s)

Competencias: AG-C08

Tópicos: **I:** Grafos: definiciones, isomorfismo, sucesiones de grados, árboles y árboles de expansión **II:** Conectividad, Teorema de Menger y flujos en redes (flujo máximo-corte mínimo) **III:** Emparejamientos (Teorema de Hall), coloraciones de grafos y el polinomio cromático **IV:** Grafos planos, fórmula de Euler, Teorema de Kuratowski y el teorema de los cuatro colores **V:** Matrices de adyacencia y laplaciana, valores propios y grafos expansores

Objetivos de Aprendizaje **I:** Identificar propiedades estructurales de grafos (conectividad, planaridad, bipartición) y aplicar la fórmula de Euler [Familiarizarse] **II:** Aplicar el Teorema de Hall y algoritmos de flujo en redes a problemas de emparejamiento y enrutamiento [Usar] **III:** Analizar el espectro de un grafo para acotar su número cromático y sus propiedades de conectividad [Evaluar]

Bibliografía: [West, 2001]

5.11 Análisis de Series Temporales, 2 hr(s)

Competencias: AG-C08,AG-C11

Tópicos: **I:** Estacionariedad, función de autocovarianza y función de autocorrelación (ACF) **II:** Modelos ARMA: identificación, estimación (Yule-Walker, MLE) y diagnóstico **III:** Densidad espectral, el periodograma y el teorema de Wiener-Khinchin **IV:** Modelos de espacio de estados y el filtro de Kalman **V:** Modelos ARIMA, ajuste estacional y predicción a múltiples pasos

Objetivos de Aprendizaje **I:** Identificar la estacionariedad y determinar el orden de un modelo ARMA a partir de gráficos ACF y PACF [Familiarizarse] **II:** Ajustar modelos ARIMA a datos de series temporales, validar residuos y producir predicciones [Usar] **III:** Aplicar el filtro de Kalman para estimar estados ocultos en un modelo lineal gaussiano de espacio de estados [Evaluar]

Bibliografía: [Shumway and Stoffer, 2017]

6. Bibliografía

[Burden and Faires, 2010] Burden, R. L. and Faires, J. D. (2010). *Numerical Analysis*. Cengage Learning.

[Chapra and Canale, 2015] Chapra, S. C. and Canale, R. P. (2015). *Numerical Methods for Engineers*. McGraw-Hill Education.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, 2nd edition.

[Pacheco and Malensek, 2022] Pacheco, P. S. and Malensek, M. (2022). *An Introduction to Parallel Programming*. Morgan Kaufmann, 2nd edition.

[Reddy, 2019] Reddy, J. N. (2019). *An Introduction to the Finite Element Method*. McGraw-Hill Education, 4th edition.

[Shumway and Stoffer, 2017] Shumway, R. H. and Stoffer, D. S. (2017). *Time Series Analysis and Its Applications: With R Examples*. Springer, 4th edition.

[Trefethen and III, 1997] Trefethen, L. N. and III, D. B. (1997). *Numerical Linear Algebra*. SIAM.

[West, 2001] West, D. B. (2001). *Introduction to Graph Theory*. Prentice Hall, 2nd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

BEI202 Inglés IV (Electivo)

1. Información general

- **Semestre**: 4to Sem. **Créditos**: 2
- **Horas del curso**: **Práctica**: 4 horas;
- **Prerrequisitos**:
  - BEI201 Inglés III (3er Sem-Pág 329)

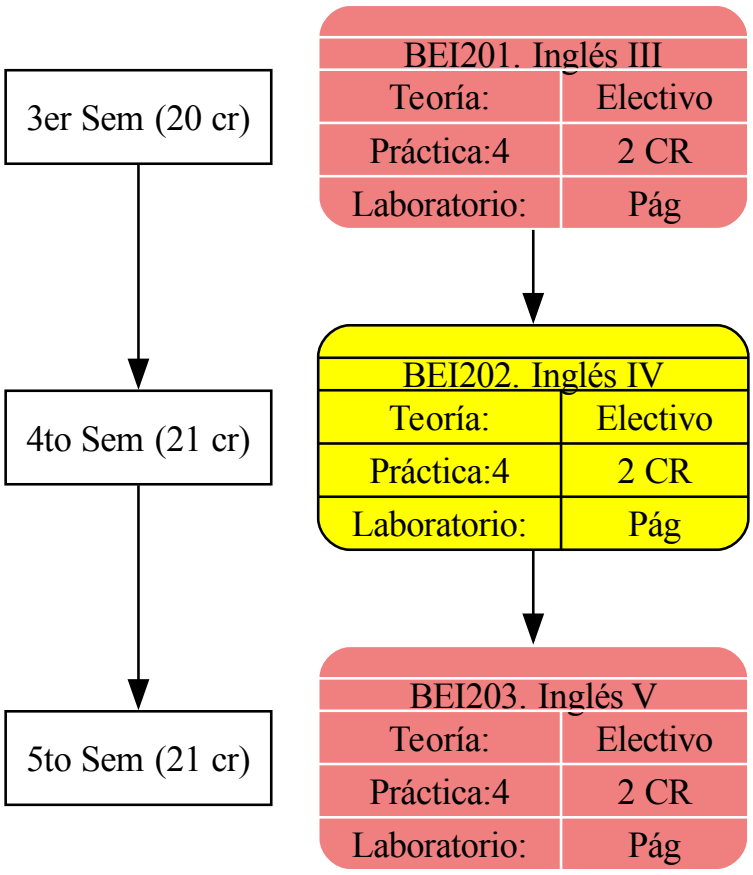

Figura 5.24: Mapa de Conexión. BEI202 Inglés IV

2. Fundamentación

En el nivel intermedio, los estudiantes aplican el inglés al rango completo de comunicación técnica requerida en la práctica de la computación y la ingeniería: lectura de artículos de investigación, redacción de documentos técnicos estructurados, análisis de estudios de caso y participación en diálogos profesionales. Las estructuras gramaticales se enseñan a través de contextos auténticos de computación e ingeniería, reforzando simultáneamente la competencia lingüística y el conocimiento disciplinar.

3. Objetivos del curso.

- Desarrollar la capacidad de escribir y leer documentos técnicos estructurados en inglés.
- Llevar al estudiante a una expresión más fluida y natural en el inglés técnico y científico.
- Dominar el vocabulario de computación, ingeniería de sistemas y ciencias aplicadas.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Familiarizarse)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Usar)

5. Contenido

5.1 Buenas Prácticas y Ética, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Modales de obligación y consejo: Should, Must, Have to en códigos de ingeniería. **II:** Prohibición y ausencia de obligación en licencias de software y privacidad de datos. **III:** Vocabulario: Ética profesional, propiedad intelectual, IA responsable. **IV:** Redacción: Documentos formales de recomendación técnica y política.

Objetivos de Aprendizaje I: Al finalizar la unidad, el estudiante utiliza correctamente los auxiliares modales para expresar estándares de ingeniería, obligaciones éticas y recomendaciones profesionales en entornos técnicos formales.

Bibliografía: [? line:47, ? line:47, ? line:47]

5.2 Fallos y Causas Raíz, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Pasado Perfecto: Estructura y uso en análisis de causa raíz y post-mortems. **II:** Contraste entre Pasado Simple, Continuo y Perfecto en cronologías de fallos. **III:** Vocabulario: Interrupciones de sistema, brechas de seguridad, fallos en cascada. **IV:** Expresiones para reportar información: estilo indirecto en informes de incidentes.

Objetivos de Aprendizaje I: El estudiante reconstruye cronologías complejas de fallos y análisis de causas raíz estableciendo secuencias cronológicas claras mediante el contraste de tiempos de pasado.

Bibliografía: [? line:60, ? line:60, ? line:60]

5.3 Lógica Si-Entonces, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Condicionales Tipo 0, 1 y 2 aplicados a la lógica computacional y decisiones de ingeniería. **II:** Oraciones con Ïfz Ünless"para restricciones y requisitos de sistemas. **III:** Vocabulario: Escenarios hipotéticos, análisis de riesgo y trade-offs de optimización. **IV:** Pensamiento de diseño hipotético: Ïf we used X, the system would..."

Objetivos de Aprendizaje I: El estudiante plantea escenarios técnicos reales e hipotéticos y sus consecuencias, utilizando estructuras condicionales para razonar sobre decisiones de diseño e ingeniería.

Bibliografía: [? line:73, ? line:73, ? line:73]

5.4 Investigación en Curso, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Presente Perfecto Continuo para describir investigación en progreso. **II:** Contraste entre Presente Perfecto Simple y Continuo en escritura científica. **III:** Vocabulario: Integración continua, desarrollo ágil, investigación iterativa. **IV:** Formación de palabras: Prefijos y sufijos en terminología científica y técnica.

Objetivos de Aprendizaje **I:** El estudiante describe actividades de investigación y procesos de desarrollo que comenzaron en el pasado y continúan en el presente, enfatizando la duración y el progreso iterativo.

Bibliografía: [**?** line:86 , **?** line:86 , **?** line:86]

5.5 Fabricación y Despliegue, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Voz Pasiva en todos los tiempos básicos para descripciones de procesos técnicos. **II:** Verbos con dos objetos en pasiva: data is sent to the server by the client. **III:** Vocabulario: Fabricación de hardware, despliegue en la nube, pipelines CI/CD. **IV:** Redacción: Descripción técnica de un proceso de despliegue o producción.

Objetivos de Aprendizaje **I:** El estudiante utiliza la voz pasiva en múltiples tiempos verbales para describir procesos técnicos donde la acción, el sistema o el resultado es el foco en lugar del agente.

Bibliografía: [**?** line:99 , **?** line:99 , **?** line:99]

5.6 Hojas de Ruta Tecnológicas, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Futuro Continuo y Futuro Perfecto para cronogramas e hitos de proyectos. **II:** Adverbios de probabilidad: perhaps, probably, definitely — en prospectiva tecnológica. **III:** Vocabulario: Hojas de ruta de producto, planificación de versiones, áreas de investigación emergente. **IV:** Presentaciones orales: .[El] estado de la IA / computación cuántica en 5 años".

Objetivos de Aprendizaje **I:** El estudiante proyecta estados técnicos, hitos futuros completados y resultados de investigación, discutiendo hojas de ruta de computación e ingeniería con vocabulario avanzado.

Bibliografía: [**?** line:112 , **?** line:112 , **?** line:112]

5.7 Equipos Técnicos, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Preguntas indirectas y etiquetas (Tag questions) en entrevistas técnicas y revisiones de pares. **II:** Phrasal Verbs comunes en entornos de ingeniería e investigación. **III:** Vocabulario: Herramientas de colaboración, prácticas de revisión de código, roles de equipo. **IV:** Redacción: Publicaciones cortas de blog técnico y actualizaciones de equipo de ingeniería.

Objetivos de Aprendizaje **I:** El estudiante utiliza formas de consulta cortés y estrategias de confirmación en entrevistas técnicas y discusiones de equipo, mejorando su fluidez y naturalidad en contextos profesionales de ingeniería.

Bibliografía: [**?** line:125 , **?** line:125 , **?** line:125]

5.8 Inglés Técnico IV: Redes y Seguridad, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Terminología de redes: protocolos, topologías, modelo OSI, enrutamiento. **II:** Vocabulario de ciberseguridad: vulnerabilidades, amenazas, cifrado, autenticación. **III:** Lectura y redacción de informes de incidentes de red y auditorías de seguridad. **IV:** Proyecto: Evaluación de seguridad para una infraestructura computacional.

Objetivos de Aprendizaje I: Al finalizar el curso, el estudiante domina la terminología técnica de redes y ciberseguridad para leer documentación especializada y comunicarse eficazmente con profesionales de TI y seguridad en inglés.

Bibliografía: [**?** line:138 , **?** line:138 , **?** line:138]

6. Bibliografía

line:143 line:143 line:143

Quinto Semestre

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS212 Análisis y Diseño de Algoritmos (Obligatorio)

1. Información general

- **Semestre**: 5to Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS210 Algoritmos y Estructuras de Datos (4to Sem-Pág 333)
  - CS211 Teoría de la Computación (4to Sem-Pág 338)

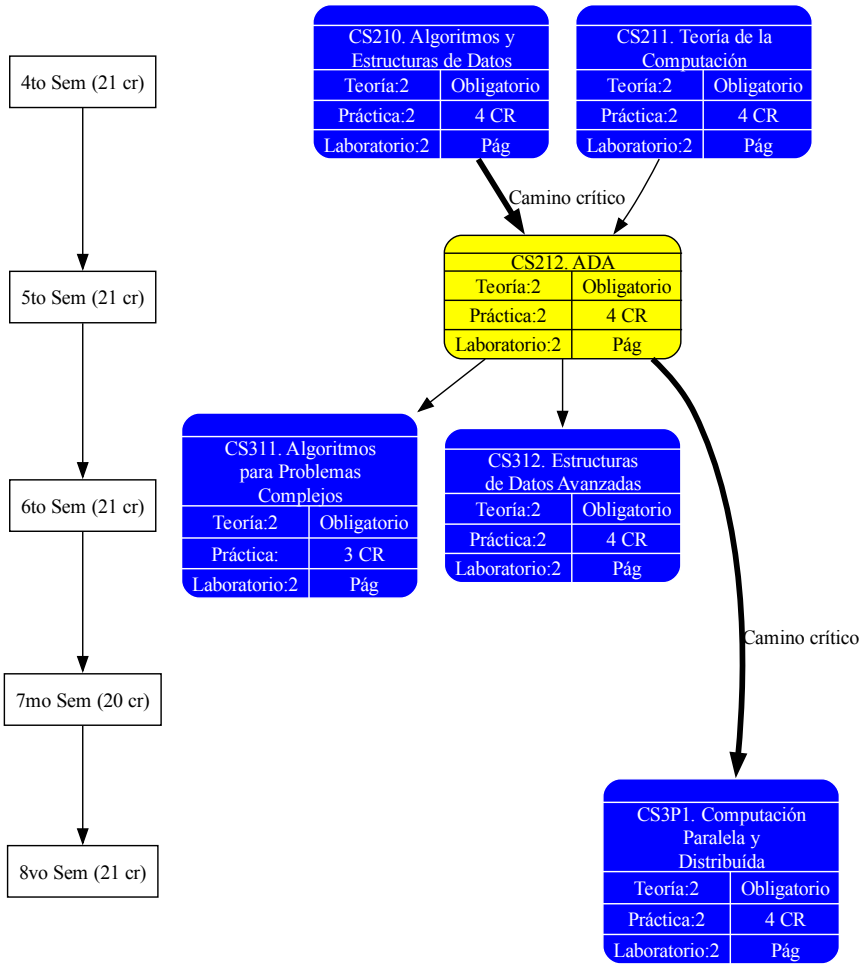

Figura 5.25: Mapa de Conexión. CS212 Análisis y Diseño de Algoritmos

2. Fundamentación

Los algoritmos son el corazón de la ciencia de la computación. Este curso proporciona las herramientas matemáticas y analíticas necesarias para evaluar la eficiencia de los algoritmos en términos de

tiempo y espacio. Los estudiantes explorarán estructuras de datos fundamentales, análisis de complejidad y varios paradigmas de diseño, como fuerza bruta, algoritmos voraces y programación dinámica. Comprender estas técnicas es esencial para construir software que no solo sea funcional, sino también escalable y eficiente en el uso de recursos.

3. Objetivos del curso.

- Dominar el uso de la notación asintótica para analizar el rendimiento de los algoritmos.
- Aplicar estructuras de datos fundamentales para resolver problemas computacionales.
- Usar estrategias algorítmicas como Voraz, Divide y Vencerás y Programación Dinámica.
- Analizar la complejidad de algoritmos en el mejor, promedio y peor de los casos.
- Desarrollar la capacidad de seleccionar el algoritmo más apropiado para un problema dado.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Estructuras de Datos Fundamentales, 6 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Tipo de Dato Abstracto (ADT) y operaciones sobre un ADT: *a*) Operaciones de diccionario (insertar, eliminar, encontrar) **II:** Arreglos: *a*) Numéricos vs no numéricos, cadenas de caracteres *b*) Unidimensionales (vector) vs multidimensionales (matriz) **III:** Registros/Estructuras/Tuplas y Objetos **IV:** Listas enlazadas (por razones históricas): *a*) Simples vs Dobles y Lineales vs Circulares **V:** Pilas **VI:** Colas y deques: *a*) Cola de prioridad basada en montículo **VII:** Tablas/mapas hash: *a*) Resolución de colisiones y complejidad (por ejemplo, sondeo, encadenamiento, rehash) **VIII:** árboles: *a*) Binarios, n-arios y árboles de búsqueda *b*) Balanceados (por ejemplo, AVL, Rojo-Negro, Montículo) **IX:** Conjuntos

Objetivos de Aprendizaje I: Para cada ADT/Estructura de Datos en esta unidad: *a*) Explicar su definición, propiedades, representación(es) y operaciones de ADT asociadas. *b*) Explicar paso a paso cómo las operaciones de ADT asociadas con la estructura de datos la transforman. [Explicar] **II:** Explicar cómo se maneja la evitación de colisiones y la resolución de colisiones en tablas hash [Explicar] **III:** Explicar la propiedad de montículo y el uso de montículos como una implementación de una cola de prioridad [Explicar]

Bibliografía: [Cormen et al., 2022, Kleinberg and Tardos, 2005]

5.2 Algoritmos Fundamentales, 6 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Grafos (por ejemplo, [no]dirigidos, [a]cíclicos, [no]conexos y [no]ponderados): *a*) Representación de grafos: lista de adyacencia vs matriz **II:** Algoritmos de búsqueda: *a*) Complejidad $O(n)$ (por ejemplo, búsqueda lineal/secuencial en arreglo/lista) *b*) Complejidad $O(\log_2 n)$ (por ejemplo, búsqueda binaria) *c*) Complejidad $O(\log_b n)$ (por ejemplo, búsqueda en árbol no informada en profundidad/amplitud) **III:** Algoritmos de ordenamiento (por ejemplo, estables, inestables): *a*) Complejidad $O(n^2)$ (por ejemplo, inserción, selección) *b*) Complejidad $O(n \log n)$ (por ejemplo, quicksort, merge, timesort) **IV:** Algoritmos de grafos: *a*) Camino más corto (por ejemplo, Dijkstra, Floyd) *b*) árbol de expansión mínima (por ejemplo, Prim, Kruskal) **V:** Algoritmos de ordenamiento: *a*) Complejidad $O(n \log n)$ heapsort *b*) Pseudo $O(n)$ complejidad (por ejemplo, bucket, counting, radix) **VI:** Algoritmos de grafos: *a*) Clausura transitiva

(por ejemplo, Warshall) *b*) Ordenamiento topológico **VII:** Emparejamiento: *a*) Emparejamiento eficiente de cadenas (por ejemplo, Boyer-Moore, Knuth-Morris-Pratt) *b*) Emparejamiento de subsecuencia común más larga *c*) Emparejamiento de expresiones regulares

Objetivos de Aprendizaje I: Para cada algoritmo en esta unidad explicar paso a paso cómo opera el algoritmo [Explicar] **II:** Para cada enfoque algorítmico (por ejemplo, ordenamiento) en esta unidad aplicar un ejemplo prototípico del enfoque (por ejemplo, ordenamiento por mezcla) [Aplicar] **III:** Dados los requisitos para un problema, desarrollar múltiples soluciones usando varias estructuras de datos y algoritmos. Posteriormente, evaluar la idoneidad, fortalezas y debilidades seleccionando un enfoque que satisfaga mejor los requisitos [Crear] **IV:** Explicar factores más allá de la eficiencia computacional que influyen en la elección de algoritmos, como el tiempo de programación, la mantenibilidad y el uso de patrones específicos de la aplicación en los datos de entrada [Explicar] **V:** Para cada uno de los algoritmos y enfoques algorítmicos en los temas del Núcleo KA: *a*) Explicar un ejemplo prototípico del algoritmo, y *b*) Explicar paso a paso cómo opera el algoritmo. [Explicar]

Bibliografía: [Cormen et al., 2022, Kleinberg and Tardos, 2005]

5.3 Algoritmos Avanzados, 4 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Algoritmos de criptografía (por ejemplo, SHA-256) **II:** Algoritmos paralelos **III:** Algoritmos de consenso (por ejemplo, Blockchain): *a*) Prueba de trabajo vs prueba de participación **IV:** Algoritmos de computación cuántica: *a*) Basados en oráculo (por ejemplo, Deutsch-Jozsa, Bernstein-Vazirani, Simon) *b*) Aceleración superpolinomial mediante QFT (por ejemplo, Shor) *c*) Aceleración polinomial mediante amplificación de amplitud (por ejemplo, Grover) **V:** Algoritmo de Transformada Rápida de Fourier (FFT) **VI:** Algoritmo de evolución diferencial

Objetivos de Aprendizaje I: Una apreciación de la computación cuántica y su aplicación a ciertos problemas [Explicar]

Bibliografía: [Cormen et al., 2022, Kleinberg and Tardos, 2005]

5.4 Estrategias Algorítmicas, 24 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Paradigmas: *a*) Fuerza Bruta (por ejemplo, búsqueda lineal, ordenamiento por selección, viajante de comercio, mochila) *b*) Disminuir y Vencer: 1) Por una Constante (por ejemplo, ordenamiento por inserción, ordenamiento topológico) 2) Por un Factor Constante (por ejemplo, búsqueda binaria) 3) Por un Tamaño Variable (por ejemplo, Euclides) *c*) Dividir y Vencer (por ejemplo, búsqueda binaria, quicksort, mergesort, Strassen) *d*) Voraz (por ejemplo, Dijkstra, Kruskal, Mochila) *e*) Transformar y Vencer: 1) Simplificación de instancia (por ejemplo, encontrar duplicados mediante preordenamiento de lista) 2) Cambio de representación (por ejemplo, heapsort) 3) Reducción de problema (por ejemplo, mínimo común múltiplo, programación lineal) 4) Programación dinámica (por ejemplo, Floyd, Marshall, Bellman-Ford) *f*) Compromisos espacio vs tiempo (por ejemplo, hash) **II:** Manejo del crecimiento exponencial (por ejemplo, heurística A*, ramificación y poda, retroceso) **III:** Iteración vs recursión (por ejemplo, factorial, búsqueda en árbol) **IV:** Paradigmas: *a*) Algoritmos de aproximación *b*) Mejora iterativa (por ejemplo, Ford-Fulkerson, simplex) *c*) Algoritmos aleatorizados/estocásticos (por ejemplo, corte máximo, bolas y cubos) **V:** Computación cuántica

Objetivos de Aprendizaje I: Para cada uno de los paradigmas en esta unidad: *a*) Explicar sus características definitorias *b*) Explicar un ejemplo que demuestre el paradigma incluyendo cómo este ejemplo satisface las características del paradigma. [Explicar] **II:** Para cada uno de los algoritmos en la unidad Estructuras de Datos Fundamentales, Algoritmos Fundamentales, Algoritmos Avanzados, explicar el paradigma utilizado por el algoritmo y cómo ejemplifica este paradigma [Explicar] **III:** Dado un algoritmo, explicar el paradigma utilizado por el algoritmo

y cómo ejemplifica este paradigma [Explicar] **IV:** Dar un problema del mundo real, evaluar paradigmas algorítmicos apropiados y algoritmos de estos paradigmas que aborden el problema incluyendo evaluar los compromisos entre los paradigmas y algoritmos seleccionados [Evaluar] **V:** Dar ejemplos de algoritmos iterativos y recursivos que resuelvan el mismo problema, explicar los beneficios y desventajas de cada enfoque [Explicar] **VI:** Evaluar si un enfoque voraz conduce a una solución óptima [Evaluar] **VII:** Explicar varios enfoques para abordar problemas computacionales cuyas soluciones algorítmicas son exponenciales [Explicar]

Bibliografía: [Cormen et al., 2022, Dasgupta et al., 2006]

5.5 Marco de Análisis de Complejidad, 6 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Marco de Análisis de Complejidad: *a*) Rendimiento de un algoritmo en el mejor caso, caso promedio y peor caso *b*) Mediciones empíricas y relativas (Orden de Crecimiento) *c*) Tamaño de entrada y operaciones primitivas *d*) Eficiencia de tiempo y espacio **II:** Mediciones empíricas de rendimiento **III:** Compromisos tiempo-espacio en algoritmos

Objetivos de Aprendizaje I: Para cada algoritmo en la unidad Estructuras de Datos Fundamentales, Algoritmos Fundamentales, Algoritmos Avanzados, explicar su clase de complejidad de tiempo de ejecución y por qué pertenece a esta clase [Explicar] **II:** Desarrollar estudios empíricos para determinar y validar hipótesis sobre la complejidad de tiempo de ejecución de varios algoritmos ejecutando algoritmos con entradas de varios tamaños y comparando el rendimiento real con el análisis teórico [Crear] **III:** Explicar ejemplos que ilustren los compromisos tiempo-espacio de algoritmos [Explicar] **IV:** Explicar cómo el balance del árbol afecta la eficiencia de las operaciones de árbol de búsqueda binaria [Explicar]

Bibliografía: [Cormen et al., 2022, Sipser, 2012]

5.6 Notación Asintótica y Clases de Complejidad, 6 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Análisis de complejidad asintótica (cotas promedio y peor caso): *a*) Notaciones formales Big-O, Big-Omega y Big-Theta *b*) Clases de Complejidad Fundamentales y Ejemplos/Problemas Representativos: 1) $O(1)$ Constante (por ejemplo, acceso a arreglo) 2) $O(\log_2 n)$ Logarítmica (por ejemplo, búsqueda binaria) 3) $O(n)$ Lineal (por ejemplo, búsqueda lineal) 4) $O(n \log_2 n)$ Log Lineal (por ejemplo, mergesort) 5) $O(n^2)$ Cuadrática (por ejemplo, ordenamiento por selección) 6) $O(n^c)$ Polinomial (por ejemplo, $O(n^3)$ eliminación gaussiana) 7) $O(2^n)$ Exponencial (por ejemplo, Mochila, Satisfactibilidad (SAT), Viajante de Comercio (TSP), todos los subconjuntos) 8) $O(n!)$ Factorial (por ejemplo, circuito hamiltoniano, todas las permutaciones)

Objetivos de Aprendizaje I: Para cada clase de complejidad fundamental en esta unidad, explicar un algoritmo que demuestre la complejidad de tiempo de ejecución asociada [Explicar] **II:** Aplicar la notación Big-O para dar cotas superiores en la complejidad tiempo/espacio de algoritmos [Aplicar]

Bibliografía: [Cormen et al., 2022, Sipser, 2012]

5.7 Análisis de Complejidad: Recursión, Amortización y Cotas Ajustadas, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Notaciones Little-o, Little-Omega y Little Theta **II:** Análisis recursivo formal **III:** Análisis amortizado

Objetivos de Aprendizaje I: Dado un problema para programar para el cual puede haber varios enfoques algorítmicos, evaluarlos y determinar cuáles son factibles, y seleccionar uno que sea óptimo en implementación y comportamiento de tiempo de ejecución [Evaluar] **II:** Usar relaciones de recurrencia para evaluar la complejidad de tiempo de algoritmos definidos recursivamente [Aplicar] **III:** Aplicar relaciones de recurrencia elementales usando una forma del Teorema Maestro [Aplicar]

Bibliografía: [Cormen et al., 2022, Sipser, 2012]

5.8 Teoría de la Complejidad Computacional, 6 hr(s)

Competencias:

Tópicos: I: Tractabilidad e intractabilidad: *a*) Clases de Complejidad P, NP y NP-Completo *b*) Problemas NP-Completos (por ejemplo, SAT, Mochila, TSP) *c*) Reducciones

Objetivos de Aprendizaje I: Explicar la importancia de la NP-Completitud [Explicar] **II:** Explicar ejemplos de problemas NP-completos [Explicar] **III:** Explicar las clases P y NP [Explicar]

Bibliografía: [Sipser, 2012, Cormen et al., 2022]

6. Bibliografía

[Cormen et al., 2022] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2022). *Introduction to Algorithms*. MIT Press, 4th edition.

[Dasgupta et al., 2006] Dasgupta, S., Papadimitriou, C., and Vazirani, U. (2006). *Algorithms*. McGraw-Hill Education.

[Kleinberg and Tardos, 2005] Kleinberg, J. and Tardos, É. (2005). *Algorithm Design*. Pearson.

[Sipser, 2012] Sipser, M. (2012). *Introduction to the Theory of Computation*. Cengage Learning, 3rd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS261 Inteligencia Artificial (Obligatorio)

1. Información general

- **Semestre**: 5to Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - STA251 Estadística y Probabilidades (3er Sem-Pág 324)

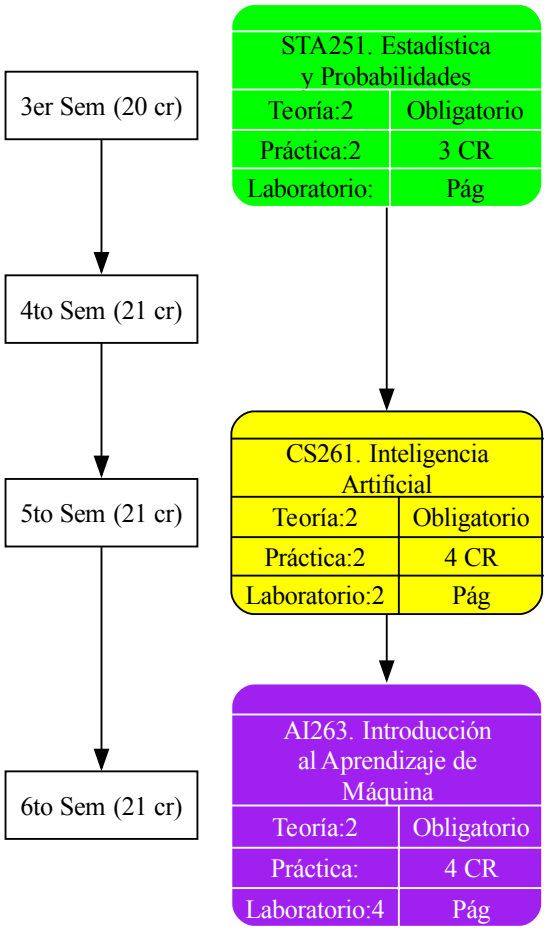

Figura 5.26: Mapa de Conexión. CS261 Inteligencia Artificial

2. Fundamentación

La Inteligencia Artificial (IA) es un campo fundamental de la ciencia de la computación que busca crear sistemas capaces de realizar tareas que normalmente requieren inteligencia humana. Este curso proporciona una introducción a las técnicas utilizadas para construir dichos sistemas, incluyendo algoritmos de búsqueda, razonamiento probabilístico y aprendizaje automático. Los estudiantes aprenderán a modelar problemas complejos e implementar agentes inteligentes, mientras consideran las implicaciones sociales y éticas de estas tecnologías.

3. Objetivos del curso.

- Comprender los conceptos fundamentales y la historia de la Inteligencia Artificial.
- Implementar y evaluar algoritmos de búsqueda para la resolución de problemas.
- Dominar los conceptos básicos del aprendizaje automático, incluyendo aprendizaje supervisado y no supervisado.
- Aplicar razonamiento probabilístico para manejar la incertidumbre en sistemas inteligentes.
- Analizar los desafíos éticos y el impacto social de la IA.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

5. Contenido

5.1 Introducción a la IA: Conceptos y Comportamiento Inteligente, 4 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: **I:** Visión general de los problemas de IA, Ejemplos de aplicaciones recientes exitosas de IA **II:** Definiciones de agentes con ejemplos (por ejemplo, reactivos, deliberativos) **III:** ¿Qué es el comportamiento inteligente?: *a*) La prueba de Turing y sus defectos *b*) Entrada y salida multimodal *c*) Simulación del comportamiento inteligente *d*) Razonamiento racional versus no racional **IV:** Visión general de las Aplicaciones de IA, crecimiento e impacto (económico, social, ético) **V:** Problemas filosóficos **VI:** Historia de la IA

Objetivos de Aprendizaje **I:** Describir la prueba de Turing y el experimento mental de la "Habitación China" [Explicar] **II:** Diferenciar entre razonamiento/comportamiento óptimo y razonamiento/comportamiento similar al humano [Evaluar] **III:** Diferenciar los términos: IA, aprendizaje automático y aprendizaje profundo [Explicar]

Bibliografía: [Russell and Norvig, 2020, Luger, 2008]

5.2 Características del Problema y Naturaleza de los Agentes, 4 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: **I:** Características del problema: *a*) Totalmente observable versus parcialmente observable *b*) Agente único versus multiagente *c*) Determinista versus estocástico *d*) Estático versus dinámico *e*) Discreto versus continuo **II:** Naturaleza de los agentes: *a*) Autonomía autónoma, semiautónoma, de iniciativa mixta *b*) Reflexivos, basados en metas y basados en utilidad *c*) Toma de decisiones bajo incertidumbre y con información incompleta *d*) La importancia de la percepción y las interacciones ambientales *e*) Agentes basados en aprendizaje *f*) Agentes corporeizados: 1) sensores, dinámicas, efectores **III:** Practicar la identificación de características del problema en entornos de ejemplo **IV:** Profundidad adicional sobre la naturaleza de los agentes con ejemplos **V:** Profundidad adicional sobre Aplicaciones de IA, Crecimiento e Impacto (económico, social, ético, seguridad)

Objetivos de Aprendizaje **I:** Enumerar las características de un problema específico [Aplicar]

Bibliografía: [Russell and Norvig, 2020, Luger, 2008]

5.3 Estado y Búsqueda No Informada, 5 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: **I:** Representación del espacio de estados de un problema: *a*) Especificación de estados, metas y operadores *b*) Factorización de estados en representaciones (espacios de hipótesis) *c*) Resolución de problemas mediante búsqueda en grafos: 1) por ejemplo, Grafos como un espacio y recorridos de árboles como exploración de ese espacio 2) Construcción dinámica del grafo (no dada de antemano) **II:** Búsqueda en grafo no informada para resolución de problemas: *a*) Búsqueda en amplitud *b*) Búsqueda en profundidad: 1) Con profundización iterativa *c*) Búsqueda de costo uniforme **III:** Complejidades de espacio y tiempo de los algoritmos de búsqueda en grafos **IV:** Búsqueda bidireccional **V:** Comprensión del espacio de búsqueda: *a*) Construcción de árboles de búsqueda *b*) Espacios de búsqueda dinámicos *c*) Explosión combinatoria del espacio de búsqueda *d*) Topología del espacio de búsqueda (por ejemplo, crestas, puntos de silla, mínimos locales)

Objetivos de Aprendizaje **I:** Diseñar la representación del espacio de estados para un rompecabezas (por ejemplo, N-reinas o problema de los 3 jarrones) [Aplicar] **II:** Seleccionar e implementar un algoritmo de búsqueda no informada apropiado para un problema (por ejemplo, tres en raya), y caracterizar sus complejidades de tiempo y espacio [Aplicar]

Bibliografía: [Russell and Norvig, 2020, Luger, 2008]

5.4 Búsqueda Heurística y CSP, 6 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: **I:** Búsqueda en grafo heurística para resolución de problemas: *a*) Construcción heurística y admisibilidad *b*) Ascenso de colinas *c*) Mínimos locales y el paisaje de búsqueda: 1) Soluciones locales versus globales *d*) Búsqueda primero el mejor voraz *e*) Búsqueda A* **II:** Búsqueda en haz **III:** Implementación de la búsqueda A* **IV:** Satisfacción de restricciones **V:** Variaciones de A* (IDA*, SMA*, RBFS)

Objetivos de Aprendizaje **I:** Seleccionar e implementar un algoritmo de búsqueda informada apropiado para un problema después de diseñar una función heurística útil (por ejemplo, un robot navegando en un mundo de cuadrícula 2D) [Aplicar] **II:** Evaluar si una heurística para un problema dado es admisible/puede garantizar una solución óptima [Evaluar] **III:** Diseñar e implementar búsqueda A*/en haz para resolver un problema, y compararla con otros algoritmos de búsqueda en términos del costo de la solución, número de nodos expandidos, etc [Aplicar] **IV:** Comparar y contrastar varias búsquedas heurísticas con respecto a la aplicabilidad a un problema dado [Evaluar] **V:** Modelar un rompecabezas lógico o Sudoku como un problema de satisfacción de restricciones, resolverlo con búsqueda con retroceso y determinar cuánto la consistencia de arcos puede reducir el espacio de búsqueda [Aplicar]

Bibliografía: [Russell and Norvig, 2020, Luger, 2008]

5.5 Búsqueda Adversarial y Optimización Estocástica, 5 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: **I:** Juegos adversariales de dos jugadores: *a*) Búsqueda minimax *b*) Poda alfa-beta: 1) Corte por profundidad **II:** Búsqueda local **III:** Búsqueda tabú **IV:** Juegos adversariales de dos jugadores: *a*) El efecto horizonte *b*) Libros de apertura/soluciones de finales *c*) Qué significa ”resolver” un juego (por ejemplo, damas) **V:** Implementación de búsqueda minimax, búsqueda en haz **VI:** Búsqueda expectimax (resolución MDP) y nodos de azar **VII:** Búsqueda estocástica: *a*) Recocido simulado *b*) Algoritmos genéticos *c*) Búsqueda en árbol Monte Carlo

Objetivos de Aprendizaje **I:** Aplicar búsqueda minimax en un juego adversarial de dos jugadores (por ejemplo, conecta 4), usando evaluación heurística en una profundidad particular para calcular las puntuaciones a propagar [Aplicar] **II:** Aplicar búsqueda minimax con poda alfa-beta para

podar el espacio de búsqueda en un juego adversarial de dos jugadores (por ejemplo, conecta 4) [Aplicar] **III:** Comparar y contrastar algoritmos genéticos con técnicas de búsqueda clásicas, explicando cuándo es más apropiado usar un algoritmo genético para aprender un modelo versus otras formas de optimización (por ejemplo, descenso de gradiente) [Evaluar] **IV:** Diseñar e implementar una solución de algoritmo genético para un problema [Aplicar] **V:** Diseñar e implementar un programa de recocido simulado para evitar mínimos locales en un problema [Aplicar]

Bibliografía: [Russell and Norvig, 2020, Luger, 2008]

5.6 Fundamentos de Aprendizaje Automático, 7 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Definición y ejemplos de una amplia variedad de tareas de aprendizaje automático: *a*) Aprendizaje supervisado: 1) Clasificación 2) Regresión *b*) Aprendizaje por refuerzo *c*) Aprendizaje no supervisado: 1) Agrupamiento **II:** Ideas fundamentales: *a*) Teorema de no hay almuerzo gratis: ningún aprendiz puede resolver todos los problemas; las decisiones de diseño de representación tienen consecuencias. *b*) Fuentes de error e indecidibilidad en el aprendizaje automático **III:** Un aprendizaje supervisado simple basado en estadísticas como regresión lineal o árboles de decisión: *a*) Enfocarse en cómo funcionan sin entrar en detalles matemáticos o de optimización; suficiente para entender y usar implementaciones existentes correctamente **IV:** El problema del sobreajuste/control de la complejidad de la solución (regularización, poda - solo intuición): *a*) El compromiso entre sesgo (subajuste) - varianza (sobreajuste)

Objetivos de Aprendizaje I: Describir las diferencias entre los tres estilos principales de aprendizaje (supervisado, por refuerzo y no supervisado) y determinar cuál es apropiado para un dominio de problema particular [Explicar] **II:** Diferenciar los términos: IA, aprendizaje automático y aprendizaje profundo [Evaluar] **III:** Formular una aplicación como un problema de clasificación, incluyendo las características de entrada disponibles y la salida a predecir (por ejemplo, identificar caracteres alfabéticos a partir de entrada de cuadrícula de píxeles) [Aplicar] **IV:** Identificar el sobreajuste en el contexto de un problema y curvas de aprendizaje y describir soluciones al sobreajuste [Evaluar] **V:** Explicar cómo funciona el aprendizaje automático como un proceso de optimización/búsqueda [Explicar]

Bibliografía: [Russell and Norvig, 2020, Alpaydin, 2020]

5.7 Aprendizaje Automático: Datos, Representaciones y Evaluación, 7 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Trabajar con Datos: *a*) Preprocesamiento de datos: 1) Importancia y peligros de las elecciones de preprocesamiento *b*) Manejo de valores faltantes (imputación, marcar como faltante): 1) Implicaciones de imputar versus marcar como faltante *c*) Codificación de variables categóricas, codificación de datos de valor real *d*) Normalización/estandarización *e*) énfasis en datos reales, no ejemplos de libros de texto **II:** Representaciones: *a*) Espacios de hipótesis y complejidad *b*) Expansión de características de base simple, como elevar al cuadrado características univariadas *c*) Representaciones de características aprendidas **III:** Evaluación del aprendizaje automático: *a*) Separación de conjuntos de entrenamiento, validación y prueba *b*) Métricas de rendimiento para clasificadores *c*) Estimación del rendimiento en prueba en datos retenidos *d*) Ajuste de los parámetros de un modelo de aprendizaje automático con un conjunto de validación *e*) Importancia de entender qué está haciendo un modelo, dónde están sus peligros/limitaciones y las implicaciones de sus decisiones

Objetivos de Aprendizaje I: Aplicar dos o más algoritmos de aprendizaje estadístico simple a una tarea de clasificación y medir la precisión de los clasificadores [Aplicar] **II:** Implementar un algoritmo de aprendizaje estadístico y el proceso de optimización correspondiente para entrenar el clasificador y obtener una predicción en nuevos datos [Aplicar] **III:** Explicar los procedimientos

adecuados de evaluación de AA, incluyendo las diferencias entre rendimiento en entrenamiento y prueba, y qué puede salir mal con el proceso de evaluación llevando a informes inexactos del rendimiento de AA [Explicar] **IV:** Comparar dos algoritmos de aprendizaje automático en un conjunto de datos, implementando la metodología de preprocesamiento de datos y evaluación (por ejemplo, métricas y manejo de divisiones entrenamiento/prueba) desde cero [Aplicar]

Bibliografía: [Russell and Norvig, 2020, Alpaydin, 2020]

5.8 Redes Neuronales y Ética, 6 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Redes neuronales básicas: *a*) Fundamentos de comprensión de cómo funcionan las redes neuronales y su proceso de entrenamiento, sin detalles de los cálculos *b*) Introducción básica a redes neuronales generativas (por ejemplo, modelos de lenguaje grandes) **II:** ética para el Aprendizaje Automático: *a*) Enfocarse en datos reales, escenarios reales y estudios de caso *b*) Sesgo de conjunto de datos/algorítmico/de evaluación y consecuencias no deseadas

Objetivos de Aprendizaje I: Describir el proceso de entrenamiento de redes neuronales y las representaciones aprendidas resultantes [Explicar]

Bibliografía: [Russell and Norvig, 2020, Alpaydin, 2020]

5.9 Formulación del Aprendizaje Automático y Evaluación, 3 hr(s)

Competencias: AG-C09

Tópicos: I: Formulación del aprendizaje automático simple como un problema de optimización, como regresión lineal por mínimos cuadrados o regresión logística: *a*) Función objetivo *b*) Descenso de gradiente *c*) Regularización para evitar sobreajuste (formulación matemática) **II:** Conjuntos de modelos: *a*) Combinación de mayoría ponderada simple **III:** Evaluación de rendimiento: *a*) Otras métricas para clasificación (por ejemplo, error, precisión, exhaustividad) *b*) Métricas de rendimiento para regresores *c*) Matriz de confusión *d*) Validación cruzada: 1) Ajuste de parámetros (búsqueda en cuadrícula/aleatoria, mediante validación cruzada)

Objetivos de Aprendizaje I: Comparar y contrastar varias técnicas de aprendizaje (por ejemplo, árboles de decisión, regresión logística, Bayes ingenuo, redes neuronales y redes de creencia), proporcionando ejemplos de cuándo cada estrategia es superior [Comparar] **II:** Evaluar el rendimiento de un sistema de aprendizaje simple en un conjunto de datos del mundo real [Evaluar] **III:** Caracterizar el estado del arte en teoría del aprendizaje, incluyendo sus logros y limitaciones [Explicar] **IV:** Explicar el problema del sobreajuste, junto con técnicas para detectar y manejar el problema [Explicar] **V:** Explicar el triple compromiso entre el tamaño de un espacio de hipótesis, el tamaño del conjunto de entrenamiento y la precisión del rendimiento [Explicar]

Bibliografía: [Goodfellow et al., 2016]

5.10 Aprendizaje Profundo, RL y Ética, 3 hr(s)

Competencias: AG-C09

Tópicos: I: Aprendizaje profundo: *a*) Redes feed-forward profundas (solo intuición, sin matemáticas) *b*) Redes neuronales convolucionales (solo intuición, sin matemáticas) *c*) Visualización de representaciones de características aprendidas de redes profundas *d*) Otras arquitecturas (NN generativas, NN recurrentes, transformers, etc.) **II:** Visión general de los métodos de aprendizaje por refuerzo **III:** Dos o más aplicaciones de algoritmos de aprendizaje automático: *a*) Por ejemplo, medicina y salud, economía, visión, lenguaje natural, robótica, juego **IV:** ética para el Aprendizaje Automático: *a*) Continuar enfocándose en datos reales, escenarios reales y estudios de caso *b*) Privacidad *c*) Equidad *d*) Propiedad intelectual *e*) Explicabilidad

Objetivos de Aprendizaje I: Visualizar el progreso del entrenamiento de una red neuronal a través de curvas de aprendizaje en un kit de herramientas establecido (por ejemplo, TensorBoard) y visualizar las características aprendidas de la red [Aplicar] **II:** Dada una aplicación real de aprendizaje automático, describir problemas éticos respecto a las elecciones de datos, pasos de preprocesamiento, selección de algoritmo y visualización/presentación de resultados [Aplicar]

Bibliografía: [Goodfellow et al., 2016]

5.11 Representación del Conocimiento y Razonamiento Probabilístico Básico, 6 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Tipos de representaciones: *a*) Simbólicas, lógicas: 1) Crear una representación a partir de un enunciado de problema en lenguaje natural *b*) Representaciones subsimbólicas aprendidas *c*) Modelos gráficos (por ejemplo, Bayes ingenuo, red bayesiana) **II:** Repaso del razonamiento probabilístico, teorema de Bayes **III:** Razonamiento bayesiano: *a*) Inferencia bayesiana **IV:** Variables aleatorias y distribuciones de probabilidad: *a*) Axiomas de probabilidad *b*) Inferencia probabilística *c*) Regla de Bayes (derivación) *d*) Inferencia bayesiana (ejemplos más complejos)

Objetivos de Aprendizaje I: Dado un enunciado de problema en lenguaje natural, codificarlo como una representación simbólica o lógica [Aplicar] **II:** Calcular una inferencia probabilística en un problema del mundo real usando el teorema de Bayes para determinar la probabilidad de una hipótesis dada la evidencia [Aplicar] **III:** Aplicar la regla de Bayes para determinar la probabilidad de una hipótesis dada la evidencia [Aplicar] **IV:** Calcular la probabilidad de resultados y probar si los resultados son independientes [Aplicar]

Bibliografía: [Russell and Norvig, 2020]

5.12 Independencia, Modelos de Markov y Toma de Decisiones, 6 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Independencia **II:** Independencia condicional **III:** Cadenas de Markov y modelos de Markov **IV:** Utilidad y toma de decisiones

Objetivos de Aprendizaje I: Explicar cómo podemos tomar decisiones bajo incertidumbre, usando conceptos como el teorema de Bayes y la utilidad [Explicar]

Bibliografía: [Russell and Norvig, 2020]

5.13 Aplicaciones e Impacto Social de la IA, 4 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Al menos una aplicación de IA a un problema y campo específico, como medicina, salud, sostenibilidad, redes sociales, economía, educación, robótica, etc. (elegir al menos uno para el Núcleo CS): *a*) Formular y evaluar una aplicación específica como un problema de IA: 1) Cómo lidiar con problemas mal especificados o mal planteados *b*) Disponibilidad/escasez y limpieza de datos: 1) Limpieza y preprocesamiento básico de datos 2) Sesgo del conjunto de datos *c*) Sesgo algorítmico *d*) Sesgo de evaluación *e*) Evaluación de las implicaciones sociales de la aplicación **II:** Modelos generativos profundos desplegados: *a*) Visión general de alto nivel de modelos generativos de imágenes profundas (por ejemplo, a 2023, DALL-E, Midjourney, Stable Diffusion, etc.), sus usos y sus limitaciones/peligros. *b*) Visión general de alto nivel de modelos de lenguaje grandes (por ejemplo, a 2023, ChatGPT, Bard, etc.), sus usos y sus limitaciones/peligros. **III:** Visión general del impacto social de la IA: *a*) ética *b*) Equidad *c*) Confianza/explicabilidad *d*) Privacidad y uso de datos de entrenamiento *e*) Autonomía humana y supervisión/regulaciones/requisitos legales *f*) Sostenibilidad

Objetivos de Aprendizaje I: Dado un dominio de aplicación y problema del mundo real, formular una solución de IA para ello, identificando datos/entrada apropiados, preprocesamiento, representaciones, técnicas de IA y métricas/metodología de evaluación [Aplicar]

Bibliografía: [Russell and Norvig, 2020, Alpaydin, 2020]

5.14 Aplicaciones Adicionales e Impacto Social de la IA, 4 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Una o más aplicaciones adicionales de IA a un amplio conjunto de problemas y diversos campos, como medicina, salud, sostenibilidad, redes sociales, economía, educación, robótica, etc. (elegir un área diferente de la elegida para el Núcleo CS): *a*) Formular y evaluar una aplicación específica como un problema de IA: 1) Cómo lidiar con problemas mal especificados o mal planteados *b*) Disponibilidad/escasez y limpieza de datos: 1) Limpieza y preprocesamiento básico de datos 2) Sesgo del conjunto de datos *c*) Sesgo algorítmico *d*) Sesgo de evaluación *e*) Evaluación de las implicaciones sociales de la aplicación **II:** Profundidad adicional sobre modelos generativos profundos desplegados: *a*) Introducción a cómo funcionan los modelos generativos de imágenes profundas (por ejemplo, a 2023, DALL-E, Midjourney, Stable Diffusion) incluyendo discusión de atención *b*) Introducción a cómo funcionan los modelos de lenguaje grandes (por ejemplo, a 2023, ChatGPT, Bard) incluyendo discusión de atención *c*) Idea de modelos fundamentales, cómo usarlos, y los beneficios/problemas con entrenarlos a partir de grandes datos **III:** Análisis y discusión del impacto social de la IA: *a*) ética *b*) Equidad *c*) Confianza/explicabilidad *d*) Privacidad y uso de datos de entrenamiento *e*) Autonomía humana y supervisión/regulaciones/requisitos legales *f*) Sostenibilidad

Objetivos de Aprendizaje I: Analizar el impacto social de una o más aplicaciones específicas de IA del mundo real, identificando problemas respecto a ética, equidad, sesgo, confianza y explicabilidad [Evaluar] **II:** Describir algunos de los modos de falla de los modelos generativos profundos actuales para lenguaje o imágenes, y cómo esto podría afectar su uso en una aplicación [Describir]

Bibliografía: [Russell and Norvig, 2020, Alpaydin, 2020]

6. Bibliografía

[Alpaydin, 2020] Alpaydin, E. (2020). *Introduction to Machine Learning*. MIT Press, 4th edition.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Luger, 2008] Luger, G. F. (2008). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Pearson, 6th edition.

[Russell and Norvig, 2020] Russell, S. and Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación
CS272 Bases de Datos II (Obligatorio)

1. Información general

- **Semestre**: 5to Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 1 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS271 Bases de Datos I (4to Sem-Pág 348)

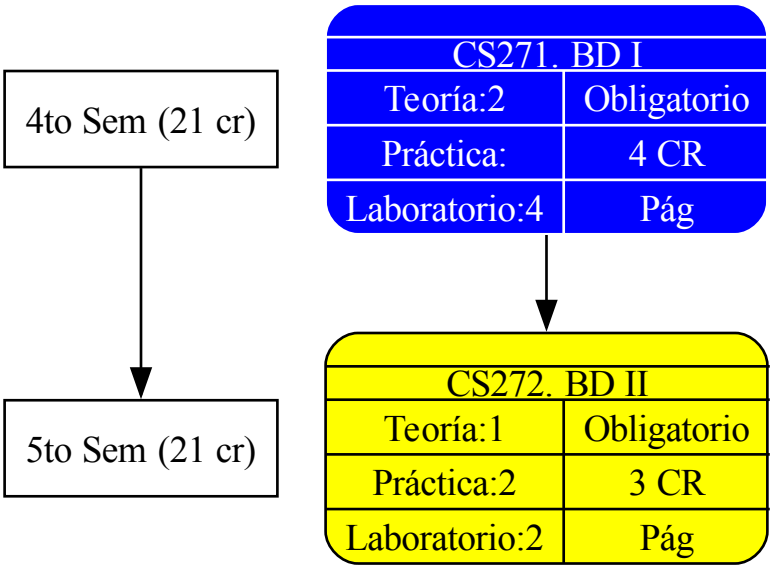

Figura 5.27: Mapa de Conexión. CS272 Bases de Datos II

2. Fundamentación

Este curso es una continuación de Gestión de Datos I, enfocándose en los mecanismos internos de los sistemas de gestión de bases de datos (DBMS) y tecnologías de bases de datos modernas. Cubre en profundidad la optimización de consultas, estructuras de indexación e integridad transaccional. Además, introduce a los estudiantes a modelos no relacionales (NoSQL) y arquitecturas distribuidas, que son esenciales para manejar datos a gran escala en entornos de nube contemporáneos.

3. Objetivos del curso.

- Comprender las estructuras de almacenamiento interno y técnicas de indexación.
- Analizar y optimizar planes de ejecución de consultas.
- Comprender protocolos de transacción avanzados y mecanismos de recuperación.
- Evaluar e implementar soluciones usando modelos de bases de datos NoSQL.
- Comprender los principios de bases de datos distribuidas y paralelas.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Procesamiento de Consultas, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Estructuras de página **II:** Estructuras de índice: *a*) árboles B+ *b*) Índices hash: estáticos y dinámicos *c*) Creación de índices en SQL **III:** Estructuras de archivos: *a*) Archivos de montón (*heap*) *b*) Archivos hash **IV:** Algoritmos para operadores de consulta: *a*) Ordenamiento externo *b*) Selección *c*) Proyección; con y sin eliminación de duplicados *d*) Uniones naturales: Bucle anidado, Mezcla-ordenada, Unión hash *e*) Análisis de eficiencia de algoritmos **V:** Transformaciones de consultas **VI:** Optimización de consultas: *a*) Rutas de acceso *b*) Construcción de plan de consulta *c*) Estimación de selectividad *d*) Planes solo de índice **VII:** Procesamiento de Consultas en Paralelo (por ejemplo, exploración paralela, unión paralela, agregación paralela) **VIII:** Afinamiento/rendimiento de base de datos: *a*) Selección de índices *b*) Impacto de índices en el rendimiento de consultas *c*) Desnormalización

Objetivos de Aprendizaje I: Describir el propósito y organización de las estructuras de índice tanto de árbol B+ como hash [Describir] **II:** Componer un comando SQL para crear un índice (de cualquier tipo) [Componer] **III:** Especificar los pasos para los diversos algoritmos de operadores de consulta: ordenamiento externo, proyección con eliminación de duplicados, unión por mezcla-ordenada, unión hash, unión por bucle anidado por bloques [Analizar] **IV:** Derivar el tiempo de ejecución (en solicitudes de E/S) para cada uno de los algoritmos anteriores [Analizar] **V:** Transformar una consulta en álgebra relacional a su equivalente apropiado para una ejecución canalizada de profundidad izquierda [Rediseñar] **VI:** Calcular estimaciones de selectividad para una operación de selección y/o unión dada [Computar/Calcular] **VII:** Describir cómo modificar una estructura de índice para facilitar una operación solo de índice para una relación dada [Describir] **VIII:** Para un escenario dado decidir qué índices soportar para la ejecución eficiente de un conjunto de consultas [Evaluar (valorar)] **IX:** Describir cómo los DBMS aprovechan el paralelismo para acelerar el procesamiento de consultas dividiendo el trabajo entre múltiples procesadores o nodos [Describir]

Bibliografía: [Silberschatz et al., 2019, Ramakrishnan and Gehrke, 2003]

5.2 Internos de DBMS, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Gestión de Búfer de DB **II:** Gestión de Transacciones: *a*) Niveles de Aislamiento *b*) ACID *c*) Serializabilidad *d*) Transacciones Distribuidas **III:** Control de Concurrencia: *a*) Bloqueo en 2 Fases *b*) Estrategias de manejo de bloqueos mutuos *c*) Modelos de consistencia basados en quórum **IV:** Gestor de Recuperación: *a*) Relación con el Gestor de Búfer **V:** Control de Concurrencia: *a*) Control de concurrencia optimista *b*) Control de concurrencia por marca de tiempo **VI:** Gestor de Recuperación: *a*) Registro por escritura anticipada *b*) Sistema de recuperación ARIES (Análisis, REDO, UNDO)

Objetivos de Aprendizaje I: Describir cómo un DBMS gestiona su Grupo de Búferes [Describir] **II:** Describir las cuatro propiedades para un gestor de transacciones correcto [Describir] **III:** Esbozar el principio de serializabilidad [Esquematizar/Esbozar]

Bibliografía: [Silberschatz et al., 2019, Garcia-Molina et al., 2008]

5.3 Bases de Datos Distribuidas/Computación en la Nube, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: DBMS Distribuido: *a*) Almacenamiento de datos distribuido *b*) Procesamiento de consultas distribuido *c*) Modelo de transacción distribuida *d*) Soluciones homogéneas y heterogéneas *e*) Bases de datos distribuidas cliente-servidor **II:** DBMS Paralelo: *a*) Arquitecturas de DBMS paralelo: memoria compartida, disco compartido, nada compartido *b*) Aceleración y escalamiento, por ejemplo, uso del modelo de procesamiento MapReduce *c*) Replicación de datos y modelos de consistencia débil

Objetivos de Aprendizaje I: Describir los componentes clave de un DBMS distribuido, incluyendo almacenamiento de datos distribuido, procesamiento de consultas y gestión de transacciones [Describir] **II:** Analizar las ventajas y desventajas entre arquitecturas de DBMS paralelo: memoria compartida, disco compartido y nada compartido [Analizar] **III:** Describir estrategias de replicación de datos y modelos de consistencia débil en sistemas de bases de datos distribuidas [Describir]

Bibliografía: [Silberschatz et al., 2019, Özsu and Valduriez, 2020]

5.4 Sistemas NoSQL, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: ¿Por qué NoSQL? (por ejemplo, Desajuste de impedancia entre Aplicación [CRUD] y RDBMS) **II:** Modelo de datos Clave-Valor y Documento **III:** Sistemas de almacenamiento (por ejemplo, sistemas Clave-Valor, Lagos de Datos (*Data Lakes*)) **IV:** Modelos de Distribución (Fragmentación y Replicación) **V:** Bases de Datos de Grafos **VI:** Modelos de Consistencia (Actualización y Lectura, consistencia de quórum, teorema CAP) **VII:** Modelo de procesamiento (por ejemplo, Map-Reduce, map-reduce multi-etapa, map-reduce incremental) **VIII:** Estudios de Caso: Sistema de almacenamiento en la nube (por ejemplo, S3); Bases de datos de grafos; "Cuándo no usar NoSQL"

Objetivos de Aprendizaje I: Desarrollar un caso de uso para el uso de NoSQL sobre RDBMS [Crear] **II:** Describir las características definitorias detrás de los modelos de datos basados en Clave-Valor y Documentos [Describir]

Bibliografía: [Sadalage and Fowler, 2012, Kleppmann, 2017]

5.5 Bases de Datos Semiestructuradas y No Estructuradas, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Datos no estructurados vectorizados (texto, video, audio, etc.) y almacenamiento vectorial: *a*) Vectorizador TF-IDF con n-grama *b*) Word2Vec *c*) Base de datos de arreglos o manejo de tipo de datos arreglo **II:** Bases de datos semiestructuradas (por ejemplo, JSON): *a*) Almacenamiento: 1) Codificación y compresión de tipos de datos anidados *b*) Indexación: 1) árbol B, índice de salto, filtro Bloom 2) Índice invertido y compresión de mapa de bits 3) Indexación por curva de llenado de espacio para datos geo-semiestructurados *c*) Procesamiento de consultas para casos de uso OLTP y OLAP: 1) Compromisos de inserción, selección, actualización/eliminación 2) Estudios de caso sobre Postgres/JSON, MongoDB y Snowflake/JSON

Objetivos de Aprendizaje I: Describir representaciones vectorizadas de datos no estructurados (por ejemplo, TF-IDF, Word2Vec) y sus modelos de almacenamiento [Describir] **II:** Aplicar técnicas de indexación (por ejemplo, árbol B, índice invertido, filtro Bloom) a datos semiestructurados almacenados en formato JSON [Aplicar] **III:** Analizar las ventajas y desventajas del procesamiento de consultas en bases de datos semiestructuradas para casos de uso OLTP y OLAP [Analizar]

Bibliografía: [Kleppmann, 2017, Manning et al., 2008, Mikolov et al., 2013, Agrawal et al., 2023, MongoDB, Inc., 2025]

6. Bibliografía

[Agrawal et al., 2023] Agrawal, D. et al. (2023). Toward a systems architecture for ai-enabled applications. *Communications of the ACM*, 66(6):54–63.

[Garcia-Molina et al., 2008] Garcia-Molina, H., Ullman, J. D., and Widom, J. (2008). *Database Systems: The Complete Book*. Pearson, 2nd edition.

[Kleppmann, 2017] Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint*. arXiv:1301.3781.

[MongoDB, Inc., 2025] MongoDB, Inc. (2025). Mongodb manual. MongoDB, Inc. `https://www.mongodb.com/docs/`.

[Özsu and Valduriez, 2020] Özsu, M. T. and Valduriez, P. (2020). *Principles of Distributed Database Systems*. Springer, 4th edition.

[Ramakrishnan and Gehrke, 2003] Ramakrishnan, R. and Gehrke, J. (2003). *Database Management Systems*. McGraw-Hill, 3rd edition.

[Sadalage and Fowler, 2012] Sadalage, P. J. and Fowler, M. (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley.

[Silberschatz et al., 2019] Silberschatz, A., Korth, H. F., and Sudarshan, S. (2019). *Database System Concepts*. McGraw-Hill, 7th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS291 Ingeniería de Software I (Obligatorio)

1. Información general

- **Semestre**: 5to Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS113 Programación Orientada a Objetos II (3er Sem-Pág 306)
  - CS271 Bases de Datos I (4to Sem-Pág 348)

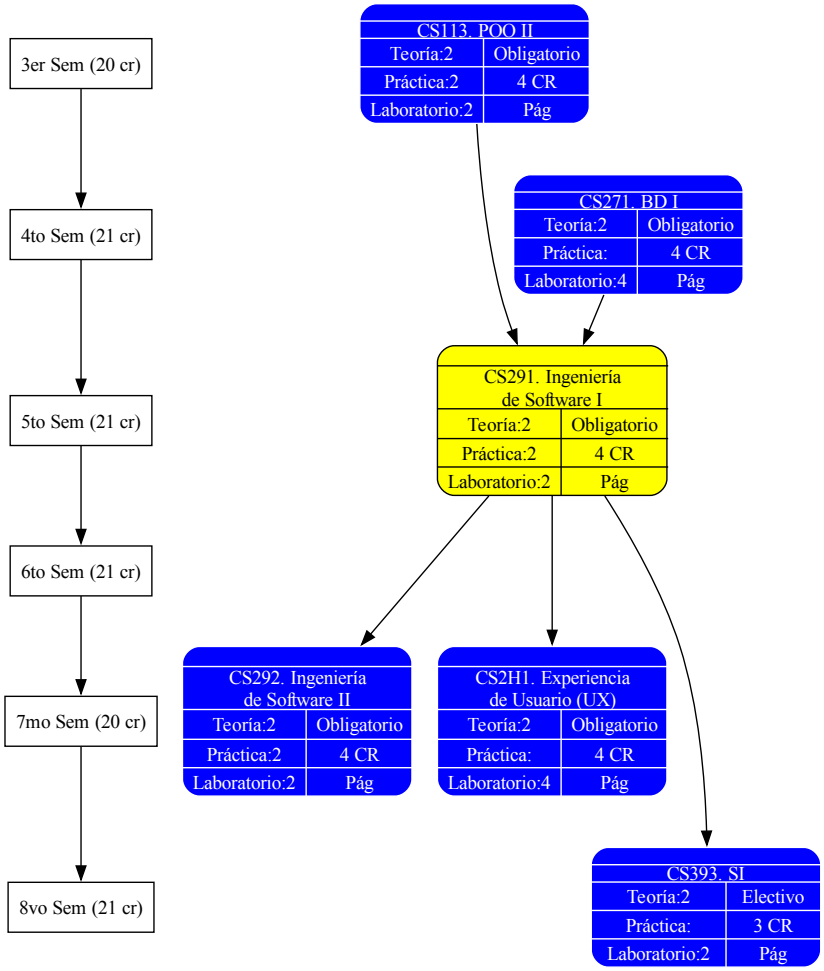

Figura 5.28: Mapa de Conexión. CS291 Ingeniería de Software I

2. Fundamentación

La ingeniería de software es una disciplina colaborativa que aplica procesos estructurados para construir sistemas de alta calidad. Este primer curso se enfoca en los aspectos fundamentales del ciclo de vida del software, desde los requisitos iniciales y el diseño centrado en el usuario hasta las pruebas sistemáticas. Al integrar principios de HCI y fundamentos de desarrollo rigurosos, los estudiantes aprenderán a crear software que sea técnicamente sólido y orientado al usuario.

3. Objetivos del curso.

- Comprender y aplicar varios modelos de proceso de software y gestión de configuración.
- Dominar la elicitación y modelado de requisitos usando notaciones estandarizadas.
- Diseñar interfaces de usuario intuitivas basadas en principios de evaluación y diseño de HCI.
- Implementar estrategias sistemáticas de prueba y comprender la calidad del software.
- Desarrollar habilidades profesionales para el desarrollo colaborativo de software.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

5. Contenido

5.1 Control de Versiones y CI/CD, 12 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Gestión de configuración de software y control de versiones: Prácticas de Desarrollo de Software *a*) Configuración en control de versiones, builds/configuración reproducibles. *b*) Estrategias de ramificación en control de versiones. Ramas de desarrollo vs ramas de lanzamiento. Desarrollo basado en tronco (*trunk-based development*). *c*) Estrategias de fusión/rebase, cuando sean relevantes. **II:** Gestión de lanzamientos (*release management*). **III:** Herramientas de prueba, incluyendo herramientas de análisis estático y dinámico. Prácticas de Desarrollo de Software, Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada **IV:** Automatización de procesos de software: *a*) Sistemas de construcción (*build systems*): el valor de builds rápidos, herméticos y reproducibles, comparar/contrastar enfoques para construir un proyecto. *b*) Integración Continua (CI): el uso de automatización y pruebas automatizadas para hacer una validación preliminar de que la revisión actual del tronco se construye y pasa las pruebas (básicas). *c*) Despliegue Continuo (CD): el uso de automatización para liberar de forma automática cada cambio que supera las pruebas hacia el entorno de producción, garantizando entregas frecuentes y confiables. *d*) Gestión de dependencias: actualización de dependencias externas/aguas arriba, gestión de paquetes, SemVer. **V:** Flujo de revisión colaborativa como puerta de calidad *a*) Revisión obligatoria de *pull requests* antes de fusionar *b*) Propiedad del código mediante reglas de tipo CODEOWNERS *c*) Convenciones de mensajes de *commit* (*conventional commits*) *d*) Reglas de protección de ramas que exigen verificaciones antes de fusionar **VI:** Escaneo de vulnerabilidades de dependencias y de la cadena de suministro integrado en los pipelines de CI

Objetivos de Aprendizaje **I:** Describir la diferencia entre la gestión de configuración de software centralizada y distribuida [Describir] **II:** Describir cómo el control de versiones puede usarse para ayudar en la gestión de lanzamientos de software [Describir] **III:** Identificar elementos de configuración y usar una herramienta de control de código fuente en un proyecto pequeño basado en equipo [Analizar] **IV:** Describir cómo las herramientas de prueba estáticas y dinámicas disponibles pueden integrarse en el entorno de desarrollo de software [Describir] **V:** Comprender el uso de sistemas de CI/CD como una fuente de verdad para el estado del código compartido del equipo (éxito en la construcción y pruebas) [Explicar] **VI:** Aplicar un flujo de revisión de *pull requests* con reglas de propiedad de código y protección de ramas para imponer puertas de calidad en un proyecto de equipo [Aplicar] **VII:** Usar un escáner de vulnerabilidades de dependencias/cadena de suministro como parte de un pipeline de CI para detectar dependencias vulnerables [Usar]

Bibliografía: [Sommerville, 2015, Pressman and Maxim, 2019]

5.2 Ingeniería de Requisitos, 14 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Describir requisitos funcionales usando, por ejemplo, casos de uso o historias de usuario. *a*) Usar al menos un método para documentar y estructurar requisitos funcionales. *b*) Comprender cómo el método apoya el diseño y la implementación. *c*) Fortalezas y debilidades de usar un enfoque específico. **II:** Propiedades de los requisitos, incluyendo consistencia, validez, completitud y viabilidad. **III:** Obtención (*elicitation*) de requisitos. *a*) Fuentes de requisitos, por ejemplo, usuarios, administradores o personal de soporte. *b*) Métodos de recopilación de requisitos, por ejemplo, encuestas, entrevistas o análisis de comportamiento. **IV:** Requisitos no funcionales, por ejemplo, seguridad, usabilidad o rendimiento, también llamados Atributos de Calidad. Diseño Sostenible y Computación Ubicua, Huella Ambiental de los Sistemas Informáticos, Efectos Sistémicos y Contexto Social **V:** Identificación y gestión de riesgos, incluyendo consideraciones éticas en torno al producto propuesto. Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo **VI:** Comunicar y/o formalizar especificaciones de requisitos.

Objetivos de Aprendizaje **I:** Comparar diferentes métodos de obtención de requisitos a lo largo de múltiples ejes [Comparar] **II:** Identificar diferencias entre dos métodos de describir requisitos funcionales (por ejemplo, entrevistas con clientes, estudios de usuarios) y las situaciones donde se preferiría cada uno [Analizar] **III:** Identificar qué comportamientos son requeridos, permitidos o prohibidos a partir de un conjunto dado de requisitos y una lista de comportamientos candidatos [Analizar] **IV:** Recopilar un conjunto de requisitos para un sistema de software simple [Analizar] **V:** Identificar áreas de un sistema de software que deben cambiarse, dada una descripción del sistema y un conjunto de nuevos requisitos a implementar [Analizar] **VI:** Identificar los requisitos funcionales y no funcionales en un conjunto de requisitos [Analizar]

Bibliografía: [Sommerville, 2015, Wiegers and Beatty, 2013]

5.3 Colaboración y Dinámica de Equipo, 14 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Comunicación efectiva, incluida la oral y escrita, así como formal (correo electrónico, documentos, comentarios, presentaciones) e informal (chat de equipo, reuniones). Comunicación Técnica y Profesional, Comunicación en Equipo y Cultura **II:** Causas comunes de conflicto en el equipo y enfoques para la resolución de conflictos. **III:** Programación cooperativa: *a*) Programación por parejas (pair programming) o Enjambre (swarming) *b*) Revisión de código *c*) Colaboración mediante control de versiones **IV:** Roles y responsabilidades en un equipo de software: Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo *a*) Ventajas del trabajo en equipo *b*) Riesgos y complejidad de dicha colaboración **V:** Procesos del equipo: responsabilidades para las tareas, estimación de esfuerzo, estructura de reuniones, cronograma de trabajo

Objetivos de Aprendizaje **I:** Seguir prácticas efectivas de comunicación en equipo [Analizar] **II:** Articular las fuentes, peligros y beneficios potenciales del conflicto en el equipo, especialmente enfocándose en el valor de estar en desacuerdo sobre ideas o propuestas sin insultar a las personas [Articular] **III:** Facilitar una estrategia de resolución de conflictos y resolución de problemas en un entorno de equipo [Analizar] **IV:** Colaborar efectivamente en desarrollo/programación cooperativa [Analizar] **V:** Proponer y delegar los roles y responsabilidades necesarios en un equipo de desarrollo de software [Proponer] **VI:** Redactar y seguir una agenda para una reunión de equipo [Componer] **VII:** Facilitar, a través de la participación en un proyecto de equipo, los elementos centrales de la formación de equipos, el establecimiento de una cultura de equipo saludable y la gestión de equipos, incluida la creación y ejecución de un plan de trabajo del equipo [Crear]

Bibliografía: [Wigdor and Wixon, 2011, Johnson, 2013]

5.4 Principios y Arquitectura de Software, 12 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Principios de diseño de sistemas. Confiabilidad del Sistema *a*) Niveles de abstracción (por ejemplo, diseño arquitectónico y diseño detallado) *b*) Separación de preocupaciones (separation of concerns) *c*) Ocultación de información (information hiding) *d*) Acoplamiento y cohesión (coupling and cohesion) **II:** Arquitectura de software. Confiabilidad del Sistema *a*) Paradigmas de diseño 1) Descomposición funcional de arriba hacia abajo/diseño por capas 2) Arquitectura orientada a datos 3) Análisis y diseño orientado a objetos 4) Diseño dirigido por eventos *b*) Arquitecturas estándar (por ejemplo, cliente-servidor y arquitecturas de microservicios incluyendo discusiones sobre REST, n-capas, tuberías y filtros, Modelo Vista Controlador) *c*) Identificar límites y dependencias de componentes **III:** Programación a gran escala vs programación a pequeña escala (programming in the large vs programming in the small). Confiabilidad del Sistema **IV:** Malos olores del código (code smells) y otras indicaciones de la calidad del código, distintas de la corrección. Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad

Objetivos de Aprendizaje **I:** Identificar la arquitectura de software estándar de un diseño de alto nivel dado [Analizar] **II:** Seleccionar y usar un paradigma de diseño apropiado para diseñar un sistema de software simple y explicar cómo se han aplicado los principios de diseño de sistemas en este diseño [Evaluar] **III:** Adaptar un diseño de sistema defectuoso para que siga mejor principios como la separación de preocupaciones o la ocultación de información [Crear] **IV:** Identificar las dependencias entre un conjunto de componentes de software en un diseño arquitectónico [Analizar]

Bibliografía: [Gamma et al., 1994, Pressman and Maxim, 2019]

5.5 Conceptos de verificación y validación, 12 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Conceptos de verificación y validación *a*) Verificación: ¿Estamos construyendo la cosa correctamente? *b*) Validación: ¿Construimos la cosa correcta? **II:** Por qué importan las pruebas: ¿El componente sigue siendo funcional a medida que el código evoluciona? **III:** Objetivos de las pruebas *a*) Usabilidad *b*) Confiabilidad *c*) Conformidad con la especificación *d*) Rendimiento *e*) Seguridad

Objetivos de Aprendizaje **I:** Explicar por qué las pruebas son importantes [Explicar] **II:** Distinguir entre validación y verificación de programas [Distinguir] **III:** Describir diferentes objetivos de las pruebas [Describir]

Bibliografía: [Sommerville, 2015, Pressman and Maxim, 2019]

5.6 Herramientas y Análisis de Pruebas, 4 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Herramientas y automatización de verificación y validación *a*) Análisis estático *b*) Cobertura de código *c*) Fuzzing *d*) Análisis dinámico y contención de fallos (sanitizadores, etc.) *e*) Registro y seguimiento de fallos **II:** Planificación y generación de pruebas *a*) Estimación de fallos y terminación de pruebas, incluyendo siembra de defectos *b*) Uso de números aleatorios y pseudoaleatorios en las pruebas **III:** Pruebas de sistemas asíncronos, paralelos y concurrentes **IV:** Verificación y validación de artefactos que no son código (documentación, materiales de capacitación) **V:** Desarrollo guiado por comportamiento (BDD) *a*) Expresar criterios de aceptación como escenarios Gherkin Given/When/Then *b*) Ejecutar escenarios BDD como pruebas automatizadas de extremo a extremo **VI:** Pruebas de mutación y análisis estático de calidad de

código *a*) Pruebas de mutación para evaluar la efectividad de una suite de pruebas *b*) Detección estática de *code smells* y complejidad ciclomática excesiva

Objetivos de Aprendizaje I: Describir y comparar diferentes herramientas para verificación y validación [Describir] **II:** Automatizar el uso de diferentes herramientas en un proyecto de software pequeño [Diseñar] **III:** Explicar cómo y cuándo se deben usar números aleatorios en las pruebas [Explicar] **IV:** Describir enfoques para la estimación de fallos [Describir] **V:** Estimar el número de fallos en una aplicación de software pequeña basándose en la densidad de fallos y la siembra de fallos [Estimar] **VI:** Describir técnicas y problemas con las pruebas de software asíncrono, concurrente y paralelo [Describir] **VII:** Crear un plan de pruebas para un segmento de código de tamaño mediano que contenga código asíncrono, concurrente y/o paralelo, incluyendo una matriz de pruebas y generación de datos y entradas de prueba [Crear] **VIII:** Describir técnicas para la verificación y validación de artefactos que no son código [Describir] **IX:** Escribir escenarios Gherkin Given/When/Then para los criterios de aceptación de una funcionalidad y ejecutarlos como pruebas automatizadas de extremo a extremo [Escribir] **X:** Analizar el puntaje de mutación de una suite de pruebas y refactorizar las pruebas para eliminar los mutantes sobrevivientes [Analizar]

Bibliografía: [Sommerville, 2015, Pressman and Maxim, 2019]

5.7 Pruebas de Rendimiento y Evaluación Comparativa, 4 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Pruebas de rendimiento y evaluación comparativa *a*) Rendimiento (*throughput*) y latencia *b*) Degradación bajo carga (pruebas de estrés, manejo FIFO vs LIFO de solicitudes) *c*) Aceleración (*speedup*) y escalado (*scaling*) 1) Ley de Amdahl 2) Ley de Gustafson 3) Escalado suave (*soft*) y débil (*weak*) *d*) Identificar y medir figuras de mérito (*figures of merit*) *e*) Cuellos de botella de rendimiento comunes 1) Limitado por cómputo (*compute-bound*) 2) Limitado por ancho de banda de memoria 3) Limitado por latencia *f*) Métodos estadísticos y mejores prácticas para la evaluación comparativa (benchmarking) 1) Estimación de la incertidumbre 2) Intervalos de confianza *g*) Análisis y presentación (gráficos, etc.) *h*) Técnicas de medición de tiempo (*timing*)

Objetivos de Aprendizaje I: Describir el rendimiento (*throughput*) y la latencia y proporcionar ejemplos de cada uno [Describir] **II:** Explicar la aceleración (speedup) y las diferentes formas de escalado (scaling) y cómo se calculan [Explicar] **III:** Describir cuellos de botella de rendimiento comunes [Describir] **IV:** Describir métodos estadísticos y mejores prácticas para la evaluación comparativa (benchmarking) de software [Describir] **V:** Explicar técnicas y desafíos para medir el tiempo al construir una evaluación comparativa (benchmark) [Explicar] **VI:** Identificar las figuras de mérito, construir y ejecutar una evaluación comparativa (benchmark) y analizar estadísticamente y visualizar los resultados para un proyecto de software pequeño [Analizar]

Bibliografía: [Bondi, 2015, Gregg, 2020]

6. Bibliografía

[Bondi, 2015] Bondi, A. B. (2015). *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice*. Addison-Wesley, Upper Saddle River, NJ.

[Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

[Gregg, 2020] Gregg, B. (2020). *Systems Performance: Enterprise and the Cloud*. Addison-Wesley Professional, Boston, MA, 2nd edition.

[Johnson, 2013] Johnson, J. (2013). *Designing with the Mind in Mind*. Morgan Kaufmann, 2nd edition.

[Pressman and Maxim, 2019] Pressman, R. S. and Maxim, B. (2019). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 9th edition.

[Sommerville, 2015] Sommerville, I. (2015). *Software Engineering*. Pearson, 10th edition.

[Wiegers and Beatty, 2013] Wiegers, K. and Beatty, J. (2013). *Software Requirements*. Microsoft Press, 3rd edition.

[Wigdor and Wixon, 2011] Wigdor, D. and Wixon, D. (2011). *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture*. Morgan Kaufmann.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS2S1 Sistemas Operativos (Obligatorio)

1. Información general

- **Semestre**: 5to Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS221 Arquitectura de Computadores (4to Sem-Pág 343)

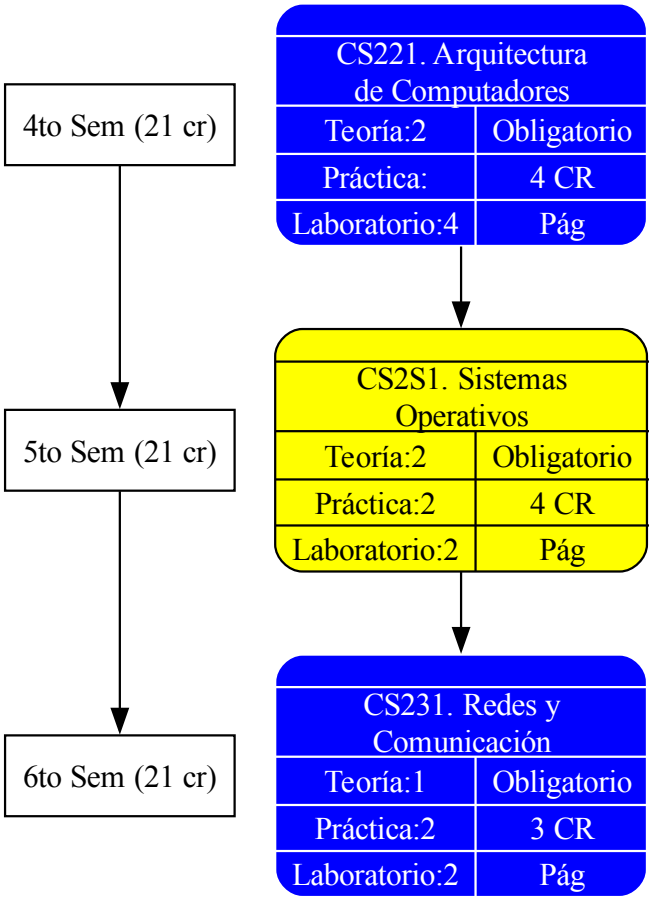

Figura 5.29: Mapa de Conexión. CS2S1 Sistemas Operativos

2. Fundamentación

El sistema operativo es una colección de servicios necesarios para interconectar de manera segura el hardware con las aplicaciones. Este curso explora los mecanismos y políticas necesarios para virtualizar el cómputo, la memoria y las E/S. Los estudiantes comprenderán conceptos centrales como concurrencia, persistencia y límites de confianza, los cuales son fundamentales para desarrollar sistemas de software robustos y eficientes.

3. Objetivos del curso.

- Comprender el rol y propósito de los sistemas operativos modernos.
- Dominar la planificación de procesos y el ciclo de vida de las unidades de ejecución.
- Analizar la gestión de memoria, incluyendo la virtualización de memoria física y virtual.
- Evaluar sistemas de almacenamiento y técnicas de gestión de archivos.
- Comprender los conceptos básicos de seguridad y protección dentro de un entorno de sistema operativo.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

5. Contenido

5.1 Rol y Propósito de los Sistemas Operativos, 8 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Los sistemas operativos median entre el hardware de propósito general y el software específico de la aplicación. **II:** Funciones universales del sistema operativo (por ejemplo, proceso, interfaces de usuario y dispositivos, persistencia de datos) **III:** Funciones extendidas y/o especializadas del sistema operativo (por ejemplo, sistemas embebidos, tipos de servidor como archivos, web, multimedia, gestores de arranque y seguridad de arranque) **IV:** Problemas de diseño (por ejemplo, eficiencia, robustez, flexibilidad, portabilidad, seguridad, compatibilidad, energía, seguridad, compensaciones entre verificación de errores y rendimiento, flexibilidad y rendimiento, y seguridad y rendimiento). Ver también: Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad **V:** Influencias de seguridad, redes, multimedia, computación paralela y distribuida **VI:** Preocupación general de seguridad/protección: Omitir considerar la seguridad en cada capa crea una oportunidad para acceder de manera inapropiada a los recursos. *a*) Se puede lograr acceso no autorizado a archivos en una unidad no cifrada moviendo el medio a otra computadora. *b*) La seguridad impuesta por los sistemas operativos puede ser derrotada infiltrando la capa de arranque antes de que se cargue el sistema operativo. *c*) El aislamiento de procesos puede ser subvertido por una verificación de autorización inadecuada en los límites de la API. *d*) Las vulnerabilidades en el firmware del sistema pueden proporcionar vectores de ataque que eviten por completo el sistema operativo. *e*) El aislamiento inadecuado de la memoria, cómputo y hardware de la máquina virtual puede exponer el sistema anfitrión a ataques desde sistemas invitados. *f*) El sistema operativo puede necesitar mitigar la explotación de vulnerabilidades de hardware y firmware, lo que lleva a posibles reducciones de rendimiento (por ejemplo, mitigaciones de Spectre y Meltdown). **VII:** Exposición de funciones del sistema operativo en shells y programación de sistemas. Ver también: Scripting de Shell

Objetivos de Aprendizaje **I:** Comprender los objetivos y funciones de los sistemas operativos modernos [Explicar] **II:** Evaluar los problemas de diseño en diferentes escenarios de uso (por ejemplo, sistema operativo en tiempo real, móvil, servidor) [Evaluar] **III:** Comprender las funciones de un sistema operativo contemporáneo con respecto a conveniencia, eficiencia y capacidad de evolucionar [Explicar] **IV:** Comprender cómo la evolución y la estabilidad son deseables y mutuamente antagónicas en las funciones del sistema operativo [Explicar]

Bibliografía: [Silberschatz et al., 2018, Tanenbaum and Bos, 2014]

5.2 Scripting de Shell, 8 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Manejo de errores/excepciones **II:** Tuberías (piping) **III:** Comandos del sistema: *a*) Interfaz con sistemas operativos **IV:** Variables de entorno **V:** Abstracción de archivo y operadores **VI:** Estructuras de datos, como arreglos y listas. **VII:** Expresiones regulares **VIII:** Programas y procesos **IX:** Flujo de trabajo

Objetivos de Aprendizaje **I:** Crear y ejecutar scripts automatizados para gestionar varias tareas del sistema [Crear] **II:** Resolver varios problemas de procesamiento de texto mediante scripting [Resolver]

Bibliografía: [Robbins and Beebe, 2005, Silberschatz et al., 2018]

5.3 Planificación, 16 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Planificación con expropiación y sin expropiación **II:** Planificadores y políticas (por ejemplo, primero en llegar primero en ser servido, trabajo más corto primero, prioridad, round robin, multinivel). Ver también: Gestión de Recursos **III:** Conceptos de planificación de multiprocesador simétrico (SMP) y coherencia de caché Ver también: Jerarquía de Memoria **IV:** Temporizadores (por ejemplo, construir muchos temporizadores a partir de temporizadores de hardware finitos). Ver también: Organización de Máquina a Nivel de Ensamblador **V:** Justicia e inanición **VI:** Subtemas de sistemas operativos como planificación consciente de energía y planificación en tiempo real. Ver también: Rendimiento y Eficiencia Energética, Características y Arquitecturas Embebidas, Tiempo Real, Memoria y Verificación, Sistemas, Programación y Sensores, Plataformas Móviles: Capacidades y Computación Asíncrona, Plataformas Móviles: Arquitectura y Restricciones, Plataformas Móviles: Fundamentos del Desarrollo **VII:** Planificación cooperativa, como futexes de Linux y planificación en espacio de usuario.

Objetivos de Aprendizaje **I:** Comparar y contrastar los algoritmos comunes utilizados para planificación con y sin expropiación de tareas en sistemas operativos, como prioridad, comparación de rendimiento y esquemas de participación justa [Comparar] **II:** Explicar las relaciones entre algoritmos de planificación y dominios de aplicación [Explicar] **III:** Explicar las distinciones entre tipos de planificadores de procesador como corto plazo, mediano plazo, largo plazo y E/S [Explicar] **IV:** Evaluar un problema o solución para determinar la idoneidad para multiprocesamiento asimétrico y/o simétrico [Evaluar] **V:** Evaluar un problema o solución para determinar la idoneidad como proceso vs hilos [Evaluar] **VI:** Listar algunos contextos que se benefician de la planificación con expropiación y con plazo límite [Listar/Enumerar] **VII:** Explicar las formas en que la lógica incorporada en los algoritmos de planificación es aplicable a otros mecanismos de sistemas operativos, como primero en llegar primero en ser servido o prioridad para E/S de disco, planificación de red, planificación de proyectos y problemas más allá de la computación [Explicar]

Bibliografía: [Silberschatz et al., 2018, Stallings, 2017]

5.4 Gestión de Memoria, 16 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Revisión de memoria física, traducción de direcciones y hardware de gestión de memoria. Ver también: Jerarquía de Memoria, Matemáticas Discretas **II:** Impacto de la jerarquía de memoria incluyendo concepto de caché, búsqueda en caché y almacenamiento en caché por CPU en mecanismos y políticas del sistema operativo Ver también: Jerarquía de Memoria, Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento **III:** Direccionamiento lógico y físico, virtualización de espacio de direcciones. Ver también: Jerarquía de Memoria, Matemáticas Discretas **IV:** Conceptos de paginación, reemplazo de páginas, trashing y asignación de páginas

y marcos **V:** Técnicas de asignación/desasignación/almacenamiento (algoritmos y estructura de datos) rendimiento y flexibilidad *a*) Arenas, asignadores de losas (slab allocators), listas libres, clases de tamaño, páginas de tamaño heterogéneo (páginas enormes) **VI:** Almacenamiento en caché de memoria y coherencia de caché y el efecto de vaciar la caché para evitar vulnerabilidades de ejecución especulativaVer también: Organización Funcional, Jerarquía de Memoria, Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento) **VII:** Mecanismos y conceptos de seguridad en gestión de memoria incluyendo sandboxing, protección, aislamiento y vectores relevantes de ataque. Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad **VIII:** Memoria virtual: aprovechando el hardware de memoria virtual para servicios del SO y eficiencia

Objetivos de Aprendizaje I: Explicar jerarquía de memoria y compensaciones costo-rendimiento [Explicar] **II:** Resumir los principios de memoria virtual aplicados a almacenamiento en caché y paginación [Resumir] **III:** Evaluar las compensaciones en términos de tamaño de memoria (memoria principal, memoria caché, memoria auxiliar) y velocidad del procesador [Evaluar] **IV:** Describir la razón y el uso de la memoria caché (rendimiento y proximidad, cómo las cachés complican el aislamiento y la abstracción de máquina virtual) [Describir] **V:** Codificar/Desarrollar programas eficientes que consideren los efectos del reemplazo de páginas y asignación de marcos en el rendimiento de un proceso y el sistema en el que se ejecuta [Crear] **VI:** Explicar cómo se utiliza el hardware para una virtualización eficiente [Explicar]

Bibliografía: [Silberschatz et al., 2018, Arpaci-Dusseau and Arpaci-Dusseau, 2018]

5.5 API de Sistemas de Archivos e Implementación, 12 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Concepto de archivo incluyendo datos, metadatos, operaciones y modo de acceso **II:** Montaje de sistema de archivos **III:** Control de acceso a archivos **IV:** Compartición de archivos **V:** Métodos básicos de asignación de archivos, incluida la tabla de asignación enlazada **VI:** Estructuras del sistema de archivos que comprenden asignación de archivos incluyendo varias estructuras de directorio y métodos para identificar archivos de manera única (por ejemplo, nombre, identificador o ubicación de almacenamiento de metadatos) **VII:** Técnicas de asignación/desasignación/almacenamiento (algoritmos y estructura de datos) impacto en rendimiento y flexibilidad (es decir, fragmentación interna y externa y compactación) **VIII:** Gestión de espacio libre como usar tablas de bits vs enlazado **IX:** Implementación de directorios para segmentar y rastrear ubicación de archivos

Objetivos de Aprendizaje I: Explicar las elecciones que deben tomarse en el diseño de sistemas de archivos [Explicar] **II:** Evaluar diferentes enfoques para organización de archivos, reconociendo las fortalezas y debilidades de cada uno [Evaluar] **III:** Aplicar constructores de software apropiadamente dado el conocimiento de la implementación del sistema de archivos [Aplicar]

Bibliografía: [Silberschatz et al., 2018, Stallings, 2017]

5.6 Protección y Seguridad, 12 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Descripción general de mecanismos de seguridad del sistema operativo. Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad **II:** Ataques y antagonismo (planificación, etc.). Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad **III:** Revisión de vulnerabilidades principales en sistemas operativos reales. Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad **IV:** Estrategias de mitigación del sistema operativo como copias de seguridad. Ver también: Confiabilidad del Sistema **V:** Separación política/mecanismo. Ver también: Gestión de Riesgos y Datos, Gobierno, Gestión y Política de Seguridad **VI:** Métodos y dispositivos de seguridad. Ver también: Seguridad en IA

y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad *a*) Anillos de protección (historia desde Multics hasta x86 virtualizado) *b*) Anillos -1 y -2 de x86_64 (hipervisor y ME/PSP) **VII:** Protección, control de acceso y autenticación. Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad, Protocolos, Aplicaciones y Criptografía Avanzada, Algoritmos Criptográficos, Fundamentos de Criptografía

Objetivos de Aprendizaje I: Comprender el requisito de mecanismos de protección y seguridad en sistemas operativos [Explicar] **II:** Listar y describir los vectores de ataque que aprovechan vulnerabilidades del SO [Listar/Enumerar] **III:** Comprender los mecanismos disponibles en un SO para controlar el acceso a recursos [Explicar] **IV:** Resumir las características y limitaciones de un sistema operativo que impactan la protección y seguridad [Resumir]

Bibliografía: [Silberschatz et al., 2018, Tanenbaum and Bos, 2014]

6. Bibliografía

[Arpaci-Dusseau and Arpaci-Dusseau, 2018] Arpaci-Dusseau, R. H. and Arpaci-Dusseau, A. C. (2018). *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books, 1.0 edition.

[Robbins and Beebe, 2005] Robbins, A. and Beebe, N. H. (2005). *Classic Shell Scripting*. O'Reilly Media.

[Silberschatz et al., 2018] Silberschatz, A., Galvin, P. B., and Gagne, G. (2018). *Operating System Concepts*. Wiley, 10th edition.

[Stallings, 2017] Stallings, W. (2017). *Operating Systems: Internals and Design Principles*. Pearson, 9th edition.

[Tanenbaum and Bos, 2014] Tanenbaum, A. S. and Bos, H. (2014). *Modern Operating Systems*. Pearson, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

FG106 Teatro (Obligatorio)

1. Información general

- **Semestre**: 5to Sem. **Créditos**: 2
- **Horas del curso**: **Teoría**: 1 horas; **Práctica**: 2 horas;
- **Prerrequisitos**: Ninguno

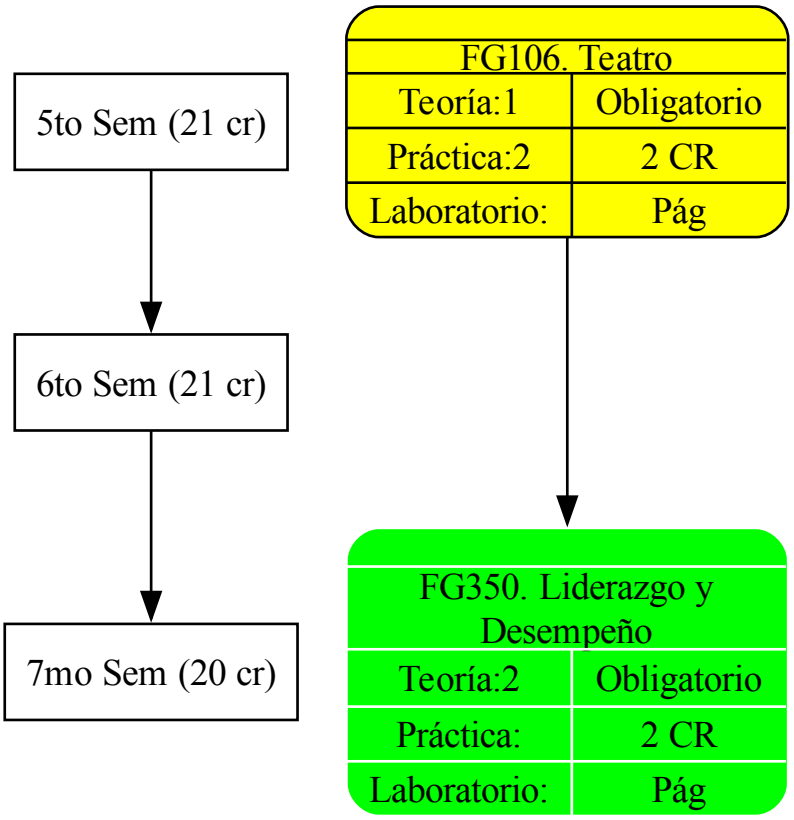

Figura 5.30: Mapa de Conexión. FG106 Teatro

2. Fundamentación

El teatro como arte escénica integral invita al estudiante a reconocerse como ser humano creativo y social. A través de la práctica teatral, el estudiante desarrolla su expresión corporal, verbal y emocional; fortalece su confianza y seguridad personal; y cultiva habilidades de comunicación, empatía y trabajo en equipo. Este curso proporciona herramientas prácticas de expresión artística que contribuyen a la formación integral del estudiante universitario, estimulando su sensibilidad, creatividad e imaginación. Al mismo tiempo, lo integra a su comunidad académica a través de experiencias vivenciales compartidas, abriendo caminos de autoconocimiento y de comunicación auténtica con los demás.

3. Objetivos del curso.

- Desarrollar la conciencia corporal y las habilidades comunicativas no verbales del estudiante mediante técnicas de expresión teatral.
- Fomentar la creatividad, la espontaneidad y el trabajo en equipo a través del juego dramático y la improvisación.
- Conocer el recorrido histórico del teatro desde sus orígenes hasta sus manifestaciones contemporáneas, incluyendo el teatro peruano.
- Valorar el arte teatral como herramienta de integración social, autoconocimiento y expresión de la identidad personal y colectiva.
- Aplicar las técnicas teatrales aprendidas en la creación y presentación pública de una propuesta escénica grupal.

4. Competencias Generales (*Outcomes*)

AG-C02) Ética: Aplica principios éticos y se compromete con la ética profesional y las normas de la práctica profesional de la computación. (Usar)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Usar)

5. Contenido

5.1 El Arte, la Creatividad y el Teatro, 6 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: **I:** ¿Qué es el Arte? Una experiencia vivencial y personal. **II:** La llave maestra: la creatividad. **III:** La importancia del teatro en la formación personal y profesional. **IV:** Utilidad y enfoque del arte teatral. **V:** El teatro como herramienta pedagógica y de desarrollo personal.

Objetivos de Aprendizaje **I:** Reconocer la vigencia del arte y la creatividad en el desarrollo personal y social [Usar]. **II:** Relacionar al estudiante con su grupo valorando la importancia de la comunicación humana y del colectivo social [Usar]. **III:** Identificar nociones básicas del teatro y su función en la vida universitaria [Usar].

Bibliografía: [Majorana, 1958, Pavis, 1998, Miravalles, 1998]

5.2 El Juego: el quehacer del actor, 6 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: **I:** Juego, luego existo. El juego como lenguaje universal. **II:** El juego del niño y el juego dramático. **III:** Juegos de integración grupal y juegos de creatividad. **IV:** La improvisación espontánea. **V:** La secuencia teatral y el calentamiento corporal.

Objetivos de Aprendizaje **I:** Reconocer el juego como herramienta fundamental del teatro [Usar]. **II:** Interiorizar y revalorar el juego como aprendizaje creativo [Usar]. **III:** Acercarse de manera espontánea y natural a la vivencia teatral [Usar].

Bibliografía: [Pavis, 1998, Miravalles, 1998]

5.3 La expresión corporal y el uso dramático del objeto, 9 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: **I:** Toma de conciencia del cuerpo: el cuerpo como instrumento. **II:** Toma de conciencia del espacio escénico. **III:** Toma de conciencia del tiempo y el ritmo. **IV:** Creación de secuencias individuales y colectivas: cuerpo, espacio y tiempo. **V:** El uso dramático del objeto: el juego teatral. **VI:** Presentaciones teatrales con el uso del objeto.

Objetivos de Aprendizaje I: Experimentar nuevas formas de expresión y comunicación corporal [Usar]. **II:** Desarrollar mecanismos de control y manejo del cuerpo en el espacio escénico [Usar]. **III:** Cultivar la imaginación y la capacidad de respuesta a estímulos auditivos, rítmicos y visuales [Usar]. **IV:** Explorar estados emocionales y climas colectivos a través del movimiento [Usar].

Bibliografía: [Pavis, 1998, Grotowski, 1970]

5.4 Comunicación No Verbal en el Teatro, 9 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: I: Relajación, concentración y respiración. **II:** Desinhibición e interacción con el grupo. **III:** La improvisación: técnicas y ejercicios. **IV:** Equilibrio, peso, tiempo y ritmo corporal. **V:** Análisis del movimiento: tipos y calidades. **VI:** La presencia escénica y la energía del actor. **VII:** La danza y la coreografía teatral. **VIII:** El gesto expresivo y el uso de la máscara.

Objetivos de Aprendizaje I: Ejercitarse en el manejo de destrezas comunicativas no verbales [Usar]. **II:** Practicar juegos y ejercicios de lenguaje corporal individual y grupal [Usar]. **III:** Expresar libre y creativamente emociones y sentimientos a través de representaciones originales [Usar]. **IV:** Distinguir los diferentes tipos de actuación y sus fundamentos técnicos [Usar].

Bibliografía: [Pavis, 1998, Stanislavski, 1936, Grotowski, 1970]

5.5 Huellas del Teatro en el Tiempo, 6 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: I: El origen del teatro: rito y representación en Grecia y Roma. **II:** El teatro medieval y la Commedia dell'Arte. **III:** Teatro del Siglo de Oro español y teatro isabelino inglés. **IV:** Romanticismo, Naturalismo y Realismo: Ibsen y Chéjov. **V:** El teatro épico de Brecht y el teatro psicológico de Stanislavski. **VI:** El teatro del absurdo, el teatro contemporáneo y el teatro total. **VII:** Teatro en el Perú: Yuyachkani, La Tarumba, Pataclaun y otros.

Objetivos de Aprendizaje I: Conocer la influencia que la sociedad ha ejercido en el teatro a lo largo de la historia [Usar]. **II:** Apreciar el valor y el aporte de dramaturgos y directores teatrales fundamentales [Usar]. **III:** Analizar el contexto social e histórico del arte teatral en distintas épocas [Usar]. **IV:** Reflexionar sobre el teatro peruano y sus expresiones contemporáneas [Usar].

Bibliografía: [Pavis, 1998, Campbell, 2001]

5.6 El Montaje Teatral, 12 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: I: Apreciación teatral: asistencia y análisis crítico de obras. **II:** El espacio escénico: tipos de escenario y su uso dramático. **III:** El texto dramático: lectura, análisis y adaptación. **IV:** Construcción del personaje: técnica y psicología. **V:** Dirección y puesta en escena: fundamentos básicos. **VI:** Elementos de producción: vestuario, maquillaje, escenografía y utilería. **VII:** Presentación pública de ejercicios teatrales en grupo.

Objetivos de Aprendizaje I: Emplear la creación teatral como manifestación de ideas y sentimientos ante la sociedad [Usar]. **II:** Aplicar las técnicas practicadas en una propuesta escénica concreta y significativa [Usar]. **III:** Intercambiar experiencias y realizar presentaciones teatrales breves en grupo frente al público [Usar].

Bibliografía: [Pavis, 1998, Miravalles, 1998, Brook, 1968, Campbell, 2001]

6. Bibliografía

[Brook, 1968] Brook, P. (1968). *The Empty Space: A Book About the Theatre*. Penguin Books.

[Campbell, 2001] Campbell, A. B. (2001). *Historia general del teatro en el Perú*. Universidad de San Martín de Porres.

[Grotowski, 1970] Grotowski, J. (1970). *Towards a Poor Theatre*. Simon and Schuster.

[Majorana, 1958] Majorana, A. (1958). *El arte de hablar en público*. La España Moderna.

[Miravalles, 1998] Miravalles, L. (1998). *Iniciación al teatro: teoría y práctica*. San Pablo.

[Pavis, 1998] Pavis, P. (1998). *Dictionary of the Theatre: Terms, Concepts, and Analysis*. University of Toronto Press.

[Stanislavski, 1936] Stanislavski, C. (1936). *An Actor Prepares*. Theatre Arts Books.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación
BEI203 Inglés V (Electivo)

1. Información general

- **Semestre**: 5to Sem. **Créditos**: 2
- **Horas del curso**: **Práctica**: 4 horas;
- **Prerrequisitos**:
  - BEI202 Inglés IV (4to Sem-Pág 362)

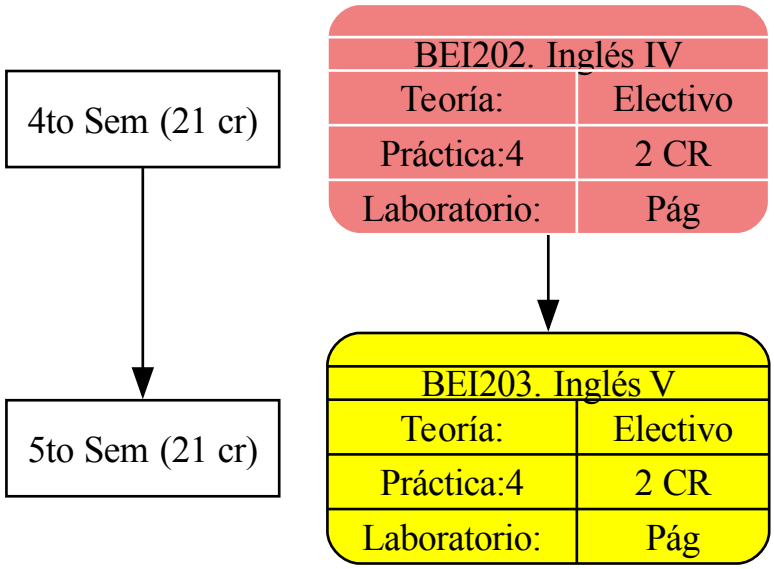

Figura 5.31: Mapa de Conexión. BEI203 Inglés V

2. Fundamentación

Una parte fundamental de la formación integral de un profesional es la capacidad de comunicarse en un idioma extranjero además del propio idioma nativo. En la carrera de ingeniería y computación, el inglés técnico es la herramienta más crítica para acceder a la vanguardia del conocimiento y la investigación, redactar artículos científicos, participar en conferencias internacionales y colaborar en equipos globales. Este curso lleva al estudiante al nivel de comunicador técnico autónomo, capaz de producir y evaluar críticamente contenido académico y profesional.

3. Objetivos del curso.

- Incrementar la capacidad y fluidez para comunicar ideas técnicas complejas en inglés.
- Lograr que los estudiantes interactúen con énfasis en la argumentación científica y la redacción técnica avanzada.
- Dominar el vocabulario de ingeniería de software, ciencia de datos e investigación de sistemas.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Familiarizarse)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Usar)

5. Contenido

5.1 Argumentación Científica, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Repaso avanzado de verbos auxiliares y tiempos verbales en escritura académica. **II:** Preguntas negativas y respuestas enfáticas en el debate científico. **III:** Formación de palabras: Prefijos y sufijos comunes en computación y ciencias. **IV:** Corrección de afirmaciones erróneas: Refutación de argumentos en discusiones técnicas.

Objetivos de Aprendizaje **I:** Al finalizar la unidad, el estudiante utiliza auxiliares y tiempos verbales con precisión para construir argumentos científicos, cuestionar afirmaciones técnicas erróneas y participar en debate académico estructurado.

Bibliografía: [? line:48 , ? line:48 , ? line:48 , ? line:48 , ? line:48]

5.2 Metodología de Investigación, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Presente Simple vs. Presente Continuo: Distinción entre hechos generales y experimentos en curso. **II:** Verbos de estado (State verbs) y verbos de acción en descripciones experimentales. **III:** Vocabulario: Diseño de investigación, variables experimentales, métodos de recolección de datos. **IV:** Redacción: Descripción de un método de investigación o procedimiento experimental.

Objetivos de Aprendizaje **I:** El estudiante diferencia entre usos estáticos y progresivos para describir con precisión leyes científicas, experimentos en curso y metodologías de investigación en computación e ingeniería.

Bibliografía: [? line:61 , ? line:61 , ? line:61 , ? line:61 , ? line:61]

5.3 Estudios de Caso, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Tiempos narrativos: Pasado Simple, Pasado Continuo y Pasado Perfecto en análisis de casos. **II:** Estructura de Üsed toz "Would"para describir sistemas heredados y prácticas pasadas. **III:** Vocabulario: Casos históricos de computación, cambios de paradigma, lecciones aprendidas. **IV:** Redacción: Informe estructurado de estudio de caso sobre un incidente o innovación computacional.

Objetivos de Aprendizaje **I:** El estudiante analiza y narra secuencias de eventos técnicos y casos históricos de computación con fluidez, aplicando correctamente los tiempos narrativos en informes profesionales de estudios de caso.

Bibliografía: [? line:74 , ? line:74 , ? line:74 , ? line:74 , ? line:74]

5.4 Propuestas de Investigación, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Predicciones y planes: Repaso del futuro para la redacción de propuestas académicas. **II:** Futuro Continuo y Futuro Perfecto en el alcance del proyecto y los resultados esperados. **III:** Vocabulario: Objetivos de investigación, contribuciones esperadas, alcance y limitaciones. **IV:** Redacción: Propuesta de investigación corta o abstract de proyecto técnico.

Objetivos de Aprendizaje I: El estudiante proyecta metas de investigación y resultados anticipados, y realiza predicciones basadas en evidencia sobre la evolución de la tecnología utilizando formas de futuro apropiadas en escritura académica formal.

Bibliografía: [? line:87, ? line:87, ? line:87, ? line:87, ? line:87]

5.5 Documentación Técnica, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Patrones verbales (Verbos seguidos de -ing o infinitivo) en escritura instruccional. **II:** Expresiones con "Like"para analogías y comparaciones técnicas. **III:** Vocabulario: Documentación de APIs, manuales de usuario, especificaciones técnicas. **IV:** Debates: Estándares de documentación de código abierto y reproducibilidad.

Objetivos de Aprendizaje I: El estudiante utiliza patrones verbales complejos para redactar documentación técnica clara y precisa, y emplea el razonamiento analógico para explicar conceptos técnicos a audiencias diversas.

Bibliografía: [? line:100, ? line:100, ? line:100, ? line:100, ? line:100]

5.6 Proyectos a Largo Plazo, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Presente Perfecto Continuo vs. Presente Perfecto Simple en informes de avance de investigación. **II:** Verbos modales de deducción (Presente y Pasado) para interpretar resultados experimentales. **III:** Vocabulario: Sistemas de largo plazo, programas de investigación sostenidos, desarrollo de startups. **IV:** Presentaciones orales: Ïnforme de avance de un proyecto de investigación en computación".

Objetivos de Aprendizaje I: El estudiante discute actividades de investigación y desarrollo a largo plazo, realiza deducciones sobre resultados técnicos y presenta informes de avance de proyectos computacionales con el registro académico apropiado.

Bibliografía: [? line:113, ? line:113, ? line:113, ? line:113, ? line:113]

5.7 Redacción Técnica Avanzada, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: I: Oraciones de relativo (Defining vs Non-defining) en la redacción de artículos técnicos. **II:** Voz pasiva avanzada en prosa científica y de ingeniería formal. **III:** Vocabulario: Revisión de pares, factor de impacto, prácticas de citación, acceso abierto. **IV:** Redacción: Abstract e introducción de un artículo de investigación en computación.

Objetivos de Aprendizaje I: El estudiante redacta textos técnicos y científicos precisos y bien estructurados utilizando cláusulas de relativo y construcciones pasivas avanzadas para añadir profundidad y claridad a la escritura de investigación.

Bibliografía: [? line:126, ? line:126, ? line:126, ? line:126, ? line:126]

5.8 Inglés para Ingeniería de Software, 12 hr(s)

Competencias: AG-C03,AG-C04

Tópicos: **I:** Metodologías Ágiles: Scrum, Kanban y XP — terminología en inglés. **II:** QA (Quality Assurance) y Testing: Planes de prueba, reportes de bugs, criterios de aceptación. **III:** Documentación de requerimientos de software: historias de usuario, casos de uso, especificaciones. **IV:** Simulación: Reunión técnica de equipo (Sprint planning y retrospectiva).

Objetivos de Aprendizaje **I:** Al finalizar el curso, el estudiante domina la terminología de ingeniería de software y gestión de proyectos, puede producir documentos técnicos profesionales y participar activamente en reuniones y revisiones técnicas internacionales en inglés.

Bibliografía: [**?** line:139 , **?** line:139 , **?** line:139 , **?** line:139 , **?** line:139]

6. Bibliografía

line:144 line:144 line:144 line:144 line:144

Sexto Semestre

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS231 Redes y Comunicación (Obligatorio)

1. Información general

- **Semestre**: 6to Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 1 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS2S1 Sistemas Operativos (5to Sem-Pág 388)

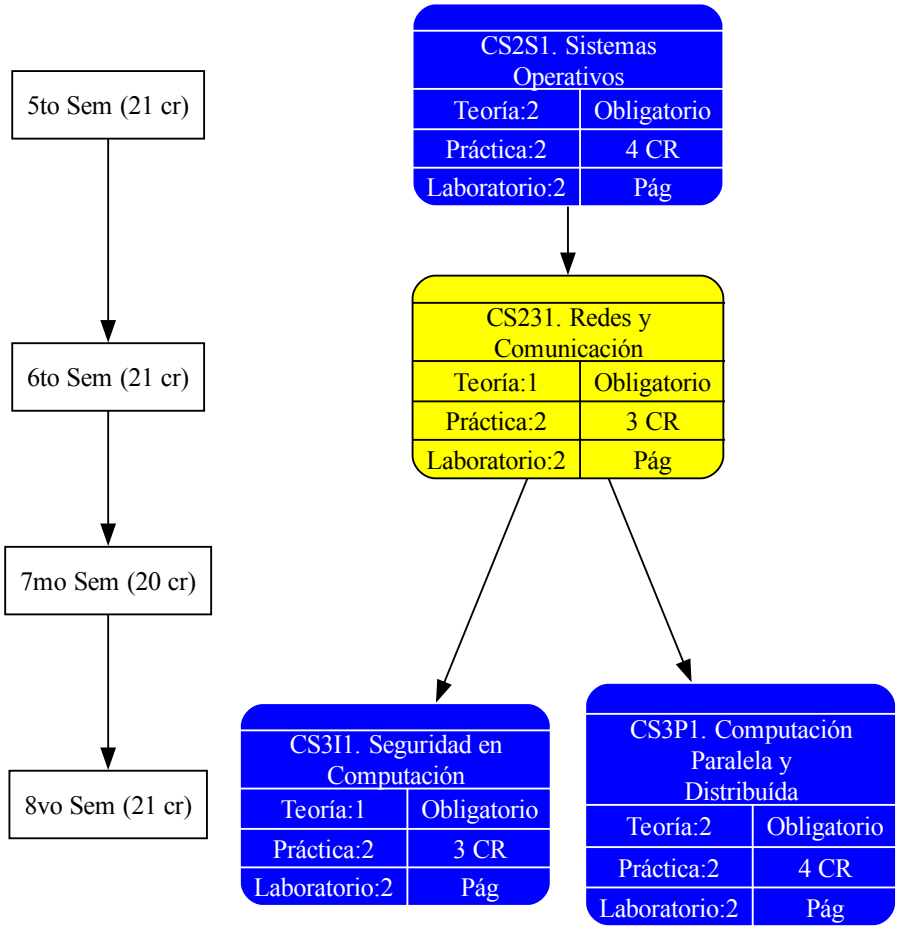

Figura 5.32: Mapa de Conexión. CS231 Redes y Comunicación

2. Fundamentación

Las redes y la comunicación son fundamentales para los sistemas de cómputo modernos, las aplicaciones distribuidas y la Internet global. Este curso ofrece una introducción completa al cuerpo de conocimiento de Redes y Comunicación del CS2023, abarcando desde la transmisión física y los protocolos de enlace de datos, pasando por el transporte confiable y el enrutamiento, hasta los servicios

de aplicación, la seguridad, la movilidad y paradigmas emergentes como SDN y redes cuánticas. Los estudiantes desarrollarán comprensión teórica y habilidades prácticas aplicables al diseño y análisis de sistemas en red modernos.

3. Objetivos del curso.

- Comprender la arquitectura en capas de los modelos TCP/IP y OSI, incluyendo el rol y responsabilidades de cada capa.
- Explicar los mecanismos de comunicación de un solo salto, incluyendo protocolos MAC, codificación, Ethernet, WiFi y topologías de red local.
- Dominar los principios de transferencia confiable de datos, control de flujo y control de congestión.
- Analizar paradigmas de enrutamiento, mecanismos de reenvío y escalabilidad de IP en redes locales y de área extensa.
- Comprender los protocolos fundamentales de la capa de aplicación (DNS, HTTP, TCP, UDP) e implementar aplicaciones basadas en sockets.
- Comprender las amenazas de seguridad en redes y las contramedidas para proteger los sistemas en red.
- Describir los principios de las redes móviles, incluyendo redes celulares, LANs inalámbricas y comunicación dispositivo a dispositivo.
- Identificar paradigmas de red emergentes como SDN, virtualización de redes y redes cuánticas.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

5. Contenido

5.1 Fundamentos de Redes y Comunicaciones, 6 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Importancia de las redes en la informática contemporánea, y desafíos asociados. Ver también: Equidad Digital y Accesibilidad, Impacto Social e IA, Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad **II:** Organización de Internet (por ejemplo, usuarios, Proveedores de Servicios de Internet, sistemas autónomos, proveedores de contenido, redes de entrega de contenido) **III:** Técnicas de conmutación (por ejemplo, circuito y paquete) **IV:** Capas y sus roles (aplicación, transporte, red, enlace de datos y física) **V:** Principios de estratificación (por ejemplo, encapsulación y modelo de reloj de arena). Ver también: Sistemas Digitales y Lógica, Procesamiento Secuencial y Paralelo, Máquinas de Estado **VI:** Elementos de red (por ejemplo, routers, switches, hubs, puntos de acceso y hosts) **VII:** Conceptos básicos de colas (por ejemplo, relación con latencia, congestión, niveles de servicio, etc.)

Objetivos de Aprendizaje **I:** Articular la organización de Internet [Articular] **II:** Listar y definir la terminología de red apropiada [Listar/Enumerar] **III:** Describir la estructura en capas de una arquitectura de red típica [Describir] **IV:** Identificar los diferentes tipos de complejidad en una red (bordes, núcleo, etc.) [Analizar]

Bibliografía: [Kurose and Ross, 2021, Tanenbaum et al., 2021, Peterson and Davie, 2022]

5.2 Comunicación de un solo salto, 6 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Introducción a modulación, ancho de banda y medios de comunicación **II:** Codificación y Entramado **III:** Control de Acceso al Medio (MAC) (por ejemplo, acceso aleatorio y acceso programado) **IV:** Ethernet y WiFi **V:** Conmutación (por ejemplo, árboles de expansión, VLANs). **VI:** Topologías de Red de área Local (por ejemplo, centro de datos, redes de campus).

Objetivos de Aprendizaje **I:** Describir algunos aspectos básicos de modulación, ancho de banda y medios de comunicación [Describir] **II:** Describir en detalle un protocolo MAC [Describir] **III:** Demostrar comprensión de las compensaciones (trade-offs) de soluciones de codificación y entramado [Demostrar] **IV:** Describir detalles de la implementación de Ethernet [Describir] **V:** Describir cómo funciona la conmutación [Describir] **VI:** Describir un tipo de topología de LAN [Describir]

Bibliografía: [Tanenbaum et al., 2021, Peterson and Davie, 2022]

5.3 Aplicaciones en Red, 6 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Esquemas de denominación y direccionamiento (por ejemplo, DNS e Identificadores Uniformes de Recursos) **II:** Paradigmas de aplicaciones distribuidas (por ejemplo, cliente/servidor, peer-to-peer, nube, edge y fog). Ver también: Comunicación, Coordinación **III:** Diversidad de demandas de aplicaciones en red (por ejemplo, latencia, ancho de banda y tolerancia a pérdidas). Ver también: Comunicación, Diseño Sostenible y Computación Ubicua, Huella Ambiental de los Sistemas Informáticos, Efectos Sistémicos y Contexto Social, Equidad Digital y Accesibilidad, Impacto Social e IA **IV:** Cobertura de protocolos de capa de aplicación (por ejemplo, HTTP) **V:** Interacciones con APIs de TCP, UDP y Socket. Ver también: Programas: Distribución, Programas: Propiedades de Ejecución, Programación GPU, Programas: Mapeos de Implementación, Programas: Paralelismo Declarativo, Programas: Inicio de Actividades

Objetivos de Aprendizaje **I:** Definir los principios de denominación, direccionamiento y localización de recursos [Definir] **II:** Analizar las necesidades de demandas específicas de aplicaciones en red [Analizar] **III:** Describir los detalles de un protocolo de capa de aplicación [Describir] **IV:** Implementar una aplicación simple cliente-servidor basada en sockets [Implementar]

Bibliografía: [Kurose and Ross, 2021, Peterson and Davie, 2022]

5.4 Soporte de Fiabilidad, 6 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Entrega no fiable (por ejemplo, UDP) **II:** Principios de fiabilidad (por ejemplo, entrega sin pérdidas, duplicación o fuera de orden). Ver también: Confiabilidad del Sistema **III:** Control de errores (por ejemplo, retransmisión, corrección de errores) **IV:** Control de flujo (por ejemplo, parar y esperar, basado en ventana) **V:** Control de congestión (por ejemplo, notificación implícita y explícita de congestión) **VI:** TCP y problemas de rendimiento (por ejemplo, Tahoe, Reno, Vegas, Cubic)

Objetivos de Aprendizaje **I:** Describir el funcionamiento de protocolos de entrega fiable [Describir] **II:** Listar los factores que afectan el rendimiento de los protocolos de entrega fiable [Listar/Enumerar] **III:** Describir algunos problemas de diseño de fiabilidad de TCP [Describir] **IV:** Diseñar e implementar un protocolo fiable simple [Diseñar]

Bibliografía: [Kurose and Ross, 2021, Tanenbaum et al., 2021]

5.5 Enrutamiento y Reenvío, 6 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Paradigmas y jerarquía de enrutamiento (por ejemplo, intra/inter dominio, centralizado y descentralizado, enrutamiento de origen, circuitos virtuales, QoS) **II:** Métodos de reenvío (por ejemplo, tablas de reenvío y algoritmos de coincidencia) **III:** IP y problemas de escalabilidad (por ejemplo, NAT, CIDR, BGP, diferentes versiones de IP)

Objetivos de Aprendizaje **I:** Describir varios paradigmas y jerarquías de enrutamiento [Describir] **II:** Describir cómo se reenvían los paquetes en una red IP [Describir] **III:** Describir cómo Internet aborda los desafíos de escalabilidad [Describir]

Bibliografía: [Kurose and Ross, 2021, Tanenbaum et al., 2021, Peterson and Davie, 2022]

5.6 Seguridad de Red, 6 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Introducción general sobre seguridad (Amenazas, vulnerabilidades y contramedidas). Ver también: Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad, Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad, Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad **II:** Amenazas específicas de red y tipos de ataque (por ejemplo, denegación de servicio, suplantación, rastreo y redireccionamiento de tráfico, atacante en el medio, ataques de integridad de mensajes, ataques de enrutamiento, ransomware y análisis de tráfico). Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad, Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad **III:** Contramedidas: *a*) Criptografía (por ejemplo, SSL, TLS, simétrica/asimétrica) *b*) Arquitecturas para redes seguras (por ejemplo, canales seguros, protocolos de enrutamiento seguro, DNS seguro, VPNs, DMZ, Zero Trust Network Access, hiper seguridad de red, protocolos de comunicación anónima, aislamiento) *c*) Monitoreo de red, detección de intrusiones, firewalls, protección contra suplantación y DoS, honeypots, trazabilidad, BGP Sec, RPKI

Objetivos de Aprendizaje **I:** Describir algunos de los modelos de amenaza de seguridad de red [Describir] **II:** Describir contramedidas específicas basadas en red [Describir] **III:** Analizar varios aspectos de seguridad de red a partir de un estudio de caso [Analizar]

Bibliografía: [Stallings, 2022, Kurose and Ross, 2021]

5.7 Soporte de Movilidad, 6 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Principios de comunicación celular (por ejemplo, 4G, 5G) **II:** Principios de LANs inalámbricas (principalmente 802.11) **III:** Comunicación dispositivo a dispositivo (por ejemplo, comunicación IoT) **IV:** Redes inalámbricas multiescalón (por ejemplo, redes ad hoc, oportunistas, tolerantes a retrasos)

Objetivos de Aprendizaje **I:** Describir algunos aspectos de comunicación celular como el registro [Describir] **II:** Describir cómo 802.11 soporta usuarios móviles [Describir] **III:** Describir usos prácticos de comunicación dispositivo a dispositivo, así como multiescalón [Describir] **IV:** Describir un tipo de red móvil como ad hoc [Describir]

Bibliografía: [Tanenbaum et al., 2021, Peterson and Davie, 2022]

5.8 Temas Emergentes, 6 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Middleboxes (por ejemplo, avances en el uso de IA, redes basadas en intención, filtrado, inspección profunda de paquetes, balanceo de carga, NAT, CDN) **II:** Virtualización de Red (por ejemplo, SDN, Redes de Centros de Datos) **III:** Redes Cuánticas (por ejemplo, Introducción al dominio, teleportación, seguridad, Internet Cuántica) **IV:** Satélite, mmWave, Luz Visible

Objetivos de Aprendizaje I: Describir el valor de los avances en middleboxes en redes [Describir] **II:** Describir la importancia de las Redes Definidas por Software [Describir] **III:** Describir algunos de los valores añadidos logrados al usar Redes Cuánticas [Describir]

Bibliografía: [Peterson and Davie, 2022, Tanenbaum et al., 2021]

6. Bibliografía

[Kurose and Ross, 2021] Kurose, J. and Ross, K. (2021). *Computer Networking: A Top-Down Approach*. Pearson, 8th edition.

[Peterson and Davie, 2022] Peterson, L. L. and Davie, B. S. (2022). *Computer Networks: A Systems Approach*. Morgan Kaufmann, 6th edition.

[Stallings, 2022] Stallings, W. (2022). *Cryptography and Network Security: Principles and Practice*. Pearson, 8th edition.

[Tanenbaum et al., 2021] Tanenbaum, A. S., Feamster, N., and Wetherall, D. J. (2021). *Computer Networks*. Pearson, 6th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS311 Algoritmos para Problemas Complejos (Obligatorio)

1. Información general

- **Semestre**: 6to Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS212 Análisis y Diseño de Algoritmos (5to Sem-Pág 366)

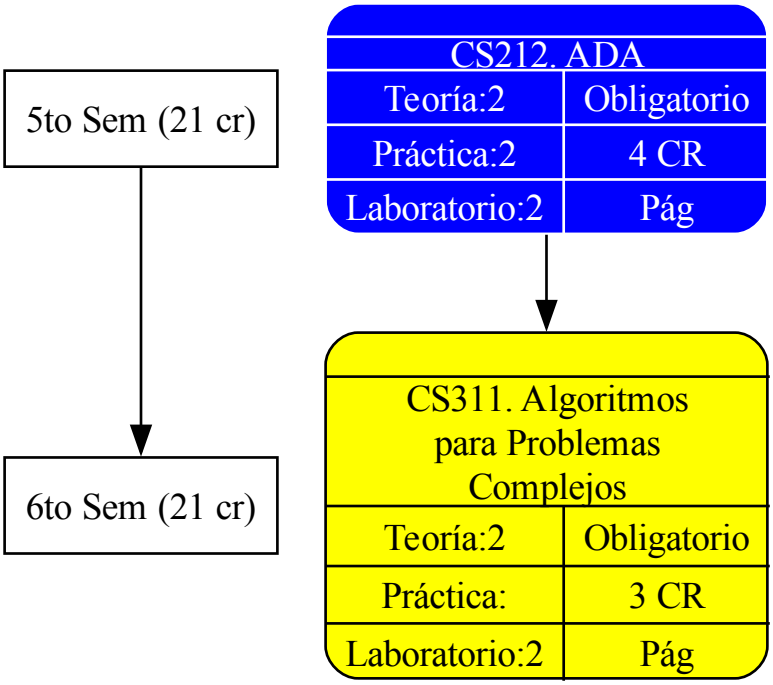

Figura 5.33: Mapa de Conexión. CS311 Algoritmos para Problemas Complejos

2. Fundamentación

La Programación Competitiva entrena a los estudiantes para resolver problemas algorítmicos complejos bajo restricciones de tiempo y recursos. Este curso cierra la brecha entre el diseño algorítmico teórico y la implementación de alto rendimiento. Cubre técnicas avanzadas en programación dinámica, teoría de grafos y matemáticas, fomentando el pensamiento analítico requerido para entrevistas de ingeniería de software de primer nivel y concursos de programación internacionales como el ICPC.

3. Objetivos del curso.

- Dominar técnicas y paradigmas avanzados de programación competitiva.
- Desarrollar la capacidad de analizar rápidamente la complejidad de soluciones potenciales.
- Implementar estructuras de datos eficientes para consultas de rango y actualizaciones dinámicas.
- Aplicar algoritmos matemáticos y geométricos especializados para resolver problemas de concursos.
- Optimizar código para velocidad de ejecución y uso de memoria en un entorno competitivo.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Paradigmas de resolución de problemas, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Búsqueda Completa: Retroceso Recursivo y Máscaras de bits. **II:** Estrategias Voraces: Propiedad de elección y subestructura óptima. **III:** Divide y Vencerás: Búsqueda Binaria sobre la respuesta y Búsqueda Ternaria. **IV:** Programación Dinámica: Reducción de estado, DP de dígitos y DP en árboles.

Objetivos de Aprendizaje I: Aplicar técnicas de poda de búsqueda para resolver problemas NP-difíciles con restricciones pequeñas [Usar] **II:** Identificar el paradigma más eficiente para un enunciado de problema dado [Evaluar]

Bibliografía: [Halim and Halim, 2020, Laaksonen, 2020]

5.2 Búsquedas por rango avanzadas, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Árboles de Fenwick (Árboles Binarios Indexados). **II:** Árboles de Segmentos: Propagación Diferida y Árboles de Segmentos Persistentes. **III:** Descomposición de Raíz Cuadrada y Algoritmo de Mo. **IV:** Tablas Dispersas para RMQ.

Objetivos de Aprendizaje I: Implementar estructuras de consulta de rango para actualizaciones de datos en tiempo real [Evaluar] **II:** Analizar las compensaciones entre diferentes estructuras de datos de consulta de rango [Usar]

Bibliografía: [Halim and Halim, 2020, Skiena, 2020]

5.3 Algoritmos avanzados de grafos, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Flujo en Redes: Flujo Máximo Corte Mínimo (Dinic y Edmonds-Karp). **II:** Componentes Fuertemente Conexas (Tarjan y Kosaraju). **III:** Emparejamiento Bipartito y Descomposición Pesado-Ligero. **IV:** Descomposición por Centroide en árboles.

Objetivos de Aprendizaje I: Para cada algoritmo en esta unidad explicar paso a paso cómo opera el algoritmo [Explicar] **II:** Para cada enfoque algorítmico (por ejemplo, ordenamiento) en esta unidad aplicar un ejemplo prototípico del enfoque (por ejemplo, ordenamiento por mezcla) [Aplicar] **III:** Dados los requisitos para un problema, desarrollar múltiples soluciones usando varias estructuras de datos y algoritmos. Posteriormente, evaluar la idoneidad, fortalezas y debilidades seleccionando un enfoque que satisfaga mejor los requisitos [Crear] **IV:** Explicar factores más allá de la eficiencia computacional que influyen en la elección de algoritmos, como el tiempo de programación, la mantenibilidad y el uso de patrones específicos de la aplicación en los datos de entrada [Explicar] **V:** Para cada uno de los algoritmos y enfoques algorítmicos en los temas del Núcleo KA: *a*) Explicar un ejemplo prototípico del algoritmo, y *b*) Explicar paso a paso cómo opera el algoritmo. [Explicar]

Bibliografía: [Laaksonen, 2020, Halim and Halim, 2020]

5.4 Matemática y Geometría, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Teoría de Números: Criba de Eratóstenes, Euclides Extendido e Inverso Modular. **II:** Combinatoria: Inclusión-Exclusión y Lema de Burnside. **III:** Geometría Computacional: Producto Cruz, Envolvente Convexa (Cadena Monótona) e Intersección de líneas. **IV:** Transformada Rápida de Fourier (FFT) para multiplicación de polinomios.

Objetivos de Aprendizaje I: Resolver problemas combinatorios usando aritmética modular [Usar] **II:** Implementar primitivas geométricas para análisis de relaciones espaciales [Evaluar]

Bibliografía: [Skiena, 2020, Laaksonen, 2020]

6. Bibliografía

[Halim and Halim, 2020] Halim, S. and Halim, F. (2020). *Competitive Programming 4: The Lower Bound of Programming Contests*. Lulu.

[Laaksonen, 2020] Laaksonen, A. (2020). *Competitive Programmer's Handbook*. Draft.

[Skiena, 2020] Skiena, S. S. (2020). *The Algorithm Design Manual*. Springer, 3rd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS312 Estructuras de Datos Avanzadas (Obligatorio)

1. Información general

- **Semestre**: 6to Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS212 Análisis y Diseño de Algoritmos (5to Sem-Pág 366)

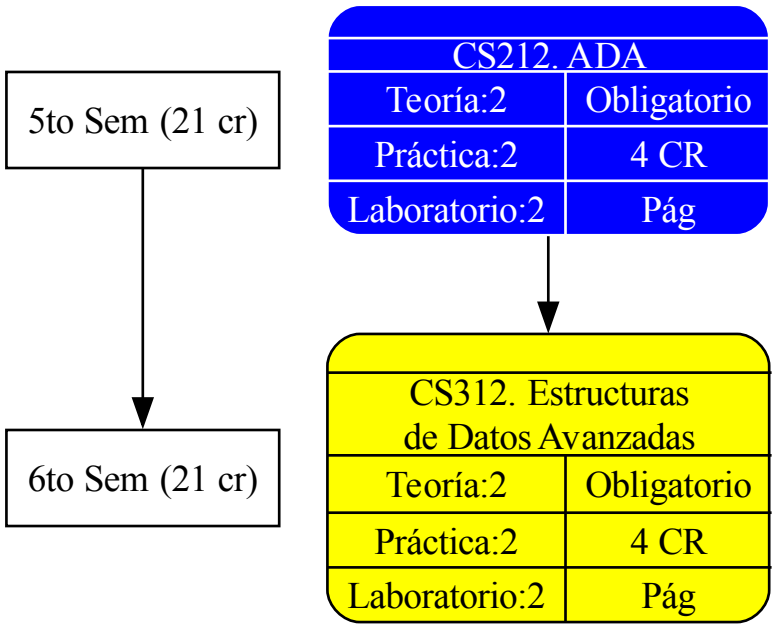

Figura 5.34: Mapa de Conexión. CS312 Estructuras de Datos Avanzadas

2. Fundamentación

Los algoritmos y estructuras de datos son una parte fundamental de la Ciencia de la Computación que nos permite organizar información de manera eficiente. Para cualquier profesional en el campo, una base sólida en esta área es crucial. Este curso se enfoca en estructuras avanzadas como Métodos de Acceso Multidimensional, Métodos de Acceso Espacio-Temporales y Métodos de Acceso Métrico, que son esenciales para aplicaciones de alto rendimiento en motores de búsqueda, sistemas de información geográfica (SIG) y procesamiento de big data.

3. Objetivos del curso.

- Comprender, diseñar e implementar estructuras de datos innovadoras para datos multidimensionales.
- Aplicar técnicas de búsqueda por similitud e indexación métrica a tipos de datos complejos.
- Analizar la ”Maldición de la Dimensionalidadz su impacto en la eficiencia de búsqueda.
- Investigar y presentar métodos de indexación contemporáneos para Big Data en inglés.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Técnicas de implementacion avanzadas, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Técnicas de implementación estáticas y dinámicas. **II:** Abstracción y encapsulación de datos. **III:** Tipos Abstractos de Datos (TAD). **IV:** Polimorfismo en tiempo de compilación vs. en tiempo de ejecución. **V:** Estructuras de datos genéricas y Plantillas de C++. **VI:** Componentes reutilizables y patrones de diseño para estructuras de datos.

Objetivos de Aprendizaje **I:** Evaluar el impacto de las técnicas estáticas y dinámicas en el diseño de estructuras de datos [Evaluar]. **II:** Implementar estructuras de datos genéricas y reutilizables usando paradigmas de programación avanzados [Usar]. **III:** Analizar las compensaciones de rendimiento entre diferentes estrategias de implementación [Evaluar].

Bibliografía: [Cuadros-Vargas et al., 2004, Knuth, 1997, Gamma et al., 1994, Vandevoorde et al., 2018]

5.2 Métodos de acceso multidimensional (Parte I), 8 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Introducción a datos multidimensionales. **II:** Representación en espacios vectoriales. **III:** La Maldición de la Dimensionalidad: fundamentos teóricos. **IV:** Impacto de la alta dimensionalidad en la indexación tradicional. **V:** Aplicaciones en el mundo real en motores de búsqueda y reconocimiento de patrones.

Objetivos de Aprendizaje **I:** Explicar la importancia de la representación de datos multidimensionales [Familiarizarse]. **II:** Analizar la complejidad y degradación del rendimiento en espacios de alta dimensión [Usar]. **III:** Discutir escenarios del mundo real donde el impacto de la dimensionalidad es crítico [Usar].

Bibliografía: [Gaede and Günther, 1998, Samet, 2006]

5.3 Métodos de acceso multidimensional (Parte II), 20 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Estructuras de datos espaciales: Quadtree y Octree. **II:** Métodos de Acceso a Puntos: Kd-Tree. **III:** Métodos de Acceso a Regiones: R-Tree (Guttman), R+ tree, R* tree. **IV:** Variaciones de R-trees y su relación con la paginación y el tamaño de bloque. **V:** Estructuras avanzadas: X-tree y SS-tree.

Objetivos de Aprendizaje **I:** Implementar estructuras de datos espaciales para indexación de datos de alto volumen [Usar]. **II:** Evaluar las limitaciones de las estructuras espaciales basadas en árboles en dominios específicos [Evaluar]. **III:** Implementar búsqueda por rango y estrategias de Vecino Más Cercano (k-NN) [Usar].

Bibliografía: [Samet, 2006]

5.4 Métodos de acceso métrico, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Espacios Métricos: definiciones y propiedades. **II:** Búsqueda por similitud vs. Búsqueda exacta. **III:** Árbol de Punto de Vista (VP-Tree). **IV:** Slim-Tree y M-Tree. **V:** Indexación basada en pivotes.

Objetivos de Aprendizaje I: Modelar problemas de búsqueda por similitud usando propiedades de espacios métricos [Usar]. **II:** Comparar métodos de acceso métrico contra métodos multidimensionales para datos no vectoriales [Evaluar].

Bibliografía: [Samet, 2006, Zezula et al., 2007]

5.5 Seminarios de investigación, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Seminario sobre artículos de investigación recientes (SIGMOD, VLDB, ICDE). **II:** Indexación para Big Data y datos en flujo. **III:** Estructuras de datos compactas y representaciones sucintas. **IV:** Tendencias futuras en búsqueda por similitud.

Objetivos de Aprendizaje I: Investigar nuevos métodos para indexar grandes volúmenes de datos complejos [Usar]. **II:** Presentar y dirigir discusiones técnicas en inglés sobre métodos de vanguardia [Evaluar]. **III:** Identificar posibles temas de tesis dentro del dominio de estructuras de datos avanzadas [Familiarizarse].

Bibliografía: [Navarro, 2016, Traina Jr et al., 2000]

6. Bibliografía

[Cuadros-Vargas et al., 2004] Cuadros-Vargas, E., Romero, R. A. F., Mock, M., and Brisaboa, N. (2004). Implementing data structures: An incremental approach. Available at: http://socios.spc.org.pe/ecuadros/cursos/pdfs/.

[Gaede and Günther, 1998] Gaede, V. and Günther, O. (1998). Multidimensional access methods. *ACM Computing Surveys*, 30(2):170–231.

[Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. M. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

[Knuth, 1997] Knuth, D. E. (1997). *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Addison-Wesley Professional, 3rd edition.

[Navarro, 2016] Navarro, G. (2016). *Compact Data Structures*. Cambridge University Press.

[Samet, 2006] Samet, H. (2006). *Foundations of Multidimensional and Metric Data Structures*. Elsevier/Morgan Kaufmann.

[Traina Jr et al., 2000] Traina Jr, C., Traina, A. J. M., Seeger, B., and Faloutsos, C. (2000). Slim-trees: High performance metric trees minimizing overlap between nodes. In *Advances in Database Technology - EDBT 2000*. Springer.

[Vandevoorde et al., 2018] Vandevoorde, D., Josuttis, N. M., and Gregor, D. (2018). *C++ Templates: The Complete Guide*. Addison-Wesley Professional.

[Zezula et al., 2007] Zezula, P., Amato, G., Dohnal, V., and Batko, M. (2007). *Similarity Search: The Metric Space Approach*. Springer.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS342 Compiladores (Obligatorio)

1. Información general

- **Semestre**: 6to Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS211 Teoría de la Computación (4to Sem-Pág 338)

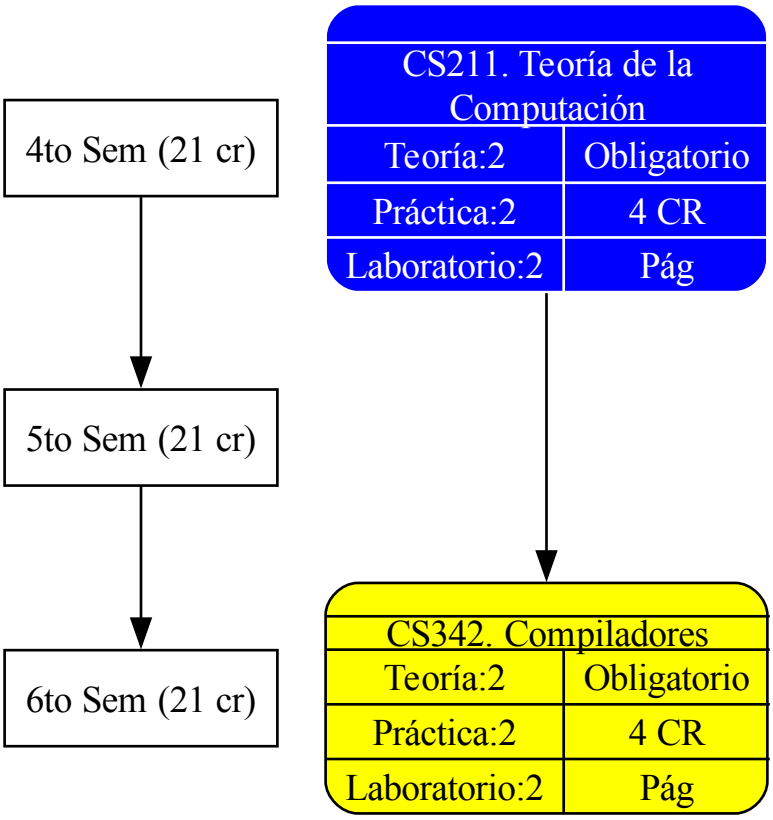

Figura 5.35: Mapa de Conexión. CS342 Compiladores

2. Fundamentación

Este curso proporciona una comprensión profunda de cómo los lenguajes de alto nivel se transforman en código máquina ejecutable. El estudio de los compiladores permite a los estudiantes comprender la semántica de los lenguajes, la gestión de memoria y la optimización de código, que son habilidades fundamentales para diseñar lenguajes específicos de dominio y realizar ingeniería de software de alto rendimiento.

3. Objetivos del curso.

- Comprender las fases de un compilador y las herramientas de construcción.

- Implementar analizadores léxicos y sintácticos basados en gramáticas.
- Generar y optimizar código intermedio para una arquitectura específica.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Análisis de Programas y Analizadores, 20 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Representaciones de programa relevantes, como bloques básicos (basic blocks), grafos de flujo de control (control-flow graphs), cadenas definición-uso (def-use chains) y asignación única estática (static single assignment) **II:** Indecidibilidad y consecuencias para el análisis de programas **III:** Análisis insensible al flujo (flow-insensitive), como verificación de tipos y análisis de punteros y alias escalables **IV:** Análisis sensible al flujo (flow-sensitive), como análisis de flujo de datos (dataflow) hacia adelante y hacia atrás **V:** Análisis sensible a la ruta (path-sensitive), como verificación de modelos de software (software model checking) y verificación de software **VI:** Herramientas y frameworks para implementar analizadores **VII:** Papel del análisis estático en la optimización de programas y el análisis de dependencia de datos durante la explotación de concurrencia **VIII:** Papel del análisis de programas en la verificación (parcial) y la búsqueda de errores (bug-finding) **IX:** Paralelización: *a*) Análisis para auto-paralelización *b*) Análisis para detectar errores de concurrencia

Objetivos de Aprendizaje **I:** Explicar la diferencia entre grafo de flujo de datos (dataflow graph) y grafo de flujo de control [Explicar] **II:** Explicar por qué los análisis de programa no triviales y sólidos (sound) deben ser aproximados [Explicar] **III:** Argumentar por qué un análisis es correcto (sound y terminante) [Argumentar] **IV:** Explicar por qué el alias potencial limita el análisis de programa sólido y cómo el análisis de alias puede ayudar [Explicar] **V:** Usar los resultados de un análisis de programa para optimización de programas y/o corrección parcial del programa [Usar]

Bibliografía: [Aho et al., 2006, Appel, 2004]

5.2 Traducción y Ejecución de Lenguajes, 24 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Modelos de ejecución para JIT (Just-In-Time), compilador, intérprete **II:** Uso de código intermedio, por ejemplo, bytecode **III:** Limitaciones y beneficios de JIT, compilador e intérprete **IV:** Compiladores/transpiladores cruzados (cross compilers/transpilers) **V:** Representación BNF y BNF extendida de gramática libre de contexto **VI:** árbol de análisis (parse tree) usando una oración simple como expresión aritmética o sentencia if-then-else **VII:** Ejecución como código nativo o dentro de una máquina virtual **VIII:** Canalización de traducción de lenguaje: análisis sintáctico, análisis (parsing), verificación de tipos opcional, generación de código/traducción y optimización, enlace (linking), carga (loading), ejecución **IX:** Representación en tiempo de ejecución de construcciones centrales del lenguaje como objetos (tablas de métodos) y funciones que pueden pasarse como parámetros a y devolverse desde funciones (clausuras) **X:** Desarrollo seguro de compiladores

Objetivos de Aprendizaje **I:** Explicar y comprender las diferencias entre implementaciones de lenguaje compilado, JIT e interpretado, incluyendo los beneficios y limitaciones de cada una [Explicar] **II:** Diferenciar sintaxis y análisis (parsing) de semántica y evaluación [Diferenciar] **III:** Usar

BNF y BNF extendida para especificar la sintaxis de construcciones simples como if-then-else, declaración de tipo y construcciones iterativas para lenguajes conocidos como C++ o Python [Usar] **IV:** Ilustrar el árbol de análisis usando una oración/expresión aritmética simple [Aplicar] **V:** Ilustrar traducción de diagramas de sintaxis a BNF/BNF extendida para construcciones simples como if-then-else, declaración de tipo, construcciones iterativas, etc [Aplicar] **VI:** Ilustrar ambigüedad en el análisis usando if-then-else anidado/expresión aritmética y mostrar resolución usando orden de precedencia [Aplicar] **VII:** Discutir los beneficios y limitaciones de la recolección de basura, incluyendo la noción de alcanzabilidad [Debatir]

Bibliografía: [Aho et al., 2006, Appel, 2004]

5.3 Comportamiento en Tiempo de Ejecución y Sistemas, 20 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Modelos de proceso usando pilas y montones para asignar y desasignar registros de activación y recuperar entornos usando punteros de marco (frame pointers) y direcciones de retorno durante una llamada a procedimiento, incluyendo ejemplos de paso de parámetros **II:** Esquemas de búsqueda de código usando tablas hash para métodos en implementaciones de programas orientados a objetos **III:** Distribución de datos para objetos y registros de activación **IV:** Asignación de objetos en el montón (heap) **V:** Implementar entidades virtuales y métodos virtuales; tablas de métodos virtuales (virtual method tables) y su aplicación **VI:** Comportamiento en tiempo de ejecución de programas orientados a objetos **VII:** Comparar y contrastar la asignación de memoria durante el intercambio de información usando paso de parámetros y variables no locales (usando cadena de enlaces estáticos). **VIII:** Enfoques y técnicas de gestión dinámica de memoria: malloc/free, recolección de basura (mark-sweep, copying, reference counting), regiones (también conocidas como arenas o zonas) **IX:** Compilación justo a tiempo (just-in-time) y recompilación dinámica **X:** Interfaz con el sistema operativo (por ejemplo, para inicialización del programa) **XI:** Interoperabilidad entre lenguajes de programación incluyendo mecanismos de paso de parámetros y representación de datos: *a*) *Big endian*, *little endian b*) Distribución de datos de tipos de datos compuestos como arreglos **XII:** Otras características comunes de las máquinas virtuales, como carga de clases (class loading), hilos (threads) y verificación de seguridad **XIII:** Sandboxing

Objetivos de Aprendizaje I: Discutir beneficios y limitaciones de la gestión automática de memoria [Debatir] **II:** Explicar el uso de metadatos en representaciones en tiempo de ejecución de objetos y registros de activación, como punteros de clase, longitudes de arreglo, direcciones de retorno y punteros de marco (frame pointers) [Explicar] **III:** Comparar y contrastar asignación estática vs asignación basada en pila vs asignación basada en montón de elementos de datos [Comparar] **IV:** Explicar por qué algunos elementos de datos no pueden desasignarse automáticamente al final de una llamada a procedimiento/método (necesidad de recolección de basura) [Explicar] **V:** Discutir ventajas, desventajas y dificultades de la compilación justo a tiempo y la recompilación dinámica [Debatir] **VI:** Discutir el uso del sandboxing en código móvil [Debatir] **VII:** Identificar los servicios proporcionados por los sistemas en tiempo de ejecución de lenguajes modernos [Analizar]

Bibliografía: [Aho et al., 2006, Cooper and Torczon, 2011]

6. Bibliografía

[Aho et al., 2006] Aho, A. V., Lam, M. S., Sethi, R., and Ullman, J. D. (2006). *Compilers: Principles, Techniques, and Tools*. Pearson, 2nd edition.

[Appel, 2004] Appel, A. W. (2004). *Modern Compiler Implementation in Java*. Cambridge University Press, 2nd edition.

[Cooper and Torczon, 2011] Cooper, K. and Torczon, L. (2011). *Engineering a Compiler*. Morgan Kaufmann, 2nd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI263 Introducción al Aprendizaje de Máquina (Obligatorio)

1. Información general

- **Semestre**: 6to Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - CS261 Inteligencia Artificial (5to Sem-Pág 371)

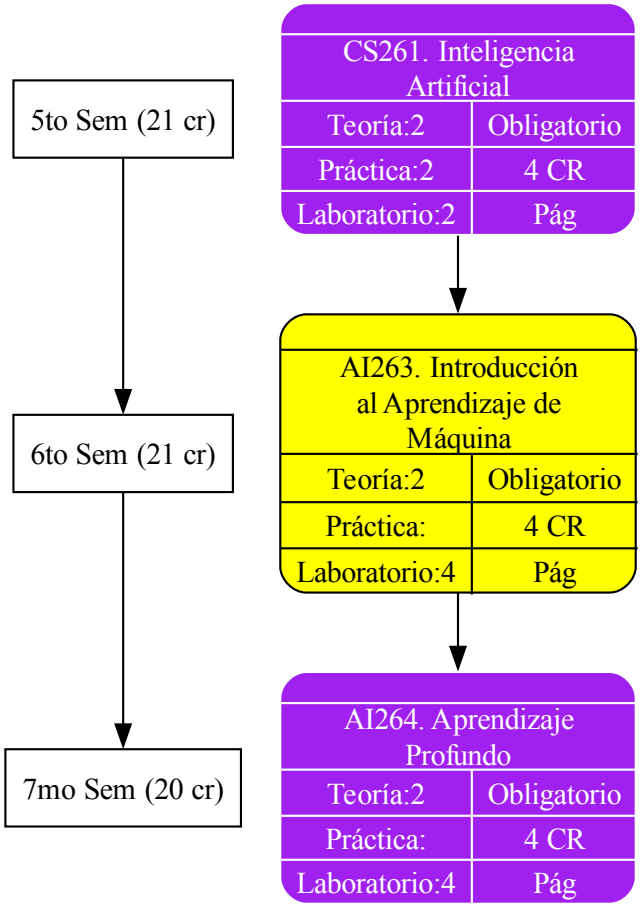

Figura 5.36: Mapa de Conexión. AI263 Introducción al Aprendizaje de Máquina

2. Fundamentación

Este curso introduce los fundamentos del aprendizaje automático, cubriendo algoritmos clásicos y modernos para problemas de clasificación, regresión y agrupamiento. Se enfoca en la implementación práctica usando scikit-learn y TensorFlow, con aplicaciones en visión por computadora y procesamiento de lenguaje natural.

3. Objetivos del curso.

- Comprender los principios matemáticos detrás de algoritmos de ML.

- Implementar pipelines completos de ML con Python.
- Evaluar y optimizar modelos usando métricas estándar.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

5. Contenido

5.1 Fundamentos de ML, 15 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Aprendizaje supervisado vs no supervisado **II:** Overfitting y regularización **III:** Validación cruzada y curvas de aprendizaje

Objetivos de Aprendizaje I: Explicar el bias-variance tradeoff [Familiarizarse] **II:** Implementar k-fold cross-validation [Usar]

Bibliografía: [Bishop, 2006, Goodfellow et al., 2016]

5.2 Modelos Lineales, 15 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Regresión lineal y logística **II:** Máquinas de vectores de soporte (SVM) **III:** Análisis discriminante lineal

Objetivos de Aprendizaje I: Programar modelos lineales con scikit-learn [Usar] **II:** Interpretar coeficientes de regresión [Evaluar]

Bibliografía: [Hastie et al., 2009, Murphy, 2012]

5.3 Ensambles y Árboles, 15 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Árboles de decisión y random forests **II:** Gradient Boosting (XGBoost, LightGBM) **III:** Stacking y votación

Objetivos de Aprendizaje I: Optimizar hiperparámetros con GridSearch [Usar] **II:** Visualizar árboles de decisión [Evaluar]

Bibliografía: [Hastie et al., 2009, Géron, 2022]

5.4 Redes Neuronales Básicas, 15 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Perceptrones multicapa (MLP) **II:** Backpropagation y optimizadores **III:** Introducción a Keras/TensorFlow

Objetivos de Aprendizaje I: Construir redes neuronales simples [Usar] **II:** Monitorizar entrenamiento con TensorBoard [Evaluar]

Bibliografía: [Goodfellow et al., 2016, Chollet, 2021]

6. Bibliografía

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[Chollet, 2021] Chollet, F. (2021). *Deep Learning with Python*. Manning, 2nd edition.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Géron, 2022] Géron, A. (2022). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly, 3rd edition.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer, 2nd edition.

[Murphy, 2012] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

FI201 Física Computacional (Obligatorio)

1. Información general

- **Semestre**: 6to Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - BFI101 Física I (2do Sem-Pág 297)

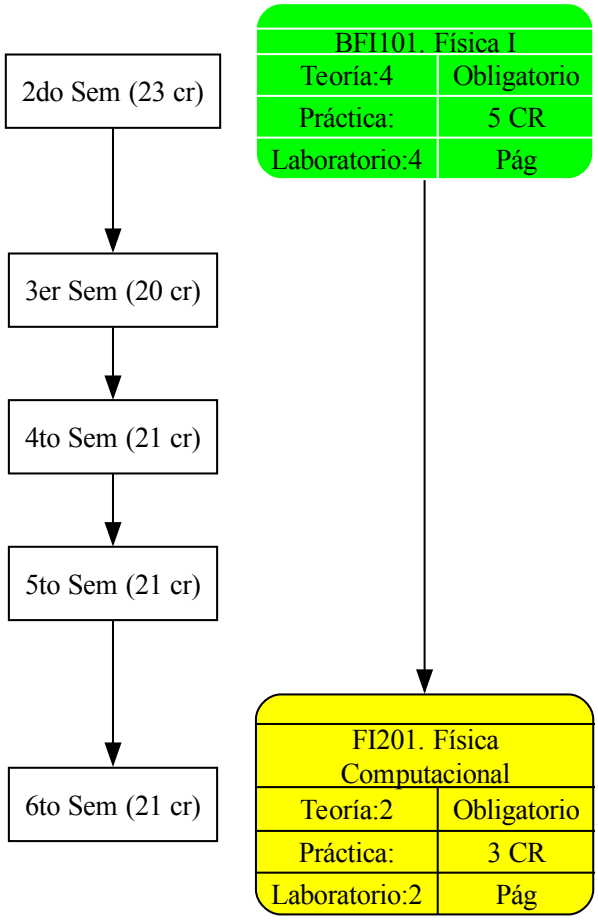

Figura 5.37: Mapa de Conexión. FI201 Física Computacional

2. Fundamentación

Este curso aplica los principios de la física a problemas computacionales, con énfasis en la luz, la propagación de ondas, colisiones y la transferencia de energía. Estos conceptos son esenciales en áreas como gráficos por computadora, simulaciones físicas y desarrollo de videojuegos.

3. Objetivos del curso.

- Comprender los principios físicos relevantes para la computación.

- Aplicar estos principios para resolver problemas computacionales específicos.
- Implementar algoritmos basados en la física para simulaciones y gráficos por computadora.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Usar)

5. Contenido

5.1 Óptica y Propagación de la Luz, 10 hr(s)

Competencias: AG-C08,AG-C11

Tópicos: I: Naturaleza de la luz. **II:** Reflexión y refracción. **III:** Lentes y espejos. **IV:** Interferencia y difracción. **V:** Modelos de iluminación (ej. Phong, Blinn-Phong).

Objetivos de Aprendizaje I: Describir las propiedades de la luz y su propagación. [Familiarizarse] **II:** Aplicar las leyes de la reflexión y refracción. [Usar] **III:** Implementar modelos de iluminación en gráficos por computadora. [Evaluar]

Bibliografía: [Young and Freedman, 2018, Hecht, 2017]

5.2 Colisiones y Transferencia de Energía, 8 hr(s)

Competencias: AG-C08,AG-C11

Tópicos: I: Impulso y momento lineal. **II:** Colisiones elásticas e inelásticas. **III:** Conservación de la energía en colisiones. **IV:** Deformación de mallas elásticas (ej. modelo de masa-resorte).

Objetivos de Aprendizaje I: Aplicar los principios de conservación del momento lineal y la energía en colisiones. [Familiarizarse] **II:** Modelar la deformación de mallas elásticas producto de un impacto. [Usar] **III:** Implementar simulaciones de colisiones en un entorno computacional. [Evaluar]

Bibliografía: [Young and Freedman, 2018, Taylor, 2005]

5.3 Mecánica de Sólidos Rígidos, 8 hr(s)

Competencias: AG-C08,AG-C11

Tópicos: I: Rotación de cuerpos rígidos. **II:** Momento de inercia. **III:** Torque y energía cinética rotacional.

Objetivos de Aprendizaje I: Describir la rotación de cuerpos rígidos. [Familiarizarse] **II:** Calcular el momento de inercia. [Usar] **III:** Aplicar las ecuaciones de la dinámica rotacional. [Evaluar]

Bibliografía: [Young and Freedman, 2018, Taylor, 2005]

5.4 Dinámica de Fluidos, 6 hr(s)

Competencias: AG-C08,AG-C11

Tópicos: I: Principios básicos de la dinámica de fluidos. **II:** Viscosidad. **III:** Flujo laminar y turbulento.

Objetivos de Aprendizaje I: Describir las propiedades de los fluidos. [Familiarizarse] **II:** Explicar los conceptos de viscosidad y flujo laminar/turbulento. [Usar] **III:** Resolver problemas sencillos de dinámica de fluidos. [Evaluar]

Bibliografía: [Young and Freedman, 2018]

5.5 Termodinámica, 6 hr(s)

Competencias: AG-C08,AG-C11

Tópicos: I: Leyes de la termodinámica. **II:** Transferencia de calor.

Objetivos de Aprendizaje I: Enunciar las leyes de la termodinámica. [Familiarizarse] **II:** Describir los mecanismos de transferencia de calor. [Usar]

Bibliografía: [Young and Freedman, 2018]

5.6 Simulación Física, 10 hr(s)

Competencias: AG-C08,AG-C11

Tópicos: I: Métodos numéricos para simulación física. **II:** Integración de Verlet. **III:** Detección de colisiones. **IV:** Sistemas de partículas. **V:** Restricciones y solvers.

Objetivos de Aprendizaje I: Implementar métodos numéricos básicos para simulación física. [Familiarizarse] **II:** Utilizar la integración de Verlet para simular el movimiento. [Usar] **III:** Implementar algoritmos de detección de colisiones. [Evaluar]

Bibliografía: [Taylor, 2005]

6. Bibliografía

[Hecht, 2017] Hecht, E. (2017). *Optics*. Pearson.

[Taylor, 2005] Taylor, J. R. (2005). *Classical Mechanics*. University Science Books.

[Young and Freedman, 2018] Young, H. D. and Freedman, R. A. (2018). *University Physics with Modern Physics*. Pearson.

Séptimo Semestre

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS251 Computación Gráfica (Obligatorio)

1. Información general

- **Semestre**: 7mo Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - MA202 Métodos Numéricos (4to Sem-Pág 357)

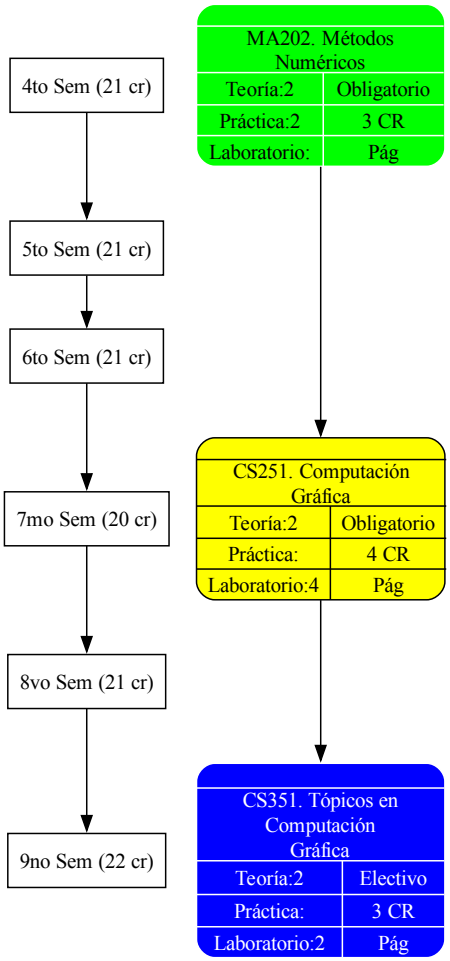

Figura 5.38: Mapa de Conexión. CS251 Computación Gráfica

2. Fundamentación

Los Gráficos por Computadora son un área fundamental de la Ciencia de la Computación que se enfoca en las técnicas matemáticas y computacionales para generar y manipular imágenes. Este curso proporciona a los estudiantes los fundamentos del modelado 2D y 3D, las canalizaciones de

renderizado y los principios físicos de la luz y el color, permitiéndoles desarrollar aplicaciones visuales, simulaciones y sistemas interactivos de alto rendimiento.

3. Objetivos del curso.

- Comprender la canalización de gráficos y las transformaciones de sistemas de coordenadas.
- Implementar algoritmos de renderizado para visualización realista y en tiempo real.
- Aplicar modelado matemático para crear y manipular objetos 3D.

4. Competencias Generales (*Outcomes*)

AG-C07) Conocimientos de Computación: Aplica conocimientos de matemáticas, ciencias y computación. (Usar)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Evaluar)

5. Contenido

5.1 Conceptos Fundamentales de Gráficos y Técnicas Interactivas, 16 hr(s)

Competencias: AG-C07,AG-C08

Tópicos: **I:** Descripción general del pipeline de gráficos por computadora y sistemas interactivos **II:** Conceptos básicos de renderizado: rasterización, trazado de rayos (*ray tracing*), modelos de sombreado **III:** Teoría del color y espacios de color **IV:** Sistemas de coordenadas y transformaciones **V:** Técnicas básicas de interacción: dispositivos de entrada, manejo de eventos **VI:** Conceptos avanzados de renderizado: iluminación global (*global illumination*), renderizado basado en física **VII:** Hardware gráfico y técnicas de aceleración **VIII:** Percepción humana y cognición visual

Objetivos de Aprendizaje **I:** Explicar las etapas básicas del pipeline de gráficos por computadora [Explicar] **II:** Describir técnicas básicas de renderizado como rasterización y trazado de rayos [Describir] **III:** Aplicar teoría del color para diseñar gráficos visualmente efectivos [Aplicar] **IV:** Implementar transformaciones de coordenadas para gráficos 2D y 3D [Implementar] **V:** Diseñar aplicaciones interactivas básicas utilizando dispositivos de entrada comunes y manejo de eventos [Diseñar] **VI:** Comparar técnicas avanzadas de renderizado como iluminación global y renderizado basado en física [Comparar] **VII:** Analizar el rol del hardware gráfico en acelerar tareas de renderizado [Analizar] **VIII:** Explicar cómo la percepción humana influye en el diseño de gráficos e interacción [Explicar]

Bibliografía: [Marschner and Shirley, 2021, Hearn et al., 2014]

5.2 Renderizado Aplicado y Técnicas, 16 hr(s)

Competencias: AG-C09

Tópicos: **I:** Algoritmos de rasterización: conversión por barrido (*scan conversion*), antialiasing **II:** Trazado de rayos (*ray tracing*): estructuras de aceleración, muestreo (*sampling*), sombreado **III:** Mapeo de texturas (*texture mapping*) y filtrado **IV:** Modelos de sombreado: Lambertiano, Phong, BRDF **V:** Técnicas avanzadas: mapeo de sombras (*shadow mapping*), mapeo ambiental (*environment mapping*), renderizado diferido (*deferred rendering*)

Objetivos de Aprendizaje **I:** Implementar algoritmos de rasterización para renderizado de primitivas [Implementar] **II:** Implementar trazado de rayos básico con estructuras de aceleración [Implementar] **III:** Aplicar técnicas de mapeo de texturas y filtrado para mejorar el realismo

visual [Aplicar] **IV:** Comparar diferentes modelos de sombreado y sus aplicaciones [Comparar] **V:** Implementar técnicas avanzadas de renderizado como mapeo de sombras o renderizado diferido [Implementar]

Bibliografía: [Marschner and Shirley, 2021, Hearn et al., 2014, Angel and Shreiner, 2014]

5.3 Modelado Geométrico, 16 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: I: Representaciones geométricas: polígonos, curvas, superficies, formas implícitas **II:** Técnicas de modelado: curvas spline, superficies de subdivisión, geometría sólida constructiva **III:** Modelado procedural y generación **IV:** Algoritmos geométricos: intersección, distancia, descomposición

Objetivos de Aprendizaje I: Describir diferentes representaciones geométricas y sus compensaciones (trade-offs) [Describir] **II:** Implementar técnicas de modelado como curvas spline o superficies de subdivisión [Implementar] **III:** Crear modelos procedurales usando algoritmos o reglas [Crear] **IV:** Aplicar algoritmos geométricos para resolver problemas como pruebas de intersección o procesamiento de mallas (mesh) [Aplicar]

Bibliografía: [Marschner and Shirley, 2021, Angel and Shreiner, 2014]

5.4 Visualización, 16 hr(s)

Competencias: AG-C09

Tópicos: I: Principios de visualización de datos: percepción, cognición y codificación efectiva **II:** Tipos de visualizaciones: científicas, de información, geográficas **III:** Técnicas de visualización para datos multivariados, temporales y jerárquicos **IV:** Visualización interactiva y exploración **V:** Evaluación de la efectividad de la visualización

Objetivos de Aprendizaje I: Aplicar principios de percepción y cognición para diseñar visualizaciones efectivas [Aplicar] **II:** Seleccionar técnicas de visualización apropiadas para diferentes tipos de datos y tareas [Evaluar] **III:** Implementar visualizaciones interactivas para exploración de datos [Implementar] **IV:** Evaluar la efectividad de una visualización usando métodos empíricos [Evaluar]

Bibliografía: [Marschner and Shirley, 2021]

5.5 Sombreado y Renderizado Avanzado, 12 hr(s)

Competencias: AG-C09

Tópicos: I: Sombreado avanzado: sombreado basado en física, dispersión subsuperficial (subsurface scattering) **II:** Iluminación global (global illumination): radiosidad, mapeo de fotones (photon mapping), trazado de caminos (path tracing) **III:** Técnicas de renderizado en tiempo real: nivel de detalle (level of detail), culling, oclusión **IV:** Programación GPU y sombreadores (shaders)

Objetivos de Aprendizaje I: Implementar modelos de sombreado avanzados para materiales realistas [Implementar] **II:** Comparar algoritmos de iluminación global y sus compensaciones [Comparar] **III:** Optimizar el renderizado en tiempo real usando técnicas como nivel de detalle y *culling* [Crear] **IV:** Desarrollar sombreadores GPU para efectos de renderizado personalizados [Crear]

Bibliografía: [Marschner and Shirley, 2021, Angel and Shreiner, 2014]

6. Bibliografía

[Angel and Shreiner, 2014] Angel, E. and Shreiner, D. (2014). *Interactive Computer Graphics: A Top-Down Approach with WebGL*. Pearson, 7th edition.

[Hearn et al., 2014] Hearn, D., Baker, M. P., and Carithers, W. (2014). *Computer Graphics with OpenGL*. Pearson, 4th edition.

[Marschner and Shirley, 2021] Marschner, S. and Shirley, P. (2021). *Fundamentals of Computer Graphics*. A K Peters/CRC Press, 5th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS292 Ingeniería de Software II (Obligatorio)

1. Información general

- **Semestre**: 7mo Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS291 Ingeniería de Software I (5to Sem-Pág 382)

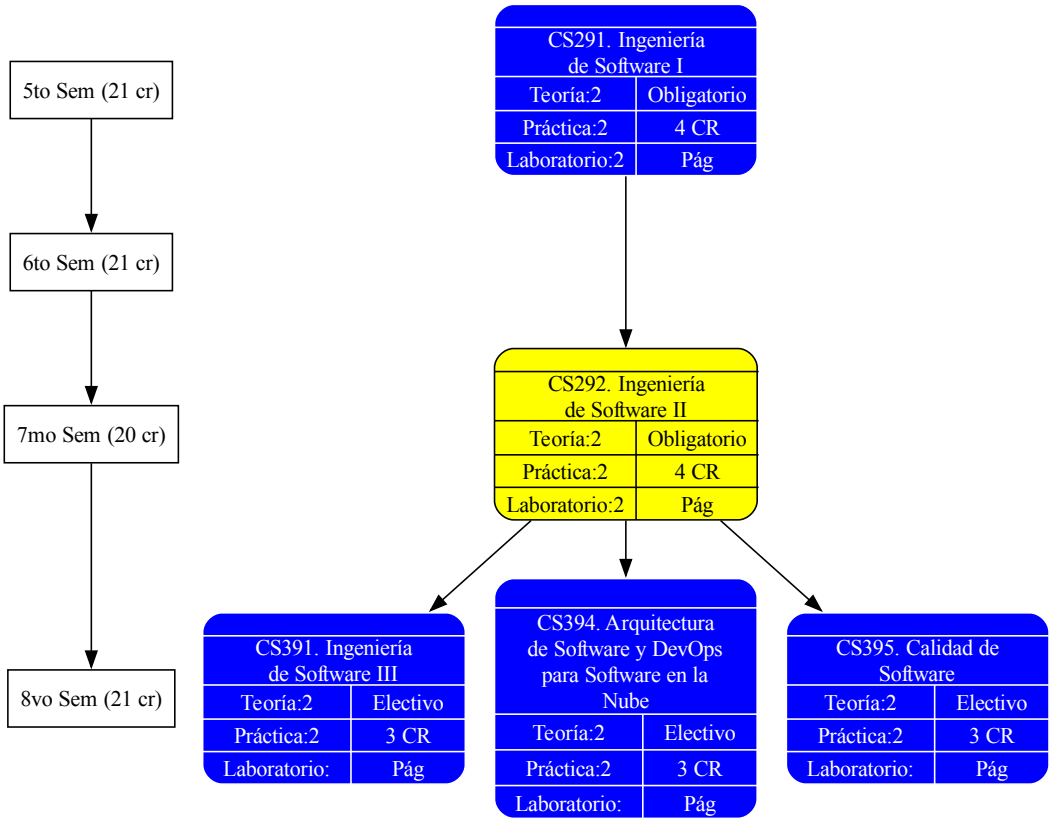

Figura 5.39: Mapa de Conexión. CS292 Ingeniería de Software II

2. Fundamentación

Ingeniería de Software II se basa en los principios fundamentales del primer curso, cambiando el enfoque hacia patrones de diseño avanzados, evolución de software y la gestión de proyectos a gran escala. Explora las complejidades de mantener sistemas heredados, asegurar la calidad del software mediante validación rigurosa, y la aplicación de técnicas profesionales de gestión de proyectos en entornos ágiles.

3. Objetivos del curso.

- Aplicar patrones de diseño orientados a objetos avanzados para resolver problemas arquitectónicos complejos.
- Dominar la gestión de configuración de software y control de versiones en entornos colaborativos.

- Comprender los principios de evolución y mantenimiento de software.
- Gestionar proyectos de software usando técnicas de estimación y análisis de riesgos.
- Implementar estrategias de prueba avanzadas y métricas de calidad.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

5. Contenido

5.1 Colaboración y Dinámica de Equipo, 14 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Comunicación efectiva, incluida la oral y escrita, así como formal (correo electrónico, documentos, comentarios, presentaciones) e informal (chat de equipo, reuniones). Comunicación Técnica y Profesional, Comunicación en Equipo y Cultura **II:** Causas comunes de conflicto en el equipo y enfoques para la resolución de conflictos. **III:** Programación cooperativa: *a*) Programación por parejas (pair programming) o Enjambre (swarming) *b*) Revisión de código *c*) Colaboración mediante control de versiones **IV:** Roles y responsabilidades en un equipo de software: Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo *a*) Ventajas del trabajo en equipo *b*) Riesgos y complejidad de dicha colaboración **V:** Procesos del equipo: responsabilidades para las tareas, estimación de esfuerzo, estructura de reuniones, cronograma de trabajo

Objetivos de Aprendizaje **I:** Seguir prácticas efectivas de comunicación en equipo [Analizar] **II:** Articular las fuentes, peligros y beneficios potenciales del conflicto en el equipo, especialmente enfocándose en el valor de estar en desacuerdo sobre ideas o propuestas sin insultar a las personas [Articular] **III:** Facilitar una estrategia de resolución de conflictos y resolución de problemas en un entorno de equipo [Analizar] **IV:** Colaborar efectivamente en desarrollo/programación cooperativa [Analizar] **V:** Proponer y delegar los roles y responsabilidades necesarios en un equipo de desarrollo de software [Proponer] **VI:** Redactar y seguir una agenda para una reunión de equipo [Componer] **VII:** Facilitar, a través de la participación en un proyecto de equipo, los elementos centrales de la formación de equipos, el establecimiento de una cultura de equipo saludable y la gestión de equipos, incluida la creación y ejecución de un plan de trabajo del equipo [Crear]

Bibliografía: [Sommerville, 2015, Pressman and Maxim, 2019]

5.2 Principios y Arquitectura de Software, 16 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Principios de diseño de sistemas. Confiabilidad del Sistema *a*) Niveles de abstracción (por ejemplo, diseño arquitectónico y diseño detallado) *b*) Separación de preocupaciones (separation of concerns) *c*) Ocultación de información (information hiding) *d*) Acoplamiento y cohesión (coupling and cohesion) **II:** Arquitectura de software. Confiabilidad del Sistema *a*) Paradigmas de diseño 1) Descomposición funcional de arriba hacia abajo/diseño por capas 2) Arquitectura orientada a datos 3) Análisis y diseño orientado a objetos 4) Diseño dirigido por eventos *b*) Arquitecturas estándar (por ejemplo, cliente-servidor y arquitecturas de microservicios incluyendo discusiones sobre REST, n-capas, tuberías y filtros, Modelo Vista Controlador) *c*) Identificar límites y dependencias de componentes **III:** Programación a gran escala vs programación a pequeña escala (programming in the large vs programming in the small). Confiabilidad del Sistema **IV:** Malos olores del código (code smells) y otras indicaciones de la calidad del código,

distintas de la corrección. Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad

Objetivos de Aprendizaje I: Identificar la arquitectura de software estándar de un diseño de alto nivel dado [Analizar] **II:** Seleccionar y usar un paradigma de diseño apropiado para diseñar un sistema de software simple y explicar cómo se han aplicado los principios de diseño de sistemas en este diseño [Evaluar] **III:** Adaptar un diseño de sistema defectuoso para que siga mejor principios como la separación de preocupaciones o la ocultación de información [Crear] **IV:** Identificar las dependencias entre un conjunto de componentes de software en un diseño arquitectónico [Analizar]

Bibliografía: [Gamma et al., 1994, Pressman and Maxim, 2019]

5.3 Prácticas de Codificación, 14 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Pruebas prácticas a pequeña escala Prácticas de Desarrollo de Software *a*) Pruebas unitarias *b*) Desarrollo guiado por pruebas (Test-driven development): esto es particularmente valioso para los estudiantes psicológicamente, ya que es mucho más fácil involucrarse constructivamente con el desafío de identificar entradas desafiantes para una API dada (casos límite, casos extremos) a priori. Si implementan primero, el instinto es a menudo evitar intentar hacer fallar su nueva creación, mientras que un enfoque de pruebas primero les da la satisfacción intelectual de detectar los casos problemáticos y luego ver cómo pasan más pruebas durante el proceso de desarrollo. **II:** Documentación Prácticas de Desarrollo de Software *a*) Documentación de la interfaz: describir los requisitos de la interfaz, potencialmente incluyendo contratos (formales o informales), pre y post condiciones, invariantes. *b*) La documentación de la implementación debe centrarse en partes de código complicadas y no obvias, ya sea porque el código usa características avanzadas del lenguaje o porque el comportamiento del código es complejo. (No agregar comentarios que reafirmen operaciones comunes/obvias y características simples del lenguaje.) 1) Aclarar el flujo de datos, cómputo, etc., centrándose en lo que el código es. 2) Identificar partes sutiles/complicadas del código y refactorizar para que sean autoexplicativas si es posible o proporcionar comentarios apropiados para aclarar. **III:** Estilo de codificación Prácticas de Desarrollo de Software *a*) Guías de estilo *b*) Comentarios *c*) Nomenclatura (*naming*) **IV:** "Mejores Prácticas" para la codificación: técnicas, patrones/idiomas, mecanismos para construir programas de calidad Prácticas de Desarrollo de Software, Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada *a*) Prácticas de codificación defensiva *b*) Prácticas y principios de codificación segura *c*) Usar mecanismos de manejo de excepciones para hacer los programas más robustos, tolerantes a fallos **V:** Depuración (debugging) Prácticas de Desarrollo de Software **VI:** Registro (logging) **VII:** Uso de bibliotecas y frameworks desarrollados por otros Prácticas de Desarrollo de Software

Objetivos de Aprendizaje I: Escribir pruebas unitarias apropiadas para un componente pequeño (varias funciones, un solo tipo, etc.) [Escribir] **II:** Escribir comentarios apropiados de interfaz y (si es necesario) de implementación para un componente pequeño [Escribir] **III:** Describir técnicas, patrones de codificación y mecanismos para implementar diseños para lograr propiedades deseadas como confiabilidad, eficiencia y robustez [Describir] **IV:** Escribir código robusto utilizando mecanismos de manejo de excepciones [Escribir] **V:** Describir prácticas de codificación segura y defensiva [Describir] **VI:** Seleccionar y usar un estándar de codificación definido en un proyecto de software pequeño [Usar] **VII:** Comparar y contrastar estrategias de integración, incluyendo integración de arriba hacia abajo (top-down), de abajo hacia arriba (bottom-up) y sándwich [Comparar] **VIII:** Describir el proceso de analizar e implementar cambios en la base de código desarrollada para un proyecto específico [Describir] **IX:** Describir el proceso de analizar e implementar cambios en una base de código existente grande [Describir]

Bibliografía: [Sommerville, 2015]

5.4 Conceptos de verificación y validación, 10 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Conceptos de verificación y validación *a*) Verificación: ¿Estamos construyendo la cosa correctamente? *b*) Validación: ¿Construimos la cosa correcta? **II:** Por qué importan las pruebas: ¿El componente sigue siendo funcional a medida que el código evoluciona? **III:** Objetivos de las pruebas *a*) Usabilidad *b*) Confiabilidad *c*) Conformidad con la especificación *d*) Rendimiento *e*) Seguridad

Objetivos de Aprendizaje **I:** Explicar por qué las pruebas son importantes [Explicar] **II:** Distinguir entre validación y verificación de programas [Distinguir] **III:** Describir diferentes objetivos de las pruebas [Describir]

Bibliografía: [Pressman and Maxim, 2019]

5.5 Herramientas y Análisis de Pruebas, 4 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Herramientas y automatización de verificación y validación *a*) Análisis estático *b*) Cobertura de código *c*) Fuzzing *d*) Análisis dinámico y contención de fallos (sanitizadores, etc.) *e*) Registro y seguimiento de fallos **II:** Planificación y generación de pruebas *a*) Estimación de fallos y terminación de pruebas, incluyendo siembra de defectos *b*) Uso de números aleatorios y pseudoaleatorios en las pruebas **III:** Pruebas de sistemas asíncronos, paralelos y concurrentes **IV:** Verificación y validación de artefactos que no son código (documentación, materiales de capacitación) **V:** Desarrollo guiado por comportamiento (BDD) *a*) Expresar criterios de aceptación como escenarios Gherkin Given/When/Then *b*) Ejecutar escenarios BDD como pruebas automatizadas de extremo a extremo **VI:** Pruebas de mutación y análisis estático de calidad de código *a*) Pruebas de mutación para evaluar la efectividad de una suite de pruebas *b*) Detección estática de *code smells* y complejidad ciclomática excesiva

Objetivos de Aprendizaje **I:** Describir y comparar diferentes herramientas para verificación y validación [Describir] **II:** Automatizar el uso de diferentes herramientas en un proyecto de software pequeño [Diseñar] **III:** Explicar cómo y cuándo se deben usar números aleatorios en las pruebas [Explicar] **IV:** Describir enfoques para la estimación de fallos [Describir] **V:** Estimar el número de fallos en una aplicación de software pequeña basándose en la densidad de fallos y la siembra de fallos [Estimar] **VI:** Describir técnicas y problemas con las pruebas de software asíncrono, concurrente y paralelo [Describir] **VII:** Crear un plan de pruebas para un segmento de código de tamaño mediano que contenga código asíncrono, concurrente y/o paralelo, incluyendo una matriz de pruebas y generación de datos y entradas de prueba [Crear] **VIII:** Describir técnicas para la verificación y validación de artefactos que no son código [Describir] **IX:** Escribir escenarios Gherkin Given/When/Then para los criterios de aceptación de una funcionalidad y ejecutarlos como pruebas automatizadas de extremo a extremo [Escribir] **X:** Analizar el puntaje de mutación de una suite de pruebas y refactorizar las pruebas para eliminar los mutantes sobrevivientes [Analizar]

Bibliografía: [Sommerville, 2015, Pressman and Maxim, 2019]

5.6 Pruebas de Rendimiento y Evaluación Comparativa, 4 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Pruebas de rendimiento y evaluación comparativa *a*) Rendimiento (*throughput*) y latencia *b*) Degradación bajo carga (pruebas de estrés, manejo FIFO vs LIFO de solicitudes) *c*) Aceleración (*speedup*) y escalado (*scaling*) 1) Ley de Amdahl 2) Ley de Gustafson 3) Escalado suave (*soft*) y débil (*weak*) *d*) Identificar y medir figuras de mérito (*figures of merit*) *e*) Cuellos de botella de rendimiento comunes 1) Limitado por cómputo (*compute-bound*) 2) Limitado por ancho de banda de memoria 3) Limitado por latencia *f*) Métodos estadísticos y mejores prácticas

para la evaluación comparativa (benchmarking) 1) Estimación de la incertidumbre 2) Intervalos de confianza *g*) Análisis y presentación (gráficos, etc.) *h*) Técnicas de medición de tiempo (*timing*)

Objetivos de Aprendizaje I: Describir el rendimiento (*throughput*) y la latencia y proporcionar ejemplos de cada uno [Describir] **II:** Explicar la aceleración (speedup) y las diferentes formas de escalado (scaling) y cómo se calculan [Explicar] **III:** Describir cuellos de botella de rendimiento comunes [Describir] **IV:** Describir métodos estadísticos y mejores prácticas para la evaluación comparativa (benchmarking) de software [Describir] **V:** Explicar técnicas y desafíos para medir el tiempo al construir una evaluación comparativa (benchmark) [Explicar] **VI:** Identificar las figuras de mérito, construir y ejecutar una evaluación comparativa (benchmark) y analizar estadísticamente y visualizar los resultados para un proyecto de software pequeño [Analizar]

Bibliografía: [Bondi, 2015, Gregg, 2020]

5.7 Conceptos de Confiabilidad, 10 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Concepto de confiabilidad como probabilidad de fallo o tiempo medio entre fallos, y fallas (faults) como causa de fallos (failures) **II:** Identificar requisitos de confiabilidad para diferentes tipos de software **III:** Fallos de software causados por defectos/errores (bugs), por lo que para alta confiabilidad el objetivo es tener un mínimo de defectos - inyectando menos defectos (mejor capacitación, educación, planificación) y eliminando la mayoría de los defectos inyectados (pruebas, revisión de código, etc.) **IV:** Confiabilidad del software, confiabilidad del sistema y comportamiento de fallos **V:** Ciclo de inyección y eliminación de defectos, y diferentes enfoques para la eliminación de defectos **VI:** Comparar el enfoque de "presupuesto de error" (error budget) para la confiabilidad con el enfoque "libre de errores" (error-free) e identificar dominios donde cada uno es relevante.

Objetivos de Aprendizaje I: Describir cómo determinar el nivel de confiabilidad requerido por un sistema de software [Describir] **II:** Explicar los problemas que existen para lograr niveles muy altos de confiabilidad [Explicar] **III:** Comprender enfoques para minimizar fallas que pueden aplicarse en cada etapa del ciclo de vida del software [Explicar]

Bibliografía: [Sommerville, 2015]

6. Bibliografía

[Bondi, 2015] Bondi, A. B. (2015). *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice*. Addison-Wesley, Upper Saddle River, NJ.

[Gamma et al., 1994] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

[Gregg, 2020] Gregg, B. (2020). *Systems Performance: Enterprise and the Cloud*. Addison-Wesley Professional, Boston, MA, 2nd edition.

[Pressman and Maxim, 2019] Pressman, R. S. and Maxim, B. (2019). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 9th edition.

[Sommerville, 2015] Sommerville, I. (2015). *Software Engineering*. Pearson, 10th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS2H1 Experiencia de Usuario (UX) (Obligatorio)

1. Información general

- **Semestre**: 7mo Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - CS291 Ingeniería de Software I (5to Sem-Pág 382)

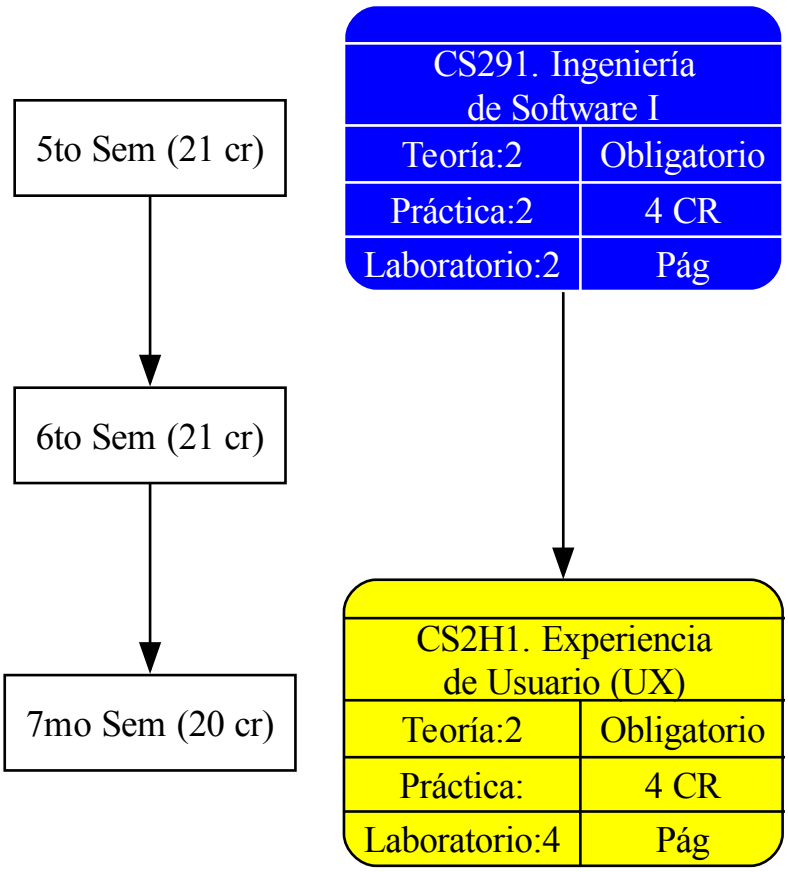

Figura 5.40: Mapa de Conexión. CS2H1 Experiencia de Usuario (UX)

2. Fundamentación

Este curso se enfoca en el diseño, evaluación e implementación de sistemas informáticos interactivos para uso humano. Más allá de los principios clásicos de HCI, aborda los desafíos emergentes de la interacción con sistemas de inteligencia artificial, agentes inteligentes y software de procesamiento de lenguaje natural. Los estudiantes desarrollarán competencias en diseño centrado en el usuario, accesibilidad, responsabilidad ética y evaluación de interfaces tradicionales y conversacionales basadas en IA, adquiriendo habilidades esenciales para diseñar tecnología responsable y centrada en las personas.

3. Objetivos del curso.

- Comprender las características cognitivas, físicas y emocionales del ser humano relevantes para el diseño de sistemas interactivos.
- Aplicar principios de responsabilidad ética y diseño centrado en valores, incluyendo equidad algorítmica y sesgo.
- Aplicar estándares de accesibilidad y diseño inclusivo para poblaciones y capacidades diversas.
- Ejecutar métodos de evaluación de usabilidad para iterar y mejorar interfaces de usuario.
- Diseñar sistemas interactivos mediante procesos de diseño centrado en el usuario e iterativo, para múltiples plataformas y dominios.
- Analizar las dimensiones sociales, éticas y profesionales de la tecnología interactiva.
- Comprender arquitecturas de agentes y aplicar principios de interacción humano-agente, incluyendo asistentes de IA y sistemas autónomos.
- Aplicar técnicas de procesamiento de lenguaje natural para diseñar y evaluar interfaces humano-software basadas en modelos de lenguaje.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

5. Contenido

5.1 Comprender al Usuario: Objetivos individuales e interacciones con otros, 8 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Capacidades y limitaciones humanas: percepción, cognición, habilidades motoras **II:** Características del usuario: demografía, experiencia, contexto cultural **III:** Objetivos, tareas y contextos de uso del usuario **IV:** Aspectos sociales y organizacionales de la interacción **V:** Modelos y teorías cognitivas relevantes para HCI **VI:** Aspectos emocionales y afectivos de la interacción **VII:** Métodos de investigación de usuarios: entrevistas, encuestas, observaciones

Objetivos de Aprendizaje **I:** Describir las capacidades y limitaciones humanas que afectan el diseño de interacción [Describir] **II:** Analizar cómo las características del usuario influyen en la interacción con sistemas informáticos [Analizar] **III:** Identificar objetivos y tareas del usuario en contextos de uso específicos [Analizar] **IV:** Explicar factores sociales y organizacionales que impactan la adopción y uso de la tecnología [Explicar] **V:** Aplicar modelos cognitivos para predecir y explicar el comportamiento del usuario [Aplicar] **VI:** Considerar factores emocionales y afectivos en el diseño de interacción [Analizar] **VII:** Realizar investigación de usuarios usando métodos apropiados como entrevistas u observaciones [Evaluar]

Bibliografía: [Norman, 2013, Johnson, 2020]

5.2 Responsabilidad y Rendición de Cuentas en el Diseño, 8 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Consideraciones éticas en HCI: equidad, transparencia, rendición de cuentas **II:** Diseño para valores y diseño sensible a valores **III:** Innovación responsable y ética del diseño **IV:** Rendición de cuentas algorítmica y sesgo en sistemas interactivos **V:** Responsabilidad profesional en la práctica de HCI **VI:** Consideraciones legales y regulatorias para sistemas interactivos

Objetivos de Aprendizaje **I:** Identificar problemas éticos en el diseño e implementación de HCI [Analizar] **II:** Aplicar enfoques de diseño sensible a valores al desarrollo tecnológico [Aplicar] **III:** Discutir prácticas de innovación responsable en HCI [Debatir] **IV:** Analizar la rendición

de cuentas algorítmica y el sesgo en sistemas interactivos [Analizar] **V:** Describir responsabilidades profesionales en la práctica de HCI [Describir] **VI:** Explicar consideraciones legales y regulatorias relevantes para sistemas interactivos [Explicar]

Bibliografía: [Shneiderman, 2022, Norman, 2013]

5.3 Accesibilidad y Diseño Inclusivo, 8 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Principios y pautas de accesibilidad **II:** Discapacidades y tecnologías de asistencia **III:** Diseño inclusivo y principios de diseño universal **IV:** Métodos de evaluación para accesibilidad **V:** Diseñar para diversas capacidades y contextos **VI:** Estándares de accesibilidad y cumplimiento

Objetivos de Aprendizaje I: Aplicar principios de accesibilidad al diseño de sistemas interactivos [Aplicar] **II:** Describir cómo diferentes discapacidades afectan la interacción con la tecnología [Describir] **III:** Diseñar sistemas inclusivos usando principios de diseño universal [Diseñar] **IV:** Realizar evaluaciones de accesibilidad de sistemas interactivos [Evaluar] **V:** Crear diseños que acomoden diversas capacidades y contextos de uso [Crear] **VI:** Garantizar el cumplimiento con estándares y regulaciones de accesibilidad [Evaluar]

Bibliografía: [Johnson, 2020]

5.4 Evaluación del Diseño, 8 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Principios y heurísticas de usabilidad **II:** Métodos de evaluación: evaluación heurística, recorrido cognitivo (cognitive walkthrough) **III:** Pruebas con usuarios: planificación, ejecución, análisis **IV:** Métodos avanzados de evaluación: experimentos controlados, estudios de campo **V:** Métricas y medición de UX **VI:** Métodos de evaluación analítica: modelado y simulación

Objetivos de Aprendizaje I: Aplicar principios y heurísticas de usabilidad para evaluar diseños [Aplicar] **II:** Realizar evaluaciones heurísticas y recorridos cognitivos [Evaluar] **III:** Planificar, conducir y analizar sesiones de pruebas con usuarios [Planear] **IV:** Diseñar y conducir experimentos controlados para investigación en HCI [Diseñar] **V:** Medir la experiencia de usuario usando métricas apropiadas [Evaluar] **VI:** Usar métodos analíticos para modelar y predecir el desempeño del usuario [Usar]

Bibliografía: [Norman, 2013, Johnson, 2020]

5.5 Diseño de Sistemas, 8 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Proceso de diseño: diseño centrado en el usuario, diseño iterativo **II:** Diseño de interacción: conceptos, principios, patrones **III:** Técnicas de prototipado: baja fidelidad, alta fidelidad, interactivo **IV:** Arquitectura de información y diseño de navegación **V:** Técnicas avanzadas de interacción: gestos, voz, multimodal **VI:** Diseño para dominios específicos: móvil, web, computación ubicua **VII:** Sistemas de computación colaborativa y social

Objetivos de Aprendizaje I: Aplicar procesos de diseño centrado en el usuario para desarrollar sistemas interactivos [Aplicar] **II:** Diseñar interacciones usando principios y patrones establecidos [Diseñar] **III:** Crear prototipos en diferentes niveles de fidelidad para exploración de diseño [Crear] **IV:** Estructurar información y navegación para una interacción efectiva del usuario [Estructurar] **V:** Implementar técnicas avanzadas de interacción como interfaces de gestos o voz [Implementar] **VI:** Diseñar sistemas interactivos para dominios y plataformas específicos [Diseñar] **VII:** Diseñar sistemas colaborativos que apoyen la interacción grupal [Diseñar]

Bibliografía: [Johnson, 2020, Norman, 2013]

5.6 Sociedad, ética y Profesión, 8 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Impacto social de las tecnologías interactivas **II:** Problemas éticos en la investigación y práctica de HCI **III:** Práctica profesional en HCI **IV:** Consideraciones culturales e interculturales en HCI **V:** Sostenibilidad y HCI **VI:** Tendencias futuras y desafíos en HCI

Objetivos de Aprendizaje **I:** Analizar el impacto social de las tecnologías interactivas en individuos y sociedad [Analizar] **II:** Abordar problemas éticos en proyectos de investigación y diseño de HCI [Evaluar] **III:** Describir prácticas profesionales y trayectorias profesionales en HCI [Describir] **IV:** Diseñar sistemas que tomen en cuenta diferencias y contextos culturales [Diseñar] **V:** Incorporar consideraciones de sostenibilidad en el diseño de HCI [Integrar] **VI:** Discutir tendencias emergentes y desafíos futuros en HCI [Debatir]

Bibliografía: [Shneiderman, 2022, Norman, 2013]

5.7 Agentes y Sistemas Cognitivos, 8 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Arquitecturas de agentes (por ejemplo, reactivas, en capas, cognitivas) **II:** Teoría de agentes (incluyendo formalismos matemáticos) **III:** Racionalidad, Teoría de juegos: *a*) Agentes decisorios-teóricos *b*) Procesos de decisión de Markov (MDP) *c*) Algoritmos de bandidos **IV:** Agentes de software, asistentes personales y acceso a información: *a*) Agentes colaborativos *b*) Agentes de recolección de información *c*) Agentes creíbles (personajes sintéticos, modelado de emociones en agentes) **V:** Agentes de aprendizaje **VI:** Sistemas cognitivos: *a*) Arquitecturas cognitivas (por ejemplo, ACT-R, SOAR, ICARUS, FORR) *b*) Capacidades (por ejemplo, percepción, toma de decisiones, predicción, mantenimiento del conocimiento) *c*) Representación, organización, utilización, adquisición y refinamiento del conocimiento *d*) Aplicaciones y evaluación de sistemas cognitivos **VII:** Sistemas multiagente: *a*) Agentes colaboradores *b*) Equipos de agentes *c*) Agentes competitivos (por ejemplo, subastas, votación) *d*) Sistemas de enjambre y modelos biológicamente inspirados *e*) Aprendizaje multiagente **VIII:** Interacción humano-agente: *a*) Metodologías de comunicación (verbal y no verbal) *b*) Problemas prácticos *c*) Aplicaciones: 1) Agentes comerciales, gestión de cadena de suministro 2) Problemas éticos de las interacciones de IA con humanos 3) Regulación y requisitos legales de los sistemas de IA para interactuar con humanos

Objetivos de Aprendizaje **I:** Caracterizar y contrastar las arquitecturas estándar de agentes [Evaluar] **II:** Describir las aplicaciones de la teoría de agentes a dominios como agentes de software, asistentes personales y agentes creíbles, y discutir las implicaciones éticas asociadas [Describir] **III:** Describir los paradigmas principales utilizados por los agentes de aprendizaje [Describir] **IV:** Demostrar usando ejemplos apropiados cómo los sistemas multiagente soportan la interacción de agentes [Demostrar] **V:** Construir un agente inteligente usando una arquitectura cognitiva establecida (ACT-R, SOAR) para resolver un problema específico [Aplicar]

Bibliografía: [Russell and Norvig, 2021, Shneiderman, 2022]

5.8 Procesamiento del Lenguaje Natural, 8 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Gramáticas deterministas y estocásticas **II:** Algoritmos de análisis sintáctico: *a*) CFGs y analizadores de tabla (por ejemplo, CYK) *b*) CFGs probabilísticos y CYK ponderado **III:** Representación del significado/Semántica: *a*) Representaciones del conocimiento basadas en lógica *b*) Roles semánticos *c*) Representaciones temporales *d*) Creencias, deseos e intenciones **IV:** Métodos basados en corpus **V:** N-gramas y HMMs **VI:** Suavizado y retroceso **VII:** Ejemplos de uso: Etiquetado POS y morfología **VIII:** Recuperación de información: *a*) Modelo de espacio vectorial: 1) TF & IDF *b*) Precisión y exhaustividad **IX:** Extracción de información

X: Traducción de lenguaje **XI:** Clasificación de texto, categorización: *a*) Modelo de bolsa de palabras **XII:** Aprendizaje profundo para PLN: *a*) RNNs *b*) Transformers *c*) Incrustaciones multimodales (por ejemplo, imágenes + texto) *d*) Modelos de lenguaje generativos

Objetivos de Aprendizaje I: Definir y contrastar gramáticas deterministas y estocásticas, proporcionando ejemplos para mostrar la adecuación de cada una [Definir] **II:** Simular, aplicar o implementar algoritmos clásicos y estocásticos para analizar lenguaje natural [Aplicar] **III:** Identificar los desafíos de representar el significado [Analizar] **IV:** Listar las ventajas de usar corpus estándar. Identificar ejemplos de corpus actuales para una variedad de tareas de PLN [Listar/Enumerar] **V:** Identificar técnicas para recuperación de información, traducción de lenguaje y clasificación de texto [Analizar] **VI:** Implementar una transformación TF/IDF, usarla para extraer características de un corpus, y entrenar un algoritmo de aprendizaje automático estándar usando esas características para hacer clasificación de texto [Aplicar]

Bibliografía: [Jurafsky and Martin, 2024, Russell and Norvig, 2021]

6. Bibliografía

[Johnson, 2020] Johnson, J. (2020). *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufmann, 3rd edition.

[Jurafsky and Martin, 2024] Jurafsky, D. and Martin, J. H. (2024). *Speech and Language Processing*. Prentice Hall, 3rd edition.

[Norman, 2013] Norman, D. (2013). *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books, 2nd edition.

[Russell and Norvig, 2021] Russell, S. and Norvig, P. (2021). *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition.

[Shneiderman, 2022] Shneiderman, B. (2022). *Human-Centered AI*. Oxford University Press, 1st edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI264 Aprendizaje Profundo (Obligatorio)

1. Información general

- **Semestre**: 7mo Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - AI263 Introducción al Aprendizaje de Máquina (6to Sem-Pág 415)

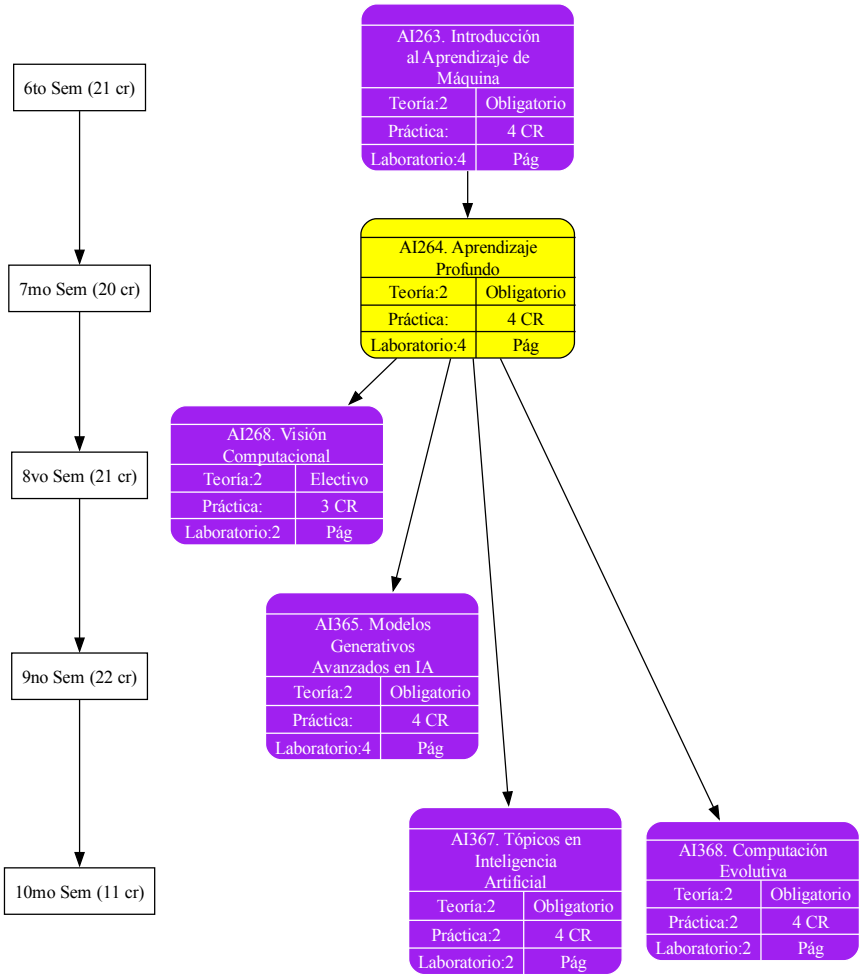

Figura 5.41: Mapa de Conexión. AI264 Aprendizaje Profundo

2. Fundamentación

Este curso cubre los fundamentos del aprendizaje profundo moderno, incluyendo redes neuronales convolucionales, arquitecturas de transformers, y técnicas de entrenamiento para sistemas de inteligencia artificial avanzada.

3. Objetivos del curso.

- Implementar redes neuronales profundas usando frameworks modernos (PyTorch/TensorFlow)

- Diseñar arquitecturas para problemas específicos (visión, lenguaje, etc.)
- Analizar limitaciones éticas de modelos profundos

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Evaluar)

5. Contenido

5.1 Entrenamiento y Optimización de Redes Neuronales, 10 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Backpropagation moderno: grafos computacionales y diferenciación automática **II:** Optimizadores avanzados: momentum, RMSProp, Adam, y programación de la tasa de aprendizaje

Objetivos de Aprendizaje **I:** Implementar una red neuronal profunda desde cero, incluyendo su paso de backpropagation [Aplicar] **II:** Seleccionar y ajustar un optimizador y su programación de tasa de aprendizaje para la estabilidad del entrenamiento [Aplicar]

Bibliografía: [Goodfellow et al., 2016]

5.2 Técnicas de Regularización para Redes Profundas, 5 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Regularización por Dropout **II:** Normalización por lotes (Batch Normalization)

Objetivos de Aprendizaje **I:** Diagnosticar el sobreajuste en un entrenamiento usando diagnósticos de regularización [Analizar] **II:** Aplicar Dropout y Batch Normalization para mejorar la generalización de un modelo [Aplicar]

Bibliografía: [Goodfellow et al., 2016]

5.3 Arquitecturas de Redes Convolucionales y Residuales, 9 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Arquitecturas de redes neuronales convolucionales profundas **II:** Conexiones residuales y ResNets

Objetivos de Aprendizaje **I:** Diseñar una arquitectura CNN profunda para una tarea de visión dada [Aplicar] **II:** Adaptar una arquitectura residual a un dominio de visión específico [Aplicar]

Bibliografía: [He et al., 2016]

5.4 Arquitecturas Transformer y Modelos Generativos, 16 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Transformers y el mecanismo de atención **II:** Modelos generativos: autoencoders variacionales (VAEs) y redes generativas adversariales (GANs)

Objetivos de Aprendizaje **I:** Entrenar un modelo transformer en un dataset real [Aplicar] **II:** Analizar y contrastar arquitecturas de modelos generativos VAE y GAN [Analizar]

Bibliografía: [Vaswani et al., 2017]

6. Bibliografía

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *CVPR*, pages 770–778.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

FG211 Ética Profesional (Obligatorio)

1. Información general

- **Semestre**: 7mo Sem. **Créditos**: 2
- **Horas del curso**: **Teoría**: 1 horas; **Práctica**: 2 horas;
- **Prerrequisitos**: Ninguno

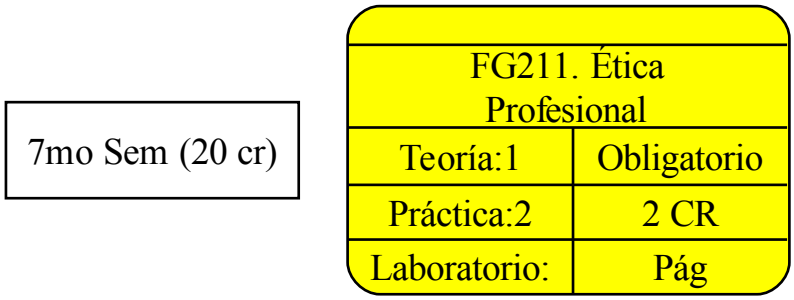

Figura 5.42: Mapa de Conexión. FG211 Ética Profesional

2. Fundamentación

Este curso adapta los principios éticos profesionales al campo de la Inteligencia Artificial, integrando los códigos de ética de ACM e IEEE con desafíos específicos de IA. Combina fundamentos regulatorios (GDPR) con casos actuales de impacto social.

3. Objetivos del curso.

- Aplicar los códigos de ética de ACM/IEEE a problemas de IA.
- Analizar el cumplimiento de regulaciones (GDPR) en sistemas autónomos.
- Diseñar protocolos éticos para proyectos de IA.

4. Competencias Generales (*Outcomes*)

AG-C02) Ética: Aplica principios éticos y se compromete con la ética profesional y las normas de la práctica profesional de la computación. (Usar)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Usar)

5. Contenido

5.1 Fundamentos Éticos en IA, 10 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: **I:** Comparación: Códigos de ética ACM vs IEEE para IA. **II:** Principios de transparencia y justicia algorítmica.

Objetivos de Aprendizaje I: Explicar diferencias entre marcos éticos [Familiarizarse]. **II:** Evaluar compliance en casos reales [Evaluar].

Bibliografía: [ACM, 2018, IEEE, 2020]

5.2 Privacidad y Regulaciones, 14 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: I: GDPR aplicado a ML: derecho a explicación. **II:** Auditoría ética de datasets (ej: ImageNet).

Objetivos de Aprendizaje I: Implementar checklist GDPR para modelos [Usar]. **II:** Identificar violaciones en estudios de caso [Evaluar].

Bibliografía: [Voigt and von dem Bussche, 2018, Müller, 2021]

5.3 Sesgos y Equidad, 18 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: I: Métricas de equidad (paridad demográfica, igualdad de oportunidades). **II:** Herramientas: IBM Fairness 360, Google What-If.

Objetivos de Aprendizaje I: Medir sesgos en modelos con Python [Usar]. **II:** Proponer estrategias de mitigación [Evaluar].

Bibliografía: [Force, 2020, Mehrabi et al., 2021]

5.4 Responsabilidad en Sistemas Autónomos, 18 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: I: Atribución legal en fallos de IA (ej: vehículos autónomos). **II:** Normas ISO para IA confiable.

Objetivos de Aprendizaje I: Redactar cláusulas de responsabilidad [Evaluar]. **II:** Analizar conflictos legales [Familiarizarse].

Bibliografía: [Bostrom, 2014, Committee, 2019]

6. Bibliografía

[ACM, 2018] ACM (2018). Acm code of ethics and professional conduct. Technical report, ACM.

[Bostrom, 2014] Bostrom, N. (2014). *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press.

[Committee, 2019] Committee, I. E. (2019). Ethical dilemmas in engineering. Technical report, IEEE.

[Force, 2020] Force, A. E. T. (2020). Case studies in computing and society. Technical report, ACM.

[IEEE, 2020] IEEE (2020). Ieee code of ethics. Technical report, IEEE.

[Mehrabi et al., 2021] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys*, 54(6).

[Müller, 2021] Müller, V. C. (2021). *Ethics of Artificial Intelligence and Robotics*. Cambridge University Press.

[Voigt and von dem Bussche, 2018] Voigt, P. and von dem Bussche, A. (2018). *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

FG350 Liderazgo y Desempeño (Obligatorio)

1. Información general

- **Semestre**: 7mo Sem. **Créditos**: 2
- **Horas del curso**: **Teoría**: 2 horas;
- **Prerrequisitos**:
  - FG106 Teatro (5to Sem-Pág 393)

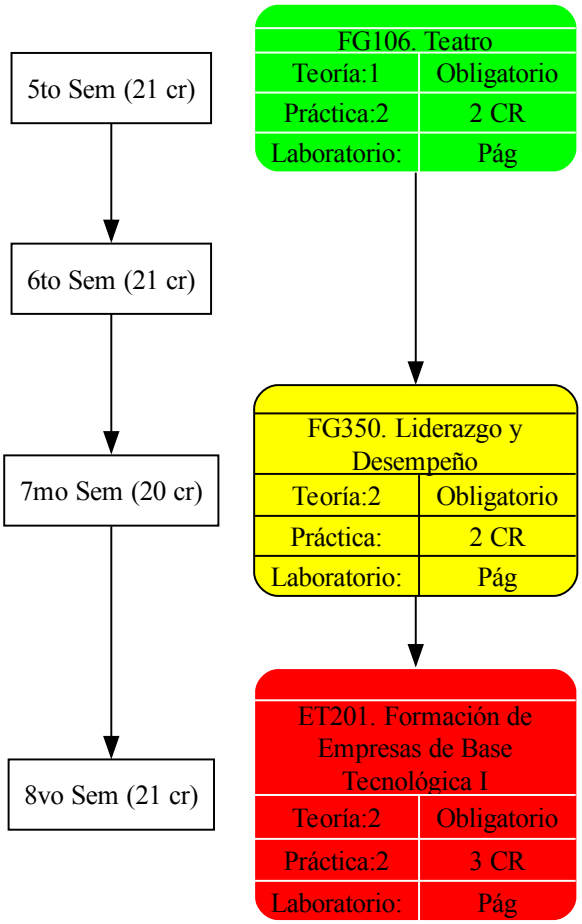

Figura 5.43: Mapa de Conexión. FG350 Liderazgo y Desempeño

2. Fundamentación

Las organizaciones contemporáneas exigen de sus integrantes el ejercicio efectivo del liderazgo: asumir con eficacia y espíritu de servicio los retos que se les encomiendan, en la búsqueda de una sociedad más justa y reconciliada. Este desafío pasa por formar personas con un recto conocimiento de sí mismas, con capacidad de juzgar objetivamente la realidad y de proponer orientaciones que modifiquen positivamente el entorno. El curso combina fundamentos teóricos con el desarrollo vivencial de competencias, capacitando al estudiante para liderar equipos con integridad, coherencia personal y genuino servicio a los demás.

3. Objetivos del curso.

- Comprender los fundamentos teóricos del liderazgo y su dimensión ética, integrándolos a un proceso de autoconocimiento.
- Identificar y desarrollar las competencias y virtudes personales necesarias para el ejercicio eficaz del liderazgo.
- Adquirir habilidades prácticas para la gestión, motivación y dirección de equipos de alto desempeño.
- Contribuir a la construcción de una sociedad más justa a través del liderazgo con sentido de servicio y responsabilidad social.

4. Competencias Generales (*Outcomes*)

AG-C02) Ética: Aplica principios éticos y se compromete con la ética profesional y las normas de la práctica profesional de la computación. (Usar)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Usar)

5. Contenido

5.1 Fundamentos del Liderazgo, 15 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: **I:** Definición y concepto de liderazgo. **II:** Principales teorías del liderazgo: transaccional, transformacional y situacional. **III:** Visión integral del ser humano: dimensiones corporal, afectiva y espiritual. **IV:** Motivos de la acción humana: extrínsecos, intrínsecos y trascendentes. **V:** La práctica de la virtud en el ejercicio del liderazgo. **VI:** El líder como modelo: ética y coherencia personal. **VII:** Estilos de liderazgo y su aplicación en distintos contextos.

Objetivos de Aprendizaje **I:** Analizar y comprender las bases teóricas del ejercicio del liderazgo [Familiarizarse]. **II:** Asumir la actitud correcta para llevar el liderazgo a la práctica [Familiarizarse]. **III:** Iniciar un proceso de autoconocimiento orientado a descubrir rasgos de liderazgo en sí mismo [Familiarizarse]. **IV:** Reconocer la dimensión ética del liderazgo y su impacto en el entorno personal y social [Familiarizarse].

Bibliografía: [Cardona and Lombardi, 2002, Ferreiro and Alcázar, 2009, Dianine-Havard, 2007, D'Souza, 1994]

5.2 Formación de Virtudes y Competencias, 15 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: **I:** Teoría de las competencias y su aplicación al liderazgo. **II:** Reconocimiento e inventario de competencias personales. **III:** Plan de desarrollo personal: diagnóstico, metas y seguimiento. **IV:** Modelos mentales y su influencia en el comportamiento. **V:** Necesidades y perfiles emocionales. **VI:** Vicios motivacionales: identificación y superación. **VII:** Inteligencia emocional aplicada al liderazgo. **VIII:** La toma de decisiones bajo presión y ambigüedad.

Objetivos de Aprendizaje **I:** Conocer y desarrollar competencias de liderazgo centradas en la eficacia y el servicio [Familiarizarse]. **II:** Reconocer las tendencias personales y grupales necesarias para el ejercicio del liderazgo [Familiarizarse]. **III:** Elaborar un plan de desarrollo personal de competencias de liderazgo [Usar].

Bibliografía: [Cardona and Wilkinson, 2009, Huete, 2008, Cardona and Lombardi, 2002, Chinchilla and Moragas, 2007, Goleman, 2006]

5.3 Gestión y Dirección de Equipos, 18 hr(s)

Competencias: AG-C02,AG-C04

Tópicos: I: La relación personal con el equipo: confianza y comunicación. **II:** Liderazgo integral: misión, visión y valores compartidos. **III:** Delegación y acompañamiento: el arte de motivar y delegar. **IV:** Coaching y mentoring como herramientas de desarrollo. **V:** Resolución de conflictos y dinámica grupal. **VI:** Las cinco disfunciones de un equipo y cómo superarlas. **VII:** Fundamentos de unidad: cohesión y cultura organizacional. **VIII:** El liderazgo de servicio y su impacto en la organización.

Objetivos de Aprendizaje I: Desarrollar habilidades para el trabajo en equipo eficaz y cohesionado [Familiarizarse]. **II:** Aplicar técnicas de liderazgo situacional para dirigir equipos en distintos contextos [Usar]. **III:** Diseñar estrategias para la resolución de conflictos y la mejora del desempeño grupal [Usar].

Bibliografía: [Goleman, 2006, Cardona and na, 2008, Hersey et al., 2007, Hunsaker, 2001, Hawkins, 2011, Lencioni, 2002]

6. Bibliografía

[Cardona and Lombardi, 2002] Cardona, P. and Lombardi, P. G. (2002). *Cómo desarrollar las Competencias de Liderazgo*. PAD, Lima, 3rd edition.

[Cardona and na, 2008] Cardona, P. and na, C. R. P. (2008). *Dirección por misiones: Cómo generar empresas de alto rendimiento*. Deusto.

[Cardona and Wilkinson, 2009] Cardona, P. and Wilkinson, H. (2009). *Creciendo como Líder*. Ediciones Universidad de Navarra EUNSA.

[Chinchilla and Moragas, 2007] Chinchilla, N. and Moragas, M. (2007). *Dueños de Nuestro Destino*. Editorial Ariel.

[Dianine-Havard, 2007] Dianine-Havard, A. (2007). *Virtuous Leadership: An Agenda for Personal Excellence*. Scepter Publishers.

[D'Souza, 1994] D'Souza, A. (1994). *Discover Your Leadership Style*. St. Pauls.

[Ferreiro and Alcázar, 2009] Ferreiro, P. and Alcázar, M. (2009). *Gobierno de Personas en la Empresa*. Ediciones Universidad de Navarra EUNSA.

[Goleman, 2006] Goleman, D. (2006). *Emotional Intelligence: Why It Can Matter More Than IQ*. Bantam Books.

[Hawkins, 2011] Hawkins, P. (2011). *Leadership Team Coaching: Developing Collective Transformational Leadership*. Kogan Page.

[Hersey et al., 2007] Hersey, P., Blanchard, K. H., and Johnson, D. E. (2007). *Management of Organizational Behavior: Leading Human Resources*. Prentice Hall, 9th edition.

[Huete, 2008] Huete, L. M. (2008). *Construye tu Sueño*. LID Editorial Empresarial.

[Hunsaker, 2001] Hunsaker, P. (2001). *Training in Management Skills*. Pearson Prentice Hall.

[Lencioni, 2002] Lencioni, P. (2002). *The Five Dysfunctions of a Team: A Leadership Fable*. Jossey-Bass.

Octavo Semestre

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS281 Computación en la Sociedad (Obligatorio)

1. Información general

- **Semestre**: 8vo Sem. **Créditos**: 2
- **Horas del curso**: **Teoría**: 2 horas;
- **Prerrequisitos**: Ninguno

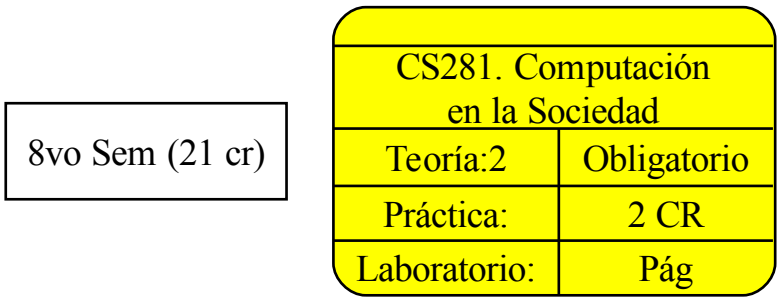

Figura 5.44: Mapa de Conexión. CS281 Computación en la Sociedad

2. Fundamentación

Este curso explora la relación multifacética entre las tecnologías de computación y la sociedad. Proporciona una perspectiva histórica sobre la evolución del campo mientras equipa a los estudiantes con marcos de análisis ético para navegar desafíos contemporáneos. Los temas incluyen propiedad intelectual, privacidad, responsabilidad profesional, sostenibilidad y el impacto socioeconómico de la automatización. Los estudiantes desarrollarán las habilidades de comunicación y sensibilidad ética requeridas para la práctica profesional en un entorno diverso y globalizado.

3. Objetivos del curso.

- Aplicar análisis ético a dilemas profesionales en computación.
- Comprender los marcos legales para Propiedad Intelectual y Privacidad.
- Evaluar la sostenibilidad ambiental y el impacto social de las TI.
- Comunicarse efectivamente respecto a problemas profesionales y éticos.

4. Competencias Generales (*Outcomes*)

AG-C01) El Profesional y el Mundo: Analiza y evalúa el impacto de las soluciones a problemas complejos de computación en el desarrollo sostenible de la sociedad. (Usar)

AG-C02) Ética: Aplica principios éticos y se compromete con la ética profesional y las normas de la práctica profesional de la computación. (Usar)

5. Contenido

5.1 Fundamentos e Historia Temprana de la Informática, 2 hr(s)

Competencias: AG-C01,AG-C02

Tópicos: **I:** La historia de la informática: hardware, software y humano/organizacional. **II:** El papel de la historia en el presente, incluyendo dentro de diferentes contextos sociales, y la relevancia de esta historia en el futuro. **III:** Edad I (Pre-digital): Computación analógica antigua (Stonehenge, mecanismo de Anticitera, reloj de la Catedral de Salisbury, etc.), tablas numéricas calculadas por humanos, Euclides, Lovelace, Babbage, Gödel, Church, Turing, hardware pre-electrónico (electromecánico y mecánico). **IV:** Edad II (Computación moderna temprana): ENIAC, UNIVAC, Bombas (Bletchley Park y descifradores de códigos), compañías informáticas (por ejemplo, IBM), mainframes, etc.

Objetivos de Aprendizaje **I:** Comprender la relevancia e impacto de la historia de la informática en eventos recientes, contexto presente y posibles resultados futuros, desde más de una perspectiva cultural [Evaluar (valorar)] **II:** Discutir cómo las perspectivas sostenidas hoy han sido moldeadas por la historia, y que existen perspectivas alternativas (por ejemplo, temores de que la IA reemplace a los trabajadores humanos vs IA que aumenta el trabajo humano, varias visiones sobre la Ley de Moore) [Debatir] **III:** Identificar tendencias formativas y consecuentes en la historia del campo de la informática [Analizar] **IV:** Identificar las contribuciones de individuos u organizaciones pioneras (laboratorios de investigación, compañías informáticas, oficinas gubernamentales) en el campo de la informática [Analizar]

Bibliografía: [Quinn, 2019, Baase and Henry, 2017]

5.2 Impacto Social e IA, 2 hr(s)

Competencias: AG-C01,AG-C02

Tópicos: **I:** Implicaciones sociales (por ejemplo, ideologías políticas y culturales) en un mundo hiperconectado donde las capacidades y el impacto de las redes sociales, la inteligencia artificial y la informática en general están evolucionando rápidamente. **II:** Impacto de las aplicaciones informáticas (por ejemplo, redes sociales, aplicaciones de inteligencia artificial) en el bienestar individual y la seguridad de todo tipo (por ejemplo, física, emocional, económica). **III:** Consecuencias de involucrar tecnologías informáticas, particularmente inteligencia artificial, tecnologías biométricas y sistemas de toma de decisiones algorítmicos, en la vida cívica (por ejemplo, tecnología de reconocimiento facial, etiquetas biométricas, algoritmos de distribución de recursos, software policial) y cómo la agencia y supervisión humana son cruciales. **IV:** Crecimiento y control de internet, datos, informática e inteligencia artificial.

Objetivos de Aprendizaje **I:** Describir las diferentes formas en que la tecnología informática (redes, computación móvil, inteligencia artificial) media la interacción social a nivel personal y colectivo [Describir] **II:** Interpretar el contexto social de un diseño dado y su implementación [Interpretar] **III:** Analizar la eficacia de un diseño e implementación dados utilizando datos empíricos [Analizar] **IV:** Comprender las implicaciones del uso de la tecnología (por ejemplo, redes sociales) para diferentes identidades, culturas y comunidades [Explicar] **V:** Describir el papel de internet en facilitar la comunicación entre ciudadanos, gobiernos y entre sí [Describir] **VI:** Analizar los efectos de la dependencia de la informática en la implementación de la democracia (por ejemplo, entrega de servicios sociales, votación electrónica) [Analizar] **VII:** Describir el impacto de la falta de representación apropiada de personas de poblaciones históricamente minorizadas en la profesión de la informática (por ejemplo, cultura de la industria, diversidad de productos) [Describir]

Bibliografía: [Quinn, 2019]

5.3 Fundamentos y Marcos del Análisis Ético, 4 hr(s)

Competencias: AG-C01,AG-C02

Tópicos: **I:** Evitar falacias y tergiversaciones en la argumentación. **II:** Teorías éticas y toma de decisiones (marcos filosóficos y sociales, por ejemplo [1]). **III:** Reconocimiento del papel que juega la cultura en nuestra comprensión, adopción, diseño y uso de la tecnología informática. **IV:** Por qué la ética es importante en la informática y cómo la ética es similar y diferente de las leyes y normas sociales. **V:** Introducción a los marcos éticos (por ejemplo, consecuencialismo como el utilitarismo, no consecuencialismo como el deber, derechos o justicia, centrado en el agente como la virtud o el feminismo, contractualismo, ética del cuidado) y su uso para analizar un dilema ético.

Objetivos de Aprendizaje **I:** Describir cómo un contexto cultural dado impacta la toma de decisiones [Describir] **II:** Expresar el uso de ejemplos y analogías en la argumentación ética [Analizar] **III:** Analizar (y evitar) falacias lógicas básicas en un argumento [Analizar] **IV:** Analizar un argumento para identificar premisas y conclusión [Analizar] **V:** Evaluar cómo y por qué la ética es tan importante en la informática y cómo se relaciona con las normas culturales, valores y leyes [Evaluar] **VI:** Justificar una decisión tomada sobre bases éticas [Justificar] **VII:** Discutir las ventajas y desventajas de los marcos éticos tradicionales [Debatir] **VIII:** Analizar dilemas éticos relacionados con la creación y uso de tecnología desde múltiples perspectivas utilizando marcos éticos [Analizar]

Bibliografía: [Quinn, 2019, Baase and Henry, 2017]

5.4 Valores, Naturaleza y Actualización Profesional, 4 hr(s)

Competencias: AG-C01,AG-C02

Tópicos: **I:** Valores de la comunidad y las leyes por las que vivimos. **II:** La naturaleza de ser un profesional, incluyendo cuidado, atención, disciplina, responsabilidad fiduciaria y tutoría (mentoring). **III:** Mantenerse actualizado como profesional de la informática en términos de familiaridad, herramientas, habilidades, marcos legales y profesionales, así como la capacidad y responsabilidad de autoevaluarse y progresar en el campo de la informática.

Objetivos de Aprendizaje **I:** Identificar problemas éticos que surgen en el diseño de software, prácticas de desarrollo y despliegue de software [Analizar] **II:** Discutir cómo abordar problemas éticos en situaciones específicas [Debatir] **III:** Describir los mecanismos que típicamente existen para que un profesional se mantenga actualizado en asuntos éticos [Describir]

Bibliografía: [Baase and Henry, 2017]

5.5 Fundamentos de la Propiedad Intelectual, 4 hr(s)

Competencias: AG-C01,AG-C02

Tópicos: **I:** Derechos de propiedad intelectual. **II:** Fundamentos legales para la protección de la propiedad intelectual. **III:** Fundamentos filosóficos de la propiedad intelectual. **IV:** Formas de propiedad intelectual (por ejemplo, derechos de autor, patentes, secretos comerciales, marcas registradas) y los derechos que protegen.

Objetivos de Aprendizaje **I:** Defender los usos legales y éticos de materiales con derechos de autor [Defender] **II:** Discutir las bases filosóficas de la propiedad intelectual en un contexto apropiado (por ejemplo, país) [Debatir] **III:** Contrastar las protecciones y obligaciones de los derechos de autor, patentes, secretos comerciales y marcas registradas [Contrastar] **IV:** Describir la justificación para la protección legal de la propiedad intelectual en el contexto apropiado (por ejemplo, país) [Describir]

Bibliografía: [Quinn, 2019]

5.6 Derechos de PI Digital y Derechos de Autor, 6 hr(s)

Competencias:

Tópicos: I: Propiedad intelectual digital intangible (IDIP). **II:** Limitaciones a las protecciones de derechos de autor, incluyendo uso justo (fair use) y la doctrina de la primera venta (first sale doctrine). **III:** Leyes y tratados de propiedad intelectual que impactan la aplicación de los derechos de autor.

Objetivos de Aprendizaje I: Describir y criticar la legislación y los precedentes destinados a las infracciones de derechos de autor digitales [Describir] **II:** Identificar ejemplos contemporáneos de propiedad intelectual digital intangible [Analizar] **III:** Discutir si un uso de material con derechos de autor es probable que sea uso justo (fair use) [Debatir] **IV:** Analizar el uso de obras con derechos de autor bajo los conceptos de uso justo (fair use) y la doctrina de la primera venta (first sale doctrine) [Analizar] **V:** Discutir la naturaleza de las leyes anti-elusión (anti-circumvention) en el contexto de la protección de derechos de autor [Debatir]

Bibliografía: [Tavani, 2013]

5.7 Datos, Vigilancia y Tecnología de Privacidad, 4 hr(s)

Competencias: AG-C01,AG-C02

Tópicos: I: Implicaciones de privacidad de la recopilación generalizada de datos, incluyendo pero no limitado a bases de datos transaccionales, almacenes de datos, sistemas de vigilancia, computación en la nube e inteligencia artificial. **II:** Concepciones de anonimato, seudonimidad e identidad. **III:** Soluciones basadas en tecnología para la protección de la privacidad (por ejemplo, encriptación de extremo a extremo y privacidad diferencial).

Objetivos de Aprendizaje I: Evaluar soluciones a las amenazas a la privacidad en bases de datos transaccionales y almacenes de datos [Evaluar] **II:** Describir el papel de la recopilación de datos en la implementación de sistemas de vigilancia generalizada (por ejemplo, RFID, reconocimiento facial, cobro de peajes, computación móvil) [Describir] **III:** Distinguir los conceptos y objetivos del anonimato y la seudonimidad [Distinguir] **IV:** Describir las ramificaciones de las protecciones de privacidad basadas en tecnología, incluyendo la privacidad diferencial y la encriptación de extremo a extremo [Describir]

Bibliografía: [Baase and Henry, 2017]

5.8 Huella Ambiental de los Sistemas Informáticos, 2 hr(s)

Competencias: AG-C01,AG-C02

Tópicos: I: Impactos ambientales, sociales y culturales de las decisiones de implementación (por ejemplo, objetivos de sostenibilidad, sesgo/resultados algorítmicos, viabilidad económica y consumo de recursos). **II:** Impactos sociales y ambientales locales/regionales/globales de los sistemas informáticos y su uso (por ejemplo, huella de carbono, uso de recursos, residuos electrónicos) debido al hardware (por ejemplo, residuos electrónicos, centros de datos, utilización de elementos y recursos raros, reciclaje) y software (por ejemplo, servicios basados en la nube, blockchain, entrenamiento y uso de modelos de IA). Esto incluye el uso diario de hardware (hardware barato reemplazado con frecuencia) y software (navegación web, correo electrónico y otros servicios con demandas computacionales ocultas/remotas).

Objetivos de Aprendizaje I: Evaluar los impactos ambientales del despliegue de un proyecto dado (por ejemplo, consumo de energía, contribución a los residuos electrónicos, impacto de la fabricación) [Evaluar (valorar)] **II:** Describir los impactos sociales y ambientales globales del uso y disposición de computadoras [Describir] **III:** Describir los impactos ambientales de las decisiones de diseño dentro del campo de la informática que se relacionan con el diseño de algoritmos, diseño de sistemas operativos, diseño de redes, diseño de bases de datos, etc [Describir] **IV:** Evaluar

aplicaciones informáticas en relación con problemas ambientales (por ejemplo, energía, contaminación, uso de recursos, reciclaje y reutilización, gestión y producción de alimentos) [Evaluar (valorar)]

Bibliografía: [Quinn, 2019]

5.9 Modelos y Estrategias Económicas, 2 hr(s)

Competencias: AG-C01,AG-C02

Tópicos: I: Modelos económicos: regulados y no regulados, monopolios, efectos de red (network effects) y mercado abierto; economías del conocimiento y de la atención (knowledge and attention economies). **II:** Estrategias de precios y despliegue: obsolescencia programada (planned obsolescence), suscripciones, freemium, licencias de software, código abierto (open-source), software libre (free software), adware. **III:** Impactos de las diferencias en el acceso a recursos informáticos y el efecto de la oferta y demanda de mano de obra calificada en la calidad de los productos informáticos.

Objetivos de Aprendizaje I: Resumir los efectos sociales de los modelos económicos (por ejemplo, las economías del conocimiento y de la atención) [Resumir] **II:** Describir las diferencias y similitudes de estrategias competitivas (por ejemplo, suscripción vs freemium vs gratis) [Describir] **III:** Discutir ejemplos de brechas digitales [Debatir]

Bibliografía: [Baase and Henry, 2017]

5.10 Delitos y Ataques Informáticos, 2 hr(s)

Competencias: AG-C01,AG-C02

Tópicos: I: Delitos informáticos, recurso legal para criminales informáticos e impacto en víctimas y sociedad. **II:** Ingeniería social, fraude habilitado por la informática, robo de identidad y recuperación de estos. **III:** Terrorismo cibernético, hacking criminal y hacktivismo. **IV:** Malware, virus, gusanos. **V:** Ataques a infraestructura crítica como redes eléctricas y oleoductos.

Objetivos de Aprendizaje I: Enumerar ejemplos clásicos de delitos informáticos e incidentes de ingeniería social con impacto social [Listar/Enumerar] **II:** Identificar problemas con las leyes que se aplican a los delitos informáticos [Analizar] **III:** Describir la motivación y ramificaciones del terrorismo cibernético, robo de datos, hacktivismo, ransomware y otros ataques [Describir] **IV:** Examinar los problemas éticos y legales que rodean el mal uso del acceso y varias violaciones de seguridad [Examinar]

Bibliografía: [Baase and Henry, 2017]

5.11 Identidad y Diversidad en la Informática, 2 hr(s)

Competencias: AG-C01,AG-C02

Tópicos: I: Cómo la identidad impacta y es impactada por las tecnologías y entornos informáticos (académicos y profesionales). **II:** Los beneficios de los equipos de desarrollo diversos y los impactos de los equipos que no son diversos.

Objetivos de Aprendizaje I: Definir y distinguir equidad, igualdad, diversidad, inclusión y accesibilidad [Definir] **II:** Identificar ejemplos de los beneficios que los equipos diversos pueden aportar a los productos de software y cómo la falta de diversidad tiene costos [Analizar] **III:** Evaluar el impacto del poder y el privilegio en la profesión de la informática en lo que respecta a la cultura, industria, productos y sociedad [Evaluar (valorar)]

Bibliografía: [Quinn, 2019]

5.12 Comunicación Técnica y Profesional, 2 hr(s)

Competencias: AG-C01,AG-C02

Tópicos: **I:** Comunicación oral, escrita y electrónica en equipo y grupo. **II:** Materiales de comunicación técnica (por ejemplo, código fuente y documentación, tutoriales, materiales de referencia, documentación de API). **III:** Comunicarse con diferentes partes interesadas, como clientes, liderazgo o el público. **IV:** Compensaciones (tradeoffs) en factores competitivos que afectan los canales y elecciones de comunicación.

Objetivos de Aprendizaje **I:** Comprender la importancia de escribir documentos técnicos concisos y precisos siguiendo estándares bien definidos para el formato y para incluir tablas, figuras y referencias apropiadas [Producir] **II:** Analizar documentación técnica escrita para verificar precisión técnica, concisión, falta de ambigüedad y conciencia de la audiencia [Analizar] **III:** Componer y entregar una presentación formal consciente de la audiencia, accesible y organizada [Componer] **IV:** Identificar y describir cualidades de la comunicación efectiva (por ejemplo, virtual, cara a cara, intragrupo, documentos compartidos) [Analizar] **V:** Elegir una forma apropiada de comunicar preocupaciones éticas delicadas [Evaluar (valorar)]

Bibliografía: [Quinn, 2019]

6. Bibliografía

[Baase and Henry, 2017] Baase, S. and Henry, T. (2017). *A Gift of Fire: Social, Legal, and Ethical Issues for Computing Technology*. Pearson, 5th edition.

[Quinn, 2019] Quinn, M. J. (2019). *Ethics for the Information Age*. Pearson, 8th edition.

[Tavani, 2013] Tavani, H. T. (2013). *Ethics and Technology: Controversies, Questions, and Strategies for Ethical Computing*. Wiley, Hoboken, NJ, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS3I1 Seguridad en Computación (Obligatorio)

1. Información general

- **Semestre**: 8vo Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 1 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS231 Redes y Comunicación (6to Sem-Pág 401)

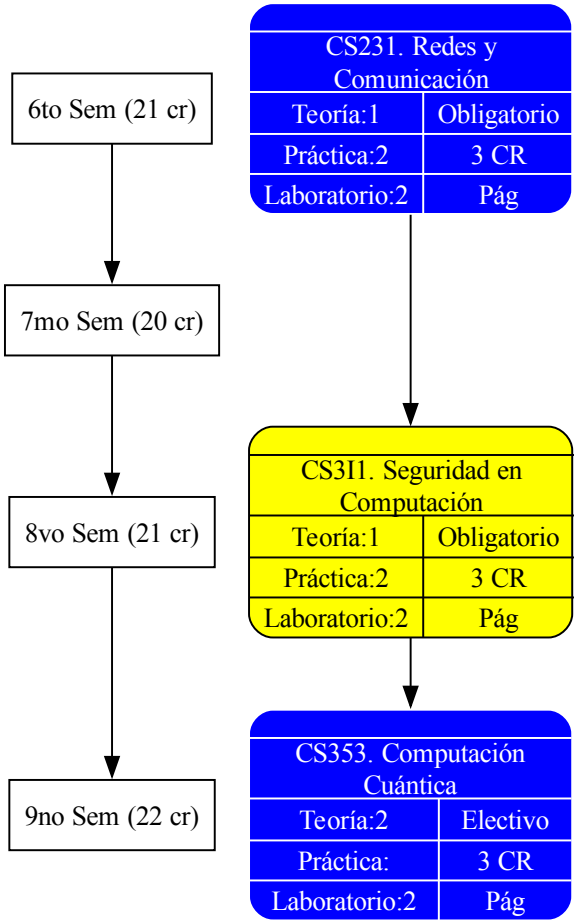

Figura 5.45: Mapa de Conexión. CS3I1 Seguridad en Computación

2. Fundamentación

En un mundo cada vez más digital, la seguridad ya no es una característica opcional sino un requisito fundamental. Este curso proporciona una introducción completa a los principios y prácticas de la ciberseguridad, cubriendo desde los fundamentos teóricos hasta la codificación segura y la informática forense digital. Los estudiantes adquirirán las habilidades necesarias para diseñar, implementar y gobiernar sistemas seguros, comprendiendo las responsabilidades éticas y profesionales involucradas en la protección de la información y la infraestructura.

3. Objetivos del curso.

- Comprender los principios fundamentales y las bases matemáticas de la seguridad.
- Aplicar técnicas de codificación segura y métodos criptográficos al software.
- Analizar vulnerabilidades del sistema y realizar investigaciones forenses.
- Evaluar la gobernanza de seguridad y la ética profesional en ciberseguridad.

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Usar)

5. Contenido

5.1 Principios y Mentalidad de Seguridad, 6 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Desarrollar una mentalidad de seguridad que incorpore conceptos transversales: confidencialidad, integridad, disponibilidad, evaluación de riesgos, pensamiento sistémico, pensamiento adversarial, pensamiento centrado en el humano. **II:** Vulnerabilidades, amenazas, superficies de ataque y vectores de ataque Protección y Seguridad. **III:** Denegación de Servicio (DoS) y Denegación de Servicio Distribuida (DDoS) Protección y Seguridad. **IV:** Principios y prácticas de protección, por ejemplo, privilegio mínimo, diseño abierto, valores predeterminados seguros ante fallos, defensa en profundidad y confianza cero; y cómo pueden implementarse Principios del Sistema Operativo, Protección y Seguridad, Prácticas de Codificación, Construcción a Escala y Proceso, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad. **V:** Consideraciones de optimización entre seguridad, privacidad, rendimiento y otros objetivos de diseño Diseño de Sistemas, Prácticas de Desarrollo de Software, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas. **VI:** Conceptos de confianza (trust) y confiabilidad (trustworthiness).

Objetivos de Aprendizaje **I:** Evaluar un sistema para posibles ataques que pueda lanzar un adversario [Evaluar] **II:** Diseñar y desarrollar enfoques para proteger un sistema de un conjunto de amenazas identificadas [Diseñar] **III:** Describir cómo se puede evitar el daño a la privacidad del usuario [Describir] **IV:** Desarrollar un sistema que incorpore varios principios de seguridad y privacidad [Crear]

Bibliografía: [Pfleeger et al., 2015, Stallings, 2017]

5.2 Fundamentos de Criptografía, 8 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Diferencias entre las vistas algorítmica, aplicada y matemática de la criptografía. **II:** Preliminares matemáticos: aritmética modular, algoritmo euclidiano, independencia probabilística, conceptos básicos de álgebra lineal, teoría de números, campos finitos, complejidad, análisis asintótico Matemáticas Discretas, álgebra Lineal. **III:** Criptografía básica: criptografía de clave simétrica y de clave pública Estructuras de Datos Fundamentales, Algoritmos Fundamentales, Algoritmos Avanzados, Matemáticas Discretas. **IV:** Criptosistemas clásicos, como cifrados de desplazamiento, sustitución, transposición, libros de códigos y máquinas Matemáticas Discretas. **V:** Principio de Kerckhoff y uso de bibliotecas evaluadas (vetted) Prácticas de Codificación, Construcción a Escala y Proceso.

Objetivos de Aprendizaje I: Explicar el papel de la criptografía en el soporte de la seguridad y la privacidad [Explicar] **II:** Discutir los riesgos de inventar los propios métodos criptográficos [Debatir] **III:** Discutir la importancia de los números primos en criptografía y explicar su uso en algoritmos criptográficos [Debatir] **IV:** Implementar y criptoanalizar cifrados clásicos [Implementar]

Bibliografía: [Stallings, 2017]

5.3 Inyección y Validación de Entrada, 8 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Vulnerabilidades y debilidades comunes. **II:** Inyección SQL y otros ataques de inyección. **III:** Técnicas de cross-site scripting y mitigaciones. **IV:** Validación de entrada y sanitización de datos Protección y Seguridad, Conceptos y Construcciones Básicas de Programación, Prácticas de Programación: Pruebas, Depuración y Documentación, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.

Objetivos de Aprendizaje I: Identificar problemas subyacentes en ejemplos dados de una enumeración de debilidades comunes y explicar cómo pueden evitarse [Analizar] **II:** Aplicar técnicas de validación de entrada y sanitización de datos para mejorar la seguridad de un programa [Aplicar] **III:** Evaluar y prevenir ataques de inyección SQL en una aplicación de base de datos [Evaluar] **IV:** Evaluar y prevenir ataques de cross-site scripting contra un sitio web [Evaluar] **V:** Construir un programa que pruebe todos los errores de manejo de entrada [Crear]

Bibliografía: [Pfleeger et al., 2015]

5.4 Análisis de Malware y Seguridad Avanzada, 6 hr(s)

Competencias: AG-C09

Tópicos: I: Técnicas de seguridad ofensiva como defensa. **II:** Técnicas de detección de malware asistidas por IA. **III:** Ransomware: creación, prevención y mitigación. **IV:** Uso seguro de componentes de terceros Prácticas de Codificación, Construcción a Escala y Proceso, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas. **V:** Malware: variedades, creación, ingeniería inversa y defensa contra ellos Traducción y Ejecución de Lenguajes, Modelo de Ejecución y Memoria de Sistemas. **VI:** Garantía (assurance): pruebas (incluyendo fuzzing y pruebas de penetración), verificación y validación Protección y Seguridad, Conceptos y Construcciones Básicas de Programación, Prácticas de Programación: Pruebas, Depuración y Documentación, Prácticas de Codificación, Construcción a Escala y Proceso, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas. **VII:** Análisis estático y dinámico Análisis de Programas y Analizadores, Protección y Seguridad, Evaluación, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas. **VIII:** Compiladores seguros y generación de código seguro Traducción y Ejecución de Lenguajes, Comportamiento en Tiempo de Ejecución y Sistemas.

Objetivos de Aprendizaje I: Describir diferentes tipos de software malicioso [Describir] **II:** Explicar los riesgos del mal uso de interfaces con código de terceros y cómo usar correctamente el código de terceros [Explicar] **III:** Discutir la necesidad de actualizar el software para corregir vulnerabilidades de seguridad y la gestión del ciclo de vida de la corrección [Debatir] **IV:** Aplicar herramientas estáticas y dinámicas para identificar fallos de programación [Aplicar] **V:** Evaluar un sistema para detectar la existencia de malware y eliminarlo [Evaluar] **VI:** Implementar técnicas preventivas para reducir la ocurrencia de ransomware [Implementar]

Bibliografía: [Sikorski and Honig, 2012, Eilam, 2005, National Institute of Standards and Technology, 2020]

5.5 Análisis de Amenazas e Ingeniería de Seguridad, 8 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Objetivos de la ingeniería de seguridad: construir sistemas que permanezcan confiables a pesar de errores, accidentes o adversarios maliciosos Prácticas de Codificación, Construcción a Escala y Proceso, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad. **II:** Análisis de problemas y análisis situacional para abordar la seguridad del sistema Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas. **III:** Análisis de seguridad, que cubre análisis de requisitos de seguridad; análisis de controles de seguridad; análisis de amenazas; y análisis de vulnerabilidades Análisis de Programas y Analizadores, Evaluación. **IV:** Dominios de ataque de seguridad y superficies de ataque, por ejemplo, comunicaciones y redes, hardware, físico, ingeniería social, software y cadena de suministro Seguridad de Red. **V:** Modos, técnicas y tácticas de ataque de seguridad, por ejemplo, abuso de autenticación; fuerza bruta; manipulación de búfer; inyección de código; inserción de contenido; denegación de servicio; escucha; omisión de función; suplantación; ataque de integridad; intercepción; phishing; análisis de protocolo; abuso de privilegio; suplantación (spoofing); e inyección de tráfico Seguridad de Red, Protección y Seguridad, Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas.

Objetivos de Aprendizaje **I:** Crear un modelo de amenazas para un sistema o diseño de sistema [Crear] **II:** Aplicar análisis situacional para desarrollar soluciones seguras bajo un escenario especificado [Aplicar] **III:** Identificar y mitigar vulnerabilidades y debilidades de seguridad en un sistema [Analizar]

Bibliografía: [Anderson, 2020]

5.6 Técnicas de Informática Forense, 6 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Principios y metodologías básicas para la informática forense. **II:** Diseño de sistemas para informática forense. **III:** Informática forense en diferentes situaciones: sistemas operativos, sistemas de archivos, informática forense de aplicaciones, informática forense web, informática forense de red, informática forense de dispositivos móviles, uso de auditoría de bases de datos Seguridad de Red. **IV:** Ataques a la informática forense y prevención de tales ataques.

Objetivos de Aprendizaje **I:** Explicar qué es una investigación digital y cómo se puede implementar Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad [Explicar] **II:** Diseñar e implementar software para apoyar la informática forense [Diseñar] **III:** Extraer datos de un disco duro para cumplir con la ley Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad [Extraer] **IV:** Recuperar datos basados en un término de búsqueda dado desde un sistema del que se ha hecho una imagen (imaged) [Analizar] **V:** Reconstruir datos y eventos a partir de un historial de aplicación, o un artefacto web, o una base de datos en la nube, o un dispositivo móvil. Plataformas Móviles: Capacidades y Computación Asíncrona, Plataformas Móviles: Arquitectura y Restricciones, Plataformas Móviles: Fundamentos del Desarrollo, Arquitectura Cloud y Gestión de Datos, Frontend, Frameworks y Estándares Web [Aplicar] **VI:** Capturar y analizar tráfico de red. Seguridad de Red [Analizar] **VII:** Desarrollar enfoques para abordar los desafíos asociados con la informática forense de dispositivos móviles [Crear] **VIII:** Aplicar herramientas de informática forense para investigar violaciones de seguridad [Aplicar] **IX:** Identificar y mitigar métodos anti-forenses [Analizar]

Bibliografía: [Pfleeger et al., 2015]

5.7 Gobierno, Gestión y Política de Seguridad, 6 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Proteger activos críticos de amenazas. **II:** Gobierno de seguridad (security governance): objetivos organizacionales y evaluación general de riesgos. **III:** Gestión de seguridad (security management): lograr y mantener niveles apropiados de confidencialidad, integridad, disponibilidad, responsabilidad (accountability), autenticidad y confiabilidad Pruebas de Rendimiento y Evaluación Comparativa, Herramientas y Análisis de Pruebas, Conceptos de verificación y validación, Planificación y Tipos de Pruebas, Prácticas Avanzadas de Pruebas. **IV:** Política de seguridad (security policy): políticas organizacionales, políticas específicas de temas, políticas específicas de sistemas.

Objetivos de Aprendizaje **I:** Describir activos críticos y cómo pueden protegerse [Describir] **II:** Diferenciar entre gobierno de seguridad, gestión y controles, dando ejemplos de cada uno [Diferenciar] **III:** Describir un control técnico e implementarlo para mitigar amenazas específicas [Describir]

Bibliografía: [Anderson, 2020]

5.8 Ética, Ley y Sociedad en Seguridad, 6 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Impactos sociales de las fallas en seguridad y privacidad Equidad Digital y Accesibilidad, Impacto Social e IA, Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad. **II:** Aplicabilidad de leyes y regulaciones sobre seguridad y privacidad Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad. **III:** Consideraciones éticas profesionales al diseñar sistemas seguros y mantener la privacidad; hacking ético Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad, Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad. **IV:** Implicaciones de seguridad y privacidad de actores maliciosos de IA/aprendizaje automático, por ejemplo, identificación de deep fakes Formulación del Aprendizaje Automático y Evaluación, Aprendizaje Profundo, RL y Ética, Introducción a la IA: Conceptos y Comportamiento Inteligente, Características del Problema y Naturaleza de los Agentes, Fundamentos de Aprendizaje Automático, Aprendizaje Automático: Datos, Representaciones y Evaluación, Redes Neuronales y Ética, Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad. **V:** Impactos sociales de los dispositivos del Internet de las Cosas (IoT) y otras tecnologías emergentes en la seguridad y la privacidad Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.

Objetivos de Aprendizaje **I:** Calcular el impacto de una falla de seguridad en un sistema dado [Calcular] **II:** Construir un sistema que se ajuste a las leyes de seguridad [Crear] **III:** Aplicar un conjunto de regulaciones de privacidad para diseñar un sistema que proteja la privacidad [Aplicar] **IV:** Evaluar las ramificaciones legales de un sistema que no se ajusta a las leyes y regulaciones aplicables [Evaluar]

Bibliografía: [Anderson, 2020, Pfleeger et al., 2015]

6. Bibliografía

[Anderson, 2020] Anderson, R. (2020). *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 3rd edition.

[Eilam, 2005] Eilam, E. (2005). *Reversing: Secrets of Reverse Engineering*. Wiley, Indianapolis, IN.

[National Institute of Standards and Technology, 2020] National Institute of Standards and Technology (2020). Security and privacy controls for information systems and organizations. Technical Report NIST SP 800-53 Rev. 5, National Institute of Standards and Technology, Gaithersburg, MD. Disponible en: `https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final`.

[Pfleeger et al., 2015] Pfleeger, C. P., Pfleeger, S. L., and Margulies, J. (2015). *Security in Computing*. Prentice Hall, 5th edition.

[Sikorski and Honig, 2012] Sikorski, M. and Honig, A. (2012). *Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software*. No Starch Press, San Francisco, CA.

[Stallings, 2017] Stallings, W. (2017). *Computer Security: Principles and Practice*. Pearson, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS3P1 Computación Paralela y Distribuída (Obligatorio)

1. Información general

- **Semestre**: 8vo Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS212 Análisis y Diseño de Algoritmos (5to Sem-Pág 366)
  - CS231 Redes y Comunicación (6to Sem-Pág 401)

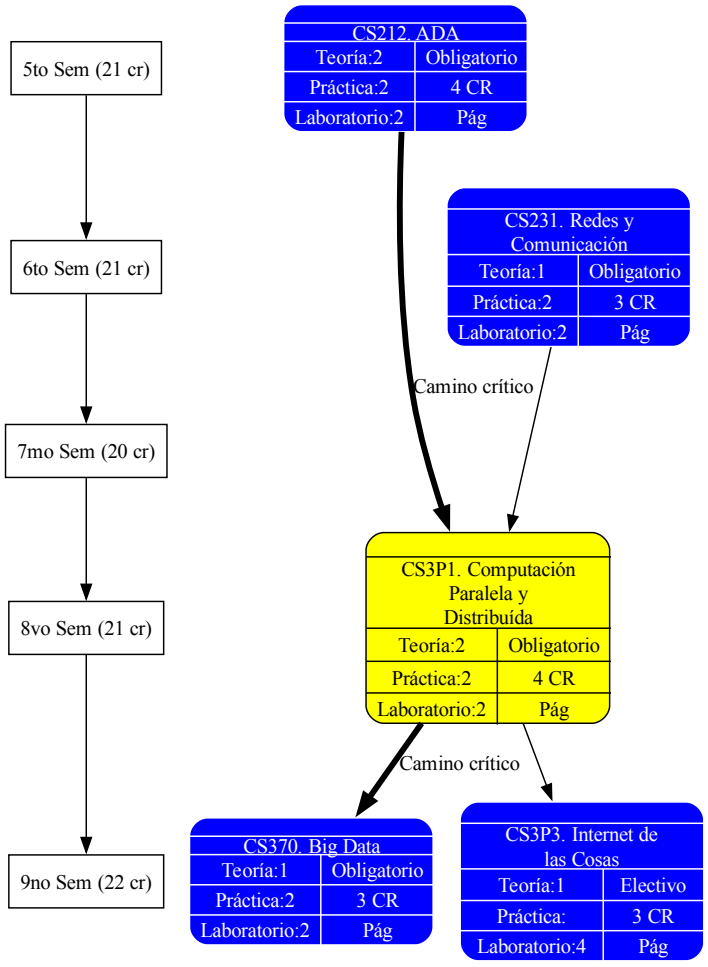

Figura 5.46: Mapa de Conexión. CS3P1 Computación Paralela y Distribuída

2. Fundamentación

Con el fin del escalado de frecuencia en procesadores de un solo núcleo, la computación paralela y distribuida se ha vuelto esencial para el desarrollo de software de alto rendimiento. Este curso introduce a los estudiantes a los modelos de programación, patrones de comunicación y mecanismos de coordinación necesarios para explotar sistemas de múltiples núcleos y clústeres distribuidos.

Los estudiantes aprenderán a diseñar algoritmos escalables y evaluar su rendimiento bajo diferentes restricciones computacionales.

3. Objetivos del curso.

- Diseñar e implementar programas paralelos utilizando memoria compartida y distribuida.
- Dominar protocolos de comunicación y sincronización en sistemas distribuidos.
- Evaluar el rendimiento y la escalabilidad mediante métricas formales.
- Aplicar algoritmos paralelos para resolver problemas computacionales complejos.

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Usar)

5. Contenido

5.1 Programas: Paralelismo Declarativo, 4 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Paralelismo *a*) Paralelismo declarativo: Determinar qué acciones pueden, o no deben, realizarse en paralelo, a nivel de instrucciones, funciones, clausuras, acciones compuestas, sesiones, tareas y servicios es la idea principal que subyace a los algoritmos PDC; no hacerlo es la principal fuente de errores. Ver también: Algoritmos *b*) Definir orden: por ejemplo, usando relaciones de "sucede antes" o grafos acíclicos dirigidos serie/paralelo que representan programas. *c*) Independencia: determinar cuándo el orden no importa, en términos de conmutatividad, dependencias, precondiciones. *d*) Garantizar el orden entre acciones que de otro modo serían paralelas cuando sea necesario, incluyendo bloqueos, publicación segura; e imponer comunicación - enviar un mensaje sucede antes de recibirlo; y relajarlo cuando no sea necesario.

Objetivos de Aprendizaje I: Mostrar gráficamente (como un Grafo Acíclico Dirigido - *Directed Acyclic Graph* (DAG)) cómo paralelizar una expresión numérica compuesta; por ejemplo, $a = (b + c) * (d + e)$ [Diseñar] **II:** Explicar por qué los conceptos de consistencia y tolerancia a fallos no surgen en programas puramente secuenciales [Explicar]

Bibliografía: [Pacheco and Malensek, 2021]

5.2 Programas: Inicio de Actividades, 3 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Iniciar actividades *a*) Las opciones que permiten que las acciones se realicen (eventualmente) en lugares van desde el cableado directo hasta scripts de configuración; también establecer comunicación y gestión de recursos; estos se expresan de manera diferente en los lenguajes y contextos, generalmente confiando en el aprovisionamiento y gestión automatizados por las plataformas. Ver también: Gestión de Recursos *b*) Procedimental: Permitir que múltiples acciones comiencen en un punto de programa dado; por ejemplo, iniciar nuevos hilos, posiblemente delimitando su alcance u organizándolos en grupos jerárquicos. *c*) Reactiva: Habilitar al ocurrir un evento instalando un manejador de eventos, con menos control de cuándo comienzan o terminan las acciones, y puede aplicarse incluso en monoprocesadores. *d*) Dependiente: Habilitar al completarse otras; por ejemplo, secuenciando conjuntos de acciones paralelas. Ver también: Coordinación. *e*) Granularidad: El costo de ejecución de los cuerpos de las acciones debe superar la sobrecarga de organizarlos.

Objetivos de Aprendizaje I: Escribir un servicio que cree un hilo (u otra forma de activación procedimental) para devolver una página web solicitada a cada nuevo cliente [Escribir]

Bibliografía: [Pacheco and Malensek, 2021]

5.3 Programas: Propiedades de Ejecución, 3 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Propiedades de Ejecución *a*) Ejecución no determinista de acciones sin orden. *b*) Consistencia: Garantizar el acuerdo entre partes sobre valores y predicados cuando sea necesario para evitar carreras, mantener seguridad y atomicidad, o llegar a consenso. *c*) Tolerancia a fallos: Manejar fallos en las partes o la comunicación, incluyendo (Bizantino) mal comportamiento debido a partes y protocolos no confiables, cuando sea necesario para mantener el progreso o la disponibilidad. Ver también: Confiabilidad del Sistema. *d*) Los compromisos son un foco de evaluación. Ver también: Evaluación.

Objetivos de Aprendizaje I: Escribir una función que cuente eventos, como recepciones de paquetes de red, de manera eficiente [Escribir]

Bibliografía: [van Steen and Tanenbaum, 2023]

5.4 Programas: Distribución, 2 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Distribución *a*) Definir lugares, como dispositivos que ejecutan acciones, incluyendo componentes de hardware, hosts remotos; también pueden incluir dispositivos, hosts y usuarios externos no controlados. Ver también: Interconexión y Comunicación *b*) Un dispositivo puede dividir el tiempo o emular múltiples acciones paralelas mediante menos procesadores mediante planificación y virtualización. Ver también: Planificación *c*) Nombrar o identificar lugares (por ejemplo, IDs de dispositivo) y acciones como partes (por ejemplo, IDs de hilo). *d*) Las actividades a través de lugares pueden comunicarse a través de medios. Ver también: Comunicación

Objetivos de Aprendizaje I: Escribir un programa de filtro/mapeo/reducción en múltiples estilos [Escribir]

Bibliografía: [van Steen and Tanenbaum, 2023]

5.5 Programas: Mapeos de Implementación, 2 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Una o más de las siguientes asignaciones y mecanismos a través de sistemas en capas: *a*) Paralelismo a nivel de instrucción y datos en la CPU. Ver también: Organización Funcional. *b*) SIMD y paralelismo de datos heterogéneo. Ver también: Arquitecturas Heterogéneas. *c*) Concurrencia planificada en multinúcleo, tareas, actores. Ver también: Planificación. *d*) Clústeres, nubes; aprovisionamiento elástico. Ver también: Aspectos Comunes: Plataformas, APIs y Restricciones, Aspectos Comunes: Lenguajes de Plataforma y Patrones Web. *e*) Sistemas distribuidos en red. Ver también: Aplicaciones en Red. *f*) Tecnologías emergentes como la computación cuántica y la computación molecular.

Objetivos de Aprendizaje I: Explicar las compensaciones entre diferentes estrategias de mapeo en términos de rendimiento, costo y complejidad de implementación [Explicar]

Bibliografía: [Pacheco and Malensek, 2021, mei W. Hwu et al., 2022]

5.6 Programación GPU, 4 hr(s)

Competencias: AG-C09

Tópicos: I: Computación de Propósito General en GPU (GPGPU) *a*) Arquitectura de GPU: multiprocesadores de flujo (SMs), núcleos CUDA y modelo de ejecución por warp. *b*) Modelo de ejecución SIMT (Single Instruction, Multiple Threads) y sus diferencias respecto al SIMD de CPU. *c*) Interacción host-dispositivo: transferencia de datos entre memoria de CPU (host) y GPU (dispositivo) vía PCIe. *d*) Casos de uso: computación científica, aprendizaje automático, procesamiento de imágenes y simulación a gran escala. **II:** Modelo de Programación CUDA *a*) Jerarquía de ejecución CUDA: grids, bloques de hilos e hilos individuales; mapeo al hardware GPU. *b*) Definición y lanzamiento de kernels; parámetros de configuración (dimensiones de grid y bloque). *c*) Indexación de hilos: threadIdx, blockIdx, blockDim, gridDim; cálculo de índices globales en 1D, 2D y 3D. *d*) Divergencia de warp: implicaciones de las bifurcaciones dentro de un warp en el rendimiento. *e*) Primitiva de sincronización intra-bloque: __syncthreads(). **III:** Jerarquía de Memoria CUDA *a*) Memoria global: grande, alta latencia, accesible por todos los hilos; patrones de acceso coalescente para el rendimiento. *b*) Memoria compartida: memoria en chip de baja latencia compartida dentro de un bloque de hilos; conflictos de banco. *c*) Registros y memoria local: almacenamiento privado por hilo. *d*) Memoria constante y de textura: cachés de solo lectura optimizadas para patrones de acceso específicos. *e*) Memoria unificada: modelo de programación simplificado con migración automática de datos entre host y dispositivo.

Objetivos de Aprendizaje I: Escribir un kernel CUDA que realice una operación data-paralela (p. ej., suma de vectores, multiplicación de matrices) y configurar correctamente las dimensiones de grid y bloque [Escribir] **II:** Analizar el impacto de los patrones de acceso a memoria global (acceso coalescente vs. acceso entrelazado) y el uso de memoria compartida en el rendimiento de un kernel GPU [Analizar] **III:** Diseñar un algoritmo paralelo usando el modelo de programación CUDA, incluyendo la jerarquía de hilos, estrategia de asignación de memoria y transferencias de datos entre host y dispositivo [Diseñar]

Bibliografía: [Kirk and mei W. Hwu, 2016, Corporation, 2024]

5.7 Comunicación, 8 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Medios *a*) Variedades: canales (paso de mensajes o E/S), memoria compartida, heterogéneos, almacenes de datos. *b*) Dependencia de la disponibilidad y naturaleza del hardware subyacente, conectividad y protocolos; soporte del lenguaje, emulación. Ver también: Interconexión y Comunicación. **II:** Canales *a*) Medios de comunicación de parte a parte explícitos (generalmente nombrados). *b*) APIs: Sockets, constructos arquitectónicos, basados en lenguaje y de kits de herramientas, como Message Passing Interface (MPI), y constructos en capas como Llamada a Procedimiento Remoto (RPC). Ver también: Fundamentos de Redes y Comunicaciones. *c*) APIs de canales de E/S. **III:** Memoria *a*) Arquitecturas de memoria compartida en las que las partes se comunican directamente solo con la memoria en direcciones dadas, con extensiones a memoria heterogénea que admite múltiples almacenes de memoria con transferencia de datos explícita entre ellos; por ejemplo, memoria local y compartida de GPU, Acceso Directo a Memoria (DMA). *b*) Jerarquías de memoria: Múltiples capas de dominios, alcances y cachés compartidos; localidad: latencia, falso uso compartido. *c*) Propiedades de consistencia: Límites de atomicidad a nivel de bits, coherencia, orden local. **IV:** Almacenes de Datos *a*) Estructuras de datos mantenidas cooperativamente que implementan mapas y TADs relacionados. *b*) Variedades: Propiedad exclusiva, compartida, fragmentada, replicada, inmutable, versionada.

Objetivos de Aprendizaje I: Explicar las similitudes y diferencias entre: (1) La parte A envía un mensaje en el canal X con contenido 1 recibido por la parte B (2) A establece la variable compartida X a 1, leída por B (3) A establece "X=1" en un mapa compartido distribuido al que accede B [Explicar] **II:** Escribir un programa que distribuya diferentes segmentos de un conjunto de datos a múltiples trabajadores y recoja los resultados (por ejemplo, sumar segmentos de un arreglo) [Escribir] **III:** Escribir un programa paralelo que solicite datos de múltiples sitios y los resuma usando alguna forma de reducción [Escribir] **IV:** Comparar el rendimiento de versiones con y sin búfer de un programa productor-consumidor [Comparar]

Bibliografía: [van Steen and Tanenbaum, 2023, Kleppmann, 2017]

5.8 Comunicación: Propiedades y Extensiones, 3 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Una o más de las siguientes propiedades y extensiones: *a*) Topologías: Unicast, Multicast, Buzones, Conmutadores; Enrutamiento a través de redes de interconexión de hardware y software. *b*) Propiedades de concurrencia de medios: Orden, consistencia, idempotencia, superposición de comunicación con computación. *c*) Rendimiento del medio: Latencia, ancho de banda (rendimiento), contención (congestión), capacidad de respuesta (vivacidad), confiabilidad (tasas de error y pérdida), progreso basado en protocolo (acks, tiempos de espera, mediación). *d*) Propiedades de seguridad del medio: integridad, privacidad, autenticación, autorización. Ver también: Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada. *e*) Formatos de datos: Serialización, validación, cifrado, compresión. *f*) Políticas de canal: Puntos finales, sesiones, almacenamiento en búfer, respuesta a saturación (esperar vs. descartar), control de velocidad. *g*) Multiplexación y demultiplexación de muchos dispositivos o partes de E/S relativamente lentos; técnicas basadas en finalización y basadas en planificador; APIs async-await, select y polling. *h*) Formalización y análisis de comunicación por canales; por ejemplo, CSP. *i*) Aplicaciones de la teoría de colas para modelar y predecir el rendimiento.

Objetivos de Aprendizaje I: Determinar si un esquema de comunicación dado proporciona propiedades de seguridad suficientes para un uso dado [Determinar] **II:** Dar un ejemplo de un escenario en el que los envíos de mensajes bloqueantes pueden causar un punto muerto (deadlock) [Crear] **III:** Describir al menos una técnica de diseño para evitar fallos de vivacidad en programas que usan múltiples bloqueos [Describir]

Bibliografía: [van Steen and Tanenbaum, 2023, Kleppmann, 2017]

5.9 Memoria y Consistencia, 3 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Modelos de memoria, consistencia de datos y tolerancia a fallos: *a*) Modelos de memoria: consistencia secuencial y de liberación/adquisición. *b*) Gestión de memoria; incluyendo la reclamación de datos compartidos; conteo de referencias y alternativas. *c*) Colocación y transferencia masiva de datos; reducción del tráfico de mensajes y mejora de la localidad; superposición de transferencia de datos y computación; impacto del diseño de datos como array de estructuras vs. estructura de arrays. *d*) Emular memoria compartida: memoria compartida distribuida, Acceso Directo a Memoria Remota (RDMA). *e*) Consistencia de almacenes de datos: Atomicidad, linealizabilidad, transaccionalidad, coherencia, orden causal, resolución de conflictos, consistencia eventual, cadenas de bloques. *f*) Fallos, particionamiento y fallos parciales; votación; protocolos como Paxos y Raft. *g*) Compromisos de diseño entre consistencia, disponibilidad, tolerancia a partición (fallos); imposibilidad de cumplir con todos a la vez. *h*) Seguridad y confianza: Fallos bizantinos, prueba de trabajo y alternativas.

Objetivos de Aprendizaje I: Dar un ejemplo de un orden de accesos entre actividades concurrentes (por ejemplo, un programa con una carrera de datos) que no sea secuencialmente consistente [Crear] **II:** Escribir un programa que ilustre el reordenamiento de acceso a memoria o de mensajes [Escribir] **III:** Describir los méritos relativos del control de concurrencia optimista versus conservador bajo diferentes tasas de contención entre actualizaciones [Describir] **IV:** Dar un ejemplo de un escenario en el que un intento de actualización optimista podría nunca completarse [Crear] **V:** Modificar un sistema concurrente para usar un almacén de datos más escalable, confiable o disponible [Crear] **VI:** Usando una plataforma existente que admita almacenes de datos replicados, escribir un programa que mantenga un mapeo clave-valor incluso cuando uno o más hosts fallen [Escribir]

Bibliografía: [van Steen and Tanenbaum, 2023, Kleppmann, 2017]

5.10 Coordinación, 7 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Dependencias *a*) La iniciación o progreso de una actividad puede depender de otras actividades, para evitar condiciones de carrera, garantizar la terminación o cumplir otros requisitos. *b*) Garantizar el progreso evitando ciclos de dependencia, usando condiciones monótonas, eliminando dependencias no esenciales. **II:** Constructos de control y patrones de diseño: *a*) Basados en finalización: Barreras, reuniones (joins), incluido el control de terminación. *b*) Habilitados por datos: Colas, diseños productor-consumidor. *c*) Basados en condición: Polling, reintentos, retrocesos, ayuda, suspensión, señalización, tiempos de espera. *d*) Reactivos: Habilitar y activar continuaciones.

Objetivos de Aprendizaje **I:** Mostrar cómo garantizar que un programa termine correctamente cuando todas las tareas concurrentes de un conjunto hayan finalizado [Diseñar] **II:** Escribir una función que cuente eventos, como entradas de sensores o recepciones de paquetes de red, de manera eficiente [Escribir] **III:** Escribir un programa de filtro/mapeo/reducción en múltiples estilos [Escribir] **IV:** Escribir un programa en el que la terminación de un conjunto de acciones paralelas sea seguida por otra [Escribir] **V:** Escribir un servicio que cree un hilo (u otra forma de activación procedimental) para devolver una página web solicitada a cada nuevo cliente [Escribir]

Bibliografía: [Herlihy et al., 2020, van Steen and Tanenbaum, 2023]

5.11 Coordinación: Sincronización y Atomicidad, 3 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Atomicidad *a*) Instrucciones atómicas, ordenamientos de acceso local impuestos. *b*) Bloqueos y exclusión mutua; granularidad de bloqueo. *c*) Uso de bloqueos en un lenguaje específico; mantener la vivacidad sin introducir carreras. *d*) Evitar punto muerto (*deadlock*): Ordenamiento, mayor granularidad, reintentos aleatorios; retrocesos, encapsulación mediante administradores de bloqueos. *e*) Errores comunes: No bloquear o desbloquear cuando es necesario, mantener bloqueos mientras se invocan operaciones desconocidas. *f*) Evitar bloqueos: replicación, solo lectura, propiedad, y construcciones no bloqueantes.

Objetivos de Aprendizaje **I:** Mostrar cómo evitar o reparar un error de carrera en un programa dado [Analizar]

Bibliografía: [Herlihy et al., 2020]

5.12 Coordinación: Propiedades Avanzadas, 4 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Una o más de las siguientes propiedades y extensiones: *a*) Propiedades de progreso incluyendo libre de bloqueo (*lock-free*), libre de espera (*wait-free*), equidad (*fairness*), planificación por prioridad, interacciones con consistencia, confiabilidad. *b*) Rendimiento con respecto a contención, granularidad, convoy, escalabilidad. *c*) Estructuras de datos y algoritmos no bloqueantes. *d*) Propiedad y control de recursos. *e*) Variantes y alternativas de bloqueos: bloqueos de secuencia, bloqueos de lectura-escritura; Actualizar-Copiando-Lectura (RCU), reentrada; tickets; control del ciclo activo (*spinning*) versus bloqueo. *f*) Control basado en transacciones: Optimista y conservador. *g*) Bloqueo distribuido: confiabilidad. *h*) Alternativas a barreras: Relojes; contadores, relojes virtuales; flujo de datos y continuaciones; futuros y RPC; basadas en consenso, recolección de resultados con reductores y colectores. *i*) Especulación, selección, cancelación; consecuencias en observabilidad y seguridad. *j*) Control de recursos usando semáforos y variables de condición. *k*) Flujo de control: Planificación de computaciones, bucles serie-paralelo con líderes (posiblemente elegidos), tuberías y flujos, paralelismo anidado. *l*) Excepciones y fallos. Manejadores, detección, tiempos de espera, tolerancia a fallos, votación.

Objetivos de Aprendizaje I: Escribir un programa que busque especulativamente una solución mediante múltiples actividades, terminando las demás cuando se encuentre una [Escribir] **II:** Escribir un programa en el que una excepción numérica (como división por cero) en una actividad cause la terminación de las demás [Escribir] **III:** Escribir un programa para que múltiples partes acuerden la hora actual del día; discutir sus limitaciones en comparación con protocolos como el protocolo de transferencia de red (NTP) [Escribir]

Bibliografía: [Herlihy et al., 2020]

5.13 Evaluación, 7 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Requisitos de seguridad (safety) y vivacidad (*liveness*) en términos de constructos de lógica temporal para expresar "siempre" y "eventualmente" Ver también: Computación Paralela y Distribuida. **II:** Identificar, probar y reparar violaciones, incluyendo formas comunes de errores como no garantizar el orden necesario (errores de carrera), atomicidad (incluyendo errores de "verificar luego actuar") y terminación (bloqueo activo). **III:** Métricas de requisitos de rendimiento para rendimiento (*throughput*), capacidad de respuesta, latencia, disponibilidad, consumo de energía, escalabilidad, uso de recursos, costos de comunicación, espera y control de velocidad, equidad; acuerdos de nivel de servicio. Ver también: Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento. **IV:** Impacto en el rendimiento de las opciones de diseño e implementación, incluyendo granularidad, sobrecarga, costos de consenso y consumo de energía. Ver también: Diseño Sostenible y Computación Ubicua, Huella Ambiental de los Sistemas Informáticos, Efectos Sistémicos y Contexto Social. **V:** Estimar limitaciones de escalabilidad, por ejemplo usando la Ley de Amdahl o la Ley de Escalabilidad Universal. Ver también: Eventos, Herramientas y Experimentación, Métricas y Benchmarks de Rendimiento, Modelos Analíticos de Rendimiento.

Objetivos de Aprendizaje I: Revisar una especificación para habilitar el paralelismo y la distribución sin violar otras propiedades o características esenciales [Rediseñar] **II:** Explicar cómo las nociones concurrentes de seguridad (safety) y vivacidad (liveness) extienden sus contrapartes secuenciales [Explicar] **III:** Especificar un conjunto de invariantes que deben mantenerse en cada paso de cómputo de paralelismo masivo [Analizar] **IV:** Escribir un programa de prueba que pueda revelar un error de carrera de datos; por ejemplo, perder una actualización cuando dos actividades intentan incrementar una variable [Escribir] **V:** En un contexto dado, explicar hasta qué punto se esperaría que introducir paralelismo en un programa por lo demás secuencial mejoraría el rendimiento (*throughput*) y/o reduciría la latencia, y cómo podría afectar la eficiencia energética [Explicar] **VI:** Mostrar cómo cambian la escalabilidad y la eficiencia para problemas de muestra con y sin el supuesto de que el tamaño del problema cambia con el número de procesadores; además, explicar si y cómo cambiaría la escalabilidad bajo relajaciones de dependencias secuenciales [Diseñar]

Bibliografía: [Sterling et al., 2024, Pacheco and Malensek, 2021]

5.14 Evaluación: Métodos Formales, 3 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Métodos formales de verificación y análisis: *a*) Extensiones a requisitos formales secuenciales como la linealizabilidad. *b*) Especificaciones de protocolo, sesión y transacciones. *c*) Uso de herramientas como Lenguaje Unificado de Modelado (UML), Lógica Temporal de Acciones (TLA), lógicas de programa. *d*) Análisis de seguridad: seguridad y vivacidad en presencia de comportamientos hostiles o con errores de otras partes; propiedades requeridas de mecanismos de comunicación (por ejemplo, ausencia de fugas entre capas), filtrado de entrada, limitación de velocidad. Ver también: Seguridad en IA y Hardware, Control de Acceso y Aplicaciones, Principios y Mentalidad de Seguridad. *e*) Análisis estático aplicado a corrección, rendimiento (*throughput*), latencia, recursos, energía. Ver también: Diseño Sostenible y Computación Ubicua,

Huella Ambiental de los Sistemas Informáticos, Efectos Sistémicos y Contexto Social. *f*) Análisis del modelo de Grafo Acíclico Dirigido (DAG) de eficiencia algorítmica (trabajo, span, caminos críticos).

Objetivos de Aprendizaje I: Especificar y medir el comportamiento cuando un servicio es solicitado por un número inesperadamente grande de clientes [Analizar] **II:** Identificar y reparar un problema de rendimiento debido a cuellos de botella secuenciales [Analizar] **III:** Comparar empíricamente el rendimiento (*throughput*) de dos implementaciones de un diseño común (quizás usando un marco de pruebas existente) [Comparar]

Bibliografía: [Sterling et al., 2024, Pacheco and Malensek, 2021]

5.15 Evaluación: Pruebas y Medición, 2 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Herramientas de prueba y técnicas de medición: *a*) Pruebas y depuración; herramientas como detectores de carreras, fuzzers, verificadores de dependencia de bloqueos, pruebas de unidad/esfuerzo/tortura, visualizaciones, integración continua (CI), despliegue continuo (CD), y generadores de pruebas. *b*) Medir y comparar rendimiento (*throughput*), sobrecarga, espera, contención, comunicación, movimiento de datos, localidad, uso de recursos, comportamiento en presencia de números excesivos de eventos, clientes o hilos. Ver también: Eventos, Herramientas y Experimentación, Métricas y Benchmarks de Rendimiento, Modelos Analíticos de Rendimiento. *c*) Análisis específicos del dominio de aplicación y técnicas de evaluación.

Objetivos de Aprendizaje I: Identificar y reparar un problema de rendimiento debido a latencia de comunicación o datos [Analizar] **II:** Identificar y reparar un problema de rendimiento debido a sobrecarga en la gestión de recursos [Analizar] **III:** Identificar y reparar un problema de confiabilidad o disponibilidad [Analizar]

Bibliografía: [Sterling et al., 2024, Pacheco and Malensek, 2021]

5.16 Algoritmos, 4 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Expresar e implementar algoritmos en lenguajes y marcos de trabajo dados, para iniciar actividades (por ejemplo hilos), usar constructos de memoria compartida, y APIs de canales, sockets y/o llamada a procedimiento remoto (RPC). Ver también: Computación Paralela y Distribuida. *a*) Ejemplos de datos paralelos incluyendo mapeo/reducción. *b*) Uso de APIs de canal, socket y/o RPC en un lenguaje dado, con control del programa para enviar (generalmente procedimental) vs recibir (generalmente reactivo o basado en RPC). *c*) Uso de bloqueos, barreras y/o sincronizadores para mantener la vivacidad sin introducir carreras.

Objetivos de Aprendizaje I: Implementar un componente paralelo/distribuido basado en un algoritmo conocido [Implementar] **II:** Escribir un programa de datos paralelos que, por ejemplo, calcule el promedio de un arreglo de números [Escribir] **III:** Escribir un programa productor-consumidor en el que un componente genere números y otro calcule su promedio. Medir las aceleraciones cuando los números son escalares pequeños versus valores de multiprecisión grandes [Escribir]

Bibliografía: [Pacheco and Malensek, 2021, Herlihy et al., 2020]

5.17 Algoritmos: Dominios de Aplicación, 4 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Panorama de dominios de aplicación comunes en multinúcleo, reactivo, datos paralelos, clúster, nube, sistemas distribuidos abiertos y marcos de trabajo (con referencia a la siguiente tabla). *a*) Multinúcleo: Agentes de ejecución típicos: Hilos. Mecanismos de comunicación típicos: Memoria compartida, Atómicos, bloqueos. Dominios algorítmicos típicos: Gestión de recursos,

procesamiento de datos. Objetivos de ingeniería típicos: Rendimiento (*throughput*), latencia, energía. *b*) Reactivo: Agentes de ejecución típicos: Manejadores, hilos. Mecanismos de comunicación típicos: Canales de E/S. Dominios algorítmicos típicos: Servicios, tiempo real. Objetivos de ingeniería típicos: Latencia. *c*) Datos paralelos: Agentes de ejecución típicos: GPU, SIMD, aceleradores, híbridos. Mecanismos de comunicación típicos: Memoria heterogénea. Dominios algorítmicos típicos: álgebra lineal, gráficos, análisis de datos. Objetivos de ingeniería típicos: Rendimiento (*throughput*), energía. *d*) Clúster: Agentes de ejecución típicos: Hosts gestionados. Mecanismos de comunicación típicos: Sockets, canales. Dominios algorítmicos típicos: Simulación, análisis de datos. Objetivos de ingeniería típicos: Rendimiento (*throughput*). *e*) Nube: Agentes de ejecución típicos: Hosts aprovisionados. Mecanismos de comunicación típicos: APIs de servicio. Dominios algorítmicos típicos: Aplicaciones web. Objetivos de ingeniería típicos: Escalabilidad. *f*) Distribuido abierto: Agentes de ejecución típicos: Hosts autónomos. Mecanismos de comunicación típicos: Sockets, Almacenes de datos. Dominios algorítmicos típicos: Almacenes de datos y servicios tolerantes a fallos. Objetivos de ingeniería típicos: Confiabilidad.

Objetivos de Aprendizaje I: Extender un programa secuencial dirigido por eventos estableciendo una nueva actividad en un manejador de eventos (por ejemplo, un nuevo hilo en un manejador de acciones de GUI) [Diseñar] **II:** Mejorar el rendimiento de un componente secuencial introduciendo paralelismo y/o distribución [Crear] **III:** Elegir entre diferentes diseños paralelos/distribuidos para componentes de un sistema dado [Evaluar (valorar)]

Bibliografía: [Pacheco and Malensek, 2021, Herlihy et al., 2020]

5.18 Algoritmos: Dominios Algorítmicos, 4 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Uno o más de los siguientes dominios algorítmicos. Ver también: Estrategias Algorítmicas: *a*) álgebra lineal: Operaciones con vectores y matrices, precisión/estabilidad numérica, aplicaciones en análisis de datos y aprendizaje automático. *b*) Procesamiento de datos: ordenación, búsqueda y recuperación, estructuras de datos concurrentes. *c*) Grafos, búsqueda y combinatoria: Marcado, paralelización de aristas, acotamiento, especulación, análisis basado en redes. *d*) Modelado y simulación: ecuaciones diferenciales; aleatorización, problemas de N-cuerpos, algoritmos genéticos. *e*) Lógica computacional: satisfactibilidad (SAT), programación lógica concurrente. *f*) Gráficos y geometría computacional: Transformaciones, renderizado, trazado de rayos. *g*) Gestión de recursos: Asignar, colocar, reciclar y planificar procesadores, memoria, canales y hosts; recursos exclusivos vs compartidos; algoritmos estáticos, dinámicos y elásticos; Restricciones de tiempo real; Lotes, priorización, partición; descentralización mediante robo de trabajo y técnicas relacionadas. *h*) Servicios: Implementar APIs web, moneda electrónica, sistemas de transacción, juegos multijugador.

Objetivos de Aprendizaje I: Diseñar, implementar, analizar y evaluar un componente o aplicación para X que opere en un contexto dado, donde X esté en uno de los dominios listados, por ejemplo, un algoritmo genético para el diseño de una planta de fábrica [Diseñar] **II:** Criticar el diseño e implementación de un componente o aplicación existente, o uno desarrollado por compañeros de clase [Críticar (análisis crítico)] **III:** Comparar el rendimiento y la eficiencia energética de múltiples implementaciones de un diseño similar, por ejemplo, multinúcleo versus clúster versus GPU [Comparar]

Bibliografía: [Pacheco and Malensek, 2021, Herlihy et al., 2020]

6. Bibliografía

[Corporation, 2024] Corporation, N. (2024). Cuda c++ programming guide. `https://docs.nvidia.com/cuda/cuda-c-programming-guide/`. Documentación oficial de NVIDIA.

[Herlihy et al., 2020] Herlihy, M., Shavit, N., Luchangco, V., and Spear, M. (2020). *The Art of Multiprocessor Programming*. Morgan Kaufmann, 2nd edition.

[Kirk and mei W. Hwu, 2016] Kirk, D. B. and mei W. Hwu, W. (2016). *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, Cambridge, MA, 3rd edition.

[Kleppmann, 2017] Kleppmann, M. (2017). *Designing Data-Intensive Applications*. O'Reilly Media, 1st edition.

[mei W. Hwu et al., 2022] mei W. Hwu, W., Kirk, D. B., and Hajj, I. E. (2022). *Programming Massively Parallel Processors: A Hands-on Approach*. Morgan Kaufmann, 4th edition.

[Pacheco and Malensek, 2021] Pacheco, P. S. and Malensek, M. (2021). *An Introduction to Parallel Programming*. Morgan Kaufmann, 2nd edition.

[Sterling et al., 2024] Sterling, T., Brodowicz, M., and Anderson, M. (2024). *High Performance Computing: Modern Systems and Practices*. Morgan Kaufmann, 2nd edition.

[van Steen and Tanenbaum, 2023] van Steen, M. and Tanenbaum, A. S. (2023). *Distributed Systems*. Maarten van Steen, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS402 Proyecto de tesis 1 (Obligatorio)

1. Información general

- **Semestre**: 8vo Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - CS401 Metodología de la Investigación (4to Sem-Pág 353)

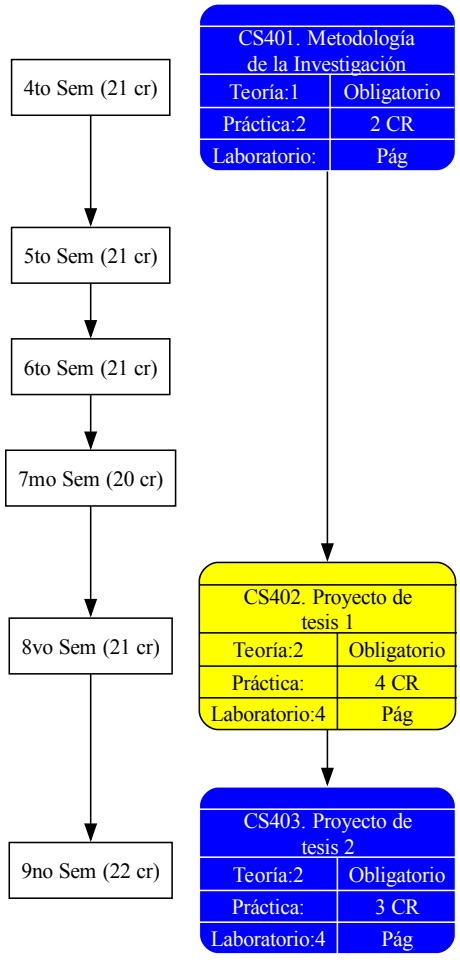

Figura 5.47: Mapa de Conexión. CS402 Proyecto de tesis 1

2. Fundamentación

Este curso tiene por objetivo que el alumno conduzca un estudio profundo del estado del arte de un tópico específico elegido para su proyecto de tesis. El alumno deberá producir un reporte técnico con experimentos preliminares comparativos y formular una propuesta de investigación, culminando junto a un asesor especializado en el área elegida.

3. Objetivos del curso.

- Realizar un estudio profundo del estado del arte en un tópico específico del área de computación.

- Demostrar dominio de la literatura relevante mediante análisis crítico y comparativo de enfoques existentes.
- Diseñar y ejecutar experimentos preliminares que reproduzcan y comparen técnicas del estado del arte.
- Redactar un reporte técnico de calidad con marco teórico, metodología y resultados experimentales.
- Formular una propuesta de investigación con pregunta, objetivos y cronograma.
- Seleccionar un asesor de tesis que domine el área de investigación elegida.
- Los entregables de este curso son:

  Avance parcial: Bibliografía sólida y borrador del marco teórico en formato LaTeX.

  Final: Reporte Técnico con experimentos preliminares comparativos y propuesta de investigación aprobada por el asesor.

4. Competencias Generales (*Outcomes*)

AG-C02) Ética: Aplica principios éticos y se compromete con la ética profesional y las normas de la práctica profesional de la computación. (Evaluar)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Evaluar)

AG-C05) Gestión de Proyectos: Aplica los principios de gestión en computación para gestionar proyectos. (Evaluar)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Evaluar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Evaluar)

5. Contenido

5.1 Profundización en el Estado del Arte Específico, 10 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: **I:** Selección y refinamiento del tópico de investigación en coordinación con el asesor. **II:** Búsqueda exhaustiva multi-fuente: IEEE Xplore, ACM Digital Library, Google Scholar, arXiv y bases de datos especializadas. **III:** Identificación de trabajos seminales y artículos más citados del área. **IV:** Construcción del grafo de citaciones y análisis de la evolución temporal del tópico. **V:** Identificación de subáreas, variantes y enfoques principales del campo.

Objetivos de Aprendizaje **I:** Construir un mapa completo de la literatura de un tópico específico identificando los trabajos fundamentales. [Evaluar] **II:** Analizar la evolución temporal de un área de investigación identificando tendencias y brechas. [Evaluar] **III:** Organizar la literatura en subcategorías coherentes que reflejen la estructura del campo. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008, CiteSeer.IST, 2008]

5.2 Análisis Crítico y Comparativo de Enfoques, 10 hr(s)

Competencias: AG-C08,AG-C09

Tópicos: **I:** Análisis crítico de propuestas: metodología, datasets, métricas y limitaciones de cada enfoque. **II:** Construcción de tablas comparativas de propuestas existentes con criterios objetivos. **III:** Identificación de brechas de investigación (research gaps) no resueltas en el área. **IV:** Formulación de la pregunta de investigación propia a partir del análisis del estado del arte. **V:** Marco teórico: síntesis estructurada y coherente del conocimiento relevante del área.

Objetivos de Aprendizaje I: Comparar sistemáticamente múltiples enfoques usando criterios objetivos y cuantificables. [Evaluar] **II:** Identificar brechas de investigación no resueltas en un área específica de computación. [Evaluar] **III:** Formular una pregunta de investigación propia sustentada en el análisis del estado del arte. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008, CiteSeer.IST, 2008]

5.3 Diseño y Ejecución de Experimentos Preliminares, 12 hr(s)

Competencias: AG-C05,AG-C08,AG-C09

Tópicos: I: Diseño experimental: variables, métricas y posibles benchmarks del área. **II:** Reproducción de resultados de trabajos relevantes del estado del arte. **III:** Implementación de baselines para comparación objetiva de enfoques. **IV:** Análisis estadístico básico: pruebas de significancia e intervalos de confianza. **V:** Gestión de datos experimentales: datasets, configuraciones, versiones y registro de resultados.

Objetivos de Aprendizaje I: Reproducir los resultados de al menos dos trabajos relevantes del estado del arte del área. [Usar] **II:** Diseñar y ejecutar un experimento comparativo preliminar con baselines definidos. [Usar] **III:** Interpretar y reportar resultados experimentales con rigor estadístico apropiado. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.4 Redacción del Reporte Técnico, 12 hr(s)

Competencias: AG-C04,AG-C09

Tópicos: I: Estructura del reporte técnico: introducción, marco teórico, metodología, experimentos, resultados y conclusiones. **II:** Redacción del marco teórico: organización temática y síntesis crítica de la literatura. **III:** Documentación de la metodología y configuración experimental en LaTeX. **IV:** Presentación de resultados: tablas, gráficas y análisis comparativos de alta calidad. **V:** Ciclos de revisión: escritura, revisión con asesor e incorporación de retroalimentación.

Objetivos de Aprendizaje I: Redactar un reporte técnico completo con marco teórico y experimentos preliminares. [Usar] **II:** Estructurar y presentar resultados experimentales con visualizaciones claras y precisas. [Usar] **III:** Aplicar ciclos de revisión para mejorar la calidad, coherencia y rigor del documento. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.5 Propuesta de Investigación y Selección de Asesor, 8 hr(s)

Competencias: AG-C02,AG-C05,AG-C09

Tópicos: I: Estructura de una propuesta de investigación: problema, motivación y objetivos. **II:** Definición del alcance del proyecto de tesis: metodología, contribución esperada y cronograma. **III:** Análisis de viabilidad: recursos, tiempo, riesgos y contingencias del proyecto. **IV:** Criterios para la selección del asesor de tesis: experiencia, disponibilidad y alineación temática. **V:** Proceso de formalización del vínculo investigador-asesor en el programa académico.

Objetivos de Aprendizaje I: Elaborar una propuesta de investigación que defina el problema, objetivos y posible solución. [Evaluar] **II:** Seleccionar un asesor con experiencia demostrada en el área de investigación elegida. [Usar] **III:** Definir un plan de tesis viable con objetivos, alcance, cronograma y análisis de riesgos. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.6 Presentación y Defensa del Estado del Arte, 8 hr(s)

Competencias: AG-C02,AG-C04,AG-C05

Tópicos: **I:** Preparación de la presentación oral del reporte técnico ante el comité evaluador. **II:** Técnicas de defensa ante preguntas y cuestionamientos de un comité académico. **III:** Elaboración de póster técnico del estado del arte para presentación en simposio. **IV:** Incorporación de retroalimentación del comité evaluador en la revisión final del reporte. **V:** Planificación de la transición a CS403: objetivos, contribución esperada y siguientes pasos.

Objetivos de Aprendizaje **I:** Presentar y defender oralmente el reporte técnico ante un comité académico con rigor. [Evaluar] **II:** Incorporar constructivamente la retroalimentación del comité en una revisión del reporte. [Usar] **III:** Planificar la transición hacia el proyecto de tesis (CS403) con objetivos y pasos claros. [Usar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

6. Bibliografía

[ACM, 2008] ACM (2008). *Digital Library.* Association for Computing Machinery. http://portal.acm.org/dl.cfm.

[CiteSeer.IST, 2008] CiteSeer.IST (2008). *Scientific Literature Digital Library.* College of Information Sciences and Technology, Penn State University. http://citeseer.ist.psu.edu.

[IEEE-Computer Society, 2008] IEEE-Computer Society (2008). *Digital Library.* IEEE-Computer Society. http://www.computer.org/publications/dlib.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

EX301 Actividades Extracurriculares (Obligatorio)

1. Información general

- **Semestre**: 8vo Sem. **Créditos**: 2
- **Horas del curso**: **Teoría**: 1 horas; **Práctica**: 2 horas;
- **Prerrequisitos**: Ninguno

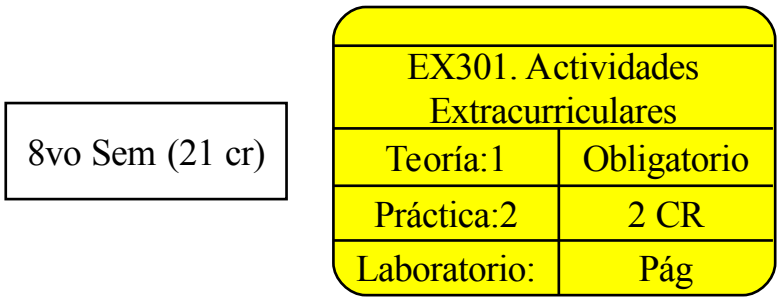

Figura 5.48: Mapa de Conexión. EX301 Actividades Extracurriculares

2. Fundamentación

Este curso registra la participación del estudiante en actividades complementarias a la formación académica, como talleres, congresos, proyección social, voluntariados o competencias técnicas. Su objetivo es fomentar el desarrollo integral, habilidades blandas y compromiso social.

3. Objetivos del curso.

- Participar en actividades que complementen su formación profesional.
- Desarrollar habilidades de comunicación y trabajo en equipo.
- Contribuir a iniciativas de impacto social o técnico.

4. Competencias Generales (*Outcomes*)

AG-C01) El Profesional y el Mundo: Analiza y evalúa el impacto de las soluciones a problemas complejos de computación en el desarrollo sostenible de la sociedad. (Familiarizarse)

AG-C06) Aprendizaje a lo largo de la vida: Reconoce la importancia del aprendizaje continuo y la adaptación a nuevas tecnologías. (Familiarizarse)

5. Contenido

5.1 Actividades Extracurriculares, 48 hr(s)

Competencias: AG-C01,AG-C06

Tópicos: **I:** Talleres técnicos o interdisciplinarios. **II:** Asistencia a congresos, seminarios o charlas. **III:** Proyección social (voluntariados, mentorías). **IV:** Competencias académicas (hackathons, ferias científicas).

Objetivos de Aprendizaje I: Demostrar participación en al menos 3 actividades [Familiarizarse]. **II:** Elaborar informes reflexivos sobre las experiencias [Usar]. **III:** Colaborar en equipos para proyectos sociales o técnicos [Evaluar].

Bibliografía: [ACM, 2019, ABET Computing Accreditation Commission, 2022]

6. Bibliografía

[ABET Computing Accreditation Commission, 2022] ABET Computing Accreditation Commission (2022). Abet criteria for student professional development. Technical report, ABET.

[ACM, 2019] ACM (2019). Guidelines for extracurricular activities in computing. Technical report, ACM.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

ET201 Formación de Empresas de Base Tecnológica I (Obligatorio)

1. Información general

- **Semestre**: 8vo Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas;
- **Prerrequisitos**:
  - FG350 Liderazgo y Desempeño (7mo Sem-Pág 440)

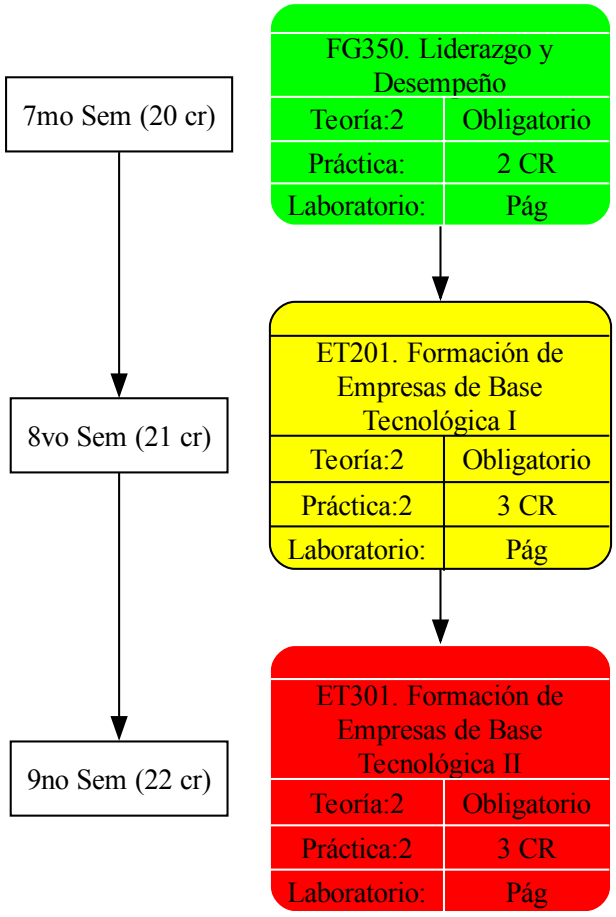

Figura 5.49: Mapa de Conexión. ET201 Formación de Empresas de Base Tecnológica I

2. Fundamentación

Este es el primer curso dentro del área de formación de empresas de base tecnológica, tiene como objetivo dotar al futuro profesional de conocimientos, actitudes y aptitudes que le permitan elaborar un plan de negocio para una empresa de base tecnológica. El curso está dividido en las siguientes unidades: Introducción, Creatividad, De la idea a la oportunidad, el modelo Canvas, Customer Development y Lean Startup, Aspectos Legales y Marketing, Finanzas de la empresa y Presentación.

Se busca aprovechar el potencial creativo e innovador y el esfuerzo de los alumnos en la creación de nuevas empresas.

3. Objetivos del curso.

- Que el alumno conozca como elaborar un plan de negocio para dar inicio a una empresa de base tecnológica.
- Que el alumno sea capaz de realizar, usando modelos de negocio, la concepción y presentación de una propuesta de negocio.

4. Competencias Generales (*Outcomes*)

Sin elementos para esta tabla

5. Contenido

5.1 Introducción, 5 hr(s)

Competencias:

Tópicos: **I:** Emprendedor, emprendedurismo e innovación tecnológica **II:** Modelos de negocio **III:** Formación de equipos

Objetivos de Aprendizaje **I:** Identificar características de los emprendedores [Familiarizarse] **II:** Introducir modelos de negocio [Familiarizarse]

Bibliografía: [Byers et al., 2010, Osterwalder and Pigneur, 2010, Garzozi-Pincay et al., 2014]

5.2 Creatividad, 5 hr(s)

Competencias:

Tópicos: **I:** Visión **II:** Misión **III:** La Propuesta de valor **IV:** Creatividad e invención **V:** Tipos y fuentes de innovación **VI:** Estrategia y Tecnología **VII:** Escala y ámbito

Objetivos de Aprendizaje **I:** Plantear correctamente la vision y misión de empresa [Usar] **II:** Caracterizar una propuesta de valor innovadora [Evaluar] **III:** Identificar los diversos tipos y fuentes de innovación [Familiarizarse]

Bibliografía: [Byers et al., 2010, Blank and Dorf, 2012, Garzozi-Pincay et al., 2014]

5.3 De la Idea a la Oportunidad, 5 hr(s)

Competencias:

Tópicos: **I:** Estrategia de la Empresa **II:** Barreras **III:** Ventaja competitiva sostenible **IV:** Alianzas **V:** Aprendizaje organizacional **VI:** Desarrollo y diseño de productos

Objetivos de Aprendizaje **I:** Conocer estrategias empresariales [Familiarizarse] **II:** Caracterizar barreras y ventajas competitivas [Familiarizarse]

Bibliografía: [Byers et al., 2010, Osterwalder and Pigneur, 2010, Ries, 2011, Garzozi-Pincay et al., 2014]

5.4 El Modelo Canvas, 20 hr(s)

Competencias:

Tópicos: **I:** Creación de un nuevo negocio **II:** El plan de negocio **III:** Canvas **IV:** Elementos del Canvas

Objetivos de Aprendizaje **I:** Conocer los elementos del modelo Canvas [Usar] **II:** Elaborar un plan de negocio basado en el modelo Canvas [Usar]

Bibliografía: [Osterwalder and Pigneur, 2010, Blank and Dorf, 2012, Garzozi-Pincay et al., 2014]

5.5 Customer Development y Lean Startup, 20 hr(s)

Competencias:

Tópicos: I: Aceleración versus incubación **II:** Customer Development **III:** Lean Startup

Objetivos de Aprendizaje I: Conocer y aplicar el modelo Customer Development [Usar] **II:** Conocer y aplicar el modelo Lean Startup [Usar]

Bibliografía: [Blank and Dorf, 2012, Ries, 2011, Garzozi-Pincay et al., 2014]

5.6 Aspectos Legales y Marketing, 5 hr(s)

Competencias:

Tópicos: I: Aspectos Legales y tributarios para la constitución de la empresa **II:** Propiedad intelectual **III:** Patentes **IV:** Copyrights y marca registrada **V:** Objetivos de marketing y segmentos de mercado **VI:** Investigación de mercado y búsqueda de clientes

Objetivos de Aprendizaje I: Conocer los aspectos legales necesarios para la formación de una empresa tecnológica [Familiarizarse] **II:** Identificar segmentos de mercado y objetivos de marketing [Familiarizarse]

Bibliografía: [Byers et al., 2010, Ries, 2011, Congreso de la Republica del Perú, 1996, de la Republica del Peru, 1997, Garzozi-Pincay et al., 2014]

5.7 Finanzas de la Empresa, 5 hr(s)

Competencias:

Tópicos: I: Modelo de costos **II:** Modelo de utilidades **III:** Precio **IV:** Plan financiero **V:** Formas de financiamiento **VI:** Fuentes de capital **VII:** Capital de riesgo

Objetivos de Aprendizaje I: Definir um modelo de costos y utilidades [Evaluar] **II:** Conocer las diversas fuentes de financiamento [Familiarizarse]

Bibliografía: [Byers et al., 2010, Blank and Dorf, 2012, Garzozi-Pincay et al., 2014]

5.8 Presentación, 5 hr(s)

Competencias:

Tópicos: I: The Elevator Pitch **II:** Presentación **III:** Negociación

Objetivos de Aprendizaje I: Conocer las diversas formas de presentar propuestas de negocio [Familiarizarse] **II:** Realizar la presentación de una propuesta de negocio [Usar]

Bibliografía: [Byers et al., 2010, Blank and Dorf, 2012, Garzozi-Pincay et al., 2014]

6. Bibliografía

[Blank and Dorf, 2012] Blank, S. and Dorf, B. (2012). *The Startup Owner's Manual: The Step-By-Step Guide for Building a Great Company*. K and S Ranch.

[Byers et al., 2010] Byers, T., Dorf, R., and Nelson, A. (2010). *Technology Ventures: From Idea to Enterprise*. McGraw-Hill Science.

[Congreso de la Republica del Perú, 1996] Congreso de la Republica del Perú (1996). *Decreto Legislativo Nº823. Ley de la Propiedad Industrial*. El Peruano.

[de la Republica del Peru, 1997] de la Republica del Peru, C. (1997). *Ley Nº26887. Ley General de Sociedades*. El Peruano.

[Garzozi-Pincay et al., 2014] Garzozi-Pincay, R., Messina-Scolaro, M., Moncada-Marino, C., Ochoa-Luna, J., Ilabel-Pérez, G., and Zambrano-Segura, R. (2014). *Planes de Negocios para Emprendedores*. Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn).

[Osterwalder and Pigneur, 2010] Osterwalder, A. and Pigneur, Y. (2010). *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. Wiley.

[Ries, 2011] Ries, E. (2011). *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI268 Visión Computacional (Electivo)

1. Información general

- **Semestre**: 8vo Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - AI264 Aprendizaje Profundo (7mo Sem-Pág 435)

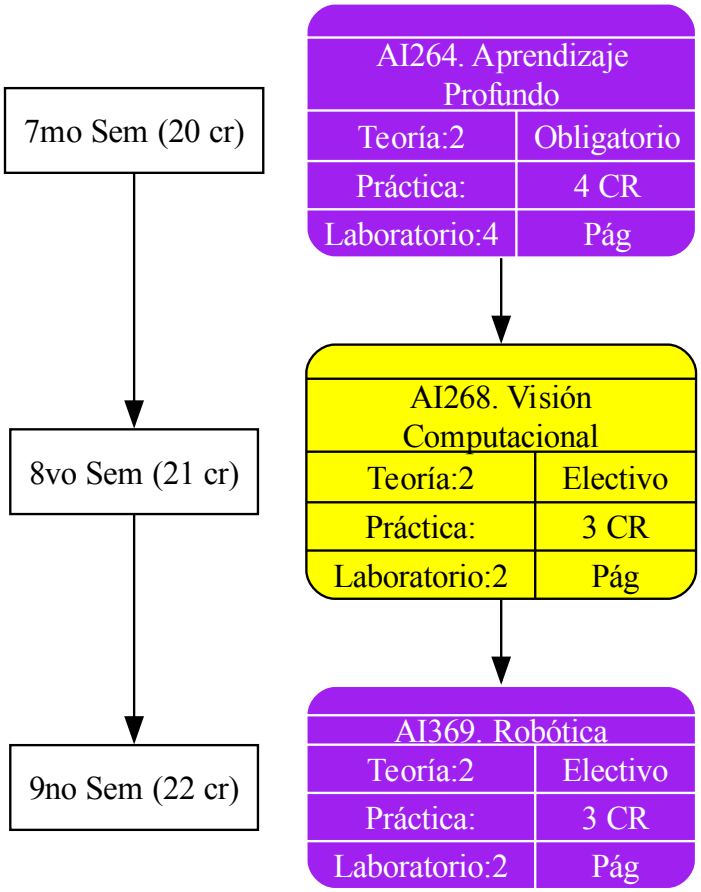

Figura 5.50: Mapa de Conexión. AI268 Visión Computacional

2. Fundamentación

La Visión por Computadora es un área clave de la Inteligencia Artificial que permite a las máquinas interpretar y comprender el mundo visual. Este curso cubre técnicas fundamentales para procesamiento de imágenes, extracción de características y reconocimiento de objetos. Con el advenimiento del aprendizaje profundo, la visión por computadora se ha vuelto esencial para vehículos autónomos, imágenes médicas y realidad aumentada. Los estudiantes aprenderán los fundamentos matemáticos y enfoques algorítmicos para resolver problemas de percepción visual del mundo real.

3. Objetivos del curso.

- Comprender los fundamentos de la formación de imágenes y la representación digital.
- Dominar técnicas de procesamiento de imágenes como filtrado y detección de bordes.
- Aplicar métodos de extracción de características para identificar patrones clave en imágenes.
- Implementar detección y reconocimiento de objetos usando modelos de aprendizaje profundo.
- Analizar técnicas de visión 3D y estimación de movimiento.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Evaluar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Introducción a la IA: Conceptos y Comportamiento Inteligente, 5 hr(s)

Competencias: AG-C08

Tópicos: **I:** Visión general de los problemas de IA, Ejemplos de aplicaciones recientes exitosas de IA **II:** Definiciones de agentes con ejemplos (por ejemplo, reactivos, deliberativos) **III:** ¿Qué es el comportamiento inteligente?: *a*) La prueba de Turing y sus defectos *b*) Entrada y salida multimodal *c*) Simulación del comportamiento inteligente *d*) Razonamiento racional versus no racional **IV:** Visión general de las Aplicaciones de IA, crecimiento e impacto (económico, social, ético) **V:** Problemas filosóficos **VI:** Historia de la IA

Objetivos de Aprendizaje **I:** Describir la prueba de Turing y el experimento mental de la "Habitación China" [Explicar] **II:** Diferenciar entre razonamiento/comportamiento óptimo y razonamiento/comportamiento similar al humano [Evaluar] **III:** Diferenciar los términos: IA, aprendizaje automático y aprendizaje profundo [Explicar]

Bibliografía: [Szeliski, 2022, Gonzalez and Woods, 2018]

5.2 Características del Problema y Naturaleza de los Agentes, 5 hr(s)

Competencias: AG-C08

Tópicos: **I:** Características del problema: *a*) Totalmente observable versus parcialmente observable *b*) Agente único versus multiagente *c*) Determinista versus estocástico *d*) Estático versus dinámico *e*) Discreto versus continuo **II:** Naturaleza de los agentes: *a*) Autonomía autónoma, semiautónoma, de iniciativa mixta *b*) Reflexivos, basados en metas y basados en utilidad *c*) Toma de decisiones bajo incertidumbre y con información incompleta *d*) La importancia de la percepción y las interacciones ambientales *e*) Agentes basados en aprendizaje *f*) Agentes corporeizados: 1) sensores, dinámicas, efectores **III:** Practicar la identificación de características del problema en entornos de ejemplo **IV:** Profundidad adicional sobre la naturaleza de los agentes con ejemplos **V:** Profundidad adicional sobre Aplicaciones de IA, Crecimiento e Impacto (económico, social, ético, seguridad)

Objetivos de Aprendizaje **I:** Enumerar las características de un problema específico [Aplicar]

Bibliografía: [Szeliski, 2022, Gonzalez and Woods, 2018]

5.3 Percepción y Visión por Computadora, 12 hr(s)

Competencias: AG-C08

Tópicos: **I:** Visión por computadora: *a*) Adquisición, representación, procesamiento y propiedades de imágenes *b*) Representación de formas, reconocimiento de objetos y segmentación *c*) Análisis de movimiento *d*) Modelos generativos **II:** Audio y reconocimiento de voz **III:** Tacto y propiocepción **IV:** Otras modalidades (por ejemplo, olfato) **V:** Modularidad en el reconocimiento **VI:** Enfoques para el reconocimiento de patrones: *a*) Algoritmos de clasificación y medidas de calidad de clasificación *b*) Técnicas estadísticas *c*) Técnicas de aprendizaje profundo

Objetivos de Aprendizaje I: Resumir la importancia del reconocimiento de imágenes y objetos en IA e indicar varias aplicaciones significativas de esta tecnología [Resumir] **II:** Listar al menos tres enfoques de segmentación de imágenes, como umbralización, algoritmos basados en bordes y basados en regiones, junto con sus características definitorias, fortalezas y debilidades [Listar/Enumerar] **III:** Implementar reconocimiento de objetos 2D basado en representaciones de forma basadas en contorno y/o región [Aplicar] **IV:** Distinguir los objetivos del reconocimiento de sonido, reconocimiento de voz y reconocimiento de hablante, e identificar cómo se manejará diferente la señal de audio cruda en cada uno de estos casos [Distinguir] **V:** Proporcionar al menos dos ejemplos de transformación de una fuente de datos de un dominio sensorial a otro, por ejemplo, datos táctiles interpretados como imágenes 2D de banda única [Generar] **VI:** Implementar un algoritmo de extracción de características en datos reales, por ejemplo, un detector de bordes o esquinas para imágenes o vectores de coeficientes de Fourier que describen un segmento corto de señal de audio [Aplicar] **VII:** Implementar un algoritmo que combine características en percepciones de nivel superior, por ejemplo, un contorno o polígono a partir de primitivas visuales o hipótesis de fonemas a partir de una señal de audio [Aplicar] **VIII:** Implementar un algoritmo de clasificación que segmenta percepciones de entrada en categorías de salida y evalúa cuantitativamente la clasificación resultante [Aplicar] **IX:** Evaluar el rendimiento de la extracción de características subyacente, relativo a al menos un enfoque alternativo posible (ya sea implementado o no) en su contribución a la tarea de clasificación (8), anterior [Evaluar] **X:** Describir al menos tres enfoques de clasificación, sus prerrequisitos para aplicabilidad, sus fortalezas y sus limitaciones [Describir] **XI:** Implementar y evaluar una solución de aprendizaje profundo para problemas en visión por computadora, como reconocimiento de objetos o escenas [Aplicar]

Bibliografía: [Szeliski, 2022, Prince, 2012]

5.4 Fundamentos de Aprendizaje Automático, 5 hr(s)

Competencias: AG-C08

Tópicos: **I:** Definición y ejemplos de una amplia variedad de tareas de aprendizaje automático: *a*) Aprendizaje supervisado: 1) Clasificación 2) Regresión *b*) Aprendizaje por refuerzo *c*) Aprendizaje no supervisado: 1) Agrupamiento **II:** Ideas fundamentales: *a*) Teorema de no hay almuerzo gratis: ningún aprendiz puede resolver todos los problemas; las decisiones de diseño de representación tienen consecuencias. *b*) Fuentes de error e indecidibilidad en el aprendizaje automático **III:** Un aprendizaje supervisado simple basado en estadísticas como regresión lineal o árboles de decisión: *a*) Enfocarse en cómo funcionan sin entrar en detalles matemáticos o de optimización; suficiente para entender y usar implementaciones existentes correctamente **IV:** El problema del sobreajuste/control de la complejidad de la solución (regularización, poda - solo intuición): *a*) El compromiso entre sesgo (subajuste) - varianza (sobreajuste)

Objetivos de Aprendizaje I: Describir las diferencias entre los tres estilos principales de aprendizaje (supervisado, por refuerzo y no supervisado) y determinar cuál es apropiado para un dominio de problema particular [Explicar] **II:** Diferenciar los términos: IA, aprendizaje automático y aprendizaje profundo [Evaluar] **III:** Formular una aplicación como un problema de clasificación, incluyendo las características de entrada disponibles y la salida a predecir (por ejemplo, identificar caracteres alfabéticos a partir de entrada de cuadrícula de píxeles) [Aplicar] **IV:** Identificar

el sobreajuste en el contexto de un problema y curvas de aprendizaje y describir soluciones al sobreajuste [Evaluar] **V:** Explicar cómo funciona el aprendizaje automático como un proceso de optimización/búsqueda [Explicar]

Bibliografía: [Goodfellow et al., 2016, Russell and Norvig, 2020]

5.5 Aprendizaje Automático: Datos, Representaciones y Evaluación, 5 hr(s)

Competencias: AG-C08

Tópicos: I: Trabajar con Datos: *a*) Preprocesamiento de datos: 1) Importancia y peligros de las elecciones de preprocesamiento *b*) Manejo de valores faltantes (imputación, marcar como faltante): 1) Implicaciones de imputar versus marcar como faltante *c*) Codificación de variables categóricas, codificación de datos de valor real *d*) Normalización/estandarización *e*) énfasis en datos reales, no ejemplos de libros de texto **II:** Representaciones: *a*) Espacios de hipótesis y complejidad *b*) Expansión de características de base simple, como elevar al cuadrado características univariadas *c*) Representaciones de características aprendidas **III:** Evaluación del aprendizaje automático: *a*) Separación de conjuntos de entrenamiento, validación y prueba *b*) Métricas de rendimiento para clasificadores *c*) Estimación del rendimiento en prueba en datos retenidos *d*) Ajuste de los parámetros de un modelo de aprendizaje automático con un conjunto de validación *e*) Importancia de entender qué está haciendo un modelo, dónde están sus peligros/limitaciones y las implicaciones de sus decisiones

Objetivos de Aprendizaje I: Aplicar dos o más algoritmos de aprendizaje estadístico simple a una tarea de clasificación y medir la precisión de los clasificadores [Aplicar] **II:** Implementar un algoritmo de aprendizaje estadístico y el proceso de optimización correspondiente para entrenar el clasificador y obtener una predicción en nuevos datos [Aplicar] **III:** Explicar los procedimientos adecuados de evaluación de AA, incluyendo las diferencias entre rendimiento en entrenamiento y prueba, y qué puede salir mal con el proceso de evaluación llevando a informes inexactos del rendimiento de AA [Explicar] **IV:** Comparar dos algoritmos de aprendizaje automático en un conjunto de datos, implementando la metodología de preprocesamiento de datos y evaluación (por ejemplo, métricas y manejo de divisiones entrenamiento/prueba) desde cero [Aplicar]

Bibliografía: [Goodfellow et al., 2016, Russell and Norvig, 2020]

5.6 Redes Neuronales y Ética, 4 hr(s)

Competencias: AG-C08

Tópicos: I: Redes neuronales básicas: *a*) Fundamentos de comprensión de cómo funcionan las redes neuronales y su proceso de entrenamiento, sin detalles de los cálculos *b*) Introducción básica a redes neuronales generativas (por ejemplo, modelos de lenguaje grandes) **II:** ética para el Aprendizaje Automático: *a*) Enfocarse en datos reales, escenarios reales y estudios de caso *b*) Sesgo de conjunto de datos/algorítmico/de evaluación y consecuencias no deseadas

Objetivos de Aprendizaje I: Describir el proceso de entrenamiento de redes neuronales y las representaciones aprendidas resultantes [Explicar]

Bibliografía: [Goodfellow et al., 2016, Russell and Norvig, 2020]

5.7 Formulación del Aprendizaje Automático y Evaluación, 3 hr(s)

Competencias: AG-C09

Tópicos: I: Formulación del aprendizaje automático simple como un problema de optimización, como regresión lineal por mínimos cuadrados o regresión logística: *a*) Función objetivo *b*) Descenso de gradiente *c*) Regularización para evitar sobreajuste (formulación matemática) **II:** Conjuntos de modelos: *a*) Combinación de mayoría ponderada simple **III:** Evaluación de rendimiento: *a*) Otras métricas para clasificación (por ejemplo, error, precisión, exhaustividad) *b*) Métricas

de rendimiento para regresores *c*) Matriz de confusión *d*) Validación cruzada: 1) Ajuste de parámetros (búsqueda en cuadrícula/aleatoria, mediante validación cruzada)

Objetivos de Aprendizaje I: Comparar y contrastar varias técnicas de aprendizaje (por ejemplo, árboles de decisión, regresión logística, Bayes ingenuo, redes neuronales y redes de creencia), proporcionando ejemplos de cuándo cada estrategia es superior [Comparar] **II:** Evaluar el rendimiento de un sistema de aprendizaje simple en un conjunto de datos del mundo real [Evaluar] **III:** Caracterizar el estado del arte en teoría del aprendizaje, incluyendo sus logros y limitaciones [Explicar] **IV:** Explicar el problema del sobreajuste, junto con técnicas para detectar y manejar el problema [Explicar] **V:** Explicar el triple compromiso entre el tamaño de un espacio de hipótesis, el tamaño del conjunto de entrenamiento y la precisión del rendimiento [Explicar]

Bibliografía: [Goodfellow et al., 2016]

5.8 Aprendizaje Profundo, RL y Ética, 3 hr(s)

Competencias: AG-C09

Tópicos: I: Aprendizaje profundo: *a*) Redes feed-forward profundas (solo intuición, sin matemáticas) *b*) Redes neuronales convolucionales (solo intuición, sin matemáticas) *c*) Visualización de representaciones de características aprendidas de redes profundas *d*) Otras arquitecturas (NN generativas, NN recurrentes, transformers, etc.) **II:** Visión general de los métodos de aprendizaje por refuerzo **III:** Dos o más aplicaciones de algoritmos de aprendizaje automático: *a*) Por ejemplo, medicina y salud, economía, visión, lenguaje natural, robótica, juego **IV:** ética para el Aprendizaje Automático: *a*) Continuar enfocándose en datos reales, escenarios reales y estudios de caso *b*) Privacidad *c*) Equidad *d*) Propiedad intelectual *e*) Explicabilidad

Objetivos de Aprendizaje I: Visualizar el progreso del entrenamiento de una red neuronal a través de curvas de aprendizaje en un kit de herramientas establecido (por ejemplo, TensorBoard) y visualizar las características aprendidas de la red [Aplicar] **II:** Dada una aplicación real de aprendizaje automático, describir problemas éticos respecto a las elecciones de datos, pasos de preprocesamiento, selección de algoritmo y visualización/presentación de resultados [Aplicar]

Bibliografía: [Goodfellow et al., 2016]

5.9 Percepción y Visión por Computadora, 12 hr(s)

Competencias: AG-C08

Tópicos: I: Visión por computadora: *a*) Adquisición, representación, procesamiento y propiedades de imágenes *b*) Representación de formas, reconocimiento de objetos y segmentación *c*) Análisis de movimiento *d*) Modelos generativos **II:** Audio y reconocimiento de voz **III:** Tacto y propiocepción **IV:** Otras modalidades (por ejemplo, olfato) **V:** Modularidad en el reconocimiento **VI:** Enfoques para el reconocimiento de patrones: *a*) Algoritmos de clasificación y medidas de calidad de clasificación *b*) Técnicas estadísticas *c*) Técnicas de aprendizaje profundo

Objetivos de Aprendizaje I: Resumir la importancia del reconocimiento de imágenes y objetos en IA e indicar varias aplicaciones significativas de esta tecnología [Resumir] **II:** Listar al menos tres enfoques de segmentación de imágenes, como umbralización, algoritmos basados en bordes y basados en regiones, junto con sus características definitorias, fortalezas y debilidades [Listar/Enumerar] **III:** Implementar reconocimiento de objetos 2D basado en representaciones de forma basadas en contorno y/o región [Aplicar] **IV:** Distinguir los objetivos del reconocimiento de sonido, reconocimiento de voz y reconocimiento de hablante, e identificar cómo se manejará diferente la señal de audio cruda en cada uno de estos casos [Distinguir] **V:** Proporcionar al menos dos ejemplos de transformación de una fuente de datos de un dominio sensorial a otro, por ejemplo, datos táctiles interpretados como imágenes 2D de banda única [Generar] **VI:** Implementar un algoritmo de extracción de características en datos reales, por ejemplo, un detector de bordes o esquinas para imágenes o vectores de coeficientes de Fourier que describen un segmento corto de señal de audio [Aplicar] **VII:** Implementar un algoritmo que combine características

en percepciones de nivel superior, por ejemplo, un contorno o polígono a partir de primitivas visuales o hipótesis de fonemas a partir de una señal de audio [Aplicar] **VIII:** Implementar un algoritmo de clasificación que segmenta percepciones de entrada en categorías de salida y evalúa cuantitativamente la clasificación resultante [Aplicar] **IX:** Evaluar el rendimiento de la extracción de características subyacente, relativo a al menos un enfoque alternativo posible (ya sea implementado o no) en su contribución a la tarea de clasificación (8), anterior [Evaluar] **X:** Describir al menos tres enfoques de clasificación, sus prerrequisitos para aplicabilidad, sus fortalezas y sus limitaciones [Describir] **XI:** Implementar y evaluar una solución de aprendizaje profundo para problemas en visión por computadora, como reconocimiento de objetos o escenas [Aplicar]

Bibliografía: [Szeliski, 2022]

6. Bibliografía

[Gonzalez and Woods, 2018] Gonzalez, R. C. and Woods, R. E. (2018). *Digital Image Processing*. Pearson, 4th edition.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Prince, 2012] Prince, S. J. D. (2012). *Computer Vision: Models, Learning, and Inference*. Cambridge University Press.

[Russell and Norvig, 2020] Russell, S. and Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition.

[Szeliski, 2022] Szeliski, R. (2022). *Computer Vision: Algorithms and Applications*. Springer, 2nd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS391 Ingeniería de Software III (Electivo)

1. Información general

- **Semestre**: 8vo Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas;
- **Prerrequisitos**:
  - CS292 Ingeniería de Software II (7mo Sem-Pág 425)

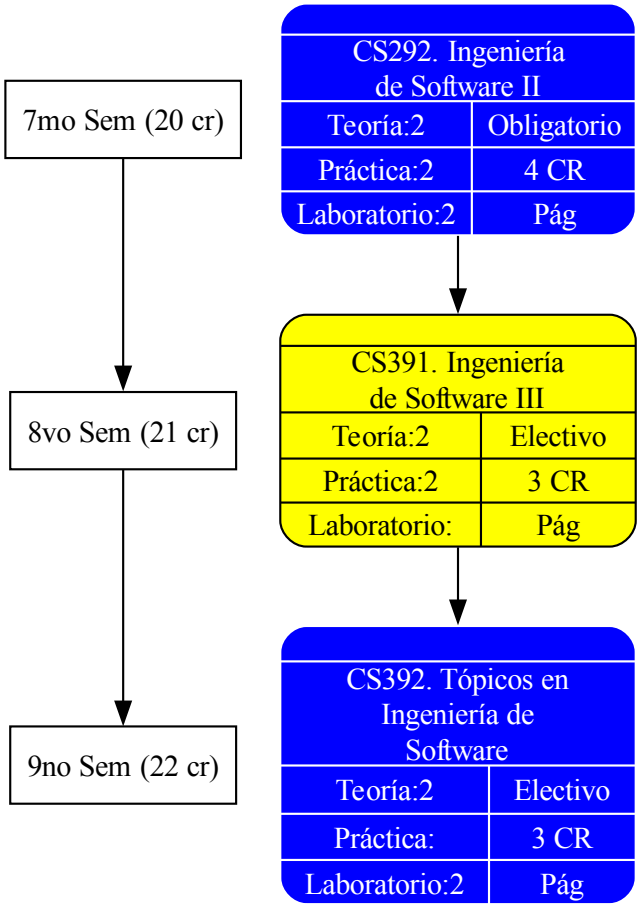

Figura 5.51: Mapa de Conexión. CS391 Ingeniería de Software III

2. Fundamentación

Ingeniería de Software III se enfoca en temas avanzados del ciclo de vida del software, haciendo énfasis en la calidad del software, seguridad por diseño y despliegue en arquitecturas modernas. Este curso prepara a los estudiantes para manejar las complejidades del desarrollo nativo de la nube y la integración de prácticas de seguridad dentro de la canalización DevOps (DevSecOps), asegurando que los productos de software no solo sean funcionales, sino también resilientes y mantenibles.

3. Objetivos del curso.

- Implementar métricas avanzadas de aseguramiento de calidad y confiabilidad.
- Integrar prácticas de seguridad a lo largo del ciclo de vida del desarrollo de software.
- Comprender y aplicar principios de DevOps y Despliegue Continuo.
- Diseñar y desplegar aplicaciones nativas de la nube usando microservicios.
- Analizar el impacto de la deuda técnica y las estrategias de rejuvenecimiento de software.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

5. Contenido

5.1 Conceptos de verificación y validación, 12 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Conceptos de verificación y validación *a*) Verificación: ¿Estamos construyendo la cosa correctamente? *b*) Validación: ¿Construimos la cosa correcta? **II:** Por qué importan las pruebas: ¿El componente sigue siendo funcional a medida que el código evoluciona? **III:** Objetivos de las pruebas *a*) Usabilidad *b*) Confiabilidad *c*) Conformidad con la especificación *d*) Rendimiento *e*) Seguridad

Objetivos de Aprendizaje I: Explicar por qué las pruebas son importantes [Explicar] **II:** Distinguir entre validación y verificación de programas [Distinguir] **III:** Describir diferentes objetivos de las pruebas [Describir]

Bibliografía: [Sommerville, 2015, Pressman and Maxim, 2019]

5.2 Herramientas y Análisis de Pruebas, 3 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Herramientas y automatización de verificación y validación *a*) Análisis estático *b*) Cobertura de código *c*) Fuzzing *d*) Análisis dinámico y contención de fallos (sanitizadores, etc.) *e*) Registro y seguimiento de fallos **II:** Planificación y generación de pruebas *a*) Estimación de fallos y terminación de pruebas, incluyendo siembra de defectos *b*) Uso de números aleatorios y pseudoaleatorios en las pruebas **III:** Pruebas de sistemas asíncronos, paralelos y concurrentes **IV:** Verificación y validación de artefactos que no son código (documentación, materiales de capacitación) **V:** Desarrollo guiado por comportamiento (BDD) *a*) Expresar criterios de aceptación como escenarios Gherkin Given/When/Then *b*) Ejecutar escenarios BDD como pruebas automatizadas de extremo a extremo **VI:** Pruebas de mutación y análisis estático de calidad de código *a*) Pruebas de mutación para evaluar la efectividad de una suite de pruebas *b*) Detección estática de *code smells* y complejidad ciclomática excesiva

Objetivos de Aprendizaje I: Describir y comparar diferentes herramientas para verificación y validación [Describir] **II:** Automatizar el uso de diferentes herramientas en un proyecto de software pequeño [Diseñar] **III:** Explicar cómo y cuándo se deben usar números aleatorios en las pruebas [Explicar] **IV:** Describir enfoques para la estimación de fallos [Describir] **V:** Estimar el número de fallos en una aplicación de software pequeña basándose en la densidad de fallos y la siembra de fallos [Estimar] **VI:** Describir técnicas y problemas con las pruebas de software asíncrono, concurrente y paralelo [Describir] **VII:** Crear un plan de pruebas para un segmento de código de tamaño mediano que contenga código asíncrono, concurrente y/o paralelo, incluyendo una

matriz de pruebas y generación de datos y entradas de prueba [Crear] **VIII:** Describir técnicas para la verificación y validación de artefactos que no son código [Describir] **IX:** Escribir escenarios Gherkin Given/When/Then para los criterios de aceptación de una funcionalidad y ejecutarlos como pruebas automatizadas de extremo a extremo [Escribir] **X:** Analizar el puntaje de mutación de una suite de pruebas y refactorizar las pruebas para eliminar los mutantes sobrevivientes [Analizar]

Bibliografía: [Pressman and Maxim, 2019]

5.3 Pruebas de Rendimiento y Evaluación Comparativa, 3 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Pruebas de rendimiento y evaluación comparativa *a*) Rendimiento (*throughput*) y latencia *b*) Degradación bajo carga (pruebas de estrés, manejo FIFO vs LIFO de solicitudes) *c*) Aceleración (*speedup*) y escalado (*scaling*) 1) Ley de Amdahl 2) Ley de Gustafson 3) Escalado suave (*soft*) y débil (*weak*) *d*) Identificar y medir figuras de mérito (*figures of merit*) *e*) Cuellos de botella de rendimiento comunes 1) Limitado por cómputo (*compute-bound*) 2) Limitado por ancho de banda de memoria 3) Limitado por latencia *f*) Métodos estadísticos y mejores prácticas para la evaluación comparativa (benchmarking) 1) Estimación de la incertidumbre 2) Intervalos de confianza *g*) Análisis y presentación (gráficos, etc.) *h*) Técnicas de medición de tiempo (*timing*)

Objetivos de Aprendizaje I: Describir el rendimiento (*throughput*) y la latencia y proporcionar ejemplos de cada uno [Describir] **II:** Explicar la aceleración (speedup) y las diferentes formas de escalado (scaling) y cómo se calculan [Explicar] **III:** Describir cuellos de botella de rendimiento comunes [Describir] **IV:** Describir métodos estadísticos y mejores prácticas para la evaluación comparativa (benchmarking) de software [Describir] **V:** Explicar técnicas y desafíos para medir el tiempo al construir una evaluación comparativa (benchmark) [Explicar] **VI:** Identificar las figuras de mérito, construir y ejecutar una evaluación comparativa (benchmark) y analizar estadísticamente y visualizar los resultados para un proyecto de software pequeño [Analizar]

Bibliografía: [Bondi, 2015, Gregg, 2020]

5.4 Compatibilidad y Versionado de API, 12 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Ley de Hyrum / Ley de las Interfaces Implícitas **II:** Compatibilidad con versiones anteriores *a*) La compatibilidad no es una propiedad de una sola entidad, es una propiedad de una relación. *b*) La compatibilidad con versiones anteriores debe evaluarse en términos de proveedor + consumidor(es) o con un modelo bien especificado de qué formas de compatibilidad aspira/promete un proveedor. **III:** Control de versiones (versioning) *a*) Versionado Semántico (SemVer) *b*) Desarrollo basado en tronco (*trunk-based development*)

Objetivos de Aprendizaje I: Identificar tanto el comportamiento explícito como el implícito de una interfaz e identificar riesgos potenciales de la Ley de Hyrum [Analizar] **II:** Identificar cambios que pueden considerarse ampliamente "compatibles con versiones anteriores", potencialmente con declaraciones explícitas sobre qué uso está o no soportado [Analizar] **III:** Evaluar si un cambio propuesto es lo suficientemente seguro dada la metodología de control de versiones en uso para un proyecto dado [Evaluar]

Bibliografía: [Sommerville, 2015]

5.5 Control de Versiones y CI/CD, 12 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Gestión de configuración de software y control de versiones: Prácticas de Desarrollo de Software *a*) Configuración en control de versiones, builds/configuración reproducibles. *b*) Estrategias de ramificación en control de versiones. Ramas de desarrollo vs ramas de lanzamiento. Desarrollo basado en tronco (*trunk-based development*). *c*) Estrategias de fusión/rebase, cuando sean relevantes. **II:** Gestión de lanzamientos (*release management*). **III:** Herramientas de prueba, incluyendo herramientas de análisis estático y dinámico. Prácticas de Desarrollo de Software, Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada **IV:** Automatización de procesos de software: *a*) Sistemas de construcción (*build systems*): el valor de builds rápidos, herméticos y reproducibles, comparar/contrastar enfoques para construir un proyecto. *b*) Integración Continua (CI): el uso de automatización y pruebas automatizadas para hacer una validación preliminar de que la revisión actual del tronco se construye y pasa las pruebas (básicas). *c*) Despliegue Continuo (CD): el uso de automatización para liberar de forma automática cada cambio que supera las pruebas hacia el entorno de producción, garantizando entregas frecuentes y confiables. *d*) Gestión de dependencias: actualización de dependencias externas/aguas arriba, gestión de paquetes, SemVer. **V:** Flujo de revisión colaborativa como puerta de calidad *a*) Revisión obligatoria de *pull requests* antes de fusionar *b*) Propiedad del código mediante reglas de tipo CODEOWNERS *c*) Convenciones de mensajes de *commit* (*conventional commits*) *d*) Reglas de protección de ramas que exigen verificaciones antes de fusionar **VI:** Escaneo de vulnerabilidades de dependencias y de la cadena de suministro integrado en los pipelines de CI

Objetivos de Aprendizaje **I:** Describir la diferencia entre la gestión de configuración de software centralizada y distribuida [Describir] **II:** Describir cómo el control de versiones puede usarse para ayudar en la gestión de lanzamientos de software [Describir] **III:** Identificar elementos de configuración y usar una herramienta de control de código fuente en un proyecto pequeño basado en equipo [Analizar] **IV:** Describir cómo las herramientas de prueba estáticas y dinámicas disponibles pueden integrarse en el entorno de desarrollo de software [Describir] **V:** Comprender el uso de sistemas de CI/CD como una fuente de verdad para el estado del código compartido del equipo (éxito en la construcción y pruebas) [Explicar] **VI:** Aplicar un flujo de revisión de *pull requests* con reglas de propiedad de código y protección de ramas para imponer puertas de calidad en un proyecto de equipo [Aplicar] **VII:** Usar un escáner de vulnerabilidades de dependencias/cadena de suministro como parte de un pipeline de CI para detectar dependencias vulnerables [Usar]

Bibliografía: [Kim et al., 2016]

5.6 Principios y Arquitectura de Software, 12 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Principios de diseño de sistemas. Confiabilidad del Sistema *a*) Niveles de abstracción (por ejemplo, diseño arquitectónico y diseño detallado) *b*) Separación de preocupaciones (separation of concerns) *c*) Ocultación de información (information hiding) *d*) Acoplamiento y cohesión (coupling and cohesion) **II:** Arquitectura de software. Confiabilidad del Sistema *a*) Paradigmas de diseño 1) Descomposición funcional de arriba hacia abajo/diseño por capas 2) Arquitectura orientada a datos 3) Análisis y diseño orientado a objetos 4) Diseño dirigido por eventos *b*) Arquitecturas estándar (por ejemplo, cliente-servidor y arquitecturas de microservicios incluyendo discusiones sobre REST, n-capas, tuberías y filtros, Modelo Vista Controlador) *c*) Identificar límites y dependencias de componentes **III:** Programación a gran escala vs programación a pequeña escala (programming in the large vs programming in the small). Confiabilidad del Sistema **IV:** Malos olores del código (code smells) y otras indicaciones de la calidad del código, distintas de la corrección. Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad

Objetivos de Aprendizaje **I:** Identificar la arquitectura de software estándar de un diseño de alto nivel dado [Analizar] **II:** Seleccionar y usar un paradigma de diseño apropiado para diseñar un sistema de software simple y explicar cómo se han aplicado los principios de diseño de sistemas en este diseño [Evaluar] **III:** Adaptar un diseño de sistema defectuoso para que siga mejor principios

como la separación de preocupaciones o la ocultación de información [Crear] **IV:** Identificar las dependencias entre un conjunto de componentes de software en un diseño arquitectónico [Analizar]

Bibliografía: [Fowler, 2017]

6. Bibliografía

[Bondi, 2015] Bondi, A. B. (2015). *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice*. Addison-Wesley, Upper Saddle River, NJ.

[Fowler, 2017] Fowler, M. (2017). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 2nd edition.

[Gregg, 2020] Gregg, B. (2020). *Systems Performance: Enterprise and the Cloud*. Addison-Wesley Professional, Boston, MA, 2nd edition.

[Kim et al., 2016] Kim, G., Humble, J., Debois, P., and Willis, J. (2016). *The DevOps Handbook*. IT Revolution Press.

[Pressman and Maxim, 2019] Pressman, R. S. and Maxim, B. (2019). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 9th edition.

[Sommerville, 2015] Sommerville, I. (2015). *Software Engineering*. Pearson, 10th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS393 Sistemas de Información (Electivo)

1. Información general

- **Semestre**: 8vo Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS291 Ingeniería de Software I (5to Sem-Pág 382)

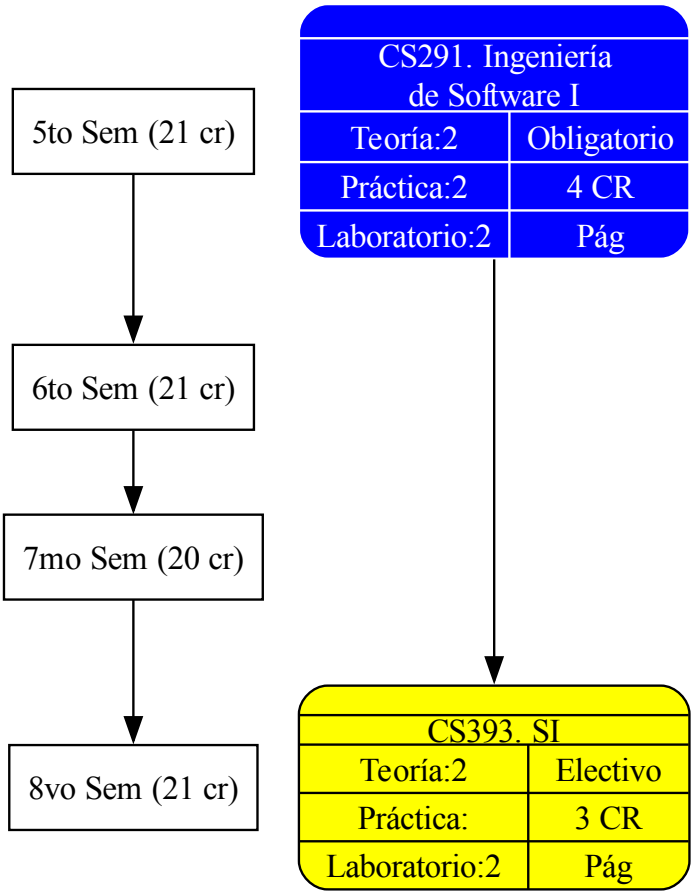

Figura 5.52: Mapa de Conexión. CS393 Sistemas de Información

2. Fundamentación

Este curso proporciona una visión general completa de cómo los sistemas de información respaldan los procesos empresariales y la toma de decisiones dentro de las organizaciones. Cubre el rol estratégico de la tecnología de la información, el ciclo de vida de los sistemas de información, estrategias de gestión de datos y la integración de aplicaciones empresariales para lograr ventajas competitivas en una economía digital globalizada.

3. Objetivos del curso.

- Comprender los componentes fundamentales y los tipos de Sistemas de Información.
- Analizar el papel de los SI en la estrategia organizacional y la toma de decisiones.
- Evaluar tecnologías emergentes y su impacto en los modelos de negocio.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

5. Contenido

5.1 Foundations of Information Systems, 12 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Componentes de los Sistemas de Información: Hardware, Software, Datos, Redes. **II:** Tipos de SI: TPS, MIS, DSS y ESS. **III:** Ventaja competitiva a través de la Tecnología de la Información. **IV:** Procesos empresariales e integración de Sistemas de Información.

Objetivos de Aprendizaje I: Identificar los componentes centrales de un sistema de información [Familiarizarse]. **II:** Explicar cómo los SI respaldan los diversos niveles de la jerarquía organizacional [Usar]. **III:** Analizar un caso de negocio para identificar usos estratégicos de TI [Evaluar].

Bibliografía: [Laudon and Laudon, 2021, Stair and Reynolds, 2017]

5.2 Data Management and Business Intelligence, 12 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Sistemas de gestión de bases de datos y almacenes de datos. **II:** Análisis de Big Data en un contexto empresarial. **III:** Inteligencia Empresarial (BI) y herramientas de Visualización de Datos. **IV:** Sistemas de Gestión del Conocimiento.

Objetivos de Aprendizaje I: Diseñar una arquitectura de datos básica para un requisito empresarial [Usar]. **II:** Evaluar la importancia de la calidad de datos y la gobernanza [Evaluar].

Bibliografía: [Laudon and Laudon, 2021, Valacich and Schneider, 2021]

5.3 Enterprise Systems and E-Commerce, 12 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Sistemas de Planificación de Recursos Empresariales (ERP). **II:** Gestión de la Cadena de Suministro (SCM) y Gestión de Relaciones con el Cliente (CRM). **III:** Modelos de comercio electrónico: B2B, B2C y C2C. **IV:** Comercio móvil y sistemas de pago digital.

Objetivos de Aprendizaje I: Describir las áreas funcionales de un sistema ERP [Familiarizarse]. **II:** Evaluar el impacto del comercio electrónico en los modelos de negocio tradicionales [Evaluar].

Bibliografía: [Stair and Reynolds, 2017, Valacich and Schneider, 2021]

5.4 IS Development and Security, 12 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Ciclo de Vida de Desarrollo de Sistemas (SDLC) y metodologías Ágiles. **II:** Externalización y servicios de SI basados en la nube. **III:** Amenazas a la seguridad de la información y gestión de riesgos. **IV:** Problemas éticos y sociales en Sistemas de Información.

Objetivos de Aprendizaje I: Comparar diferentes metodologías de desarrollo de sistemas [Usar]. **II:** Formular pautas éticas para la privacidad y seguridad de los datos [Evaluar].

Bibliografía: [Laudon and Laudon, 2021, Stair and Reynolds, 2017]

6. Bibliografía

[Laudon and Laudon, 2021] Laudon, K. C. and Laudon, J. P. (2021). *Management Information Systems: Managing the Digital Firm*. Pearson, 17th edition.

[Stair and Reynolds, 2017] Stair, R. and Reynolds, G. (2017). *Principles of Information Systems*. Cengage Learning, 13th edition.

[Valacich and Schneider, 2021] Valacich, J. and Schneider, C. (2021). *Information Systems Today: Managing in the Digital World*. Pearson, 9th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS394 Arquitectura de Software y DevOps para Software en la Nube (Electivo)

1. Información general

- **Semestre**: 8vo Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas;
- **Prerrequisitos**:
  - CS292 Ingeniería de Software II (7mo Sem-Pág 425)

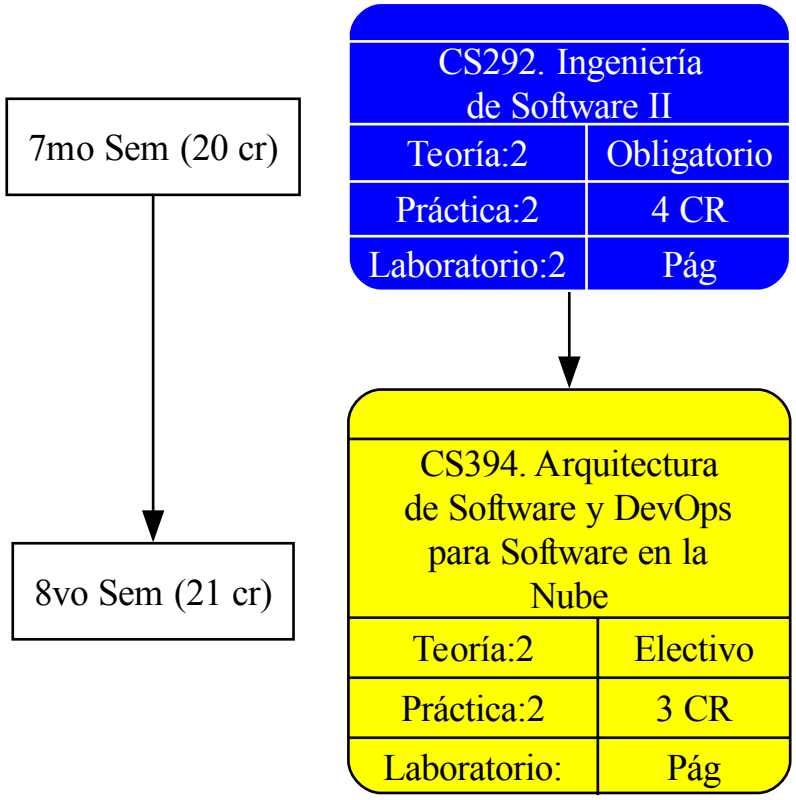

Figura 5.53: Mapa de Conexión. CS394 Arquitectura de Software y DevOps para Software en la Nube

2. Fundamentación

El diseño de software moderno exige combinar principios sólidos de arquitectura con prácticas ágiles de entrega continua y operación en la nube. Este curso integra tres áreas que en la industria actual son inseparables: arquitectura de software, DevOps y computación en la nube.

El estudiante aprenderá a diseñar sistemas a escala usando patrones arquitectónicos reconocidos, a automatizar los ciclos de construcción, prueba y despliegue mediante pipelines de CI/CD, y a provisionar y operar infraestructura en plataformas de nube pública mediante contenedores, orquestación y herramientas de infraestructura como código.

El curso pone especial énfasis en la confiabilidad y observabilidad de sistemas distribuidos, cubriendo métricas, trazas distribuidas, presupuestos de error y prácticas de ingeniería de confiabilidad del sitio (SRE), tal como se aplican en organizaciones de tecnología de alto rendimiento.

3. Objetivos del curso.

- Diseñar sistemas de software escalables y mantenibles aplicando patrones arquitectónicos modernos.
- Implementar pipelines de integración y entrega continua (CI/CD) para acelerar y asegurar el ciclo de vida del software.
- Desplegar y gestionar aplicaciones en contenedores sobre plataformas de nube pública usando herramientas estándar de la industria.
- Provisionar infraestructura de nube de forma reproducible y versionada mediante infraestructura como código (IaC).
- Aplicar prácticas de observabilidad y SRE para garantizar la confiabilidad de sistemas en producción.

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Evaluar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Evaluar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Principios y Arquitectura de Software, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Principios de diseño de sistemas. Confiabilidad del Sistema *a*) Niveles de abstracción (por ejemplo, diseño arquitectónico y diseño detallado) *b*) Separación de preocupaciones (separation of concerns) *c*) Ocultación de información (information hiding) *d*) Acoplamiento y cohesión (coupling and cohesion) **II:** Arquitectura de software. Confiabilidad del Sistema *a*) Paradigmas de diseño 1) Descomposición funcional de arriba hacia abajo/diseño por capas 2) Arquitectura orientada a datos 3) Análisis y diseño orientado a objetos 4) Diseño dirigido por eventos *b*) Arquitecturas estándar (por ejemplo, cliente-servidor y arquitecturas de microservicios incluyendo discusiones sobre REST, n-capas, tuberías y filtros, Modelo Vista Controlador) *c*) Identificar límites y dependencias de componentes **III:** Programación a gran escala vs programación a pequeña escala (programming in the large vs programming in the small). Confiabilidad del Sistema **IV:** Modelos estructurales y de comportamiento de diseños de software **V:** Modelado de diseño, por ejemplo con diagramas de clases, diagramas de entidad-relación o diagramas de secuencia **VI:** Principios de diseño de API *a*) Consistencia 1) Las API consistentes son más fáciles de aprender y menos propensas a errores 2) La consistencia es tanto interna (entre diferentes partes de la API) como externa (siguiendo patrones comunes de API) *b*) Capacidad de composición (*composability*) *c*) Documentación de contratos 1) Las operaciones de la API deben describir su efecto en el sistema, pero generalmente no su implementación 2) Precondiciones, postcondiciones e invariantes *d*) Capacidad de expansión (*expandability*) *e*) Reporte de errores 1) Los errores deben ser claros, predecibles y accionables 2) La entrada que no coincida con el contrato debe producir un error 3) Los errores que pueden gestionarse de manera confiable sin reportarse deben gestionarse

Objetivos de Aprendizaje **I:** Identificar la arquitectura de software estándar de un diseño de alto nivel dado [Analizar] **II:** Identificar las dependencias entre un conjunto de componentes de software en un diseño arquitectónico [Analizar] **III:** Seleccionar y usar un paradigma de diseño apropiado

para diseñar un sistema de software simple y explicar cómo se han aplicado los principios de diseño de sistemas en este diseño [Evaluar] **IV:** Traducir un diseño de software en lenguaje natural a diagramas de clases [Traducir]

Bibliografía: [Bass et al., 2021, Richards and Ford, 2020]

5.2 Patrones y Estilos Arquitectónicos, 8 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: I: Evaluar compensaciones de diseño (tradeoffs) (por ejemplo, eficiencia vs confiabilidad, seguridad vs usabilidad) **II:** Estilos arquitectónicos: en capas, orientado a eventos, microkernel, orientado a servicios, pipeline. **III:** Atributos de calidad: disponibilidad, escalabilidad, rendimiento, seguridad, mantenibilidad. **IV:** Trazabilidad de requisitos *a*) Comprender qué requisitos son satisfechos por un diseño **V:** Medición y análisis de la calidad del diseño

Objetivos de Aprendizaje I: Contrastar dos diseños de software a través de diferentes cualidades, como eficiencia o usabilidad [Contrastar] **II:** Identificar qué requisitos son satisfechos por un diseño de software proporcionado [Analizar] **III:** Adaptar un diseño de sistema defectuoso para que siga mejor principios como la separación de preocupaciones o la ocultación de información [Crear] **IV:** Seleccionar el estilo arquitectónico más adecuado dados los atributos de calidad requeridos por un sistema. [Evaluar]

Bibliografía: [Richards and Ford, 2020]

5.3 Arquitectura de Microservicios, 6 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Principios de microservicios: cohesión, acoplamiento débil y fronteras de servicio. **II:** Descomposición de monolitos: diseño orientado al dominio (DDD). **III:** Principios de diseño de API *a*) Consistencia 1) Las API consistentes son más fáciles de aprender y menos propensas a errores 2) La consistencia es tanto interna (entre diferentes partes de la API) como externa (siguiendo patrones comunes de API) *b*) Capacidad de composición (*composability*) *c*) Documentación de contratos 1) Las operaciones de la API deben describir su efecto en el sistema, pero generalmente no su implementación 2) Precondiciones, postcondiciones e invariantes *d*) Capacidad de expansión (*expandability*) *e*) Reporte de errores 1) Los errores deben ser claros, predecibles y accionables 2) La entrada que no coincida con el contrato debe producir un error 3) Los errores que pueden gestionarse de manera confiable sin reportarse deben gestionarse **IV:** API Gateway y malla de servicios (*service mesh*). **V:** Patrones de resiliencia: Circuit Breaker, Retry, Bulkhead, Timeout. **VI:** Almacenes de Datos *a*) Estructuras de datos mantenidas cooperativamente que implementan mapas y TADs relacionados. *b*) Variedades: Propiedad exclusiva, compartida, fragmentada, replicada, inmutable, versionada.

Objetivos de Aprendizaje I: Diseñar una API para un solo componente de un sistema de software grande, incluyendo la identificación y documentación de las invariantes, el contrato y las condiciones de error de cada operación [Diseñar] **II:** Evaluar una descripción de API en términos de consistencia, capacidad de composición y capacidad de expansión [Evaluar] **III:** Diseñar un sistema basado en microservicios descomponiendo una aplicación monolítica en servicios independientes. [Usar] **IV:** Aplicar patrones de resiliencia para tolerar fallos en sistemas distribuidos. [Usar]

Bibliografía: [Newman, 2021]

5.4 Contenedores y Docker, 6 hr(s)

Competencias: AG-C11,AG-C12

Tópicos: **I:** Contenedores y su comparación con máquinas virtuales. **II:** Tipos de virtualización (incluyendo hardware/software, SO, servidor, servicio, red). Ver también: Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento **III:** Virtualización portátil; emulación vs aislamiento. Ver también: Latencia, Caché y Jerarquía de Memoria, Virtualización y Aislamiento **IV:** Máquinas virtuales y escapes de contenedores, peligros desde una perspectiva de seguridad. Ver también: Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad **V:** Dockerfile: buenas prácticas para construir imágenes eficientes y seguras. **VI:** Composición de aplicaciones multi-contenedor con docker-compose.

Objetivos de Aprendizaje **I:** Explicar la diferencia entre emulación y aislamiento [Explicar] **II:** Evaluar las compensaciones de la virtualización [Evaluar] **III:** Construir y desplegar aplicaciones en contenedores usando Docker. [Usar] **IV:** Aplicar buenas prácticas de seguridad en la construcción de imágenes Docker. [Usar]

Bibliografía: [Burns et al., 2022]

5.5 Orquestación de Contenedores con Kubernetes, 6 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Arquitectura de Kubernetes: plano de control y nodos trabajadores. **II:** Objetos fundamentales: Pods, Deployments, Services, ConfigMaps, Secrets. **III:** Escalado horizontal automático (*HPA*) y autocuración. **IV:** Dependencias *a*) La iniciación o progreso de una actividad puede depender de otras actividades, para evitar condiciones de carrera, garantizar la terminación o cumplir otros requisitos. *b*) Garantizar el progreso evitando ciclos de dependencia, usando condiciones monótonas, eliminando dependencias no esenciales. **V:** Gestión de paquetes con Helm.

Objetivos de Aprendizaje **I:** Desplegar y gestionar aplicaciones en contenedores sobre un clúster de Kubernetes. [Usar] **II:** Configurar el escalado automático y la autocuración para cargas de trabajo nativas de la nube. [Usar] **III:** Modificar un sistema concurrente para usar un almacén de datos más escalable, confiable o disponible [Crear]

Bibliografía: [Burns et al., 2022]

5.6 Control de Versiones y CI/CD, 6 hr(s)

Competencias: AG-C11,AG-C12

Tópicos: **I:** Gestión de configuración de software y control de versiones: Prácticas de Desarrollo de Software *a*) Configuración en control de versiones, builds/configuración reproducibles. *b*) Estrategias de ramificación en control de versiones. Ramas de desarrollo vs ramas de lanzamiento. Desarrollo basado en tronco (*trunk-based development*). *c*) Estrategias de fusión/rebase, cuando sean relevantes. **II:** Gestión de lanzamientos (*release management*). **III:** Automatización de procesos de software: *a*) Sistemas de construcción (*build systems*): el valor de builds rápidos, herméticos y reproducibles, comparar/contrastar enfoques para construir un proyecto. *b*) Integración Continua (CI): el uso de automatización y pruebas automatizadas para hacer una validación preliminar de que la revisión actual del tronco se construye y pasa las pruebas (básicas). *c*) Despliegue Continuo (CD): el uso de automatización para liberar de forma automática cada cambio que supera las pruebas hacia el entorno de producción, garantizando entregas frecuentes y confiables. *d*) Gestión de dependencias: actualización de dependencias externas/aguas arriba, gestión de paquetes, SemVer. **IV:** Herramientas de prueba, incluyendo herramientas de análisis estático y dinámico. Prácticas de Desarrollo de Software, Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada **V:** Conceptos y mecanismos de integración de herramientas. Prácticas de Desarrollo de Software **VI:** Cultura DevOps: flujo, retroalimentación y aprendizaje continuo.

Objetivos de Aprendizaje I: Describir la diferencia entre la gestión de configuración de software centralizada y distribuida [Describir] **II:** Describir cómo el control de versiones puede usarse para ayudar en la gestión de lanzamientos de software [Describir] **III:** Comprender el uso de sistemas de CI/CD como una fuente de verdad para el estado del código compartido del equipo (éxito en la construcción y pruebas) [Explicar] **IV:** Demostrar la capacidad de usar herramientas de software para apoyar el desarrollo de un producto de software de tamaño mediano [Demostrar]

Bibliografía: [Kim et al., 2021]

5.7 Infraestructura como Código (IaC), 6 hr(s)

Competencias: AG-C11,AG-C12

Tópicos: I: Principios de IaC: reproducibilidad, idempotencia y versionamiento de infraestructura. **II:** Enfoques declarativo vs. imperativo. **III:** Terraform: proveedores, recursos, módulos y gestión de estado. **IV:** Herramientas nativas de nube: AWS CloudFormation, Azure ARM Templates, GCP Deployment Manager. **V:** Gestión de configuración de software y control de versiones: Prácticas de Desarrollo de Software *a*) Configuración en control de versiones, builds/configuración reproducibles. *b*) Estrategias de ramificación en control de versiones. Ramas de desarrollo vs ramas de lanzamiento. Desarrollo basado en tronco (*trunk-based development*). *c*) Estrategias de fusión/rebase, cuando sean relevantes.

Objetivos de Aprendizaje I: Escribir y aplicar configuraciones de Terraform para aprovisionar infraestructura en la nube. [Usar] **II:** Identificar elementos de configuración y usar una herramienta de control de código fuente en un proyecto pequeño basado en equipo [Analizar] **III:** Comparar herramientas de IaC según sus capacidades de gestión de estado y modularidad. [Evaluar]

Bibliografía: [Morris, 2021, Kim et al., 2021]

5.8 Plataformas de Nube y Servicios Nativos, 6 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Modelos de servicio en la nube: IaaS, PaaS, SaaS, FaaS. **II:** Paradigmas de aplicaciones distribuidas (por ejemplo, cliente/servidor, peer-to-peer, nube, edge y fog). Ver también: Comunicación, Coordinación **III:** Principales proveedores: AWS, Google Cloud, Azure — servicios clave. **IV:** Servicios de cómputo: máquinas virtuales, funciones serverless, contenedores administrados. **V:** Servicios de almacenamiento: objetos, bloque y bases de datos administradas. **VI:** Redes en la nube: VPCs, balanceadores de carga, CDN.

Objetivos de Aprendizaje I: Analizar las necesidades de demandas específicas de aplicaciones en red [Analizar] **II:** Comparar los modelos IaaS, PaaS, SaaS y FaaS para seleccionar el más adecuado a una arquitectura dada. [Evaluar] **III:** Diseñar una aplicación nativa de la nube usando servicios administrados de un proveedor principal. [Usar]

Bibliografía: [Kim et al., 2021, Burns et al., 2022]

5.9 Observabilidad: Métricas, Logs y Trazas, 6 hr(s)

Competencias: AG-C11,AG-C12

Tópicos: I: Análisis basado en mediciones de la confiabilidad del software - telemetría, monitorización y alertas, paneles de control, métricas de calificación de lanzamientos, etc. **II:** Los tres pilares de la observabilidad: métricas, logs y trazas distribuidas. **III:** Trazas distribuidas e identificadores de correlación. **IV:** Prácticas de ingeniería de confiabilidad de software - incluyendo revisiones, pruebas, verificación de modelos práctica (practical model checking) **V:** Estrategias de alertas y prácticas de guardia (*on-call*).

Objetivos de Aprendizaje I: Describir cómo determinar el nivel de confiabilidad requerido por un sistema de software [Describir] **II:** Identificar métodos que conducirán a la realización de una arquitectura de software que logre un nivel especificado de confiabilidad [Analizar] **III:** Implementar observabilidad en una aplicación de nube usando métricas, logs y trazas distribuidas. [Usar]

Bibliografía: [Kim et al., 2021]

5.10 Ingeniería de Confiabilidad del Sitio (SRE), 6 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: I: Concepto de confiabilidad como probabilidad de fallo o tiempo medio entre fallos, y fallas (faults) como causa de fallos (failures) **II:** Comparar el enfoque de "presupuesto de error" (error budget) para la confiabilidad con el enfoque "libre de errores" (error-free) e identificar dominios donde cada uno es relevante. **III:** Técnicas y modelos de tolerancia a fallos de software *a*) Diferencias contextuales en la tolerancia a fallos (por ejemplo, hacer fallar un sistema crítico para la aviación se evita firmemente, hacer fallar un sistema de procesamiento de datos antes de que se escriban datos corruptos en el almacenamiento es muy valioso) **IV:** Confiabilidad mediante redundancia: *a*) verificación y reintento (*check and retry*) Soporte de Fiabilidad, Tolerancia a fallos *b*) codificación redundante (códigos de corrección de errores, CRC, FEC, RAID) Jerarquía de Memoria, Bases de Datos Distribuidas/Computación en la Nube, Soporte de Fiabilidad *c*) duplicación/reflejo (*mirroring*)/réplicas Bases de Datos Distribuidas/Computación en la Nube **V:** Confiabilidad del software, confiabilidad del sistema y comportamiento de fallos **VI:** Indicadores de nivel de servicio (SLI), objetivos (SLO) y acuerdos (SLA).

Objetivos de Aprendizaje I: Identificar formas de aplicar redundancia para lograr tolerancia a fallos [Analizar] **II:** Identificar dependencias de punto único de fallo (single-point-of-failure, SPF) en un diseño de sistema [Analizar] **III:** Explicar los problemas que existen para lograr niveles muy altos de confiabilidad [Explicar] **IV:** Definir SLIs, SLOs y presupuestos de error para un servicio en producción. [Usar]

Bibliografía: [Kim et al., 2021]

6. Bibliografía

[Bass et al., 2021] Bass, L., Clements, P., and Kazman, R. (2021). *Software Architecture in Practice*. Addison-Wesley Professional, 4th edition.

[Burns et al., 2022] Burns, B., Beda, J., Hightower, K., and Evenson, L. (2022). *Kubernetes: Up and Running*. O'Reilly Media, 3rd edition.

[Kim et al., 2021] Kim, G., Humble, J., Debois, P., Willis, J., and Forsgren, N. (2021). *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, 2nd edition.

[Morris, 2021] Morris, K. (2021). *Infrastructure as Code: Dynamic Systems for the Cloud Age*. O'Reilly Media, 2nd edition.

[Newman, 2021] Newman, S. (2021). *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, 2nd edition.

[Richards and Ford, 2020] Richards, M. and Ford, N. (2020). *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media, 1st edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS395 Calidad de Software (Electivo)

1. Información general

- **Semestre**: 8vo Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas;
- **Prerrequisitos**:
  - CS292 Ingeniería de Software II (7mo Sem-Pág 425)

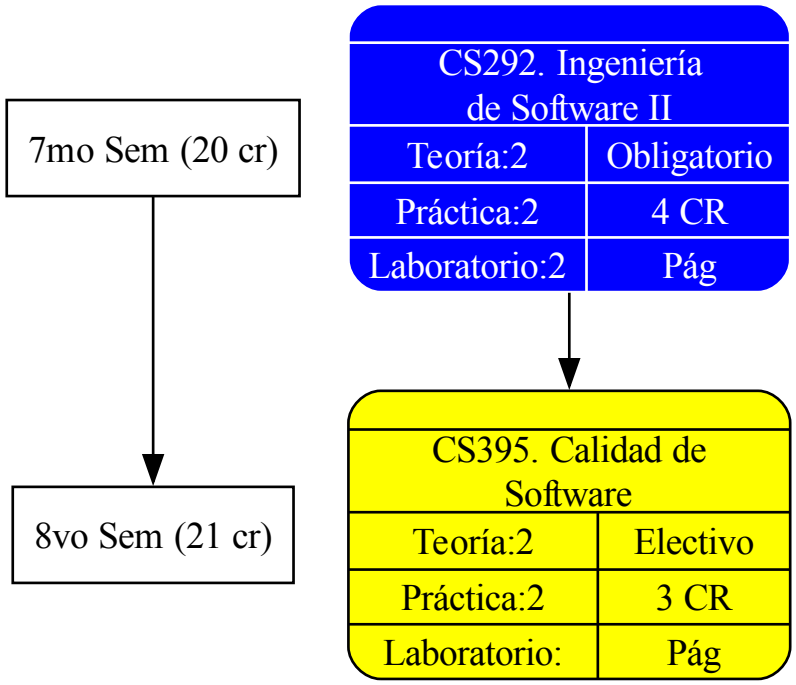

Figura 5.54: Mapa de Conexión. CS395 Calidad de Software

2. Fundamentación

Entregar software meramente ”funcionalza no es suficiente para la industria: las organizaciones modernas exigen sistemas confiables, seguros, mantenibles y construidos mediante procesos de ingeniería disciplinados y auditables. Este curso enseña la calidad del software como una disciplina de ingeniería práctica y de primer nivel, ejercida mediante las mismas herramientas y flujos de trabajo usados por equipos profesionales en GitHub: protección de ramas, revisión de código obligatoria, pruebas automatizadas en todos los niveles, integración continua/entrega continua (CI/CD), análisis estático y observabilidad en producción.

Los estudiantes avanzan desde modelos fundamentales de calidad hacia flujos de trabajo colaborativos avanzados, estrategias de pruebas automatizadas (unitarias, de integración, de extremo a extremo y guiadas por comportamiento), pipelines de CI/CD, análisis estático y refactorización, y aspectos de calidad en producción como seguridad de la cadena de suministro, rendimiento e ingeniería de confiabilidad. El curso cierra con énfasis en la cultura DevOps: las prácticas y valores compartidos del

equipo (Definition of Done, postmortems sin culpa, aprendizaje continuo) que sostienen la calidad a escala.

3. Objetivos del curso.

- Evaluar la calidad del software usando un modelo de características de calidad reconocido internacionalmente y razonar sobre el costo de la mala calidad y la deuda técnica.
- Aplicar un flujo de trabajo profesional de Git/GitHub basado en revisión, incluyendo protección de ramas, CODEOWNERS y conventional commits.
- Diseñar e implementar suites de pruebas unitarias, de integración, de extremo a extremo y guiadas por comportamiento, incluyendo desarrollo guiado por pruebas.
- Construir y operar pipelines de CI/CD que impongan puertas de calidad automatizadas, escaneo de dependencias y versionado semántico.
- Aplicar análisis estático, pruebas de mutación y técnicas de refactorización para identificar y eliminar code smells y deuda técnica.
- Analizar aspectos de calidad en producción, incluyendo pruebas de rendimiento y prácticas de confiabilidad/observabilidad.
- Adoptar prácticas de cultura DevOps – Definition of Done, postmortems sin culpa y aprendizaje continuo del equipo – para sostener la calidad.

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Evaluar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Evaluar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Calidad y Evaluación del Diseño, 4 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: **I:** Diseño de datos Modelado de Datos *a*) Estructuras de datos *b*) Sistemas de almacenamiento **II:** Trazabilidad de requisitos *a*) Comprender qué requisitos son satisfechos por un diseño **III:** Modelado de diseño, por ejemplo con diagramas de clases, diagramas de entidad-relación o diagramas de secuencia **IV:** Medición y análisis de la calidad del diseño **V:** Principios de diseño y codificación seguros Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad *a*) Principio de privilegio mínimo *b*) Principio de valores predeterminados seguros ante fallos (fail-safe defaults) *c*) Principio de aceptabilidad psicológica **VI:** Evaluar compensaciones de diseño (tradeoffs) (por ejemplo, eficiencia vs confiabilidad, seguridad vs usabilidad) **VII:** Modelo de características de calidad ISO/IEC 25010 y costo de la mala calidad *a*) Características de calidad: adecuación funcional, eficiencia de desempeño, compatibilidad, usabilidad, confiabilidad, seguridad, mantenibilidad, portabilidad *b*) Costo de la mala calidad y la deuda técnica como preocupación de negocio/económica, no solo técnica

Objetivos de Aprendizaje **I:** Diseñar un conjunto de estructuras de datos para implementar una superficie de API proporcionada [Diseñar] **II:** Identificar qué requisitos son satisfechos por un diseño de software proporcionado [Analizar] **III:** Traducir un diseño de software en lenguaje natural a diagramas de clases [Traducir] **IV:** Adaptar un diseño de sistema defectuoso para que siga mejor los principios de privilegio mínimo y valores predeterminados seguros ante fallos [Crear] **V:** Contrastar dos diseños de software a través de diferentes cualidades, como eficiencia o usabilidad [Contrastar] **VI:** Evaluar un sistema de software frente a las características de calidad ISO/IEC 25010 y estimar el costo de negocio de la mala calidad/deuda técnica [Evaluar]

Bibliografía: [International Organization for Standardization, 2011, Sommerville, 2015]

5.2 Control de Versiones y CI/CD, 12 hr(s)

Competencias: AG-C11,AG-C12

Tópicos: **I:** Gestión de configuración de software y control de versiones: Prácticas de Desarrollo de Software *a*) Configuración en control de versiones, builds/configuración reproducibles. *b*) Estrategias de ramificación en control de versiones. Ramas de desarrollo vs ramas de lanzamiento. Desarrollo basado en tronco (*trunk-based development*). *c*) Estrategias de fusión/rebase, cuando sean relevantes. **II:** Gestión de lanzamientos (*release management*). **III:** Herramientas de prueba, incluyendo herramientas de análisis estático y dinámico. Prácticas de Desarrollo de Software, Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada **IV:** Automatización de procesos de software: *a*) Sistemas de construcción (*build systems*): el valor de builds rápidos, herméticos y reproducibles, comparar/contrastar enfoques para construir un proyecto. *b*) Integración Continua (CI): el uso de automatización y pruebas automatizadas para hacer una validación preliminar de que la revisión actual del tronco se construye y pasa las pruebas (básicas). *c*) Despliegue Continuo (CD): el uso de automatización para liberar de forma automática cada cambio que supera las pruebas hacia el entorno de producción, garantizando entregas frecuentes y confiables. *d*) Gestión de dependencias: actualización de dependencias externas/aguas arriba, gestión de paquetes, SemVer. **V:** Flujo de revisión colaborativa como puerta de calidad *a*) Revisión obligatoria de *pull requests* antes de fusionar *b*) Propiedad del código mediante reglas de tipo CODEOWNERS *c*) Convenciones de mensajes de *commit* (*conventional commits*) *d*) Reglas de protección de ramas que exigen verificaciones antes de fusionar **VI:** Escaneo de vulnerabilidades de dependencias y de la cadena de suministro integrado en los pipelines de CI

Objetivos de Aprendizaje **I:** Describir la diferencia entre la gestión de configuración de software centralizada y distribuida [Describir] **II:** Describir cómo el control de versiones puede usarse para ayudar en la gestión de lanzamientos de software [Describir] **III:** Identificar elementos de configuración y usar una herramienta de control de código fuente en un proyecto pequeño basado en equipo [Analizar] **IV:** Describir cómo las herramientas de prueba estáticas y dinámicas disponibles pueden integrarse en el entorno de desarrollo de software [Describir] **V:** Comprender el uso de sistemas de CI/CD como una fuente de verdad para el estado del código compartido del equipo (éxito en la construcción y pruebas) [Explicar] **VI:** Aplicar un flujo de revisión de *pull requests* con reglas de propiedad de código y protección de ramas para imponer puertas de calidad en un proyecto de equipo [Aplicar] **VII:** Usar un escáner de vulnerabilidades de dependencias/cadena de suministro como parte de un pipeline de CI para detectar dependencias vulnerables [Usar]

Bibliografía: [Kim et al., 2021]

5.3 Planificación y Tipos de Pruebas, 8 hr(s)

Competencias: AG-C09

Tópicos: **I:** Tipos de pruebas *a*) Unitarias *b*) Integración *c*) Validación *d*) Sistema **II:** Diferencias estilísticas entre el código de prueba y el código de producción: DAMP vs DRY - se justifica más duplicación en el código de prueba. **III:** Planificación y generación de pruebas *a*) Generación de casos de prueba, a partir de modelos formales, especificaciones, etc. *b*) Cobertura de pruebas 1) Matrices de prueba 2) Cobertura de código - ¿cuánto del código está probado? 3) Cobertura de entorno - ¿cuántas arquitecturas de hardware, sistemas operativos, navegadores, etc. están probados? *c*) Datos y entradas de prueba

Objetivos de Aprendizaje **I:** Comparar y contrastar los diferentes tipos y niveles de pruebas (regresión, unitarias, integración, sistemas y aceptación) [Comparar] **II:** Describir técnicas para crear un plan de pruebas y generar casos de prueba [Describir] **III:** Crear un plan de pruebas para un segmento de código de tamaño mediano que incluya una matriz de pruebas y generación de datos

y entradas de prueba [Crear] **IV:** Implementar un plan de pruebas para un segmento de código de tamaño mediano [Implementar]

Bibliografía: [Sommerville, 2015]

5.4 Prácticas Avanzadas de Pruebas, 4 hr(s)

Competencias: AG-C11

Tópicos: **I:** Desarrollo de pruebas *a*) Desarrollo guiado por pruebas *b*) Pruebas orientadas a objetos, uso de mocks e inyección de dependencias *c*) Técnicas de prueba de caja opaca (anteriormente, caja negra) y caja transparente (anteriormente, caja blanca) *d*) Herramientas de prueba, incluyendo cobertura de código, análisis estático y fuzzing **II:** Verificación y validación en el ciclo de desarrollo *a*) Revisiones de código *b*) Automatización de pruebas, incluyendo automatización de herramientas *c*) Pruebas pre-commit y post-commit *d*) Compensaciones entre cobertura de pruebas y rendimiento/latencia de las pruebas *e*) Seguimiento y priorización de defectos: reproducibilidad de los defectos reportados **III:** Desafíos de verificación y validación específicos del dominio *a*) Pruebas de rendimiento y evaluación comparativa (benchmarking) *b*) Asincronía, paralelismo y concurrencia *c*) Crítico para la seguridad (*safety-critical*) *d*) Numérico

Objetivos de Aprendizaje **I:** Identificar los principios fundamentales de los métodos de desarrollo guiado por pruebas y explicar el papel de las pruebas automatizadas en estos métodos [Analizar] **II:** Discutir problemas relacionados con las pruebas de software orientado a objetos [Debatir] **III:** Describir el uso de mocks y la inyección de dependencias y su aplicación [Describir] **IV:** Realizar, como parte de una actividad en equipo, una revisión de código de un segmento de código de tamaño mediano [Evaluar] **V:** Describir el papel que las herramientas pueden desempeñar en la validación del software [Describir] **VI:** Automatizar las pruebas en un proyecto de software pequeño [Diseñar] **VII:** Explicar los roles, pros y contras de las pruebas pre-commit y post-commit [Explicar] **VIII:** Discutir las compensaciones entre la cobertura de pruebas y el rendimiento/latencia de las pruebas y cómo esto puede afectar la verificación [Debatir] **IX:** Usar una herramienta de seguimiento de defectos para gestionar defectos de software en un proyecto de software pequeño [Usar] **X:** Discutir las limitaciones de las pruebas en ciertos dominios [Debatir]

Bibliografía: [Beck, 2002]

5.5 Herramientas y Análisis de Pruebas, 8 hr(s)

Competencias: AG-C11

Tópicos: **I:** Herramientas y automatización de verificación y validación *a*) Análisis estático *b*) Cobertura de código *c*) Fuzzing *d*) Análisis dinámico y contención de fallos (sanitizadores, etc.) *e*) Registro y seguimiento de fallos **II:** Planificación y generación de pruebas *a*) Estimación de fallos y terminación de pruebas, incluyendo siembra de defectos *b*) Uso de números aleatorios y pseudoaleatorios en las pruebas **III:** Pruebas de sistemas asíncronos, paralelos y concurrentes **IV:** Verificación y validación de artefactos que no son código (documentación, materiales de capacitación) **V:** Desarrollo guiado por comportamiento (BDD) *a*) Expresar criterios de aceptación como escenarios Gherkin Given/When/Then *b*) Ejecutar escenarios BDD como pruebas automatizadas de extremo a extremo **VI:** Pruebas de mutación y análisis estático de calidad de código *a*) Pruebas de mutación para evaluar la efectividad de una suite de pruebas *b*) Detección estática de *code smells* y complejidad ciclomática excesiva

Objetivos de Aprendizaje **I:** Describir y comparar diferentes herramientas para verificación y validación [Describir] **II:** Automatizar el uso de diferentes herramientas en un proyecto de software pequeño [Diseñar] **III:** Explicar cómo y cuándo se deben usar números aleatorios en las pruebas [Explicar] **IV:** Describir enfoques para la estimación de fallos [Describir] **V:** Estimar el número de fallos en una aplicación de software pequeña basándose en la densidad de fallos y la siembra de fallos [Estimar] **VI:** Describir técnicas y problemas con las pruebas de software asíncrono,

concurrente y paralelo [Describir] **VII:** Crear un plan de pruebas para un segmento de código de tamaño mediano que contenga código asíncrono, concurrente y/o paralelo, incluyendo una matriz de pruebas y generación de datos y entradas de prueba [Crear] **VIII:** Describir técnicas para la verificación y validación de artefactos que no son código [Describir] **IX:** Escribir escenarios Gherkin Given/When/Then para los criterios de aceptación de una funcionalidad y ejecutarlos como pruebas automatizadas de extremo a extremo [Escribir] **X:** Analizar el puntaje de mutación de una suite de pruebas y refactorizar las pruebas para eliminar los mutantes sobrevivientes [Analizar]

Bibliografía: [Wynne and Hellesøy, 2012]

5.6 Compatibilidad y Versionado de API, 4 hr(s)

Competencias: AG-C11,AG-C12

Tópicos: I: Ley de Hyrum / Ley de las Interfaces Implícitas **II:** Compatibilidad con versiones anteriores *a*) La compatibilidad no es una propiedad de una sola entidad, es una propiedad de una relación. *b*) La compatibilidad con versiones anteriores debe evaluarse en términos de proveedor + consumidor(es) o con un modelo bien especificado de qué formas de compatibilidad aspira/promete un proveedor. **III:** Control de versiones (versioning) *a*) Versionado Semántico (SemVer) *b*) Desarrollo basado en tronco (*trunk-based development*)

Objetivos de Aprendizaje I: Identificar tanto el comportamiento explícito como el implícito de una interfaz e identificar riesgos potenciales de la Ley de Hyrum [Analizar] **II:** Identificar cambios que pueden considerarse ampliamente "compatibles con versiones anteriores", potencialmente con declaraciones explícitas sobre qué uso está o no soportado [Analizar] **III:** Evaluar si un cambio propuesto es lo suficientemente seguro dada la metodología de control de versiones en uso para un proyecto dado [Evaluar]

Bibliografía: [Sommerville, 2015]

5.7 Técnicas de Refactorización, 4 hr(s)

Competencias: AG-C09,AG-C12

Tópicos: I: Refactorización *a*) Patrones de refactorización estándar (renombrar, integrar, extraer, etc.) *b*) Uso de herramientas de refactorización en el IDE *c*) Aplicación de herramientas de análisis estático (para identificar código que necesita refactorización, generar cambios, etc.) *d*) Valor de la refactorización como remedio para la deuda técnica **II:** Refactorización "a Gran Escala": técnicas cuando un cambio de refactorización es demasiado grande para comprometerlo de forma segura (proyectos grandes), o cuando es imposible sincronizar el cambio entre proveedor + todos los consumidores (múltiples repositorios, consumidores con código privado). *a*) Expresar tanto las API antiguas como las nuevas para que puedan coexistir. *b*) Minimizar el tamaño de los cambios de comportamiento. *c*) ¿Por qué se requieren estas técnicas?. Por ejemplo, "consumidores de API que puedo ver" vs "consumidores que no puedo ver").

Objetivos de Aprendizaje I: Considerar las entradas de herramientas de análisis estático y/o los principios de Diseño de Software para identificar código que necesita refactorización [Analizar] **II:** Refactorizar la implementación de una interfaz para mejorar el diseño, claridad, etc., con impacto mínimo/cero en los usuarios existentes [Crear] **III:** Planificar una refactorización compleja de múltiples pasos para cambiar el comportamiento predeterminado de una API de manera segura [Planear]

Bibliografía: [Fowler, 2017]

5.8 Pruebas de Rendimiento y Evaluación Comparativa, 4 hr(s)

Competencias:

Tópicos: **I:** Pruebas de rendimiento y evaluación comparativa *a*) Rendimiento (*throughput*) y latencia *b*) Degradación bajo carga (pruebas de estrés, manejo FIFO vs LIFO de solicitudes) *c*) Aceleración (*speedup*) y escalado (*scaling*) 1) Ley de Amdahl 2) Ley de Gustafson 3) Escalado suave (*soft*) y débil (*weak*) *d*) Identificar y medir figuras de mérito (*figures of merit*) *e*) Cuellos de botella de rendimiento comunes 1) Limitado por cómputo (*compute-bound*) 2) Limitado por ancho de banda de memoria 3) Limitado por latencia *f*) Métodos estadísticos y mejores prácticas para la evaluación comparativa (benchmarking) 1) Estimación de la incertidumbre 2) Intervalos de confianza *g*) Análisis y presentación (gráficos, etc.) *h*) Técnicas de medición de tiempo (*timing*)

Objetivos de Aprendizaje **I:** Describir el rendimiento (*throughput*) y la latencia y proporcionar ejemplos de cada uno [Describir] **II:** Explicar la aceleración (speedup) y las diferentes formas de escalado (scaling) y cómo se calculan [Explicar] **III:** Describir cuellos de botella de rendimiento comunes [Describir] **IV:** Describir métodos estadísticos y mejores prácticas para la evaluación comparativa (benchmarking) de software [Describir] **V:** Explicar técnicas y desafíos para medir el tiempo al construir una evaluación comparativa (benchmark) [Explicar] **VI:** Identificar las figuras de mérito, construir y ejecutar una evaluación comparativa (benchmark) y analizar estadísticamente y visualizar los resultados para un proyecto de software pequeño [Analizar]

Bibliografía: [Bondi, 2015, Gregg, 2020]

5.9 Ingeniería de Confiabilidad, 4 hr(s)

Competencias: AG-C11,AG-C12

Tópicos: **I:** Modelos de confiabilidad de software **II:** Técnicas y modelos de tolerancia a fallos de software *a*) Diferencias contextuales en la tolerancia a fallos (por ejemplo, hacer fallar un sistema crítico para la aviación se evita firmemente, hacer fallar un sistema de procesamiento de datos antes de que se escriban datos corruptos en el almacenamiento es muy valioso) **III:** Prácticas de ingeniería de confiabilidad de software - incluyendo revisiones, pruebas, verificación de modelos práctica (practical model checking) **IV:** Identificación de dominios de fallos dependientes e independientes, y su impacto en la confiabilidad del sistema **V:** Análisis basado en mediciones de la confiabilidad del software - telemetría, monitorización y alertas, paneles de control, métricas de calificación de lanzamientos, etc.

Objetivos de Aprendizaje **I:** Demostrar la capacidad de aplicar múltiples métodos para desarrollar estimaciones de confiabilidad para un sistema de software [Demostrar] **II:** Identificar métodos que conducirán a la realización de una arquitectura de software que logre un nivel especificado de confiabilidad [Analizar] **III:** Identificar formas de aplicar redundancia para lograr tolerancia a fallos [Analizar] **IV:** Identificar dependencias de punto único de fallo (single-point-of-failure, SPF) en un diseño de sistema [Analizar]

Bibliografía: [Kim et al., 2021, Forsgren et al., 2018]

5.10 Construcción a Escala y Proceso, 4 hr(s)

Competencias: AG-C12

Tópicos: **I:** Pruebas a mayor escala *a*) Dobles de prueba (test doubles) (*stubs*, *mocks*, *fakes*) *b*) Inyección de dependencias (*dependency injection*) **II:** Secuenciación del trabajo, incluyendo identificación de dependencias, hitos y retiro de riesgo *a*) Identificación de dependencias: identificar las dependencias entre diferentes tareas *b*) Hitos: una colección de tareas que sirve como marcador de progreso cuando se completa. Idealmente, el hito abarca una unidad útil de funcionalidad. *c*) Retiro de riesgo: identificar qué elementos de un proyecto son riesgosos y priorizar la finalización de tareas que aborden esos riesgos. **III:** Problemas potenciales de seguridad en programas Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada *a*) Desbordamientos de búfer y otros tipos de desbordamientos *b*) Condiciones de carrera (race conditions) *c*) Inicialización incorrecta,

incluida la elección de privilegios *d*) Validación de entrada (input validation) **IV:** Documentación (autogenerada) **V:** Contexto de desarrollo: "campo verde" (*green field*) vs base de código existente *a*) Análisis de impacto del cambio (*change impact analysis*) *b*) Realización del cambio (*change actualization*) **VI:** Gestión de lanzamientos (release management) **VII:** Prácticas de DevOps *a*) Definition of Done como umbral de calidad compartido para el trabajo terminado *b*) Postmortems sin culpa (*blameless postmortems*) para aprender de incidentes/fallas *c*) Cultura de calidad en equipo: propiedad compartida, retroalimentación y aprendizaje continuo

Objetivos de Aprendizaje I: Reescribir un programa simple para eliminar vulnerabilidades comunes, como desbordamientos de búfer, desbordamientos de enteros y condiciones de carrera [Aplicar] **II:** Escribir un componente de software que realice alguna tarea no trivial y sea resistente a errores de entrada y de tiempo de ejecución [Escribir]

Bibliografía: [Forsgren et al., 2018]

5.11 Proyecto Final: Demostración de un Pipeline de Calidad de Extremo a Extremo, 8 hr(s)

Competencias: AG-C09

Tópicos: I: Integración de todas las prácticas del curso en un único repositorio en vivo: protección de ramas, revisión obligatoria de PRs, suites de pruebas automatizadas (unitarias, de integración, E2E/BDD), pipeline de CI/CD con puertas de calidad, escaneo de dependencias, análisis estático y observabilidad. **II:** Revisión en vivo del repositorio: demostración y defensa del pipeline de calidad del equipo ante compañeros/instructor. **III:** Retrospectiva y postmortem sin culpa del recorrido de calidad del proyecto.

Objetivos de Aprendizaje I: Integrar control de versiones, pruebas automatizadas, CI/CD, análisis estático y observabilidad en un único pipeline de calidad de extremo a extremo para un repositorio real. [Usar] **II:** Defender el diseño y la efectividad del pipeline de calidad de un equipo mediante una revisión en vivo del repositorio. [Usar] **III:** Realizar un postmortem sin culpa para identificar mejoras de proceso para proyectos futuros. [Evaluar]

Bibliografía: [Forsgren et al., 2018]

6. Bibliografía

[Beck, 2002] Beck, K. (2002). *Test-Driven Development: By Example*. Addison-Wesley.

[Bondi, 2015] Bondi, A. B. (2015). *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice*. Addison-Wesley, Upper Saddle River, NJ.

[Forsgren et al., 2018] Forsgren, N., Humble, J., and Kim, G. (2018). *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. IT Revolution Press.

[Fowler, 2017] Fowler, M. (2017). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 2nd edition.

[Gregg, 2020] Gregg, B. (2020). *Systems Performance: Enterprise and the Cloud*. Addison-Wesley Professional, Boston, MA, 2nd edition.

[International Organization for Standardization, 2011] International Organization for Standardization (2011). Iso/iec 25010:2011 – systems and software engineering – systems and software quality requirements and evaluation (square) – system and software quality models. ISO/IEC.

[Kim et al., 2021] Kim, G., Humble, J., Debois, P., Willis, J., and Forsgren, N. (2021). *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, 2nd edition.

[Sommerville, 2015] Sommerville, I. (2015). *Software Engineering*. Pearson, 10th edition.

[Wynne and Hellesøy, 2012] Wynne, M. and Hellesøy, A. (2012). *The Cucumber Book: Behaviour-Driven Development for Testers and Developers*. Pragmatic Bookshelf.

Noveno Semestre

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS370 Big Data (Obligatorio)

1. Información general

- **Semestre**: 9no Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 1 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS3P1 Computación Paralela y Distribuída (8vo Sem-Pág 455)

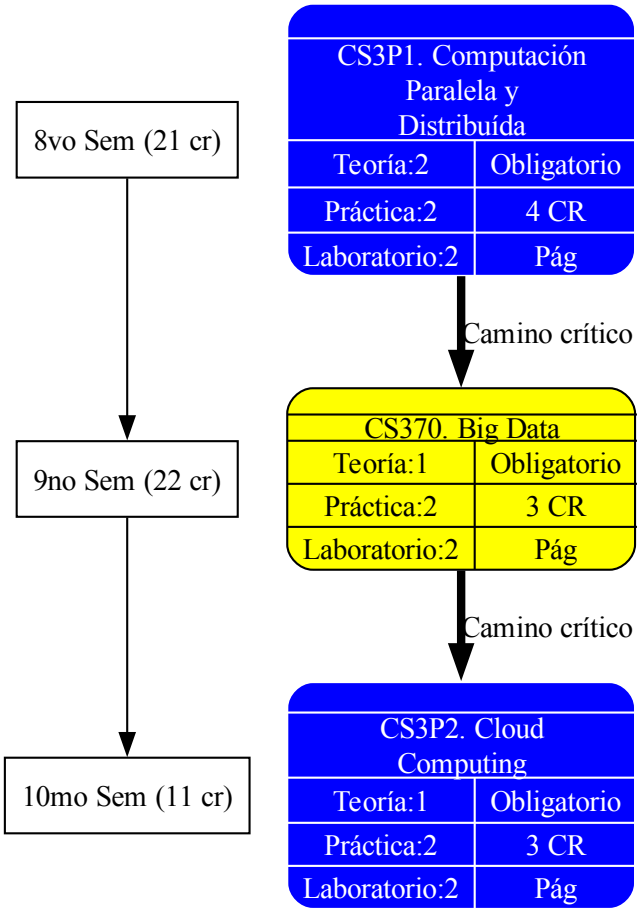

Figura 5.55: Mapa de Conexión. CS370 Big Data

2. Fundamentación

En la era actual, el volumen, velocidad y variedad de los datos superan las capacidades de los sistemas tradicionales de almacenamiento y procesamiento. Este curso introduce los conceptos y tecnologías fundamentales del Big Data. Cubre sistemas de archivos distribuidos, frameworks de procesamiento

paralelo, bases de datos NoSQL y flujo de datos en tiempo real. Los estudiantes aprenderán a diseñar arquitecturas escalables para extraer valor de conjuntos de datos masivos.

3. Objetivos del curso.

- Comprender las 5 V's del Big Data y su impacto en el diseño de sistemas.
- Diseñar e implementar soluciones escalables usando el paradigma MapReduce.
- Dominar el uso de sistemas de archivos distribuidos como HDFS.
- Analizar diferentes patrones arquitectónicos para Big Data (Lambda y Kappa).
- Aplicar técnicas de procesamiento de flujos para análisis de datos en tiempo real.

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Usar)

5. Contenido

5.1 Conceptos Fundamentales de Sistemas de Bases de Datos, 12 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Propósito y ventajas de los sistemas de bases de datos **II:** Componentes de los sistemas de bases de datos **III:** Diseño de funciones fundamentales de DBMS (por ejemplo, mecanismos de consulta, gestión de transacciones, gestión de búfer, métodos de acceso) **IV:** Arquitectura de base de datos, independencia de datos y abstracción de datos **V:** Gestión de transacciones **VI:** Normalización **VII:** Enfoques para gestionar grandes volúmenes de datos (por ejemplo, sistemas de bases de datos NoSQL, uso de MapReduce) **VIII:** Cómo soportar aplicaciones solo de CRUD **IX:** Bases de datos distribuidas/sistemas basados en la nube **X:** Datos estructurados, semiestructurados y no estructurados **XI:** Uso de un lenguaje de consulta declarativo **XII:** Sistemas que soportan contenido estructurado y/o de flujo

Objetivos de Aprendizaje **I:** Identificar al menos cuatro ventajas que proporciona el uso de un sistema de base de datos [Analizar] **II:** Enumerar los componentes de un sistema de base de datos (relacional) [Enumerar] **III:** Seguir una consulta a medida que es procesada por los componentes de un sistema de base de datos (relacional) [Analizar] **IV:** Defender el valor de la independencia de datos [Defender] **V:** Componer una consulta simple de selección-proyección-unión en SQL [Componer] **VI:** Enumerar las cuatro propiedades de un gestor de transacciones correcto [Enumerar] **VII:** Describir las ventajas de eliminar datos duplicados repetidos [Describir] **VIII:** Esbozar cómo MapReduce usa paralelismo para procesar datos eficientemente [Esquematizar/Esbozar] **IX:** Evaluar las diferencias entre bases de datos estructuradas y semiestructuradas/no estructuradas [Evaluar]

Bibliografía: [White, 2015, Karau et al., 2017]

5.2 Bases de Datos Distribuidas/Computación en la Nube, 12 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** DBMS Distribuido: *a*) Almacenamiento de datos distribuido *b*) Procesamiento de consultas distribuido *c*) Modelo de transacción distribuida *d*) Soluciones homogéneas y heterogéneas *e*) Bases de datos distribuidas cliente-servidor **II:** DBMS Paralelo: *a*) Arquitecturas de DBMS paralelo: memoria compartida, disco compartido, nada compartido *b*) Aceleración y escalamiento, por ejemplo, uso del modelo de procesamiento MapReduce *c*) Replicación de datos y modelos de consistencia débil

Objetivos de Aprendizaje I: Describir los componentes clave de un DBMS distribuido, incluyendo almacenamiento de datos distribuido, procesamiento de consultas y gestión de transacciones [Describir] **II:** Analizar las ventajas y desventajas entre arquitecturas de DBMS paralelo: memoria compartida, disco compartido y nada compartido [Analizar] **III:** Describir estrategias de replicación de datos y modelos de consistencia débil en sistemas de bases de datos distribuidas [Describir]

Bibliografía: [White, 2015, Lin and Dyer, 2010]

5.3 Sistemas NoSQL, 12 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: ¿Por qué NoSQL? (por ejemplo, Desajuste de impedancia entre Aplicación [CRUD] y RDBMS) **II:** Modelo de datos Clave-Valor y Documento **III:** Sistemas de almacenamiento (por ejemplo, sistemas Clave-Valor, Lagos de Datos (*Data Lakes*)) **IV:** Modelos de Distribución (Fragmentación y Replicación) **V:** Bases de Datos de Grafos **VI:** Modelos de Consistencia (Actualización y Lectura, consistencia de quórum, teorema CAP) **VII:** Modelo de procesamiento (por ejemplo, Map-Reduce, map-reduce multi-etapa, map-reduce incremental) **VIII:** Estudios de Caso: Sistema de almacenamiento en la nube (por ejemplo, S3); Bases de datos de grafos; "Cuándo no usar NoSQL"

Objetivos de Aprendizaje I: Desarrollar un caso de uso para el uso de NoSQL sobre RDBMS [Crear] **II:** Describir las características definitorias detrás de los modelos de datos basados en Clave-Valor y Documentos [Describir]

Bibliografía: [Sadalage and Fowler, 2012, Kleppmann, 2017]

5.4 Análisis de Datos, 12 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Técnicas exploratorias de datos (motivación, representación, estadísticas descriptivas, visualizaciones) **II:** Ciclo de vida de ciencia de datos: comprensión del negocio, comprensión de datos, preparación de datos, modelado, evaluación, despliegue y aceptación del usuario **III:** Algoritmos de minería de datos y aprendizaje automático: por ejemplo, clasificación, agrupamiento, asociación, regresión **IV:** Adquisición y gobernanza de datos **V:** Consideraciones de seguridad y privacidad de datos **VI:** Equidad y sesgo de datos **VII:** Técnicas de visualización de datos y su uso en análisis de datos **VIII:** Resolución de Entidades

Objetivos de Aprendizaje I: Describir varios enfoques de exploración de datos, incluyendo visualización, para comprender conjuntos de datos no familiares [Describir] **II:** Aplicar varios enfoques de exploración de datos para comprender conjuntos de datos no familiares [Aplicar] **III:** Describir algoritmos básicos de aprendizaje automático/minería de datos y cuándo son apropiados para su uso [Describir] **IV:** Aplicar varios algoritmos de aprendizaje automático/minería de datos [Aplicar] **V:** Describir consideraciones legales y éticas en la adquisición, uso y modificación de conjuntos de datos [Describir] **VI:** Describir problemas de equidad y sesgo en la recolección y uso de datos [Describir]

Bibliografía: [Matei et al., 2012, Bill and Matei, 2018, Kleppmann, 2017, Psaltis, 2017]

6. Bibliografía

[Bill and Matei, 2018] Bill, C. and Matei, Z. (2018). *Spark: The Definitive Guide: Big Data Processing Made Simple*. O'Reilly Media, Sebastopol, CA.

[Karau et al., 2017] Karau, H., Konwinski, A., Wendell, P., and Zaharia, M. (2017). *Learning Spark: Lightning-Fast Big Data Analysis*. O'Reilly Media.

[Kleppmann, 2017] Kleppmann, M. (2017). *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media.

[Lin and Dyer, 2010] Lin, J. and Dyer, C. (2010). *Data-Intensive Text Processing with MapReduce*. Morgan and Claypool Publishers.

[Matei et al., 2012] Matei, Z., Mosharaf, C., Tathagata, D., Ankur, D., Justin, M., Murphy, M., J, F. M., Scott, S., and Ion, S. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA. USENIX Association.

[Psaltis, 2017] Psaltis, A. (2017). *Streaming Data: Understanding the real-time pipeline*. Manning Publications.

[Sadalage and Fowler, 2012] Sadalage, P. J. and Fowler, M. (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley.

[White, 2015] White, T. (2015). *Hadoop: The Definitive Guide*. O'Reilly Media, 4th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS400 Prácticas Pre-profesionales (Obligatorio)

1. Información general

- **Semestre**: 9no Sem. **Créditos**: 2
- **Horas del curso**: **Teoría**: 1 horas; **Práctica**: 2 horas;
- **Prerrequisitos**:
  - 110Cr

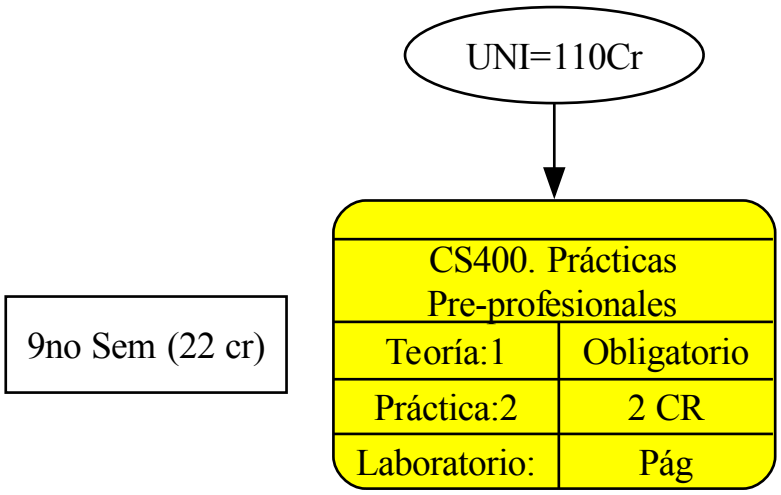

Figura 5.56: Mapa de Conexión. CS400 Prácticas Pre-profesionales

2. Fundamentación

Este curso permite a los estudiantes aplicar los conocimientos adquiridos durante su formación académica en un entorno laboral real, bajo la supervisión de una empresa y la universidad. Las prácticas pre-profesionales son esenciales para desarrollar competencias profesionales, éticas y técnicas, asegurando que el estudiante adquiera experiencia en proyectos relacionados con la carrera.

3. Objetivos del curso.

- Evaluar el desempeño del estudiante en un entorno laboral real, aplicando conocimientos de Ciencia de la Computación.
- Desarrollar habilidades profesionales, éticas y de trabajo en equipo en un contexto empresarial.
- Garantizar que las actividades realizadas estén alineadas con los objetivos de la carrera.

4. Competencias Generales (*Outcomes*)

AG-C01) El Profesional y el Mundo: Analiza y evalúa el impacto de las soluciones a problemas complejos de computación en el desarrollo sostenible de la sociedad. (Usar)

AG-C02) Ética: Aplica principios éticos y se compromete con la ética profesional y las normas de la práctica profesional de la computación. (Evaluar)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Evaluar)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Evaluar)

AG-C05) Gestión de Proyectos: Aplica los principios de gestión en computación para gestionar proyectos. (Usar)

AG-C06) Aprendizaje a lo largo de la vida: Reconoce la importancia del aprendizaje continuo y la adaptación a nuevas tecnologías. (Evaluar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Evaluar)

5. Contenido

5.1 Evaluación de Desempeño en Entorno Laboral, 48 hr(s)

Competencias: AG-C02,AG-C04,AG-C05,AG-C09

Tópicos: I: Integración del estudiante en un equipo de trabajo profesional. **II:** Aplicación de conocimientos técnicos en proyectos reales. **III:** Evaluación de competencias éticas y profesionales en el ámbito laboral. **IV:** Presentación de informes y retroalimentación por parte de la empresa y la universidad.

Objetivos de Aprendizaje I: Demostrar habilidades técnicas en un entorno laboral real [Evaluar]. **II:** Aplicar principios éticos y profesionales en sus actividades [Evaluar]. **III:** Trabajar efectivamente en equipo y comunicarse de manera profesional [Evaluar].

Bibliografía: [ACM Education Committee, 2022, IEEE Computer Society, 2021]

6. Bibliografía

[ACM Education Committee, 2022] ACM Education Committee (2022). Guide to successful pre-professional internships in computer science. Technical report, Association for Computing Machinery. Official document with standards for internships in computing.

[IEEE Computer Society, 2021] IEEE Computer Society (2021). *Professional Practices in Computing: Ethics and Management*. IEEE Press. Best-practices manual for IT work environments.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS403 Proyecto de tesis 2 (Obligatorio)

1. Información general

- **Semestre**: 9no Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - CS402 Proyecto de tesis 1 (8vo Sem-Pág 465)

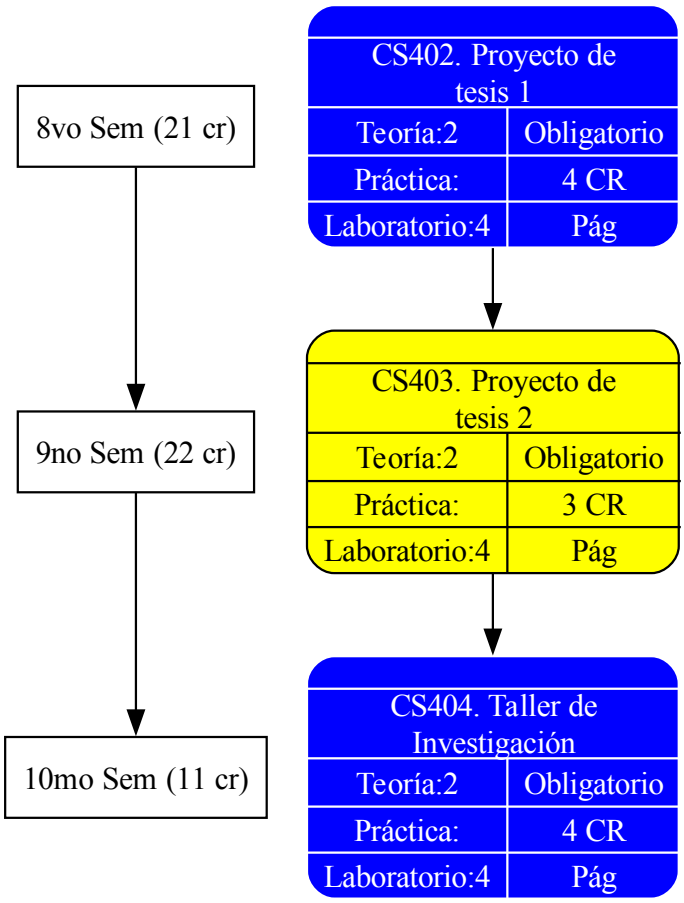

Figura 5.57: Mapa de Conexión. CS403 Proyecto de tesis 2

2. Fundamentación

Este curso tiene por objetivo que el alumno desarrolle su propuesta de investigación y ejecute los primeros experimentos formales, produciendo los capítulos centrales de su tesis bajo la supervisión del asesor.

3. Objetivos del curso.

- Formalizar el plan de tesis en el formato institucional con objetivos, cronograma y análisis de viabilidad.

- Redactar el marco teórico y el capítulo de trabajos relacionados con síntesis crítica de la literatura.
- Diseñar la metodología original e implementar un prototipo funcional de la propuesta de investigación.
- Ejecutar experimentos preliminares comparativos con rigor estadístico e interpretación fundamentada de resultados.
- Presentar y defender el avance de tesis ante el comité evaluador incorporando su retroalimentación.
- Los entregables de este curso son:

  Avance parcial: Plan de tesis en formato institucional, capítulo de marco teórico y trabajos relacionados en LaTeX.

  Final: Capítulos de metodología, propuesta y experimentos preliminares aprobados por el asesor.

4. Competencias Generales (*Outcomes*)

AG-C02) Ética: Aplica principios éticos y se compromete con la ética profesional y las normas de la práctica profesional de la computación. (Evaluar)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Evaluar)

AG-C05) Gestión de Proyectos: Aplica los principios de gestión en computación para gestionar proyectos. (Evaluar)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Evaluar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Evaluar)

5. Contenido

5.1 Formalización del Plan de Tesis, 10 hr(s)

Competencias: AG-C05,AG-C08,AG-C09

Tópicos: **I:** Estructura del plan de tesis según el formato institucional. **II:** Definición precisa del problema de investigación y pregunta central de tesis. **III:** Formulación de objetivos generales y específicos verificables. **IV:** Cronograma de actividades con hitos hasta la defensa final de tesis. **V:** Análisis de viabilidad: recursos, tiempos, riesgos y contingencias del proyecto.

Objetivos de Aprendizaje **I:** Producir el plan de tesis en el formato exigido por la institución académica. [Evaluar] **II:** Definir un problema de investigación original y objetivos verificables. [Evaluar] **III:** Elaborar un cronograma detallado con hitos y análisis de riesgos hasta la defensa de tesis. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.2 Marco Teórico y Estado del Arte Avanzado, 10 hr(s)

Competencias: AG-C04,AG-C08,AG-C09

Tópicos: **I:** Organización temática y síntesis crítica de la literatura del área de tesis. **II:** Redacción del capítulo de marco teórico en LaTeXcon estructura coherente. **III:** Identificación y justificación de las brechas de investigación del problema. **IV:** Análisis comparativo de enfoques: tablas y diagramas de clasificación de la literatura. **V:** Integración de referencias con BibTeX y gestión de versiones con Git.

Objetivos de Aprendizaje I: Redactar el capítulo de marco teórico con síntesis estructurada y crítica de la literatura. [Evaluar] **II:** Justificar la originalidad del problema en relación con el estado del arte. [Evaluar] **III:** Construir tablas comparativas que evidencien las brechas de investigación. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008, CiteSeer.IST, 2008]

5.3 Diseño de la Metodología Original, 10 hr(s)

Competencias: AG-C05,AG-C08,AG-C09

Tópicos: I: Definición del enfoque metodológico: experimental, analítico o constructivo. **II:** Especificación de la contribución original: algoritmo, sistema, modelo o metodología. **III:** Diseño del protocolo experimental: variables, métricas, datasets y benchmarks. **IV:** Criterios de validación y generalización de resultados de la propuesta. **V:** Alineación de la metodología con los objetivos del plan de tesis.

Objetivos de Aprendizaje I: Diseñar la metodología de investigación alineada con los objetivos. [Evaluar] **II:** Especificar la contribución original del proyecto con criterios verificables. [Evaluar] **III:** Definir el protocolo experimental completo con variables, métricas y benchmarks. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.4 Implementación y Experimentos Preliminares, 12 hr(s)

Competencias: AG-C05,AG-C08,AG-C09

Tópicos: I: Implementación de la propuesta original: prototipo, algoritmo o modelo de tesis. **II:** Reproducción de baselines relevantes del estado del arte para comparación. **III:** Ejecución de experimentos preliminares con análisis estadístico básico. **IV:** Registro y documentación de configuraciones, versiones y resultados. **V:** Identificación de limitaciones y ajuste iterativo de la propuesta.

Objetivos de Aprendizaje I: Implementar un prototipo funcional de la propuesta original de tesis. [Usar] **II:** Ejecutar experimentos preliminares comparativos con baselines definidos. [Usar] **III:** Analizar e interpretar resultados con rigor estadístico apropiado. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.5 Redacción de Capítulos Centrales de la Tesis, 10 hr(s)

Competencias: AG-C02,AG-C04,AG-C09

Tópicos: I: Redacción del capítulo de metodología y diseño experimental en LaTeX. **II:** Redacción del capítulo de trabajos relacionados con análisis crítico comparativo. **III:** Presentación de resultados preliminares: tablas, figuras y análisis comparativo. **IV:** Ciclos de revisión con el asesor: incorporación de retroalimentación estructurada. **V:** Coherencia y consistencia narrativa entre capítulos de la tesis.

Objetivos de Aprendizaje I: Redactar los capítulos de metodología y trabajos relacionados con rigor académico. [Evaluar] **II:** Presentar resultados preliminares con visualizaciones claras y análisis comparativo. [Usar] **III:** Aplicar ciclos de revisión con el asesor para mejorar la calidad del documento. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.6 Presentación del Avance y Planificación Final, 8 hr(s)

Competencias: AG-C02,AG-C04,AG-C05

Tópicos: **I:** Presentación oral del avance de tesis ante el comité evaluador. **II:** Defensa de metodología y resultados preliminares ante preguntas del comité. **III:** Incorporación de retroalimentación del comité en la revisión del documento. **IV:** Planificación de actividades restantes hacia la defensa final de tesis. **V:** Gestión ética: autoría, integridad académica y confidencialidad de datos.

Objetivos de Aprendizaje **I:** Presentar y defender oralmente el avance de tesis ante un comité académico. [Evaluar] **II:** Incorporar la retroalimentación del comité en una revisión fundamentada del documento. [Evaluar] **III:** Planificar las actividades restantes con objetivos claros hacia la defensa final. [Usar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

6. Bibliografía

[ACM, 2008] ACM (2008). *Digital Library*. Association for Computing Machinery. http://portal.acm.org/dl.cfm.

[CiteSeer.IST, 2008] CiteSeer.IST (2008). *Scientific Literature Digital Library*. College of Information Sciences and Technology, Penn State University. http://citeseer.ist.psu.edu.

[IEEE-Computer Society, 2008] IEEE-Computer Society (2008). *Digital Library*. IEEE-Computer Society. http://www.computer.org/publications/dlib.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI365 Modelos Generativos Avanzados en IA (Obligatorio)

1. Información general

- **Semestre**: 9no Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - AI264 Aprendizaje Profundo (7mo Sem-Pág 435)

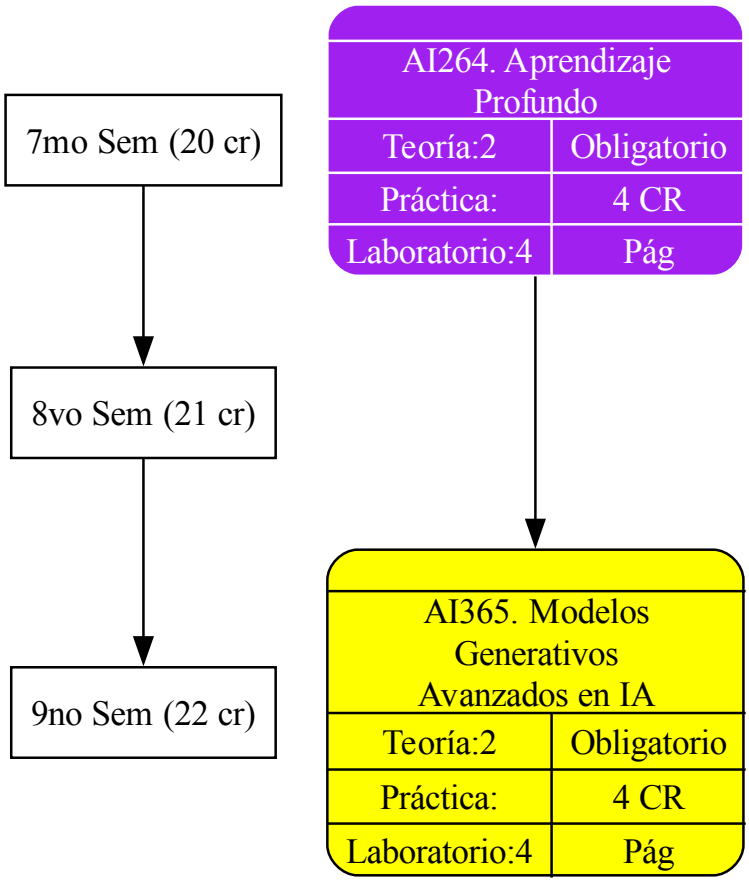

Figura 5.58: Mapa de Conexión. AI365 Modelos Generativos Avanzados en IA

2. Fundamentación

Este curso explora modelos generativos modernos (GANs, Diffusion Models, LLMs) para creación de contenido multimodal, abordando tanto fundamentos matemáticos como aplicaciones prácticas y consideraciones éticas en generación sintética.

3. Objetivos del curso.

- Implementar pipelines de generación de imágenes/texto/video

- Diseñar sistemas de edición asistida por IA
- Evaluar riesgos de contenido sintético

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Evaluar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Evaluar)

5. Contenido

5.1 Fundamentos Generativos, 12 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Teoría de GANs (minimax, Wasserstein) **II:** Autoregressive models (PixelCNN, Transformers) **III:** Métricas de evaluación (FID, Inception Score)

Objetivos de Aprendizaje I: Entrenar GANs básicas [Usar] **II:** Analizar modos de falla en generación [Evaluar]

Bibliografía: [Goodfellow et al., 2014]

5.2 Modelos de Difusión, 24 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Procesos de difusión (forward/reverse) **II:** Stable Diffusion y arquitecturas latentes **III:** Técnicas de control (ControlNet, LoRA)

Objetivos de Aprendizaje I: Implementar samplers de difusión [Usar] **II:** Fine-tuning para dominios específicos [Evaluar]

Bibliografía: [Sohl-Dickstein et al., 2023, Rombach et al., 2022]

5.3 Ética en Medios Sintéticos, 8 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Deepfakes y desinformación **II:** Derechos de autor en contenido generado **III:** Marcado de autenticidad (C2PA, watermarking)

Objetivos de Aprendizaje I: Detectar contenido sintético [Familiarizarse] **II:** Implementar sistemas de trazabilidad [Usar]

Bibliografía: [Sohl-Dickstein et al., 2023]

6. Bibliografía

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. In *NeurIPS*.

[Rombach et al., 2022] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. *CVPR*.

[Sohl-Dickstein et al., 2023] Sohl-Dickstein, J., Song, Y., and Ermon, S. (2023). *Diffusion Models: A Comprehensive Practical Guide*. AI Press.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CB309 Bioinformática (Obligatorio)

1. Información general

- **Semestre**: 9no Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 4 horas;
- **Prerrequisitos**: Ninguno

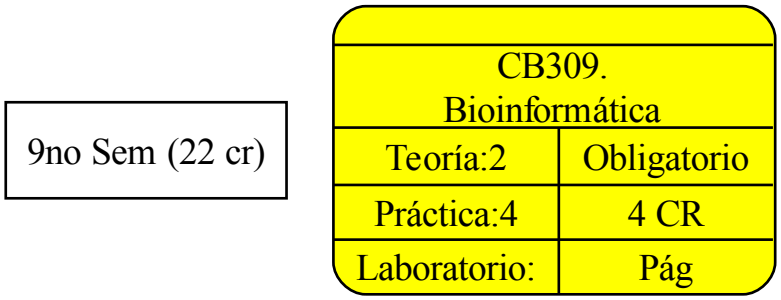

Figura 5.59: Mapa de Conexión. CB309 Bioinformática

2. Fundamentación

El uso de métodos computacionales en las ciencias biológicas se ha convertido en una de las herramientas claves para el campo de la biología molecular, siendo parte fundamental en las investigaciones de esta área.
En Biología Molecular, existen diversas aplicaciones que involucran tanto al ADN, al análisis de proteínas o al secuenciamiento del genoma humano, que dependen de métodos computacionales. Muchos de estos problemas son realmente complejos y tratan con grandes conjuntos de datos.
Este curso puede ser aprovechado para ver casos de uso concretos de varias áreas de conocimiento de Ciencia de la Computación como: Lenguajes de Programación (PL), Algoritmos y Complejidad (AL), Probabilidades y Estadística, Manejo de Información (IM), Inteligencia Artificial (IA).

3. Objetivos del curso.

- Que el alumno tenga un conocimiento sólido de los problemas biológicos moleculares que desafían a la computación.
- Que el alumno sea capaz de abstraer la esencia de los diversos problemas biológicos para plantear soluciones usando sus conocimientos de Ciencia de la Computación

4. Competencias Generales (*Outcomes*)

AG-C10) Indagación: Estudia problemas complejos de computación usando métodos de ciencias de la información. (Usar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Usar)

5. Contenido

5.1 Introducción a la Biología Molecular, 4 hr(s)

Competencias: AG-C10,AG-C11

Tópicos: **I:** Revisión de la química orgánica: moléculas y macromoléculas, azúcares, acidos nucleicos, nuclótidos, ARN, ADN, proteínas, aminoácidos y nivels de estructura en las proteinas. **II:** El dogma de la vida: del ADN a las proteinas, transcripción, traducción, síntesis de proteinas **III:** Estudio del genoma: Mapas y secuencias, técnicas específicas

Objetivos de Aprendizaje **I:** Lograr un conocimiento general de los tópicos más importantes en Biología Molecular. [Familiarizarse] **II:** Entender que los problemas biológicos son un desafío al mundo computacional. [Evaluar]

Bibliografía: [Clote and Backofen, 2000, Setubal and Meidanis, 1997]

5.2 Comparación de Secuencias, 4 hr(s)

Competencias: AG-C10,AG-C11

Tópicos: **I:** Secuencias de nucléotidos y secuencias de aminoácidos. **II:** Alineamiento de secuencias, el problema de alineamiento por pares, búsqueda exhaustiva, Programación dinámica, alineamiento global, alineamiento local, penalización por gaps **III:** Comparación de múltiples secuencias: suma de pares, análisis de complejidad por programación dinámica, heurísticas de alineamiento, algoritmo estrella, algoritmos de alineamiento progresivo.

Objetivos de Aprendizaje **I:** Entender y solucionar el problema de alineamiento de un par de secuencias. [Usar] **II:** Comprender y solucionar el problema de alineamiento de múltiples secuencias. [Usar] **III:** Conocer los diversos algoritmos de alineamiento de secuencias existentes en la literatura. [Familiarizarse]

Bibliografía: [Clote and Backofen, 2000, Setubal and Meidanis, 1997, Pevzner, 2000]

5.3 Árboles Filogenéticos, 4 hr(s)

Competencias: AG-C10,AG-C11

Tópicos: **I:** Filogenia: Introducción y relaciones filogenéticas. **II:** Arboles Filogenéticos: definición, tipo de árboles, problema de búsqueda y reconstrucción de árboles **III:** Métodos de Reconstrucción: métodos por parsimonia, métodos por distancia, métodos por máxima verosimilitud, confianza de los árboles reconstruidos

Objetivos de Aprendizaje **I:** Comprender el concepto de filogenia, árboles filogenéticos y la diferencia metodológica entre biología y biología molecular. [Familiarizarse] **II:** Comprender el problema de reconstrucción de árboles filogenéticos, conocer y aplicar los principales algoritmos para reconstrucción de árboles filogenéticos. [Evaluar]

Bibliografía: [Clote and Backofen, 2000, Setubal and Meidanis, 1997, Pevzner, 2000]

5.4 Ensamblaje de Secuencias de ADN, 4 hr(s)

Competencias: AG-C10,AG-C11

Tópicos: **I:** Fundamento biológico: caso ideal, dificultades, métodos alternativos para secuenciamiento de ADN **II:** Modelos formales de ensamblaje: *Shortest Common Superstring*, *Reconstruction*, *Multicontig* **III:** Algoritmos para ensamblaje de secuencias: representación de overlaps, caminos para crear *superstrings*, algoritmo voraz, grafos acíclicos. **IV:** Heurísticas para ensamblaje: búsqueda de sobreposiciones, ordenación de fragmentos, alineamientos y consenso.

Objetivos de Aprendizaje I: Comprender el desafío computacional que ofrece el problema de Ensamblaje de Secuencias. [Familiarizarse] **II:** Entender el principio de modelo formal para ensamblaje. [Evaluar] **III:** Conocer las principales heurísticas para el problema de ensambjale de secuencias ADN [Usar]

Bibliografía: [Setubal and Meidanis, 1997, Aluru, 2006]

5.5 Estructuras secundarias y terciarias, 4 hr(s)

Competencias: AG-C10,AG-C11

Tópicos: I: Estructuras moleculares: primaria, secundaria, terciaria, cuaternaria. **II:** Predicción de estructuras secundarias de ARN: modelo formal, energia de pares, estructuras con bases independientes, solución con Programación Dinámica, estructuras con bucles. **III:** *Protein folding*: Estructuras en proteinas, problema de *protein folding*. **IV:** *Protein Threading*: Definiciones, Algoritmo *Branch & Bound*, *Branch & Bound* para *protein threading*. **V:** *Structural Alignment*: definiciones, algoritmo DALI

Objetivos de Aprendizaje I: Conocer las estructuras protéicas y la necesidad de métodos computacionales para la predicción de la geometría. [Familiarizarse] **II:** Conocer los algoritmos de solución de problemas de predicción de estructuras secundarias ARN, y de estructuras en proteínas. [Evaluar]

Bibliografía: [Setubal and Meidanis, 1997, Clote and Backofen, 2000, Aluru, 2006]

5.6 Modelos Probabilísticos en Biología Molecular, 4 hr(s)

Competencias: AG-C10,AG-C11

Tópicos: I: Probabilidad: Variables aleatorias, Cadenas de Markov, Algoritmo de Metropoli-Hasting, Campos Aleatorios de Markov y Muestreador de Gibbs, Máxima Verosimilitud. **II:** Modelos Ocultos de Markov (HMM), estimación de parámetros, algoritmo de Viterbi y método Baul-Welch, Aplicación en alineamientos de pares y múltiples, en detección de Motifs en proteínas, en ADN eucariótico, en familias de secuencias. **III:** Filogenia Probabilística: Modelos probabilísticos de evolución, verosimilitud de alineamientos, verosimilitud para inferencia, comparación de métodos probailísticos y no probabilísticos

Objetivos de Aprendizaje I: Revisar conceptos de Modelos Probabilísticos y comprender su importancia en Biología Molecular Computacional. [Evaluar] **II:** Conocer y aplicar Modelos Ocultos de Markov para varios análisis en Biología Molecular. [Usar] **III:** Conocer la aplicación de modelos probabilísticos en Filogenia y comparalos con modelos no probabilísticos [Evaluar]

Bibliografía: [R et al., 1998, Clote and Backofen, 2000, Aluru, 2006, Krogh et al., 1994]

6. Bibliografía

[Aluru, 2006] Aluru, S., editor (2006). *Handbook of Computational Molecular Biology*. Computer and Information Science Series. Chapman & Hall, CRC, Boca Raton, FL.

[Clote and Backofen, 2000] Clote, P. and Backofen, R. (2000). *Computational Molecular Biology: An Introduction*. John Wiley & Sons Ltd. 279 pages.

[Krogh et al., 1994] Krogh, A., Brown, M., Mian, I. S., Sjölander, K., and Haussler, D. (1994). Hidden markov models in computational biology, applications to protein modeling. *J Mol. Biol*, 235:1501–1531.

[Pevzner, 2000] Pevzner, P. A. (2000). *Computational Molecular Biology: an Algorithmic Approach*. The MIT Press, Cambridge, Massachusetts.

[R et al., 1998] R, D., S.R, E., A, K., and G, M. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.

[Setubal and Meidanis, 1997] Setubal, J. C. and Meidanis, J. (1997). *Introduction to computational molecular biology*. Boston: PWS Publishing Company.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

ET301 Formación de Empresas de Base Tecnológica II (Obligatorio)

1. Información general

- **Semestre**: 9no Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas;
- **Prerrequisitos**:
  - ET201 Formación de Empresas de Base Tecnológica I (8vo Sem-Pág 471)

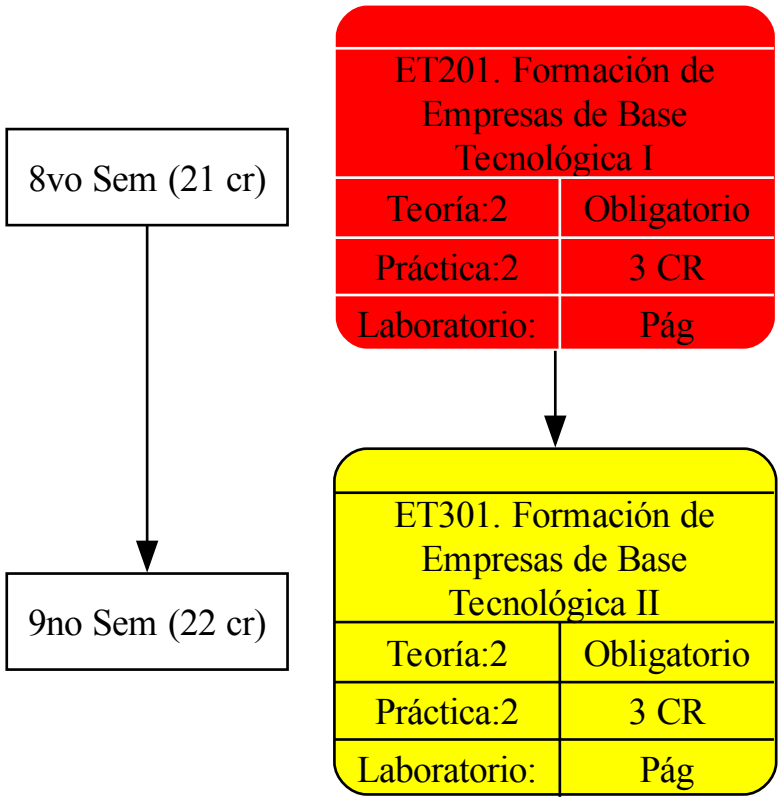

Figura 5.60: Mapa de Conexión. ET301 Formación de Empresas de Base Tecnológica II

2. Fundamentación

Este curso tiene como objetivo dotar al futuro profesional de conocimientos, actitudes y aptitudes que le permitan formar su propia empresa de desarrollo de software y/o consultoría en informática. El curso está dividido en tres unidades: Valorización de Proyectos, Marketing de Servicios y Negociaciones. En la primera unidad se busca que el alumno pueda analizar y tomar decisiones en relación a la viabilidad de un proyecto y/o negocio.

En la segunda unidad se busca preparar al alumno para que este pueda llevar a cabo un plan de marketing satisfactorio del bien o servicio que su empresa pueda ofrecer al mercado. La tercera unidad busca desarrollar la capacidad negociadora de los participantes a través del entrenamiento vivencial y práctico y de los conocimientos teóricos que le permitan cerrar contrataciones donde tanto el cliente

como el proveedor resulten ganadores. Consideramos estos temas sumamente críticos en las etapas de lanzamiento, consolidación y eventual relanzamiento de una empresa de base tecnológica.

3. Objetivos del curso.

- Que el alumno comprenda y aplique la terminología y conceptos fundamentales de ingeniería económica que le permitan valorizar un proyecto para tomar la mejor decisión económica.
- Que el alumno adquiera las bases para formar su propia empresa de base tecnológica.

4. Competencias Generales (*Outcomes*)

Sin elementos para esta tabla

5. Contenido

5.1 Valorización de Proyectos, 20 hr(s)

Competencias:

Tópicos: **I:** Introducción **II:** Proceso de toma de decisiones **III:** El valor del dinero en el tiempo **IV:** Tasa de interés y tasa de rendimiento **V:** Interés simple e interés compuesto **VI:** Identificación de costos **VII:** Flujo de Caja Neto **VIII:** Tasa de Retorno de Inversión (TIR) **IX:** Valor Presente Neto (VPN) **X:** Valorización de Proyectos

Objetivos de Aprendizaje **I:** Permitir al alumno tomar decisiones sobre como invertir mejor los fondos disponibles, fundamentadas en el análisis de los factores tanto económicos como no económicos que determinen la viabilidad de un emprendimiento. [Evaluar]

Bibliografía: [Blank and Tarkin, 2006]

5.2 Marketing de Servicios, 30 hr(s)

Competencias:

Tópicos: **I:** Introducción **II:** Importancia del marketing en las empresas de servicios **III:** El Proceso estratégico. **IV:** El Plan de Marketing **V:** Marketing estratégico y marketing operativo **VI:** Segmentación, targeting y posicionamiento de servicios en mercados competitivos **VII:** Ciclo de vida del producto **VIII:** Aspectos a considerar en la fijación de precios en servicios **IX:** El rol de la publicidad, las ventas y otras formas de comunicación **X:** El comportamiento del consumidor en servicios **XI:** Fundamentos de marketing de servicios **XII:** Creación del modelo de servicio **XIII:** Gestión de la calidad de servicio

Objetivos de Aprendizaje **I:** Brindar las herramientas al alumno para que pueda identificar, analizar y aprovechar las oportunidades de marketing que generan valor en un emprendimiento. [Usar] **II:** Lograr que el alumno conozca, entienda e identifique criterios, habilidades, métodos y procedimientos que permitan una adecuada formulación de estrategias de marketing en sectores y medios específicos como lo es una empresa de base tecnológica. [Usar]

Bibliografía: [Kotler and Keller, 2006, Lovelock and Wirtz, 2009]

5.3 Negociaciones, 10 hr(s)

Competencias:

Tópicos: **I:** Introducción. ¿Qué es una negociación? **II:** Teoría de las necesidades de la negociación **III:** La proceso de la negociación **IV:** Estilos de negociación **V:** Teoría de juegos **VI:** El método Harvard de negociación

Objetivos de Aprendizaje **I:** Conocer los puntos clave en el proceso de negociación. [Usar] **II:** Establecer una metodología de negociación eficaz. [Usar] **III:** Desarrollar destrezas y habilidades que permitan llevar a cabo una negociación exitosa. [Usar]

Bibliografía: [Fisher et al., 1996, de Manuel Dasí and Martínez, 2006]

6. Bibliografía

[Blank and Tarkin, 2006] Blank, L. and Tarkin, A. (2006). *Ingeniería Económica*. McGraw Hill, México D.F., México.

[de Manuel Dasí and Martínez, 2006] de Manuel Dasí, F. and Martínez, R. M.-V. (2006). *Técnicas de Negociación. Un método práctico*. Esic, Madrid.

[Fisher et al., 1996] Fisher, R., Ury, W., and Patton, B. (1996). *Si... ¡de acuerdo! Cómo negociar sin ceder*. Norma, Barcelona.

[Kotler and Keller, 2006] Kotler, P. and Keller, K. L. (2006). *Dirección de Marketing*. Prentice Hall, México.

[Lovelock and Wirtz, 2009] Lovelock, C. and Wirtz, J. (2009). *Marketing de servicios. Personal, tecnología y estrategia*. Prentice Hall, México.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI369 Robótica (Electivo)

1. Información general

- **Semestre**: 9no Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - AI268 Visión Computacional (8vo Sem-Pág 475)

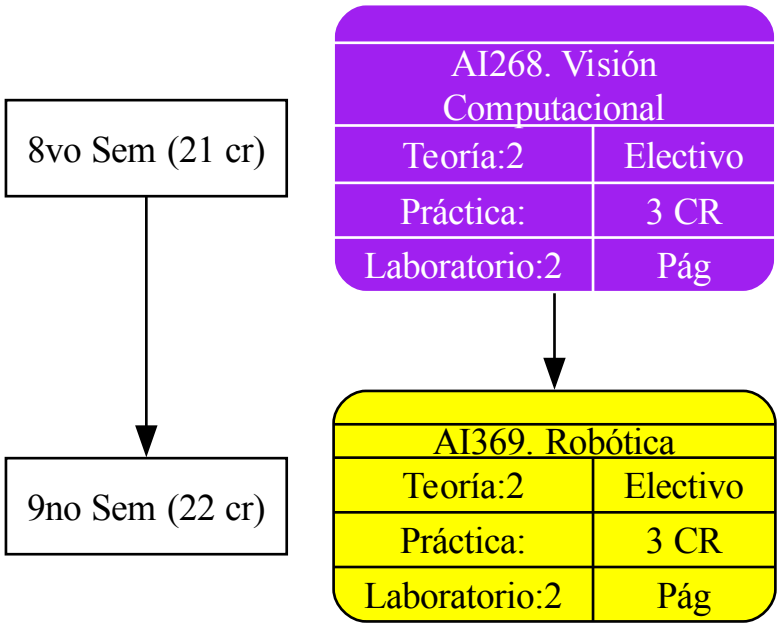

Figura 5.61: Mapa de Conexión. AI369 Robótica

2. Fundamentación

La Robótica es la manifestación física de la Inteligencia Artificial. Este curso introduce los conceptos fundamentales del diseño de robots, incluyendo cinemática, percepción y control. Los estudiantes explorarán cómo los robots interactúan con el mundo físico a través de sensores y actuadores, y cómo toman decisiones autónomas usando técnicas probabilísticas de mapeo y localización. El curso cierra la brecha entre algoritmos abstractos y ejecución física.

3. Objetivos del curso.

- Comprender los fundamentos mecánicos y matemáticos de la cinemática del robot.
- Dominar técnicas de fusión de sensores para percepción robótica.
- Implementar algoritmos de navegación y planificación de trayectorias.
- Analizar la integración del aprendizaje automático en el control robótico.
- Diseñar y simular agentes autónomos para entornos específicos.

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Evaluar)

5. Contenido

5.1 Robótica, 12 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Visión general: problemas y progreso: *a*) Sistemas robóticos de última generación, incluyendo sus sensores y una visión general de su procesamiento sensorial *b*) Arquitecturas de control de robots, por ejemplo, control deliberativo versus reactivo y vehículos de Braitenberg *c*) Modelado del mundo y modelos del mundo *d*) Incertidumbre inherente en la percepción y en el control **II:** Sensores y efectores: *a*) Sensores: por ejemplo, LIDAR, sonar, visión, profundidad, estereoscópico, cámaras de eventos, micrófonos, háptica *b*) Efectores: por ejemplo, ruedas, brazos, pinzas **III:** Marcos de coordenadas, traslación y rotación (2D y 3D) **IV:** Espacio de configuración y mapas ambientales **V:** Interpretación de datos sensoriales inciertos **VI:** Localización y mapeo **VII:** Navegación y control **VIII:** Cinemática directa e inversa **IX:** Planificación de trayectorias de movimiento y optimización de trayectorias **X:** Manipulación y agarre **XI:** Control y dinámica de articulaciones **XII:** Control basado en visión **XIII:** Coordinación y colaboración múltiple de robots **XIV:** Interacción humano-robot: *a*) Espacios de trabajo compartidos *b*) Equipos humano-robot e HRI físico *c*) Robots de asistencia social *d*) Predicción de movimiento/tarea/meta *e*) Colaboración y comunicación (explícita vs implícita, verbal o simbólica vs no verbal o visual) *f*) Confianza **XV:** Aplicaciones y Problemas Sociales, Económicos y éticos: *a*) Implicaciones sociales, económicas, de derecho al trabajo *b*) Implicaciones éticas y de privacidad de las aplicaciones robóticas *c*) Responsabilidad en robótica autónoma *d*) Armas autónomas y ética *e*) Supervisión y control humano

Objetivos de Aprendizaje **I:** Listar capacidades y limitaciones de los sistemas robóticos de última generación actuales, incluyendo sus sensores y el procesamiento sensorial crucial que informa a esos sistemas [Listar/Enumerar] **II:** Integrar sensores, actuadores y software en un robot diseñado para emprender una tarea específica [Aplicar] **III:** Programar un robot para realizar tareas simples usando arquitecturas de control deliberativas, reactivas y/o híbridas [Aplicar] **IV:** Implementar algoritmos fundamentales de planificación de movimiento dentro de un espacio de configuración de robot [Aplicar] **V:** Caracterizar las incertidumbres asociadas con sensores y actuadores robóticos comunes; articular estrategias para mitigar estas incertidumbres [Caracterizar] **VI:** Listar las diferencias entre las representaciones del entorno externo de los robots, incluyendo sus fortalezas y limitaciones [Listar/Enumerar] **VII:** Comparar y contrastar al menos tres estrategias para la navegación de robots dentro de entornos conocidos y/o desconocidos, incluyendo sus fortalezas y limitaciones [Comparar] **VIII:** Describir al menos un enfoque para coordinar las acciones y percepción de varios robots para realizar una única tarea [Describir] **IX:** Comparar y contrastar un enfoque de coordinación multirobot y un enfoque de colaboración humano-robot y atribuir sus diferencias a diferencias entre los entornos del problema [Comparar] **X:** Analizar los problemas sociales, económicos y éticos de una aplicación robótica del mundo real [Analizar]

Bibliografía: [Siegwart et al., 2011, Craig, 2017]

5.2 Percepción y Visión por Computadora, 12 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Visión por computadora: *a*) Adquisición, representación, procesamiento y propiedades de imágenes *b*) Representación de formas, reconocimiento de objetos y segmentación *c*) Análisis de

movimiento *d*) Modelos generativos **II:** Audio y reconocimiento de voz **III:** Tacto y propiocepción **IV:** Otras modalidades (por ejemplo, olfato) **V:** Modularidad en el reconocimiento **VI:** Enfoques para el reconocimiento de patrones: *a*) Algoritmos de clasificación y medidas de calidad de clasificación *b*) Técnicas estadísticas *c*) Técnicas de aprendizaje profundo

Objetivos de Aprendizaje I: Resumir la importancia del reconocimiento de imágenes y objetos en IA e indicar varias aplicaciones significativas de esta tecnología [Resumir] **II:** Listar al menos tres enfoques de segmentación de imágenes, como umbralización, algoritmos basados en bordes y basados en regiones, junto con sus características definitorias, fortalezas y debilidades [Listar/Enumerar] **III:** Implementar reconocimiento de objetos 2D basado en representaciones de forma basadas en contorno y/o región [Aplicar] **IV:** Distinguir los objetivos del reconocimiento de sonido, reconocimiento de voz y reconocimiento de hablante, e identificar cómo se manejará diferente la señal de audio cruda en cada uno de estos casos [Distinguir] **V:** Proporcionar al menos dos ejemplos de transformación de una fuente de datos de un dominio sensorial a otro, por ejemplo, datos táctiles interpretados como imágenes 2D de banda única [Generar] **VI:** Implementar un algoritmo de extracción de características en datos reales, por ejemplo, un detector de bordes o esquinas para imágenes o vectores de coeficientes de Fourier que describen un segmento corto de señal de audio [Aplicar] **VII:** Implementar un algoritmo que combine características en percepciones de nivel superior, por ejemplo, un contorno o polígono a partir de primitivas visuales o hipótesis de fonemas a partir de una señal de audio [Aplicar] **VIII:** Implementar un algoritmo de clasificación que segmenta percepciones de entrada en categorías de salida y evalúa cuantitativamente la clasificación resultante [Aplicar] **IX:** Evaluar el rendimiento de la extracción de características subyacente, relativo a al menos un enfoque alternativo posible (ya sea implementado o no) en su contribución a la tarea de clasificación (8), anterior [Evaluar] **X:** Describir al menos tres enfoques de clasificación, sus prerrequisitos para aplicabilidad, sus fortalezas y sus limitaciones [Describir] **XI:** Implementar y evaluar una solución de aprendizaje profundo para problemas en visión por computadora, como reconocimiento de objetos o escenas [Aplicar]

Bibliografía: [Thrun et al., 2005, Siegwart et al., 2011]

5.3 Robótica, 24 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Visión general: problemas y progreso: *a*) Sistemas robóticos de última generación, incluyendo sus sensores y una visión general de su procesamiento sensorial *b*) Arquitecturas de control de robots, por ejemplo, control deliberativo versus reactivo y vehículos de Braitenberg *c*) Modelado del mundo y modelos del mundo *d*) Incertidumbre inherente en la percepción y en el control **II:** Sensores y efectores: *a*) Sensores: por ejemplo, LIDAR, sonar, visión, profundidad, estereoscópico, cámaras de eventos, micrófonos, háptica *b*) Efectores: por ejemplo, ruedas, brazos, pinzas **III:** Marcos de coordenadas, traslación y rotación (2D y 3D) **IV:** Espacio de configuración y mapas ambientales **V:** Interpretación de datos sensoriales inciertos **VI:** Localización y mapeo **VII:** Navegación y control **VIII:** Cinemática directa e inversa **IX:** Planificación de trayectorias de movimiento y optimización de trayectorias **X:** Manipulación y agarre **XI:** Control y dinámica de articulaciones **XII:** Control basado en visión **XIII:** Coordinación y colaboración múltiple de robots **XIV:** Interacción humano-robot: *a*) Espacios de trabajo compartidos *b*) Equipos humano-robot e HRI físico *c*) Robots de asistencia social *d*) Predicción de movimiento/tarea/meta *e*) Colaboración y comunicación (explícita vs implícita, verbal o simbólica vs no verbal o visual) *f*) Confianza **XV:** Aplicaciones y Problemas Sociales, Económicos y éticos: *a*) Implicaciones sociales, económicas, de derecho al trabajo *b*) Implicaciones éticas y de privacidad de las aplicaciones robóticas *c*) Responsabilidad en robótica autónoma *d*) Armas autónomas y ética *e*) Supervisión y control humano

Objetivos de Aprendizaje I: Listar capacidades y limitaciones de los sistemas robóticos de última generación actuales, incluyendo sus sensores y el procesamiento sensorial crucial que informa a esos sistemas [Listar/Enumerar] **II:** Integrar sensores, actuadores y software en un robot diseñado para emprender una tarea específica [Aplicar] **III:** Programar un robot para realizar

tareas simples usando arquitecturas de control deliberativas, reactivas y/o híbridas [Aplicar] **IV:** Implementar algoritmos fundamentales de planificación de movimiento dentro de un espacio de configuración de robot [Aplicar] **V:** Caracterizar las incertidumbres asociadas con sensores y actuadores robóticos comunes; articular estrategias para mitigar estas incertidumbres [Caracterizar] **VI:** Listar las diferencias entre las representaciones del entorno externo de los robots, incluyendo sus fortalezas y limitaciones [Listar/Enumerar] **VII:** Comparar y contrastar al menos tres estrategias para la navegación de robots dentro de entornos conocidos y/o desconocidos, incluyendo sus fortalezas y limitaciones [Comparar] **VIII:** Describir al menos un enfoque para coordinar las acciones y percepción de varios robots para realizar una única tarea [Describir] **IX:** Comparar y contrastar un enfoque de coordinación multirobot y un enfoque de colaboración humano-robot y atribuir sus diferencias a diferencias entre los entornos del problema [Comparar] **X:** Analizar los problemas sociales, económicos y éticos de una aplicación robótica del mundo real [Analizar]

Bibliografía: [Thrun et al., 2005, Siegwart et al., 2011]

6. Bibliografía

[Craig, 2017] Craig, J. J. (2017). *Introduction to Robotics: Mechanics and Control*. Pearson, 4th edition.

[Siegwart et al., 2011] Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots*. MIT Press, 2nd edition.

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. MIT Press.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS351 Tópicos en Computación Gráfica (Electivo)

1. Información general

- **Semestre**: 9no Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS251 Computación Gráfica (7mo Sem-Pág 421)

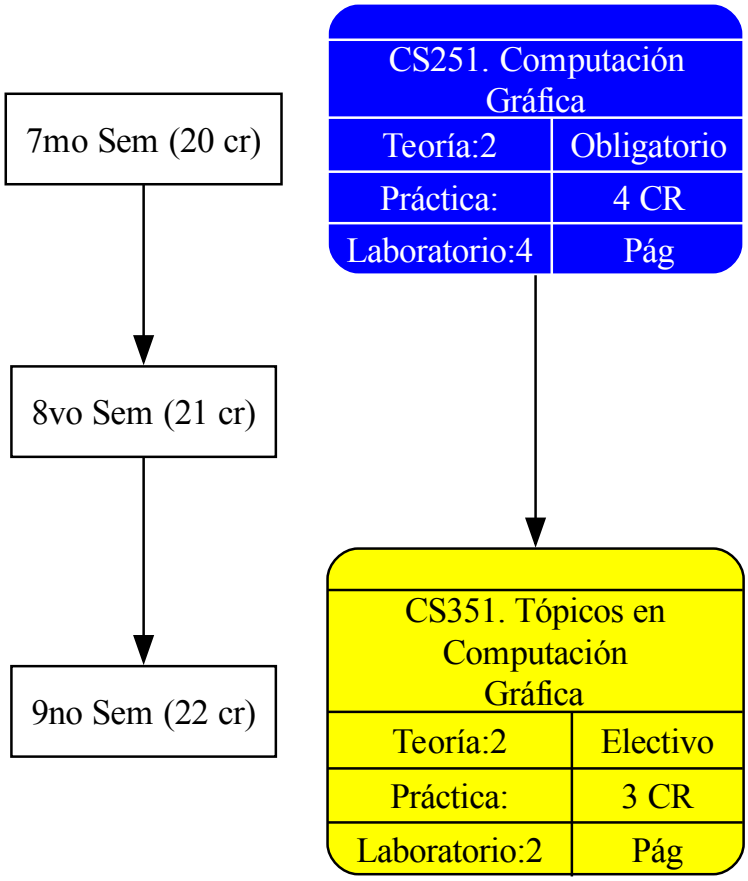

Figura 5.62: Mapa de Conexión. CS351 Tópicos en Computación Gráfica

2. Fundamentación

Los gráficos por computadora modernos van más allá de la simple generación de imágenes, abarcando interacción en tiempo real, simulación física y entornos inmersivos. Este curso explora temas avanzados en la canalización de gráficos, enfocándose en sombreado programable, modelado geométrico complejo y animación por computadora. Además, aborda la intersección de los gráficos con la computación física y las implicaciones éticas de la generación de contenido sintético.

3. Objetivos del curso.

- Dominar la programación avanzada de sombreadores y el renderizado no fotorrealista.
- Implementar simulaciones basadas en física y animaciones complejas.
- Diseñar experiencias inmersivas e interactivas usando computación tangible.
- Evaluar el impacto ético y social de las imágenes generadas por computadora.

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Usar)

5. Contenido

5.1 Sombreado y Renderizado Avanzado, 12 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Sombreado avanzado: sombreado basado en física, dispersión subsuperficial (subsurface scattering) **II:** Iluminación global (global illumination): radiosidad, mapeo de fotones (photon mapping), trazado de caminos (path tracing) **III:** Técnicas de renderizado en tiempo real: nivel de detalle (level of detail), culling, oclusión **IV:** Programación GPU y sombreadores (shaders)

Objetivos de Aprendizaje **I:** Implementar modelos de sombreado avanzados para materiales realistas [Implementar] **II:** Comparar algoritmos de iluminación global y sus compensaciones [Comparar] **III:** Optimizar el renderizado en tiempo real usando técnicas como nivel de detalle y *culling* [Crear] **IV:** Desarrollar sombreadores GPU para efectos de renderizado personalizados [Crear]

Bibliografía: [Marschner and Shirley, 2021a, Marschner and Shirley, 2021b]

5.2 Modelado Geométrico, 10 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Representaciones geométricas: polígonos, curvas, superficies, formas implícitas **II:** Técnicas de modelado: curvas spline, superficies de subdivisión, geometría sólida constructiva **III:** Modelado procedural y generación **IV:** Algoritmos geométricos: intersección, distancia, descomposición

Objetivos de Aprendizaje **I:** Describir diferentes representaciones geométricas y sus compensaciones (trade-offs) [Describir] **II:** Implementar técnicas de modelado como curvas spline o superficies de subdivisión [Implementar] **III:** Crear modelos procedurales usando algoritmos o reglas [Crear] **IV:** Aplicar algoritmos geométricos para resolver problemas como pruebas de intersección o procesamiento de mallas (mesh) [Aplicar]

Bibliografía: [Marschner and Shirley, 2021a]

5.3 Animación por Computadora, 8 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Principios de animación: tiempo (timing), espaciado (spacing), anticipación, seguimiento (follow-through) **II:** Animación por fotogramas clave (keyframe) e interpolación **III:** Animación procedural: sistemas de partículas, comportamiento de bandadas (flocking), basada en física **IV:** Animación de personajes: animación esquelética, skinning, cinemática inversa **V:** Captura de movimiento (motion capture) y retargeting

Objetivos de Aprendizaje I: Aplicar principios de animación para crear movimiento creíble [Aplicar] **II:** Implementar animación por fotogramas clave con técnicas de interpolación [Implementar] **III:** Crear animaciones procedurales como sistemas de partículas o simulaciones basadas en física [Crear] **IV:** Implementar animación de personajes usando sistemas esqueléticos y skinning [Implementar] **V:** Procesar datos de captura de movimiento para retargeting de animación [Implementar]

Bibliografía: [Parent, 2012]

5.4 Simulación, 8 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Simulación basada en física: cuerpos rígidos, fluidos, tela **II:** Métodos numéricos para simulación: integración, detección de colisiones **III:** Restricciones y optimizaciones de simulación en tiempo real **IV:** Aplicaciones en videojuegos, efectos visuales y visualización científica

Objetivos de Aprendizaje I: Implementar simulaciones basadas en física para cuerpos rígidos o fluidos [Implementar] **II:** Aplicar métodos de integración numérica para simulaciones dinámicas [Aplicar] **III:** Optimizar simulaciones para rendimiento en tiempo real [Crear] **IV:** Desarrollar simulaciones para aplicaciones específicas como videojuegos o visualización científica [Crear]

Bibliografía: [Parent, 2012]

5.5 Inmersión, 5 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Conceptos de realidad virtual (VR) y realidad aumentada (AR) **II:** Tecnologías de visualización inmersiva: pantallas montadas en la cabeza (head-mounted displays), CAVEs **III:** Seguimiento (tracking) y entrada para entornos inmersivos **IV:** Presencia y comodidad en sistemas inmersivos

Objetivos de Aprendizaje I: Explicar los conceptos centrales y diferencias entre VR y AR [Explicar] **II:** Evaluar tecnologías de visualización inmersiva y sus aplicaciones [Evaluar] **III:** Implementar soluciones de seguimiento y entrada para entornos inmersivos [Implementar] **IV:** Diseñar experiencias inmersivas que maximicen la presencia y minimicen la incomodidad [Diseñar]

Bibliografía: [LaValle, 2023]

5.6 Interacción, 5 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Paradigmas de interacción: manipulación directa, basada en gestos, tangible **II:** Diseño de interfaz de usuario para aplicaciones gráficas **III:** Interacción multimodal: combinando retroalimentación visual, auditiva, háptica **IV:** Evaluación de técnicas de interacción: usabilidad, experiencia de usuario

Objetivos de Aprendizaje I: Comparar diferentes paradigmas de interacción y su idoneidad para varias tareas [Comparar] **II:** Diseñar interfaces de usuario para aplicaciones intensivas en gráficos [Diseñar] **III:** Implementar sistemas de interacción multimodal incorporando múltiples canales de retroalimentación [Implementar] **IV:** Realizar evaluaciones de usabilidad de técnicas de interacción [Evaluar]

Bibliografía: [LaValle, 2023]

5.7 Computación Tangible/Física, 8 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Interfaces tangibles y computación física **II:** Sensores y actuadores para sistemas interactivos **III:** Sistemas embebidos para arte y diseño interactivo **IV:** Instalaciones y entornos interactivos

Objetivos de Aprendizaje **I:** Diseñar interfaces tangibles que unan interacciones físicas y digitales [Diseñar] **II:** Integrar sensores y actuadores para crear sistemas responsivos [Integrar] **III:** Desarrollar sistemas embebidos para proyectos de arte o diseño interactivo [Crear] **IV:** Crear instalaciones interactivas que involucren a los usuarios en espacios físicos [Crear]

Bibliografía: [Noble, 2012]

5.8 Sociedad, ética y Profesión, 8 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Consideraciones éticas en gráficos e interacción: representación, accesibilidad, privacidad **II:** Impacto social de las tecnologías inmersivas y la visualización **III:** Práctica profesional en industrias de gráficos e interactivas

Objetivos de Aprendizaje **I:** Analizar problemas éticos relacionados con representación, accesibilidad y privacidad en gráficos e interacción [Analizar] **II:** Discutir las implicaciones sociales de las tecnologías inmersivas y la visualización de datos [Debatir] **III:** Describir prácticas y estándares profesionales en industrias de gráficos e interactivas [Describir]

Bibliografía: [Marschner and Shirley, 2021a]

6. Bibliografía

[LaValle, 2023] LaValle, S. M. (2023). *Virtual Reality*. Cambridge University Press.

[Marschner and Shirley, 2021a] Marschner, S. and Shirley, P. (2021a). *Fundamentals of Computer Graphics*. CRC Press, 5th edition.

[Marschner and Shirley, 2021b] Marschner, S. and Shirley, P. (2021b). *Fundamentals of Computer Graphics*. CRC Press, 5th edition. Same book as Marschner2021; alternate citation key using lead author Shirley.

[Noble, 2012] Noble, J. (2012). *Programming Interactivity: A Designer's Guide to Processing, Arduino, and OpenFrameworks*. O'Reilly Media, 2nd edition.

[Parent, 2012] Parent, R. (2012). *Computer Animation: Algorithms and Techniques*. Morgan Kaufmann, 3rd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS353 Computación Cuántica (Electivo)

1. Información general

- **Semestre**: 9no Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS3I1 Seguridad en Computación (8vo Sem-Pág 449)

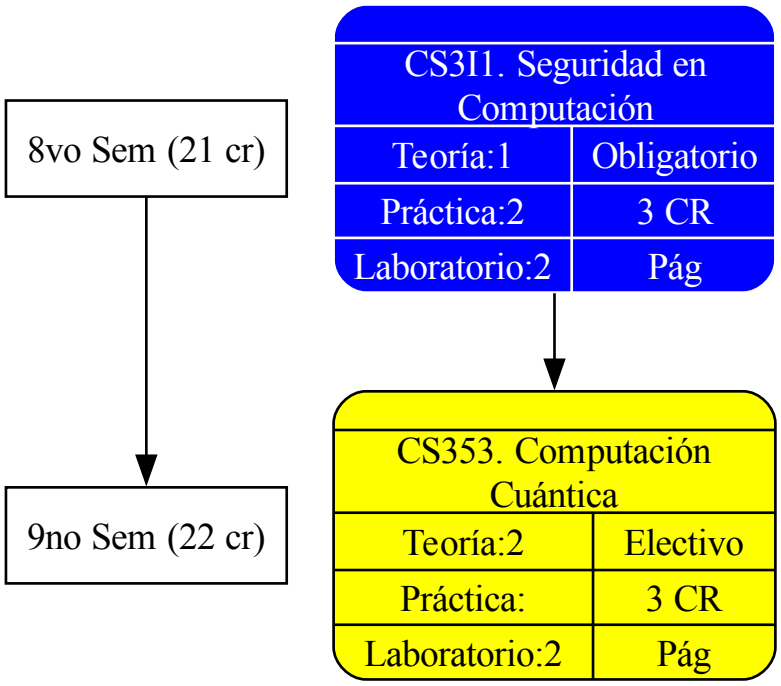

Figura 5.63: Mapa de Conexión. CS353 Computación Cuántica

2. Fundamentación

Este curso introduce los principios fundamentales de la computación cuántica, incluyendo cúbits, superposición, entrelazamiento, algoritmos cuánticos y hardware. Los estudiantes contrastarán modelos clásicos y cuánticos, explorando aplicaciones en criptografía, optimización, simulación física y aprendizaje automático cuántico, usando frameworks como Qiskit o Cirq. Incluye un módulo práctico sobre arquitecturas de hardware (superconductores, trampas de iones) y su impacto en el diseño de algoritmos.

3. Objetivos del curso.

- Comprender los postulados de la mecánica cuántica aplicados a la computación.
- Implementar circuitos cuánticos básicos y algoritmos de optimización.
- Analizar el impacto ético, técnico y comercial de la computación cuántica.
- Evaluar las limitaciones del hardware en aplicaciones del mundo real.

4. Competencias Generales (*Outcomes*)

AG-C10) Indagación: Estudia problemas complejos de computación usando métodos de ciencias de la información. (Usar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Usar)

5. Contenido

5.1 Conceptos fundamentales, 10 hr(s)

Competencias: AG-C10,AG-C11

Tópicos: I: Cúbits y representación en la esfera de Bloch. **II:** Postulados de la mecánica cuántica. **III:** Compuertas cuánticas (Hadamard, CNOT, Toffoli).

Objetivos de Aprendizaje I: Modelar matemáticamente un cúbit [Usar]. **II:** Simular operaciones básicas en Qiskit [Evaluar].

Bibliografía: [Nielsen and Chuang, 2010, Team, 2023]

5.2 Arquitecturas Cuánticas, 10 hr(s)

Competencias: AG-C10,AG-C11

Tópicos: I: Principios: *a*) El principio de dualidad onda-partícula *b*) El principio de incertidumbre en el experimento de doble rendija *c*) ¿Qué es un Qubit? Superposición, interferencia y medición. Fotones como qubits *d*) Sistemas de dos qubits, Entrelazamiento, estados de Bell, El teorema de No Señalización **II:** Axiomas de la MC: principio de superposición, axioma de medición, evolución unitaria **III:** Compuertas de un solo qubit para el modelo de circuito de computación cuántica: X, Z, H **IV:** Compuertas de dos qubits y productos tensoriales, trabajando con matrices **V:** El Teorema de No Clonación. El protocolo de Teleportación Cuántica **VI:** Algoritmos: *a*) Algoritmos cuánticos simples: Bernstein-Vazirani, algoritmo de Simon *b*) Implementar Deutsch-Josza con Interferómetros Mach-Zehnder *c*) Factorización cuántica (Algoritmo de Shor) *d*) Búsqueda cuántica (Algoritmo de Grover) **VII:** Aspectos de implementación: *a*) La implementación física de qubits *b*) Control clásico de una Unidad de Procesamiento Cuántico (QPU) *c*) Mitigación y control de errores, NISQ y más allá *d*) Enfoques de medición **VIII:** Aplicaciones Emergentes: *a*) Cifrado post-cuántico *b*) La Internet Cuántica *c*) Computación cuántica adiabática (AQC) y recocido cuántico

Objetivos de Aprendizaje I: Discutir cómo un objeto cuántico producido como partícula se propaga como una onda y se detecta como partícula con una distribución de probabilidad correspondiente a la onda [Debatir] **II:** Discutir la naturaleza a nivel cuántico que es inherentemente probabilística [Debatir] **III:** Expresar tu punto de vista sobre el entrelazamiento que puede usarse para crear correlaciones no clásicas, pero no hay forma de usar el entrelazamiento cuántico para enviar mensajes más rápido que la velocidad de la luz [Evaluar] **IV:** Describir el paralelismo cuántico y el papel de la interferencia constructiva vs destructiva en algoritmos cuánticos dada la naturaleza probabilística de las mediciones [Describir] **V:** Analizar un fragmento de código proporcionando el papel de la transformada cuántica de Fourier (QFT) en el algoritmo de Shor [Analizar] **VI:** Escribir un programa para implementar el algoritmo de Shor en un simulador, destacando los componentes clásicos y aspectos del algoritmo de Shor [Escribir] **VII:** Enumerar los detalles de cada modalidad de qubit (por ejemplo, ion atrapado, superconductor, espín de silicio, fotónico, punto cuántico, átomo neutro, topológico, centro de color, electrón sobre helio) [Enumerar] **VIII:** Contrastar AQC con el modelo de compuertas de computación cuántica y los problemas para los que cada uno es más adecuado resolver [Contrastar]

Bibliografía: [Möttönen and Vartiainen, 2023, IBM Research, 2023]

5.3 Estrategias Algorítmicas, 12 hr(s)

Competencias: AG-C10,AG-C11

Tópicos: I: Computación cuántica **II:** Algoritmo de Deutsch-Jozsa. **III:** Búsqueda de Grover (aplicada a optimización combinatoria). **IV:** Factorización de Shor y su impacto en RSA.

Objetivos de Aprendizaje I: Implementar Grover para problemas SAT [Usar].

Bibliografía: [Preskill, 2018, Team, 2023]

5.4 Lenguajes Formales y Autómatas, 10 hr(s)

Competencias: AG-C10,AG-C11

Tópicos: I: Computación cuántica: *a*) Postulados de la mecánica cuántica: 1) Espacio de estados 2) Evolución del estado 3) Composición de estados 4) Medición del estado *b*) Representaciones de vector columna de qubits *c*) Representaciones matriciales de operaciones cuánticas *d*) Compuertas cuánticas simples (por ejemplo, XNOT, CNOT) **II:** Algoritmo de Optimización Cuántica Aproximada (QAOA). **III:** Aplicaciones en logística/finanzas (Optimización de Cartera).

Objetivos de Aprendizaje I: Para un sistema cuántico dar ejemplos que expliquen los siguientes postulados: *a*) Espacio de Estados - estado del sistema representado como un vector unitario en espacio de Hilbert *b*) Evolución del Estado - el uso de operadores unitarios para evolucionar el estado del sistema *c*) Composición de Estados - el uso de producto tensorial para componer estados del sistema *d*) Medición del Estado - la salida probabilística de medir un estado del sistema. [Familiarizarse]. **II:** Diseñar un QAOA para problemas de enrutamiento [Evaluar].

Bibliografía: [Schuld and Petruccione, 2021, Bravyi et al., 2022]

5.5 Ethics and Post-Quantum Security, 6 hr(s)

Competencias: AG-C10,AG-C11

Tópicos: I: Criptografía post-cuántica (Estandarización NIST PQC). **II:** Supremacía cuántica: debate técnico/ético. **III:** Patentes y geopolítica cuántica.

Objetivos de Aprendizaje I: Proponer estrategias de migración post-cuántica [Familiarizarse].

Bibliografía: [Team, 2019]

6. Bibliografía

[Bravyi et al., 2022] Bravyi, S., Dial, O., Gambetta, J. M., Gil, D., and Nazario, Z. (2022). Quantum algorithms for fixed qubit architectures. *Nature Reviews Physics*, 4(8):499–512. Survey on quantum algorithms for optimization and simulation with near-term devices.

[IBM Research, 2023] IBM Research (2023). Ibm quantum hardware roadmap. Technical report, IBM Research. Accessed: 2026-02-13. The original page has been moved or updated.

[Möttönen and Vartiainen, 2023] Möttönen, M. and Vartiainen, J. (2023). *Quantum Computing: From Qubits to Quantum Machines*. Springer.

[Nielsen and Chuang, 2010] Nielsen, M. A. and Chuang, I. L. (2010). *Quantum Computation and Quantum Information*. Cambridge University Press.

[Preskill, 2018] Preskill, J. (2018). Lecture notes on quantum computation. Ph219/CS219 course, California Institute of Technology. Material available online. The original URL `http://theory.caltech.edu/~preskill/ph219/` is no longer active (February 2026). Check the author's website for the current version.

[Schuld and Petruccione, 2021] Schuld, M. and Petruccione, F. (2021). *Machine Learning with Quantum Computers*. Springer, Cham, Switzerland, 2nd edition. Comprehensive introduction to quantum machine learning algorithms and implementations.

[Team, 2019] Team, G. A. Q. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, 574.

[Team, 2023] Team, I. Q. (2023). *Qiskit Textbook*. IBM.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS392 Tópicos en Ingeniería de Software (Electivo)

1. Información general

- **Semestre**: 9no Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS391 Ingeniería de Software III (8vo Sem-Pág 481)

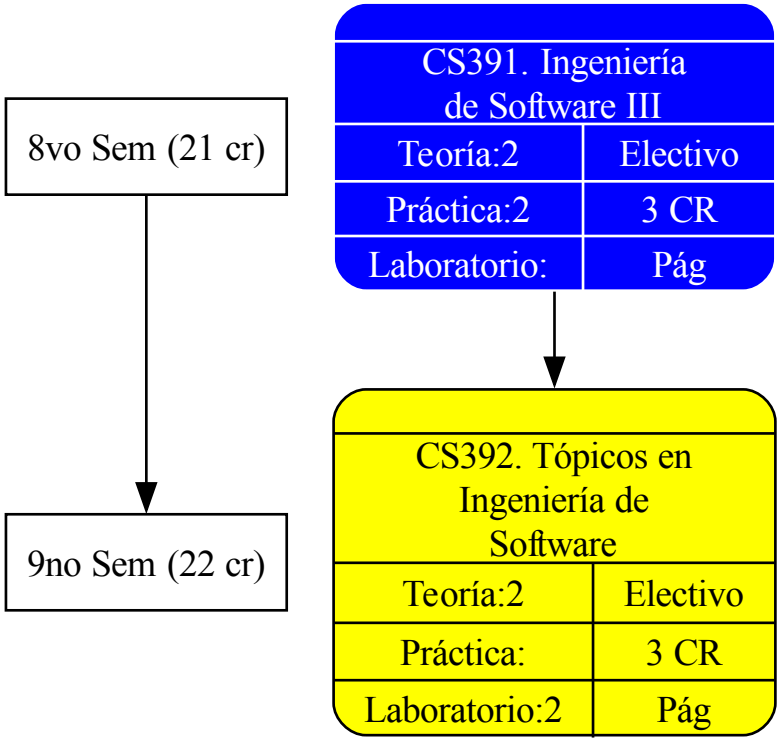

Figura 5.64: Mapa de Conexión. CS392 Tópicos en Ingeniería de Software

2. Fundamentación

El desarrollo de software moderno requiere prácticas de ingeniería rigurosas para garantizar la calidad, mantenibilidad y confiabilidad. Este curso explora temas avanzados en el ciclo de vida del software, enfocándose en la colaboración en equipo, verificación formal y refactorización avanzada. Los estudiantes dominarán el uso de herramientas profesionales para diseñar, construir y validar sistemas complejos mientras aseguran altos estándares de confiabilidad y corrección formal.

3. Objetivos del curso.

- Aplicar técnicas avanzadas de construcción y refactorización de software para mejorar la calidad del código.
- Usar métodos formales y estrategias de validación para garantizar la confiabilidad del sistema.

- Gestionar requisitos complejos y diseñar arquitecturas que cumplan con las necesidades de las partes interesadas.
- Liderar equipos de software usando herramientas profesionales y flujos de trabajo colaborativos.

4. Competencias Generales (*Outcomes*)

AG-C03) Trabajo Individual y en Equipo: Se desempeña efectivamente como individuo y como miembro o líder en equipos diversos. (Usar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

5. Contenido

5.1 Colaboración y Dinámica de Equipo, 4 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Comunicación efectiva, incluida la oral y escrita, así como formal (correo electrónico, documentos, comentarios, presentaciones) e informal (chat de equipo, reuniones). Comunicación Técnica y Profesional, Comunicación en Equipo y Cultura **II:** Causas comunes de conflicto en el equipo y enfoques para la resolución de conflictos. **III:** Programación cooperativa: *a*) Programación por parejas (pair programming) o Enjambre (swarming) *b*) Revisión de código *c*) Colaboración mediante control de versiones **IV:** Roles y responsabilidades en un equipo de software: Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo *a*) Ventajas del trabajo en equipo *b*) Riesgos y complejidad de dicha colaboración **V:** Procesos del equipo: responsabilidades para las tareas, estimación de esfuerzo, estructura de reuniones, cronograma de trabajo

Objetivos de Aprendizaje **I:** Seguir prácticas efectivas de comunicación en equipo [Analizar] **II:** Articular las fuentes, peligros y beneficios potenciales del conflicto en el equipo, especialmente enfocándose en el valor de estar en desacuerdo sobre ideas o propuestas sin insultar a las personas [Articular] **III:** Facilitar una estrategia de resolución de conflictos y resolución de problemas en un entorno de equipo [Analizar] **IV:** Colaborar efectivamente en desarrollo/programación cooperativa [Analizar] **V:** Proponer y delegar los roles y responsabilidades necesarios en un equipo de desarrollo de software [Proponer] **VI:** Redactar y seguir una agenda para una reunión de equipo [Componer] **VII:** Facilitar, a través de la participación en un proyecto de equipo, los elementos centrales de la formación de equipos, el establecimiento de una cultura de equipo saludable y la gestión de equipos, incluida la creación y ejecución de un plan de trabajo del equipo [Crear]

Bibliografía: [Sommerville, 2015]

5.2 Control de Versiones y CI/CD, 4 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Gestión de configuración de software y control de versiones: Prácticas de Desarrollo de Software *a*) Configuración en control de versiones, builds/configuración reproducibles. *b*) Estrategias de ramificación en control de versiones. Ramas de desarrollo vs ramas de lanzamiento. Desarrollo basado en tronco (*trunk-based development*). *c*) Estrategias de fusión/rebase, cuando sean relevantes. **II:** Gestión de lanzamientos (*release management*). **III:** Herramientas de prueba, incluyendo herramientas de análisis estático y dinámico. Prácticas de Desarrollo de Software, Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada **IV:** Automatización de procesos de software: *a*) Sistemas de construcción (*build systems*): el valor de builds rápidos, herméticos y reproducibles, comparar/contrastar enfoques para construir un proyecto. *b*) Integración

Continua (CI): el uso de automatización y pruebas automatizadas para hacer una validación preliminar de que la revisión actual del tronco se construye y pasa las pruebas (básicas). *c*) Despliegue Continuo (CD): el uso de automatización para liberar de forma automática cada cambio que supera las pruebas hacia el entorno de producción, garantizando entregas frecuentes y confiables. *d*) Gestión de dependencias: actualización de dependencias externas/aguas arriba, gestión de paquetes, SemVer. **V:** Flujo de revisión colaborativa como puerta de calidad *a*) Revisión obligatoria de *pull requests* antes de fusionar *b*) Propiedad del código mediante reglas de tipo CODEOWNERS *c*) Convenciones de mensajes de *commit* (*conventional commits*) *d*) Reglas de protección de ramas que exigen verificaciones antes de fusionar **VI:** Escaneo de vulnerabilidades de dependencias y de la cadena de suministro integrado en los pipelines de CI

Objetivos de Aprendizaje I: Describir la diferencia entre la gestión de configuración de software centralizada y distribuida [Describir] **II:** Describir cómo el control de versiones puede usarse para ayudar en la gestión de lanzamientos de software [Describir] **III:** Identificar elementos de configuración y usar una herramienta de control de código fuente en un proyecto pequeño basado en equipo [Analizar] **IV:** Describir cómo las herramientas de prueba estáticas y dinámicas disponibles pueden integrarse en el entorno de desarrollo de software [Describir] **V:** Comprender el uso de sistemas de CI/CD como una fuente de verdad para el estado del código compartido del equipo (éxito en la construcción y pruebas) [Explicar] **VI:** Aplicar un flujo de revisión de *pull requests* con reglas de propiedad de código y protección de ramas para imponer puertas de calidad en un proyecto de equipo [Aplicar] **VII:** Usar un escáner de vulnerabilidades de dependencias/cadena de suministro como parte de un pipeline de CI para detectar dependencias vulnerables [Usar]

Bibliografía: [Pressman and Maxim, 2014]

5.3 Ingeniería de Requisitos, 6 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Describir requisitos funcionales usando, por ejemplo, casos de uso o historias de usuario. *a*) Usar al menos un método para documentar y estructurar requisitos funcionales. *b*) Comprender cómo el método apoya el diseño y la implementación. *c*) Fortalezas y debilidades de usar un enfoque específico. **II:** Propiedades de los requisitos, incluyendo consistencia, validez, completitud y viabilidad. **III:** Obtención (*elicitation*) de requisitos. *a*) Fuentes de requisitos, por ejemplo, usuarios, administradores o personal de soporte. *b*) Métodos de recopilación de requisitos, por ejemplo, encuestas, entrevistas o análisis de comportamiento. **IV:** Requisitos no funcionales, por ejemplo, seguridad, usabilidad o rendimiento, también llamados Atributos de Calidad. Diseño Sostenible y Computación Ubicua, Huella Ambiental de los Sistemas Informáticos, Efectos Sistémicos y Contexto Social **V:** Identificación y gestión de riesgos, incluyendo consideraciones éticas en torno al producto propuesto. Códigos de Ética y Responsabilidad Legal, Responsabilidad Profesional y Política Pública, Valores, Naturaleza y Actualización Profesional, Disidencia Ética y Contexto Cultural, Ética en el Lugar de Trabajo **VI:** Comunicar y/o formalizar especificaciones de requisitos.

Objetivos de Aprendizaje I: Comparar diferentes métodos de obtención de requisitos a lo largo de múltiples ejes [Comparar] **II:** Identificar diferencias entre dos métodos de describir requisitos funcionales (por ejemplo, entrevistas con clientes, estudios de usuarios) y las situaciones donde se preferiría cada uno [Analizar] **III:** Identificar qué comportamientos son requeridos, permitidos o prohibidos a partir de un conjunto dado de requisitos y una lista de comportamientos candidatos [Analizar] **IV:** Recopilar un conjunto de requisitos para un sistema de software simple [Analizar] **V:** Identificar áreas de un sistema de software que deben cambiarse, dada una descripción del sistema y un conjunto de nuevos requisitos a implementar [Analizar] **VI:** Identificar los requisitos funcionales y no funcionales en un conjunto de requisitos [Analizar]

Bibliografía: [Sommerville, 2015]

5.4 Principios y Arquitectura de Software, 6 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Principios de diseño de sistemas. Confiabilidad del Sistema *a*) Niveles de abstracción (por ejemplo, diseño arquitectónico y diseño detallado) *b*) Separación de preocupaciones (separation of concerns) *c*) Ocultación de información (information hiding) *d*) Acoplamiento y cohesión (coupling and cohesion) **II:** Arquitectura de software. Confiabilidad del Sistema *a*) Paradigmas de diseño 1) Descomposición funcional de arriba hacia abajo/diseño por capas 2) Arquitectura orientada a datos 3) Análisis y diseño orientado a objetos 4) Diseño dirigido por eventos *b*) Arquitecturas estándar (por ejemplo, cliente-servidor y arquitecturas de microservicios incluyendo discusiones sobre REST, n-capas, tuberías y filtros, Modelo Vista Controlador) *c*) Identificar límites y dependencias de componentes **III:** Programación a gran escala vs programación a pequeña escala (programming in the large vs programming in the small). Confiabilidad del Sistema **IV:** Malos olores del código (code smells) y otras indicaciones de la calidad del código, distintas de la corrección. Diseño e Ingeniería de Controles de Seguridad, Análisis de Amenazas e Ingeniería de Seguridad, Computación Confiable e Ingeniería de Privacidad

Objetivos de Aprendizaje **I:** Identificar la arquitectura de software estándar de un diseño de alto nivel dado [Analizar] **II:** Seleccionar y usar un paradigma de diseño apropiado para diseñar un sistema de software simple y explicar cómo se han aplicado los principios de diseño de sistemas en este diseño [Evaluar] **III:** Adaptar un diseño de sistema defectuoso para que siga mejor principios como la separación de preocupaciones o la ocultación de información [Crear] **IV:** Identificar las dependencias entre un conjunto de componentes de software en un diseño arquitectónico [Analizar]

Bibliografía: [Sommerville, 2015]

5.5 Prácticas de Codificación, 6 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: **I:** Pruebas prácticas a pequeña escala Prácticas de Desarrollo de Software *a*) Pruebas unitarias *b*) Desarrollo guiado por pruebas (Test-driven development): esto es particularmente valioso para los estudiantes psicológicamente, ya que es mucho más fácil involucrarse constructivamente con el desafío de identificar entradas desafiantes para una API dada (casos límite, casos extremos) a priori. Si implementan primero, el instinto es a menudo evitar intentar hacer fallar su nueva creación, mientras que un enfoque de pruebas primero les da la satisfacción intelectual de detectar los casos problemáticos y luego ver cómo pasan más pruebas durante el proceso de desarrollo. **II:** Documentación Prácticas de Desarrollo de Software *a*) Documentación de la interfaz: describir los requisitos de la interfaz, potencialmente incluyendo contratos (formales o informales), pre y post condiciones, invariantes. *b*) La documentación de la implementación debe centrarse en partes de código complicadas y no obvias, ya sea porque el código usa características avanzadas del lenguaje o porque el comportamiento del código es complejo. (No agregar comentarios que reafirmen operaciones comunes/obvias y características simples del lenguaje.) 1) Aclarar el flujo de datos, cómputo, etc., centrándose en lo que el código es. 2) Identificar partes sutiles/complicadas del código y refactorizar para que sean autoexplicativas si es posible o proporcionar comentarios apropiados para aclarar. **III:** Estilo de codificación Prácticas de Desarrollo de Software *a*) Guías de estilo *b*) Comentarios *c*) Nomenclatura (*naming*) **IV:** "Mejores Prácticas" para la codificación: técnicas, patrones/idiomas, mecanismos para construir programas de calidad Prácticas de Desarrollo de Software, Flujo de Información y No Interferencia, Inyección y Validación de Entrada, Seguridad de Memoria y Tipos, Análisis de Malware y Seguridad Avanzada *a*) Prácticas de codificación defensiva *b*) Prácticas y principios de codificación segura *c*) Usar mecanismos de manejo de excepciones para hacer los programas más robustos, tolerantes a fallos **V:** Depuración (debugging) Prácticas de Desarrollo de Software **VI:** Registro (logging) **VII:** Uso de bibliotecas y frameworks desarrollados por otros Prácticas de Desarrollo de Software

Objetivos de Aprendizaje I: Escribir pruebas unitarias apropiadas para un componente pequeño (varias funciones, un solo tipo, etc.) [Escribir] **II:** Escribir comentarios apropiados de interfaz y (si es necesario) de implementación para un componente pequeño [Escribir] **III:** Describir técnicas, patrones de codificación y mecanismos para implementar diseños para lograr propiedades deseadas como confiabilidad, eficiencia y robustez [Describir] **IV:** Escribir código robusto utilizando mecanismos de manejo de excepciones [Escribir] **V:** Describir prácticas de codificación segura y defensiva [Describir] **VI:** Seleccionar y usar un estándar de codificación definido en un proyecto de software pequeño [Usar] **VII:** Comparar y contrastar estrategias de integración, incluyendo integración de arriba hacia abajo (top-down), de abajo hacia arriba (bottom-up) y sándwich [Comparar] **VIII:** Describir el proceso de analizar e implementar cambios en la base de código desarrollada para un proyecto específico [Describir] **IX:** Describir el proceso de analizar e implementar cambios en una base de código existente grande [Describir]

Bibliografía: [Fowler, 2018]

5.6 Compatibilidad y Versionado de API, 6 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Ley de Hyrum / Ley de las Interfaces Implícitas **II:** Compatibilidad con versiones anteriores *a*) La compatibilidad no es una propiedad de una sola entidad, es una propiedad de una relación. *b*) La compatibilidad con versiones anteriores debe evaluarse en términos de proveedor + consumidor(es) o con un modelo bien especificado de qué formas de compatibilidad aspira/promete un proveedor. **III:** Control de versiones (versioning) *a*) Versionado Semántico (SemVer) *b*) Desarrollo basado en tronco (*trunk-based development*)

Objetivos de Aprendizaje I: Identificar tanto el comportamiento explícito como el implícito de una interfaz e identificar riesgos potenciales de la Ley de Hyrum [Analizar] **II:** Identificar cambios que pueden considerarse ampliamente ”compatibles con versiones anteriores”, potencialmente con declaraciones explícitas sobre qué uso está o no soportado [Analizar] **III:** Evaluar si un cambio propuesto es lo suficientemente seguro dada la metodología de control de versiones en uso para un proyecto dado [Evaluar]

Bibliografía: [Fowler, 2018]

5.7 Conceptos de verificación y validación, 6 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Conceptos de verificación y validación *a*) Verificación: ¿Estamos construyendo la cosa correctamente? *b*) Validación: ¿Construimos la cosa correcta? **II:** Por qué importan las pruebas: ¿El componente sigue siendo funcional a medida que el código evoluciona? **III:** Objetivos de las pruebas *a*) Usabilidad *b*) Confiabilidad *c*) Conformidad con la especificación *d*) Rendimiento *e*) Seguridad

Objetivos de Aprendizaje I: Explicar por qué las pruebas son importantes [Explicar] **II:** Distinguir entre validación y verificación de programas [Distinguir] **III:** Describir diferentes objetivos de las pruebas [Describir]

Bibliografía: [Pressman and Maxim, 2014]

5.8 Herramientas y Análisis de Pruebas, 3 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Herramientas y automatización de verificación y validación *a*) Análisis estático *b*) Cobertura de código *c*) Fuzzing *d*) Análisis dinámico y contención de fallos (sanitizadores, etc.) *e*) Registro y seguimiento de fallos **II:** Planificación y generación de pruebas *a*) Estimación de fallos y terminación de pruebas, incluyendo siembra de defectos *b*) Uso de números aleatorios y

pseudoaleatorios en las pruebas **III:** Pruebas de sistemas asíncronos, paralelos y concurrentes **IV:** Verificación y validación de artefactos que no son código (documentación, materiales de capacitación) **V:** Desarrollo guiado por comportamiento (BDD) *a*) Expresar criterios de aceptación como escenarios Gherkin Given/When/Then *b*) Ejecutar escenarios BDD como pruebas automatizadas de extremo a extremo **VI:** Pruebas de mutación y análisis estático de calidad de código *a*) Pruebas de mutación para evaluar la efectividad de una suite de pruebas *b*) Detección estática de *code smells* y complejidad ciclomática excesiva

Objetivos de Aprendizaje I: Describir y comparar diferentes herramientas para verificación y validación [Describir] **II:** Automatizar el uso de diferentes herramientas en un proyecto de software pequeño [Diseñar] **III:** Explicar cómo y cuándo se deben usar números aleatorios en las pruebas [Explicar] **IV:** Describir enfoques para la estimación de fallos [Describir] **V:** Estimar el número de fallos en una aplicación de software pequeña basándose en la densidad de fallos y la siembra de fallos [Estimar] **VI:** Describir técnicas y problemas con las pruebas de software asíncrono, concurrente y paralelo [Describir] **VII:** Crear un plan de pruebas para un segmento de código de tamaño mediano que contenga código asíncrono, concurrente y/o paralelo, incluyendo una matriz de pruebas y generación de datos y entradas de prueba [Crear] **VIII:** Describir técnicas para la verificación y validación de artefactos que no son código [Describir] **IX:** Escribir escenarios Gherkin Given/When/Then para los criterios de aceptación de una funcionalidad y ejecutarlos como pruebas automatizadas de extremo a extremo [Escribir] **X:** Analizar el puntaje de mutación de una suite de pruebas y refactorizar las pruebas para eliminar los mutantes sobrevivientes [Analizar]

Bibliografía: [Pressman and Maxim, 2014]

5.9 Pruebas de Rendimiento y Evaluación Comparativa, 3 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Pruebas de rendimiento y evaluación comparativa *a*) Rendimiento (*throughput*) y latencia *b*) Degradación bajo carga (pruebas de estrés, manejo FIFO vs LIFO de solicitudes) *c*) Aceleración (*speedup*) y escalado (*scaling*) 1) Ley de Amdahl 2) Ley de Gustafson 3) Escalado suave (*soft*) y débil (*weak*) *d*) Identificar y medir figuras de mérito (*figures of merit*) *e*) Cuellos de botella de rendimiento comunes 1) Limitado por cómputo (*compute-bound*) 2) Limitado por ancho de banda de memoria 3) Limitado por latencia *f*) Métodos estadísticos y mejores prácticas para la evaluación comparativa (benchmarking) 1) Estimación de la incertidumbre 2) Intervalos de confianza *g*) Análisis y presentación (gráficos, etc.) *h*) Técnicas de medición de tiempo (*timing*)

Objetivos de Aprendizaje I: Describir el rendimiento (*throughput*) y la latencia y proporcionar ejemplos de cada uno [Describir] **II:** Explicar la aceleración (speedup) y las diferentes formas de escalado (scaling) y cómo se calculan [Explicar] **III:** Describir cuellos de botella de rendimiento comunes [Describir] **IV:** Describir métodos estadísticos y mejores prácticas para la evaluación comparativa (benchmarking) de software [Describir] **V:** Explicar técnicas y desafíos para medir el tiempo al construir una evaluación comparativa (benchmark) [Explicar] **VI:** Identificar las figuras de mérito, construir y ejecutar una evaluación comparativa (benchmark) y analizar estadísticamente y visualizar los resultados para un proyecto de software pequeño [Analizar]

Bibliografía: [Bondi, 2015, Gregg, 2020]

5.10 Conceptos de Confiabilidad, 5 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Concepto de confiabilidad como probabilidad de fallo o tiempo medio entre fallos, y fallas (faults) como causa de fallos (failures) **II:** Identificar requisitos de confiabilidad para diferentes tipos de software **III:** Fallos de software causados por defectos/errores (bugs), por lo que para alta confiabilidad el objetivo es tener un mínimo de defectos - inyectando menos defectos

(mejor capacitación, educación, planificación) y eliminando la mayoría de los defectos inyectados (pruebas, revisión de código, etc.) **IV:** Confiabilidad del software, confiabilidad del sistema y comportamiento de fallos **V:** Ciclo de inyección y eliminación de defectos, y diferentes enfoques para la eliminación de defectos **VI:** Comparar el enfoque de "presupuesto de error" (error budget) para la confiabilidad con el enfoque "libre de errores" (error-free) e identificar dominios donde cada uno es relevante.

Objetivos de Aprendizaje I: Describir cómo determinar el nivel de confiabilidad requerido por un sistema de software [Describir] **II:** Explicar los problemas que existen para lograr niveles muy altos de confiabilidad [Explicar] **III:** Comprender enfoques para minimizar fallas que pueden aplicarse en cada etapa del ciclo de vida del software [Explicar]

Bibliografía: [Pressman and Maxim, 2014]

5.11 Métodos Formales, 5 hr(s)

Competencias: AG-C03,AG-C09

Tópicos: I: Especificación formal de interfaces *a*) Especificación de pre y post condiciones *b*) Lenguajes formales para escribir y analizar pre y post condiciones. **II:** áreas de problemas bien atendidas por los métodos formales *a*) Programación libre de bloqueos (*lock-free*), condiciones de carrera de datos *b*) Sistemas asíncronos y distribuidos, bloqueos (*deadlock*), bloqueos vivos (*livelock*), etc. **III:** Comparación con otras herramientas y técnicas para la detección de defectos *a*) Pruebas *b*) *Fuzzing* **IV:** Enfoques formales para el modelado y análisis de software *a*) Verificadores de modelos (model checkers) *b*) Buscadores de modelos (model finders)

Objetivos de Aprendizaje I: Describir el papel que las técnicas de especificación y análisis formal pueden desempeñar en el desarrollo de software complejo y comparar su uso como técnicas de validación y verificación con las pruebas [Describir] **II:** Aplicar técnicas de especificación y análisis formal a diseños y programas de software de baja complejidad [Aplicar] **III:** Explicar los beneficios y desventajas potenciales del uso de lenguajes de especificación formal [Explicar]

Bibliografía: [Sommerville, 2015]

6. Bibliografía

[Bondi, 2015] Bondi, A. B. (2015). *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice*. Addison-Wesley, Upper Saddle River, NJ.

[Fowler, 2018] Fowler, M. (2018). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 2nd edition.

[Gregg, 2020] Gregg, B. (2020). *Systems Performance: Enterprise and the Cloud*. Addison-Wesley Professional, Boston, MA, 2nd edition.

[Pressman and Maxim, 2014] Pressman, R. S. and Maxim, B. (2014). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 8th edition.

[Sommerville, 2015] Sommerville, I. (2015). *Software Engineering*. Pearson, 10th edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS3P3 Internet de las Cosas (Electivo)

1. Información general

- **Semestre**: 9no Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 1 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - CS3P1 Computación Paralela y Distribuída (8vo Sem-Pág 455)

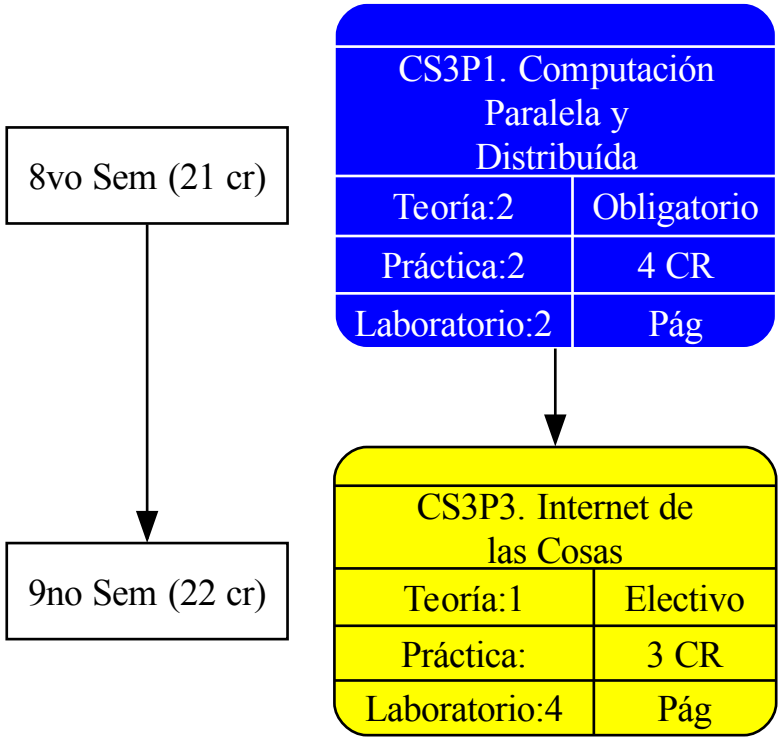

Figura 5.65: Mapa de Conexión. CS3P3 Internet de las Cosas

2. Fundamentación

El Internet de las Cosas (IoT) ha transformado cómo interactuamos con el mundo físico al conectar miles de millones de dispositivos. Este curso explora los fundamentos arquitectónicos del IoT, haciendo énfasis en la movilidad de dispositivos, la comunicación inalámbrica y las implicaciones sociales de la percepción generalizada. Los estudiantes analizarán los desafíos técnicos de los sistemas restringidos y las responsabilidades éticas asociadas con la recolección de datos a gran escala en entornos interconectados.

3. Objetivos del curso.

- Comprender la arquitectura de los sistemas IoT y el rol de la movilidad en el diseño de redes.
- Evaluar los impactos sociales y éticos de las tecnologías IoT.

- Implementar protocolos de comunicación para dispositivos restringidos y móviles.
- Analizar los riesgos de seguridad en el ciclo de vida de un ecosistema IoT.

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Usar)

5. Contenido

5.1 Soporte de Movilidad, 12 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Principios de comunicación celular (por ejemplo, 4G, 5G) **II:** Principios de LANs inalámbricas (principalmente 802.11) **III:** Comunicación dispositivo a dispositivo (por ejemplo, comunicación IoT) **IV:** Redes inalámbricas multiescalón (por ejemplo, redes ad hoc, oportunistas, tolerantes a retrasos)

Objetivos de Aprendizaje I: Describir algunos aspectos de comunicación celular como el registro [Describir] **II:** Describir cómo 802.11 soporta usuarios móviles [Describir] **III:** Describir usos prácticos de comunicación dispositivo a dispositivo, así como multiescalón [Describir] **IV:** Describir un tipo de red móvil como ad hoc [Describir]

Bibliografía: [Hanes et al., 2017, Buyya and Dastjerdi, 2016]

5.2 Ética, Ley y Sociedad en Seguridad, 12 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Impactos sociales de los dispositivos del Internet de las Cosas (IoT) y otras tecnologías emergentes en la seguridad y la privacidad Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad. **II:** Privacidad por diseño (privacy by design) e ingeniería de la privacidad Libertades Civiles y Expresión, Datos, Vigilancia y Tecnología de Privacidad, Fundamentos Filosóficos y Legislación de Privacidad, Delitos y Ataques Informáticos, Leyes, Políticas y Responsabilidad en Seguridad.

Objetivos de Aprendizaje I: Analizar los impactos sociales de la vigilancia generalizada por IoT [Evaluar]. **II:** Aplicar principios de privacidad por diseño a arquitecturas de datos de IoT [Usar].

Bibliografía: [Hanes et al., 2017]

5.3 Aplicaciones en Red, 24 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Paradigmas de aplicaciones distribuidas (por ejemplo, cliente/servidor, peer-to-peer, nube, edge y fog). Ver también: Comunicación, Coordinación **II:** Protocolos para redes restringidas (CoAP, MQTT, LoRaWAN).

Objetivos de Aprendizaje I: Definir los principios de denominación, direccionamiento y localización de recursos [Definir] **II:** Analizar las necesidades de demandas específicas de aplicaciones en red [Analizar] **III:** Describir los detalles de un protocolo de capa de aplicación [Describir] **IV:** Implementar una aplicación simple cliente-servidor basada en sockets [Implementar]

Bibliografía: [Hanes et al., 2017]

6. Bibliografía

[Buyya and Dastjerdi, 2016] Buyya, R. and Dastjerdi, A. V. (2016). *Internet of Things: Principles and Paradigms*. Morgan Kaufmann.

[Hanes et al., 2017] Hanes, D., Salgueiro, G., and Grossetete, P. (2017). *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*. Cisco Press.

Décimo Semestre

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS3P2 Cloud Computing (Obligatorio)

1. Información general

- **Semestre**: 10mo Sem. **Créditos**: 3
- **Horas del curso**: **Teoría**: 1 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - CS370 Big Data (9no Sem-Pág 503)

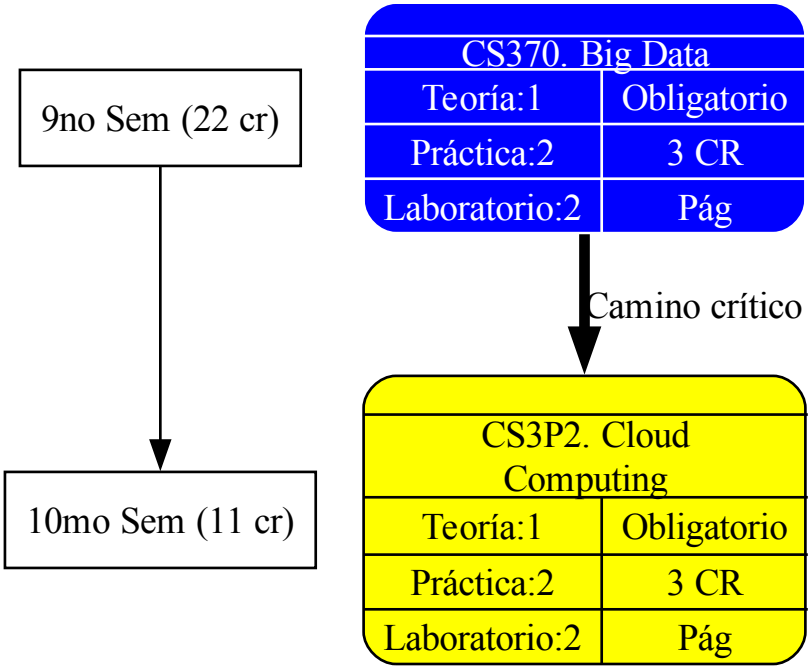

Figura 5.66: Mapa de Conexión. CS3P2 Cloud Computing

2. Fundamentación

La computación moderna depende en gran medida de arquitecturas distribuidas y plataformas nativas de la nube. Este curso proporciona una exploración en profundidad de los servicios de computación en la nube, conceptos de bases de datos distribuidas y plataformas web. Los estudiantes aprenderán a diseñar arquitecturas escalables, comprender las implicaciones de la consistencia de datos distribuidos y evaluar los desafíos de privacidad y seguridad inherentes a los entornos basados en la nube.

3. Objetivos del curso.

- Comprender la arquitectura y los modelos de despliegue de la computación en la nube.
- Diseñar soluciones de bases de datos distribuidas considerando consistencia y disponibilidad.
- Desarrollar aplicaciones utilizando plataformas y frameworks web modernos basados en la nube.
- Analizar las implicaciones de seguridad y privacidad en los servicios distribuidos y en la nube.

4. Competencias Generales (*Outcomes*)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Usar)

AG-C11) Uso de Herramientas: Aplica herramientas modernas de computación en la resolución de problemas. (Usar)

5. Contenido

5.1 Arquitectura Cloud y Gestión de Datos, 12 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Plataformas, frameworks o meta-frameworks web *a*) Servicios en la nube *b*) API, Componentes Web (Web Components) **II:** Servicios de computación Sistemas NoSQL *a*) Alojamiento en la Nube (Cloud Hosting) *b*) Escalabilidad (por ejemplo, Autoescalado (*Autoscaling*), Clústeres) *c*) Estimación de costos para servicios **III:** Arquitectura *a*) Monolitos vs Microservicios *b*) Micro-frontends *c*) Arquitecturas Dirigidas por Eventos (Event-Driven) vs RESTful: ventajas y desventajas *d*) Sin servidor (Serverless), computación en la nube bajo demanda

Objetivos de Aprendizaje **I:** Diseñar e implementar una aplicación basada en web utilizando un diseño de arquitectura de microservicios [Diseñar] **II:** Describir las restricciones, como alojamiento, servicios y escalabilidad, relacionadas con las plataformas web [Describir]

Bibliografía: [Marinescu, 2022, Buyya et al., 2013]

5.2 Bases de Datos Distribuidas/Computación en la Nube, 10 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Bases de datos distribuidas/sistemas basados en la nube

Objetivos de Aprendizaje **I:** Describir los componentes clave de un DBMS distribuido, incluyendo almacenamiento de datos distribuido, procesamiento de consultas y gestión de transacciones [Describir] **II:** Analizar las ventajas y desventajas entre arquitecturas de DBMS paralelo: memoria compartida, disco compartido y nada compartido [Analizar] **III:** Describir estrategias de replicación de datos y modelos de consistencia débil en sistemas de bases de datos distribuidas [Describir]

Bibliografía: [Buyya et al., 2013]

5.3 Computación Paralela y Distribuida, 8 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: **I:** Seguridad (safety) y vivacidad (liveness): *a*) Condiciones de carrera (race conditions) *b*) Dependencias/precondiciones *c*) Modelos de fallos (fault models) *d*) Terminación **II:** Modelos de programación: *a*) Modelo de actores (*actor models*) *b*) Modelos procedurales y reactivos *c*) Modelos de programación síncrona/asíncrona *d*) Paralelismo de datos (*data parallelism*) **III:** Propiedades: *a*) Propiedades basadas en orden: 1) Conmutatividad 2) Independencia *b*) Propiedades basadas en consistencia: 1) Atomicidad 2) Consenso **IV:** Control de ejecución: *a*) Espera asíncrona *Async await b*) Promesas (*promises*) *c*) Hilos (*threads*) **V:** Comunicación y coordinación: *a*) Mutexes *b*) Paso de mensajes (*message-passing*) *c*) Memoria compartida (*shared memory*) *d*) Cobegin-coend *e*) Monitores *f*) Canales (*channels*) *g*) Hilos (*threads*) *h*) Guardas (*guards*) **VI:** Futuros (futures) **VII:** Soporte del lenguaje para paralelismo de datos como forall, desenrollado de bucles (loop unrolling), map/reduce **VIII:** Efecto de los modelos de consistencia de memoria en la semántica del lenguaje y la generación correcta de código **IX:** Interfaces de Programación de Aplicaciones de Transferencia de Estado Representacional (REST APIs) **X:** Tecnologías y enfoques: computación en la nube (*cloud computing*), computación

de alto rendimiento, computación cuántica, computación ubicua **XI:** Sobrecarga del paso de mensajes **XII:** Granularidad del programa para una explotación eficiente de la concurrencia **XIII:** Concurrencia y otros paradigmas de programación (por ejemplo, funcional)

Objetivos de Aprendizaje I: Explicar por qué los lenguajes de programación no garantizan la consistencia secuencial en presencia de condiciones de carrera (data races) y qué deben hacer los programadores como resultado [Explicar] **II:** Implementar programas concurrentes correctos usando múltiples modelos de programación, como memoria compartida, actores, futuros, construcciones de sincronización y primitivas de paralelismo de datos [Implementar] **III:** Usar un modelo de paso de mensajes para analizar un protocolo de comunicación [Usar] **IV:** Usar construcciones de sincronización como métodos monitorizados/sincronizados en un programa simple [Usar] **V:** Modelar dependencia de datos usando construcciones de programación simples que involucran variables, lectura y escritura [Implementar] **VI:** Modelar dependencia de control usando construcciones simples como selección e iteración [Implementar] **VII:** Explicar cómo las APIs REST integran aplicaciones y automatizan procesos [Explicar] **VIII:** Explicar beneficios, limitaciones y desafíos relacionados con la computación distribuida y paralela [Explicar]

Bibliografía: [Marinescu, 2022]

5.4 Aplicaciones en Red, 6 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Paradigmas de aplicaciones distribuidas (por ejemplo, cliente/servidor, peer-to-peer, nube, edge y fog). Ver también: Comunicación, Coordinación

Objetivos de Aprendizaje I: Definir los principios de denominación, direccionamiento y localización de recursos [Definir] **II:** Analizar las necesidades de demandas específicas de aplicaciones en red [Analizar] **III:** Describir los detalles de un protocolo de capa de aplicación [Describir] **IV:** Implementar una aplicación simple cliente-servidor basada en sockets [Implementar]

Bibliografía: [Marinescu, 2022]

5.5 Algoritmos: Dominios de Aplicación, 4 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Panorama de dominios de aplicación comunes en multinúcleo, reactivo, datos paralelos, clúster, nube, sistemas distribuidos abiertos y marcos de trabajo (con referencia a la siguiente tabla). *a*) Multinúcleo: Agentes de ejecución típicos: Hilos. Mecanismos de comunicación típicos: Memoria compartida, Atómicos, bloqueos. Dominios algorítmicos típicos: Gestión de recursos, procesamiento de datos. Objetivos de ingeniería típicos: Rendimiento (*throughput*), latencia, energía. *b*) Reactivo: Agentes de ejecución típicos: Manejadores, hilos. Mecanismos de comunicación típicos: Canales de E/S. Dominios algorítmicos típicos: Servicios, tiempo real. Objetivos de ingeniería típicos: Latencia. *c*) Datos paralelos: Agentes de ejecución típicos: GPU, SIMD, aceleradores, híbridos. Mecanismos de comunicación típicos: Memoria heterogénea. Dominios algorítmicos típicos: álgebra lineal, gráficos, análisis de datos. Objetivos de ingeniería típicos: Rendimiento (*throughput*), energía. *d*) Clúster: Agentes de ejecución típicos: Hosts gestionados. Mecanismos de comunicación típicos: Sockets, canales. Dominios algorítmicos típicos: Simulación, análisis de datos. Objetivos de ingeniería típicos: Rendimiento (*throughput*). *e*) Nube: Agentes de ejecución típicos: Hosts aprovisionados. Mecanismos de comunicación típicos: APIs de servicio. Dominios algorítmicos típicos: Aplicaciones web. Objetivos de ingeniería típicos: Escalabilidad. *f*) Distribuido abierto: Agentes de ejecución típicos: Hosts autónomos. Mecanismos de comunicación típicos: Sockets, Almacenes de datos. Dominios algorítmicos típicos: Almacenes de datos y servicios tolerantes a fallos. Objetivos de ingeniería típicos: Confiabilidad.

Objetivos de Aprendizaje I: Extender un programa secuencial dirigido por eventos estableciendo una nueva actividad en un manejador de eventos (por ejemplo, un nuevo hilo en un manejador de acciones de GUI) [Diseñar] **II:** Mejorar el rendimiento de un componente secuencial introduciendo

paralelismo y/o distribución [Crear] **III:** Elegir entre diferentes diseños paralelos/distribuidos para componentes de un sistema dado [Evaluar (valorar)]

Bibliografía: [Marinescu, 2022]

5.6 Principios y Mentalidad de Seguridad, 8 hr(s)

Competencias: AG-C09,AG-C11

Tópicos: I: Aplicaciones de una mentalidad de seguridad: web, nube y dispositivos móviles Diseño del Sistema, Aspectos Comunes: Plataformas, APIs y Restricciones, Aspectos Comunes: Lenguajes de Plataforma y Patrones Web. **II:** Implicaciones de privacidad de la recopilación generalizada de datos, incluyendo pero no limitado a bases de datos transaccionales, almacenes de datos, sistemas de vigilancia, computación en la nube e inteligencia artificial.

Objetivos de Aprendizaje I: Analizar los riesgos de seguridad asociados con aplicaciones en la nube distribuidas [Usar]. **II:** Evaluar las implicaciones de privacidad del almacenamiento de datos en entornos de nube multiinquilino [Evaluar].

Bibliografía: [Marinescu, 2022]

6. Bibliografía

[Buyya et al., 2013] Buyya, R., Vecchiola, C., and Selvi, S. T. (2013). *Mastering Cloud Computing: Foundations and Applications Programming*. Morgan Kaufmann, 1st edition.

[Marinescu, 2022] Marinescu, D. C. (2022). *Cloud Computing: Theory and Practice*. Morgan Kaufmann, 3rd edition.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

CS404 Taller de Investigación (Obligatorio)

1. Información general

- **Semestre**: 10mo Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Laboratorio**: 4 horas;
- **Prerrequisitos**:
  - CS403 Proyecto de tesis 2 (9no Sem-Pág 509)

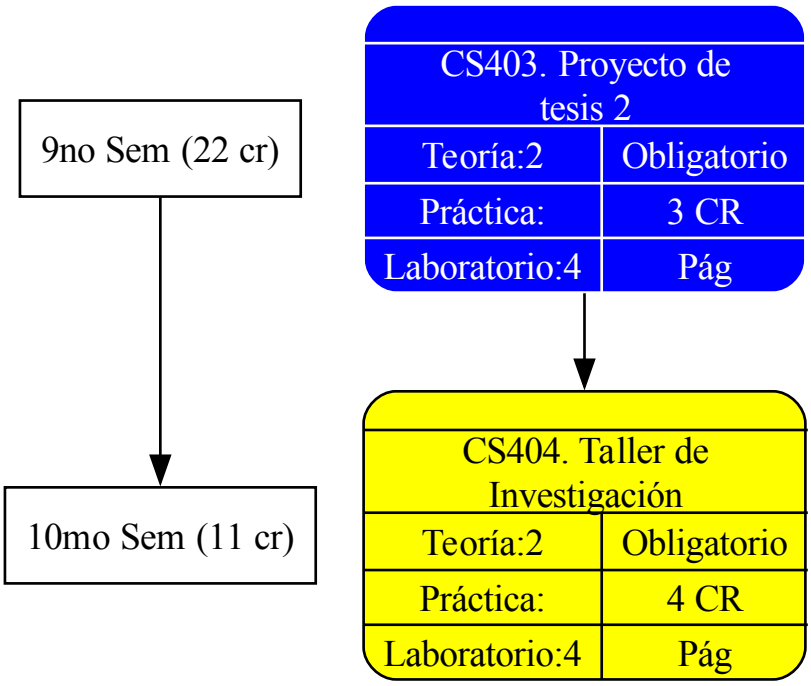

Figura 5.67: Mapa de Conexión. CS404 Taller de Investigación

2. Fundamentación

Este curso tiene por objetivo que el alumno concluya su tesis con calidad suficiente para la sustentación pública, completando experimentos, capítulos finales, revisiones con el asesor y la entrega formal.

3. Objetivos del curso.

- Concluir la parte experimental de la tesis con rigor estadístico y análisis completo de resultados.
- Redactar los capítulos finales de la tesis: conclusiones, contribuciones y trabajo futuro.
- Asegurar que el documento cumpla con el formato institucional y tenga coherencia narrativa completa.
- Incorporar las revisiones del asesor hasta obtener un documento aprobado para sustentación.
- Preparar y realizar la presentación formal de la tesis ante el comité de la facultad.
- Los entregables de este curso son:

Parcial: Borrador avanzado con experimentos concluidos, capítulos de introducción, marco teórico, propuesta y resultados en LaTeX.

Final: Documento de tesis completo, aprobado por el asesor y listo para sustentación pública.

4. Competencias Generales (*Outcomes*)

AG-C02) Ética: Aplica principios éticos y se compromete con la ética profesional y las normas de la práctica profesional de la computación. (Evaluar)

AG-C04) Comunicación: Se comunica de forma efectiva en actividades complejas de computación. (Evaluar)

AG-C05) Gestión de Proyectos: Aplica los principios de gestión en computación para gestionar proyectos. (Evaluar)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Evaluar)

AG-C09) Diseño y Desarrollo de Soluciones: Diseña, implementa y evalúa soluciones para problemas complejos de computación. (Evaluar)

5. Contenido

5.1 Conclusión de Experimentos y Análisis de Resultados, 12 hr(s)

Competencias: AG-C05,AG-C08,AG-C09

Tópicos: **I:** Ejecución completa del protocolo experimental definido en el plan de tesis. **II:** Análisis estadístico final: pruebas de significancia, intervalos de confianza y efecto. **III:** Comparación objetiva con baselines y estado del arte usando métricas establecidas. **IV:** Interpretación de resultados: hallazgos, limitaciones y amenazas a la validez. **V:** Documentación reproducible: código, datasets, configuraciones y resultados.

Objetivos de Aprendizaje **I:** Completar la ejecución del protocolo experimental con todos los casos definidos. [Evaluar] **II:** Analizar resultados con rigor estadístico e interpretar hallazgos con respecto a la hipótesis. [Evaluar] **III:** Documentar de forma reproducible los experimentos, configuraciones y resultados. [Usar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.2 Redacción de Capítulos Finales de la Tesis, 12 hr(s)

Competencias: AG-C02,AG-C04,AG-C09

Tópicos: **I:** Redacción del capítulo de resultados: tablas, figuras y análisis comparativo de alta calidad. **II:** Redacción del capítulo de conclusiones: hallazgos, contribuciones originales y limitaciones. **III:** Redacción del capítulo de trabajo futuro: extensiones y problemas abiertos. **IV:** Redacción de la introducción final: motivación, problema y resumen de contribuciones. **V:** Redacción del resumen ejecutivo (abstract) en español e inglés.

Objetivos de Aprendizaje **I:** Redactar el capítulo de resultados con visualizaciones claras y análisis comparativo. [Evaluar] **II:** Formular las conclusiones y contribuciones originales de la tesis con precisión. [Evaluar] **III:** Escribir el abstract y la introducción final articulando el problema y los aportes. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.3 Formato, Integración y Calidad del Documento, 10 hr(s)

Competencias: AG-C02,AG-C04,AG-C09

Tópicos: I: Verificación del cumplimiento del formato institucional de tesis en LaTeX. **II:** Revisión de calidad de figuras, tablas y ecuaciones en el documento final. **III:** Consistencia narrativa y terminológica entre todos los capítulos del documento. **IV:** Revisión y actualización de la bibliografía: cobertura, formato BibTeX y citas correctas. **V:** Generación del documento final: PDF, portada, índices y apéndices.

Objetivos de Aprendizaje I: Verificar que el documento cumple íntegramente con el formato institucional de tesis. [Evaluar] **II:** Corregir la consistencia narrativa y terminológica de todos los capítulos. [Evaluar] **III:** Producir el documento PDF final con portada, índices y bibliografía completos. [Usar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.4 Revisiones con el Asesor y Ciclos de Corrección, 10 hr(s)

Competencias: AG-C02,AG-C08,AG-C09

Tópicos: I: Dinámica de revisión estructurada: entrega, revisión y respuesta al asesor. **II:** Incorporación de observaciones del asesor: priorización y respuesta argumentada. **III:** Revisión de la solidez de la propuesta y la validez de los experimentos según el asesor. **IV:** Manejo de desacuerdos académicos con el asesor de forma constructiva y ética. **V:** Criterios de aprobación del asesor para la autorización de sustentación.

Objetivos de Aprendizaje I: Incorporar sistemáticamente las observaciones del asesor con respuestas argumentadas. [Evaluar] **II:** Mejorar la solidez científica del documento a través de ciclos de revisión. [Evaluar] **III:** Obtener la aprobación formal del asesor para la sustentación del documento. [Evaluar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.5 Preparación y Ensayo de la Sustentación, 8 hr(s)

Competencias: AG-C02,AG-C04,AG-C05

Tópicos: I: Diseño de la presentación oral: estructura, narrativa y manejo del tiempo. **II:** Preparación de diapositivas técnicas: figuras, resultados y comparaciones clave. **III:** Técnicas de defensa ante preguntas del jurado: argumentación y manejo de objeciones. **IV:** Ensayo de la sustentación con el asesor y/o pares académicos. **V:** Gestión del estrés y comunicación efectiva en contextos de evaluación académica.

Objetivos de Aprendizaje I: Diseñar y exponer una presentación oral clara y estructurada del trabajo de tesis. [Evaluar] **II:** Defender la metodología y resultados ante preguntas críticas del jurado. [Evaluar] **III:** Demostrar dominio completo del tema mediante ensayos de sustentación previos. [Usar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

5.6 Entrega Formal y Pre-Sustentación, 8 hr(s)

Competencias: AG-C02,AG-C04,AG-C05

Tópicos: I: Proceso institucional de entrega de la tesis: documentación y plazos reglamentarios. **II:** Pre-sustentación ante autoridades de la facultad: evaluación del borrador final. **III:** Incorporación de correcciones del comité pre-sustentador en el documento. **IV:** Trámites administrativos: designación de jurado y programación de la sustentación. **V:** Ética en la diseminación: acceso abierto, repositorio institucional y derechos de autoría.

Objetivos de Aprendizaje I: Entregar el documento de tesis completo cumpliendo todos los requisitos institucionales. [Evaluar] **II:** Presentar el borrador final ante el comité pre-sustentador e incorporar sus correcciones. [Evaluar] **III:** Completar los trámites administrativos para la programación de la sustentación pública. [Usar]

Bibliografía: [IEEE-Computer Society, 2008, ACM, 2008]

6. Bibliografía

[ACM, 2008] ACM (2008). *Digital Library*. Association for Computing Machinery. http://portal.acm.org/dl.cfm.

[IEEE-Computer Society, 2008] IEEE-Computer Society (2008). *Digital Library*. IEEE-Computer Society. http://www.computer.org/publications/dlib.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI367 Tópicos en Inteligencia Artificial (Obligatorio)

1. Información general

- **Semestre**: 10mo Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - AI264 Aprendizaje Profundo (7mo Sem-Pág 435)

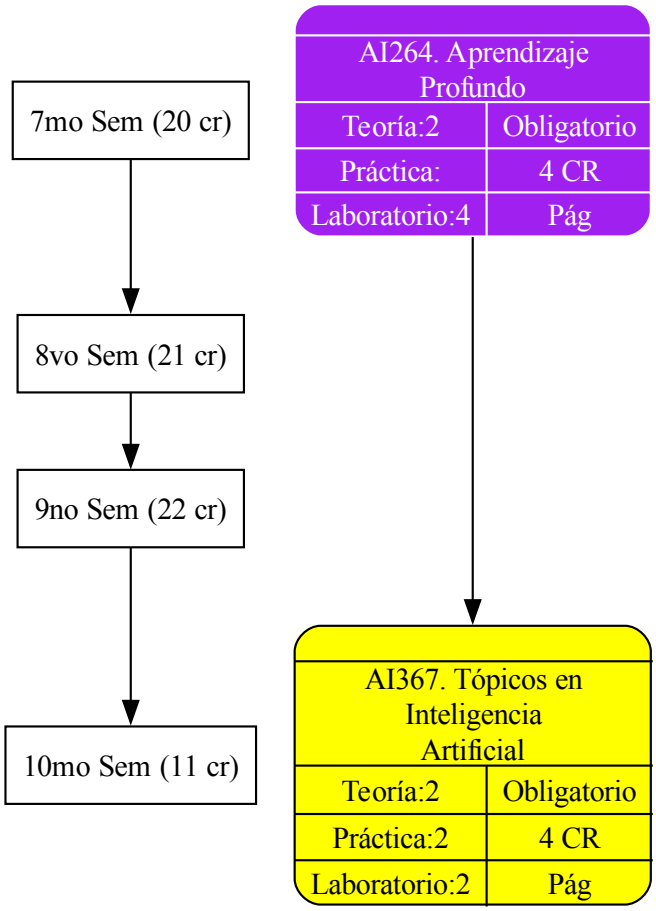

Figura 5.68: Mapa de Conexión. AI367 Tópicos en Inteligencia Artificial

2. Fundamentación

La Inteligencia Artificial es un campo en rápida evolución. Este curso explora temas avanzados y contemporáneos que están dando forma actualmente a la industria y la investigación. Se pone énfasis en la IA Generativa, Modelos de Lenguaje Grandes (LLMs) y Aprendizaje por Refuerzo Avanzado. Los estudiantes analizarán las arquitecturas detrás de los sistemas modernos de IA, como los Transformers, y discutirán los desafíos técnicos y sociales de desplegar agentes inteligentes autónomos a gran escala.

3. Objetivos del curso.

- Comprender la arquitectura y el proceso de entrenamiento de Modelos de Lenguaje Grandes.
- Dominar los principios de la IA Generativa para síntesis de texto e imágenes.
- Implementar algoritmos avanzados de Aprendizaje por Refuerzo para la toma de decisiones.
- Aplicar técnicas de ingeniería de prompts y ajuste fino a modelos existentes.
- Evaluar críticamente los últimos artículos de investigación y tendencias en IA.

4. Competencias Generales (*Outcomes*)

AG-C06) Aprendizaje a lo largo de la vida: Reconoce la importancia del aprendizaje continuo y la adaptación a nuevas tecnologías. (Usar)

5. Contenido

5.1 Fundamentos de Aprendizaje Automático, 6 hr(s)

Competencias: AG-C06

Tópicos: **I:** Definición y ejemplos de una amplia variedad de tareas de aprendizaje automático: *a*) Aprendizaje supervisado: 1) Clasificación 2) Regresión *b*) Aprendizaje por refuerzo *c*) Aprendizaje no supervisado: 1) Agrupamiento **II:** Ideas fundamentales: *a*) Teorema de no hay almuerzo gratis: ningún aprendiz puede resolver todos los problemas; las decisiones de diseño de representación tienen consecuencias. *b*) Fuentes de error e indecidibilidad en el aprendizaje automático **III:** Un aprendizaje supervisado simple basado en estadísticas como regresión lineal o árboles de decisión: *a*) Enfocarse en cómo funcionan sin entrar en detalles matemáticos o de optimización; suficiente para entender y usar implementaciones existentes correctamente **IV:** El problema del sobreajuste/control de la complejidad de la solución (regularización, poda - solo intuición): *a*) El compromiso entre sesgo (subajuste) - varianza (sobreajuste)

Objetivos de Aprendizaje **I:** Describir las diferencias entre los tres estilos principales de aprendizaje (supervisado, por refuerzo y no supervisado) y determinar cuál es apropiado para un dominio de problema particular [Explicar] **II:** Diferenciar los términos: IA, aprendizaje automático y aprendizaje profundo [Evaluar] **III:** Formular una aplicación como un problema de clasificación, incluyendo las características de entrada disponibles y la salida a predecir (por ejemplo, identificar caracteres alfabéticos a partir de entrada de cuadrícula de píxeles) [Aplicar] **IV:** Identificar el sobreajuste en el contexto de un problema y curvas de aprendizaje y describir soluciones al sobreajuste [Evaluar] **V:** Explicar cómo funciona el aprendizaje automático como un proceso de optimización/búsqueda [Explicar]

Bibliografía: [Goodfellow et al., 2016, Foster, 2022]

5.2 Aprendizaje Automático: Datos, Representaciones y Evaluación, 5 hr(s)

Competencias: AG-C06

Tópicos: **I:** Trabajar con Datos: *a*) Preprocesamiento de datos: 1) Importancia y peligros de las elecciones de preprocesamiento *b*) Manejo de valores faltantes (imputación, marcar como faltante): 1) Implicaciones de imputar versus marcar como faltante *c*) Codificación de variables categóricas, codificación de datos de valor real *d*) Normalización/estandarización *e*) énfasis en datos reales, no ejemplos de libros de texto **II:** Representaciones: *a*) Espacios de hipótesis y complejidad *b*) Expansión de características de base simple, como elevar al cuadrado características univariadas *c*) Representaciones de características aprendidas **III:** Evaluación del aprendizaje automático: *a*) Separación de conjuntos de entrenamiento, validación y prueba *b*) Métricas de rendimiento para clasificadores *c*) Estimación del rendimiento en prueba en datos retenidos *d*) Ajuste de los parámetros de un modelo de aprendizaje automático con un conjunto de validación *e*) Importancia de entender qué está haciendo un modelo, dónde están sus peligros/limitaciones y las implicaciones de sus decisiones

Objetivos de Aprendizaje I: Aplicar dos o más algoritmos de aprendizaje estadístico simple a una tarea de clasificación y medir la precisión de los clasificadores [Aplicar] **II:** Implementar un algoritmo de aprendizaje estadístico y el proceso de optimización correspondiente para entrenar el clasificador y obtener una predicción en nuevos datos [Aplicar] **III:** Explicar los procedimientos adecuados de evaluación de AA, incluyendo las diferencias entre rendimiento en entrenamiento y prueba, y qué puede salir mal con el proceso de evaluación llevando a informes inexactos del rendimiento de AA [Explicar] **IV:** Comparar dos algoritmos de aprendizaje automático en un conjunto de datos, implementando la metodología de preprocesamiento de datos y evaluación (por ejemplo, métricas y manejo de divisiones entrenamiento/prueba) desde cero [Aplicar]

Bibliografía: [Goodfellow et al., 2016, Foster, 2022]

5.3 Redes Neuronales y Ética, 5 hr(s)

Competencias: AG-C06

Tópicos: I: Redes neuronales básicas: *a*) Fundamentos de comprensión de cómo funcionan las redes neuronales y su proceso de entrenamiento, sin detalles de los cálculos *b*) Introducción básica a redes neuronales generativas (por ejemplo, modelos de lenguaje grandes) **II:** ética para el Aprendizaje Automático: *a*) Enfocarse en datos reales, escenarios reales y estudios de caso *b*) Sesgo de conjunto de datos/algorítmico/de evaluación y consecuencias no deseadas

Objetivos de Aprendizaje I: Describir el proceso de entrenamiento de redes neuronales y las representaciones aprendidas resultantes [Explicar]

Bibliografía: [Goodfellow et al., 2016, Foster, 2022]

5.4 Formulación del Aprendizaje Automático y Evaluación, 3 hr(s)

Competencias:

Tópicos: I: Formulación del aprendizaje automático simple como un problema de optimización, como regresión lineal por mínimos cuadrados o regresión logística: *a*) Función objetivo *b*) Descenso de gradiente *c*) Regularización para evitar sobreajuste (formulación matemática) **II:** Conjuntos de modelos: *a*) Combinación de mayoría ponderada simple **III:** Evaluación de rendimiento: *a*) Otras métricas para clasificación (por ejemplo, error, precisión, exhaustividad) *b*) Métricas de rendimiento para regresores *c*) Matriz de confusión *d*) Validación cruzada: 1) Ajuste de parámetros (búsqueda en cuadrícula/aleatoria, mediante validación cruzada)

Objetivos de Aprendizaje I: Comparar y contrastar varias técnicas de aprendizaje (por ejemplo, árboles de decisión, regresión logística, Bayes ingenuo, redes neuronales y redes de creencia), proporcionando ejemplos de cuándo cada estrategia es superior [Comparar] **II:** Evaluar el rendimiento de un sistema de aprendizaje simple en un conjunto de datos del mundo real [Evaluar] **III:** Caracterizar el estado del arte en teoría del aprendizaje, incluyendo sus logros y limitaciones [Explicar] **IV:** Explicar el problema del sobreajuste, junto con técnicas para detectar y manejar el problema [Explicar] **V:** Explicar el triple compromiso entre el tamaño de un espacio de hipótesis, el tamaño del conjunto de entrenamiento y la precisión del rendimiento [Explicar]

Bibliografía: [Goodfellow et al., 2016]

5.5 Aprendizaje Profundo, RL y Ética, 3 hr(s)

Competencias:

Tópicos: I: Aprendizaje profundo: *a*) Redes feed-forward profundas (solo intuición, sin matemáticas) *b*) Redes neuronales convolucionales (solo intuición, sin matemáticas) *c*) Visualización de representaciones de características aprendidas de redes profundas *d*) Otras arquitecturas (NN generativas, NN recurrentes, transformers, etc.) **II:** Visión general de los métodos de aprendizaje por refuerzo **III:** Dos o más aplicaciones de algoritmos de aprendizaje automático: *a*) Por

ejemplo, medicina y salud, economía, visión, lenguaje natural, robótica, juego **IV:** ética para el Aprendizaje Automático: *a*) Continuar enfocándose en datos reales, escenarios reales y estudios de caso *b*) Privacidad *c*) Equidad *d*) Propiedad intelectual *e*) Explicabilidad

Objetivos de Aprendizaje I: Visualizar el progreso del entrenamiento de una red neuronal a través de curvas de aprendizaje en un kit de herramientas establecido (por ejemplo, TensorBoard) y visualizar las características aprendidas de la red [Aplicar] **II:** Dada una aplicación real de aprendizaje automático, describir problemas éticos respecto a las elecciones de datos, pasos de preprocesamiento, selección de algoritmo y visualización/presentación de resultados [Aplicar]

Bibliografía: [Goodfellow et al., 2016]

5.6 Procesamiento del Lenguaje Natural, 20 hr(s)

Competencias: AG-C06

Tópicos: I: Gramáticas deterministas y estocásticas **II:** Algoritmos de análisis sintáctico: *a*) CFGs y analizadores de tabla (por ejemplo, CYK) *b*) CFGs probabilísticos y CYK ponderado **III:** Representación del significado/Semántica: *a*) Representaciones del conocimiento basadas en lógica *b*) Roles semánticos *c*) Representaciones temporales *d*) Creencias, deseos e intenciones **IV:** Métodos basados en corpus **V:** N-gramas y HMMs **VI:** Suavizado y retroceso **VII:** Ejemplos de uso: Etiquetado POS y morfología **VIII:** Recuperación de información: *a*) Modelo de espacio vectorial: 1) TF & IDF *b*) Precisión y exhaustividad **IX:** Extracción de información **X:** Traducción de lenguaje **XI:** Clasificación de texto, categorización: *a*) Modelo de bolsa de palabras **XII:** Aprendizaje profundo para PLN: *a*) RNNs *b*) Transformers *c*) Incrustaciones multimodales (por ejemplo, imágenes + texto) *d*) Modelos de lenguaje generativos

Objetivos de Aprendizaje I: Definir y contrastar gramáticas deterministas y estocásticas, proporcionando ejemplos para mostrar la adecuación de cada una [Definir] **II:** Simular, aplicar o implementar algoritmos clásicos y estocásticos para analizar lenguaje natural [Aplicar] **III:** Identificar los desafíos de representar el significado [Analizar] **IV:** Listar las ventajas de usar corpus estándar. Identificar ejemplos de corpus actuales para una variedad de tareas de PLN [Listar/Enumerar] **V:** Identificar técnicas para recuperación de información, traducción de lenguaje y clasificación de texto [Analizar] **VI:** Implementar una transformación TF/IDF, usarla para extraer características de un corpus, y entrenar un algoritmo de aprendizaje automático estándar usando esas características para hacer clasificación de texto [Aplicar]

Bibliografía: [Tunstall et al., 2022, Russell and Norvig, 2020]

5.7 Fundamentos de Aprendizaje Automático, 6 hr(s)

Competencias: AG-C06

Tópicos: I: Definición y ejemplos de una amplia variedad de tareas de aprendizaje automático: *a*) Aprendizaje supervisado: 1) Clasificación 2) Regresión *b*) Aprendizaje por refuerzo *c*) Aprendizaje no supervisado: 1) Agrupamiento **II:** Ideas fundamentales: *a*) Teorema de no hay almuerzo gratis: ningún aprendiz puede resolver todos los problemas; las decisiones de diseño de representación tienen consecuencias. *b*) Fuentes de error e indecidibilidad en el aprendizaje automático **III:** Un aprendizaje supervisado simple basado en estadísticas como regresión lineal o árboles de decisión: *a*) Enfocarse en cómo funcionan sin entrar en detalles matemáticos o de optimización; suficiente para entender y usar implementaciones existentes correctamente **IV:** El problema del sobreajuste/control de la complejidad de la solución (regularización, poda - solo intuición): *a*) El compromiso entre sesgo (subajuste) - varianza (sobreajuste)

Objetivos de Aprendizaje I: Describir las diferencias entre los tres estilos principales de aprendizaje (supervisado, por refuerzo y no supervisado) y determinar cuál es apropiado para un dominio de problema particular [Explicar] **II:** Diferenciar los términos: IA, aprendizaje automático y aprendizaje profundo [Evaluar] **III:** Formular una aplicación como un problema de clasificación,

incluyendo las características de entrada disponibles y la salida a predecir (por ejemplo, identificar caracteres alfabéticos a partir de entrada de cuadrícula de píxeles) [Aplicar] **IV:** Identificar el sobreajuste en el contexto de un problema y curvas de aprendizaje y describir soluciones al sobreajuste [Evaluar] **V:** Explicar cómo funciona el aprendizaje automático como un proceso de optimización/búsqueda [Explicar]

Bibliografía: [Sutton and Barto, 2018]

5.8 Aprendizaje Automático: Datos, Representaciones y Evaluación, 5 hr(s)

Competencias: AG-C06

Tópicos: I: Trabajar con Datos: *a*) Preprocesamiento de datos: 1) Importancia y peligros de las elecciones de preprocesamiento *b*) Manejo de valores faltantes (imputación, marcar como faltante): 1) Implicaciones de imputar versus marcar como faltante *c*) Codificación de variables categóricas, codificación de datos de valor real *d*) Normalización/estandarización *e*) énfasis en datos reales, no ejemplos de libros de texto **II:** Representaciones: *a*) Espacios de hipótesis y complejidad *b*) Expansión de características de base simple, como elevar al cuadrado características univariadas *c*) Representaciones de características aprendidas **III:** Evaluación del aprendizaje automático: *a*) Separación de conjuntos de entrenamiento, validación y prueba *b*) Métricas de rendimiento para clasificadores *c*) Estimación del rendimiento en prueba en datos retenidos *d*) Ajuste de los parámetros de un modelo de aprendizaje automático con un conjunto de validación *e*) Importancia de entender qué está haciendo un modelo, dónde están sus peligros/limitaciones y las implicaciones de sus decisiones

Objetivos de Aprendizaje I: Aplicar dos o más algoritmos de aprendizaje estadístico simple a una tarea de clasificación y medir la precisión de los clasificadores [Aplicar] **II:** Implementar un algoritmo de aprendizaje estadístico y el proceso de optimización correspondiente para entrenar el clasificador y obtener una predicción en nuevos datos [Aplicar] **III:** Explicar los procedimientos adecuados de evaluación de AA, incluyendo las diferencias entre rendimiento en entrenamiento y prueba, y qué puede salir mal con el proceso de evaluación llevando a informes inexactos del rendimiento de AA [Explicar] **IV:** Comparar dos algoritmos de aprendizaje automático en un conjunto de datos, implementando la metodología de preprocesamiento de datos y evaluación (por ejemplo, métricas y manejo de divisiones entrenamiento/prueba) desde cero [Aplicar]

Bibliografía: [Sutton and Barto, 2018]

5.9 Redes Neuronales y Ética, 5 hr(s)

Competencias: AG-C06

Tópicos: I: Redes neuronales básicas: *a*) Fundamentos de comprensión de cómo funcionan las redes neuronales y su proceso de entrenamiento, sin detalles de los cálculos *b*) Introducción básica a redes neuronales generativas (por ejemplo, modelos de lenguaje grandes) **II:** ética para el Aprendizaje Automático: *a*) Enfocarse en datos reales, escenarios reales y estudios de caso *b*) Sesgo de conjunto de datos/algorítmico/de evaluación y consecuencias no deseadas

Objetivos de Aprendizaje I: Describir el proceso de entrenamiento de redes neuronales y las representaciones aprendidas resultantes [Explicar]

Bibliografía: [Sutton and Barto, 2018]

5.10 Aplicaciones e Impacto Social de la IA, 6 hr(s)

Competencias: AG-C06

Tópicos: I: Al menos una aplicación de IA a un problema y campo específico, como medicina, salud, sostenibilidad, redes sociales, economía, educación, robótica, etc. (elegir al menos uno para el Núcleo CS): *a*) Formular y evaluar una aplicación específica como un problema de IA: 1) Cómo

lidiar con problemas mal especificados o mal planteados *b*) Disponibilidad/escasez y limpieza de datos: 1) Limpieza y preprocesamiento básico de datos 2) Sesgo del conjunto de datos *c*) Sesgo algorítmico *d*) Sesgo de evaluación *e*) Evaluación de las implicaciones sociales de la aplicación **II:** Modelos generativos profundos desplegados: *a*) Visión general de alto nivel de modelos generativos de imágenes profundas (por ejemplo, a 2023, DALL-E, Midjourney, Stable Diffusion, etc.), sus usos y sus limitaciones/peligros. *b*) Visión general de alto nivel de modelos de lenguaje grandes (por ejemplo, a 2023, ChatGPT, Bard, etc.), sus usos y sus limitaciones/peligros. **III:** Visión general del impacto social de la IA: *a*) ética *b*) Equidad *c*) Confianza/explicabilidad *d*) Privacidad y uso de datos de entrenamiento *e*) Autonomía humana y supervisión/regulaciones/requisitos legales *f*) Sostenibilidad

Objetivos de Aprendizaje I: Dado un dominio de aplicación y problema del mundo real, formular una solución de IA para ello, identificando datos/entrada apropiados, preprocesamiento, representaciones, técnicas de IA y métricas/metodología de evaluación [Aplicar]

Bibliografía: [Russell and Norvig, 2020, Bostrom, 2014]

5.11 Aplicaciones Adicionales e Impacto Social de la IA, 6 hr(s)

Competencias: AG-C06

Tópicos: I: Una o más aplicaciones adicionales de IA a un amplio conjunto de problemas y diversos campos, como medicina, salud, sostenibilidad, redes sociales, economía, educación, robótica, etc. (elegir un área diferente de la elegida para el Núcleo CS): *a*) Formular y evaluar una aplicación específica como un problema de IA: 1) Cómo lidiar con problemas mal especificados o mal planteados *b*) Disponibilidad/escasez y limpieza de datos: 1) Limpieza y preprocesamiento básico de datos 2) Sesgo del conjunto de datos *c*) Sesgo algorítmico *d*) Sesgo de evaluación *e*) Evaluación de las implicaciones sociales de la aplicación **II:** Profundidad adicional sobre modelos generativos profundos desplegados: *a*) Introducción a cómo funcionan los modelos generativos de imágenes profundas (por ejemplo, a 2023, DALL-E, Midjourney, Stable Diffusion) incluyendo discusión de atención *b*) Introducción a cómo funcionan los modelos de lenguaje grandes (por ejemplo, a 2023, ChatGPT, Bard) incluyendo discusión de atención *c*) Idea de modelos fundamentales, cómo usarlos, y los beneficios/problemas con entrenarlos a partir de grandes datos **III:** Análisis y discusión del impacto social de la IA: *a*) ética *b*) Equidad *c*) Confianza/explicabilidad *d*) Privacidad y uso de datos de entrenamiento *e*) Autonomía humana y supervisión/regulaciones/requisitos legales *f*) Sostenibilidad

Objetivos de Aprendizaje I: Analizar el impacto social de una o más aplicaciones específicas de IA del mundo real, identificando problemas respecto a ética, equidad, sesgo, confianza y explicabilidad [Evaluar] **II:** Describir algunos de los modos de falla de los modelos generativos profundos actuales para lenguaje o imágenes, y cómo esto podría afectar su uso en una aplicación [Describir]

Bibliografía: [Russell and Norvig, 2020, Bostrom, 2014]

6. Bibliografía

[Bostrom, 2014] Bostrom, N. (2014). *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press.

[Foster, 2022] Foster, D. (2022). *Generative Deep Learning*. O'Reilly Media, 2nd edition.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Russell and Norvig, 2020] Russell, S. and Norvig, P. (2020). *Artificial Intelligence: A Modern Approach*. Pearson, 4th edition.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition.

[Tunstall et al., 2022] Tunstall, L., von Werra, L., and Wolf, T. (2022). *Natural Language Processing with Transformers*. O'Reilly Media.

Universidad Nacional de Ingeniería (UNI)
Facultad de Ciencias
Escuela Profesional de Ciencia de la Computación

AI368 Computación Evolutiva (Obligatorio)

1. Información general

- **Semestre**: 10mo Sem. **Créditos**: 4
- **Horas del curso**: **Teoría**: 2 horas; **Práctica**: 2 horas; **Laboratorio**: 2 horas;
- **Prerrequisitos**:
  - AI264 Aprendizaje Profundo (7mo Sem-Pág 435)

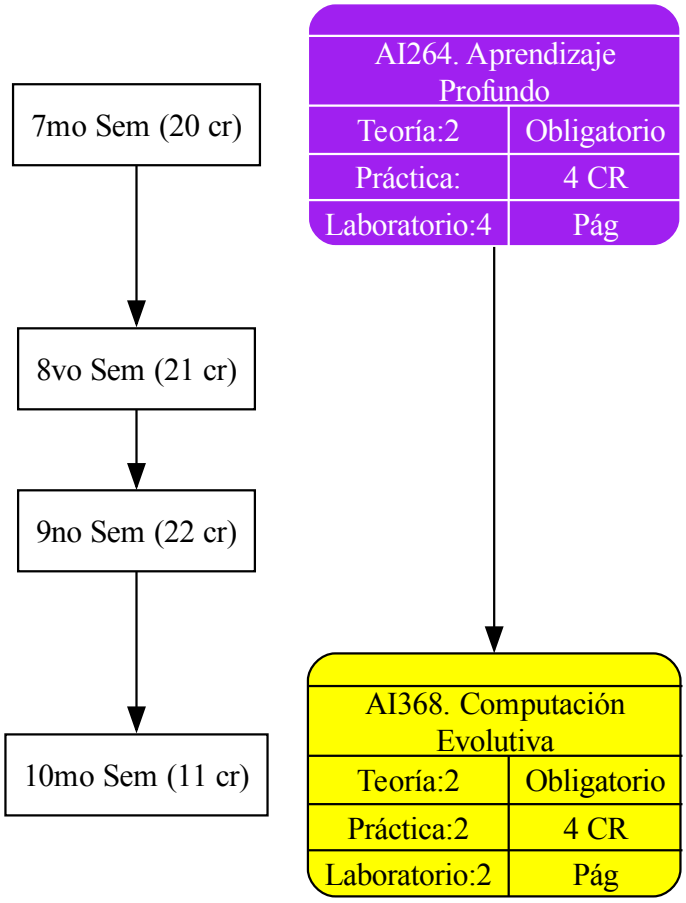

Figura 5.69: Mapa de Conexión. AI368 Computación Evolutiva

2. Fundamentación

La Computación Evolutiva es un subcampo de la Inteligencia Artificial inspirado en la selección natural y la evolución biológica. Este curso explora algoritmos metaheurísticos de optimización como Algoritmos Genéticos, Estrategias de Evolución y Programación Genética. Estas técnicas son esenciales para resolver problemas complejos de optimización y búsqueda donde los métodos analíticos tradicionales son insuficientes. Los estudiantes aprenderán a diseñar, implementar y analizar sistemas evolutivos aplicados a ingeniería, ciencia de datos y agentes autónomos.

3. Objetivos del curso.

- Entender los principios de la selección natural aplicados a la computación.
- Dominar el diseño de representaciones, operadores y funciones de aptitud (fitness).
- Implementar y ajustar Algoritmos Genéticos y Estrategias de Evolución.
- Analizar la convergencia y el rendimiento de algoritmos bioinspirados.
- Aplicar técnicas evolutivas a problemas de optimización multiobjetivo.

4. Competencias Generales (*Outcomes*)

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usar)

AG-C12) Aplica la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computadora. (Usar)

5. Contenido

5.1 Fundamentos de la Computación Evolutiva, 12 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Introducción a los Algoritmos Evolutivos (EAs) y metáforas biológicas. **II:** El ciclo general del Algoritmo Evolutivo: Inicialización, Selección, Variación, Reemplazo. **III:** Componentes: Poblaciones, cromosomas, genes, alelos. **IV:** Paisajes de aptitud (*fitness, landscapes*), mapeo genotipo-fenotipo y espacios de búsqueda.

Objetivos de Aprendizaje **I:** Explicar la relación entre la evolución biológica y los procesos de búsqueda artificial [Familiarizarse] **II:** Identificar los componentes de un algoritmo evolutivo genérico y sus roles [Usar] **III:** Describir el concepto de paisajes de aptitud y su impacto en el rendimiento de los EAs [Familiarizarse]

Bibliografía: [Eiben and Smith, 2015, Back, 1996]

5.2 Algoritmos Genéticos y sus Variantes, 16 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Algoritmos Genéticos Canónicos (CGA): representación binaria, esquemas de selección (ruleta, torneo). **II:** Operadores Genéticos: Cruza (un punto, uniforme, aritmética), Mutación (bit-flip, Gaussiana). **III:** Teorema de Esquemas e Hipótesis de Bloques de Construcción. **IV:** Variantes avanzadas de AG: AG con codificación real, Programación Genética (GP) con representaciones de árbol.

Objetivos de Aprendizaje **I:** Diseñar representaciones y operadores apropiados para problemas específicos de optimización [Evaluar] **II:** Implementar un Algoritmo Genético Canónico para optimización de funciones [Usar] **III:** Aplicar Programación Genética a tareas de regresión simbólica o programación automática [Evaluar]

Bibliografía: [Eiben and Smith, 2015, Mitchell, 1998]

5.3 Estrategias de Evolución y Programación Evolutiva, 18 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: **I:** Estrategias de Evolución (ES): mecanismos de selección (μ, λ) y $(\mu + \lambda)$. **II:** Autoadaptación de parámetros de estrategia (tamaños de paso de mutación). **III:** Adaptación de Matriz de Covarianza (CMA-ES). **IV:** Programación Evolutiva (EP) y su enfoque en la evolución de comportamiento.

Objetivos de Aprendizaje I: Implementar Estrategias de Evolución con mutación auto-adaptativa para optimización continua [Evaluar] **II:** Comparar el rendimiento de AG vs ES en problemas de referencia [Usar] **III:** Explicar el papel de la adaptación de parámetros de estrategia en la velocidad de convergencia [Familiarizarse]

Bibliografía: [Eiben and Smith, 2015, Deb, 2001]

5.4 Algoritmos Evolutivos Multiobjetivo e Híbridos, 18 hr(s)

Competencias: AG-C08,AG-C12

Tópicos: I: Optimización Multiobjetivo: optimalidad de Pareto, relaciones de dominancia. **II:** EAs Multiobjetivo: NSGA-II, SPEA2, MOEA/D. **III:** Mecanismos de preservación de diversidad y estrategias de archivo. **IV:** Algoritmos Meméticos: Hibridización con búsqueda local. **V:** Técnicas de manejo de restricciones en búsqueda evolutiva.

Objetivos de Aprendizaje I: Resolver problemas multiobjetivo usando métodos de selección basados en Pareto [Evaluar] **II:** Diseñar un algoritmo memético combinando búsqueda global y local [Evaluar] **III:** Evaluar las compensaciones entre convergencia y diversidad en MOEAs [Usar]

Bibliografía: [Eiben and Smith, 2015, Deb, 2001]

6. Bibliografía

[Back, 1996] Back, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press.

[Deb, 2001] Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley.

[Eiben and Smith, 2015] Eiben, A. E. and Smith, J. E. (2015). *Introduction to Evolutionary Computing*. Springer, 2nd edition.

[Mitchell, 1998] Mitchell, M. (1998). *An Introduction to Genetic Algorithms*. MIT Press.

Capítulo 6

Profesores & Cursos

Cursos asignados por profesor

- **Dr. Ernesto Cuadros-Vargas**
  - CS312 Estructuras de Datos Avanzadas
  - CS395 Calidad de Software

Profesor asignado por curso

- Primer Semestre
  - BIC101. Introducción a la Computación (Pág 257)
    - No hay profesor asignado a esta materia
  - BMA101. Algebra Lineal (Pág 262)
    - No hay profesor asignado a esta materia
  - BMA102. Cálculo Diferencial (Pág 264)
    - No hay profesor asignado a esta materia
  - BCH101. Química I (Pág 267)
    - No hay profesor asignado a esta materia
  - BEI101. Inglés I (Pág 272)
    - No hay profesor asignado a esta materia
- Segundo Semestre
  - CS100. Introducción a la Ciencia de la Computación (Pág 276)
    - No hay profesor asignado a esta materia
  - CS112. Programación Orientada a Objetos I (Pág 287)
    - No hay profesor asignado a esta materia
  - CS1D1. Estructuras Discretas (Pág 292)
    - No hay profesor asignado a esta materia
  - BMA103. Cálculo Integral (Pág 295)
    - No hay profesor asignado a esta materia
  - BFI101. Física I (Pág 297)
    - No hay profesor asignado a esta materia
  - BEI102. Inglés II (Pág 302)
    - No hay profesor asignado a esta materia
- Tercer Semestre
  - CS113. Programación Orientada a Objetos II (Pág 306)
    - No hay profesor asignado a esta materia
  - CS2B1. Desarrollo Basado en Plataformas (Pág 312)
    - No hay profesor asignado a esta materia
  - AI161. IA Aplicada (Pág 316)
    - No hay profesor asignado a esta materia
  - BMA104. Cálculo Diferencial e Integral Avanzado (Pág 320)

- No hay profesor asignado a esta materia
- STA251. Estadística y Probabilidades (Pág 324)
  - No hay profesor asignado a esta materia
- BEI201. Inglés III (Pág 329)
  - No hay profesor asignado a esta materia

- Cuarto Semestre
  - CS210. Algoritmos y Estructuras de Datos (Pág 333)
    - No hay profesor asignado a esta materia
  - CS211. Teoría de la Computación (Pág 338)
    - No hay profesor asignado a esta materia
  - CS221. Arquitectura de Computadores (Pág 343)
    - No hay profesor asignado a esta materia
  - CS271. Bases de Datos I (Pág 348)
    - No hay profesor asignado a esta materia
  - CS401. Metodología de la Investigación (Pág 353)
    - No hay profesor asignado a esta materia
  - MA202. Métodos Numéricos (Pág 357)
    - No hay profesor asignado a esta materia
  - BEI202. Inglés IV (Pág 362)
    - No hay profesor asignado a esta materia
- Quinto Semestre
  - CS212. Análisis y Diseño de Algoritmos (Pág 366)
    - No hay profesor asignado a esta materia
  - CS261. Inteligencia Artificial (Pág 371)
    - No hay profesor asignado a esta materia
  - CS272. Bases de Datos II (Pág 378)
    - No hay profesor asignado a esta materia
  - CS291. Ingeniería de Software I (Pág 382)
    - No hay profesor asignado a esta materia
  - CS2S1. Sistemas Operativos (Pág 388)
    - No hay profesor asignado a esta materia
  - FG106. Teatro (Pág 393)
    - No hay profesor asignado a esta materia
  - BEI203. Inglés V (Pág 397)
    - No hay profesor asignado a esta materia
- Sexto Semestre
  - CS231. Redes y Comunicación (Pág 401)
    - No hay profesor asignado a esta materia
  - CS311. Algoritmos para Problemas Complejos (Pág 406)
    - No hay profesor asignado a esta materia
  - CS312. Estructuras de Datos Avanzadas (Pág 409)
    - Ernesto Cuadros-Vargas (ecuadrosv@uni.edu.pe)
  - CS342. Compiladores (Pág 412)
    - No hay profesor asignado a esta materia
  - AI263. Introducción al Aprendizaje de Máquina (Pág 415)
    - No hay profesor asignado a esta materia
  - FI201. Física Computacional (Pág 418)
    - No hay profesor asignado a esta materia
- Séptimo Semestre
  - CS251. Computación Gráfica (Pág 421)
    - No hay profesor asignado a esta materia
  - CS292. Ingeniería de Software II (Pág 425)
    - No hay profesor asignado a esta materia
  - CS2H1. Experiencia de Usuario (UX) (Pág 430)
    - No hay profesor asignado a esta materia
  - AI264. Aprendizaje Profundo (Pág 435)

- No hay profesor asignado a esta materia
- FG211. Ética Profesional (Pág 438)
  - No hay profesor asignado a esta materia
- FG350. Liderazgo y Desempeño (Pág 440)
  - No hay profesor asignado a esta materia

- Octavo Semestre
  - CS281. Computación en la Sociedad (Pág 443)
    - No hay profesor asignado a esta materia
  - CS3I1. Seguridad en Computación (Pág 449)
    - No hay profesor asignado a esta materia
  - CS3P1. Computación Paralela y Distribuída (Pág 455)
    - No hay profesor asignado a esta materia
  - CS402. Proyecto de tesis 1 (Pág 465)
    - No hay profesor asignado a esta materia
  - EX301. Actividades Extracurriculares (Pág 469)
    - No hay profesor asignado a esta materia
  - ET201. Formación de Empresas de Base Tecnológica I (Pág 471)
    - No hay profesor asignado a esta materia
  - AI268. Visión Computacional (Pág 475)
    - No hay profesor asignado a esta materia
  - CS391. Ingeniería de Software III (Pág 481)
    - No hay profesor asignado a esta materia
  - CS393. Sistemas de Información (Pág 486)
    - No hay profesor asignado a esta materia
  - CS394. Arquitectura de Software y DevOps para Software en la Nube (Pág 489)
    - No hay profesor asignado a esta materia
  - CS395. Calidad de Software (Pág 495)
    - Ernesto Cuadros-Vargas (ecuadrosv@uni.edu.pe)
- Noveno Semestre
  - CS370. Big Data (Pág 503)
    - No hay profesor asignado a esta materia
  - CS400. Prácticas Pre-profesionales (Pág 507)
    - No hay profesor asignado a esta materia
  - CS403. Proyecto de tesis 2 (Pág 509)
    - No hay profesor asignado a esta materia
  - AI365. Modelos Generativos Avanzados en IA (Pág 513)
    - No hay profesor asignado a esta materia
  - CB309. Bioinformática (Pág 515)
    - No hay profesor asignado a esta materia
  - ET301. Formación de Empresas de Base Tecnológica II (Pág 519)
    - No hay profesor asignado a esta materia
  - AI369. Robótica (Pág 522)
    - No hay profesor asignado a esta materia
  - CS351. Tópicos en Computación Gráfica (Pág 526)
    - No hay profesor asignado a esta materia
  - CS353. Computación Cuántica (Pág 530)
    - No hay profesor asignado a esta materia
  - CS392. Tópicos en Ingeniería de Software (Pág 534)
    - No hay profesor asignado a esta materia
  - CS3P3. Internet de las Cosas (Pág 541)
    - No hay profesor asignado a esta materia
- Décimo Semestre
  - CS3P2. Cloud Computing (Pág 544)
    - No hay profesor asignado a esta materia
  - CS404. Taller de Investigación (Pág 548)
    - No hay profesor asignado a esta materia

- AI367. Tópicos en Inteligencia Artificial (Pág 552)
  - No hay profesor asignado a esta materia
- AI368. Computación Evolutiva (Pág 559)
  - No hay profesor asignado a esta materia

Capítulo 7

Equivalencias con otros planes curriculares

A continuación se pueden observar las equivalencias de la presente malla con el(los) Plan(es) curricular(es) anterior(ES). Para ver mayores detalles de la malla propuesta observar la sección 4.2 (Pág. 204).

Equivalencia del Plan 2027 al 2018

Primer Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
BIC101	Introducción a la Computación	4	BIC01	Introducción a la Computación	1er Sem	2
BMA101	Algebra Lineal	4	BMA03	Álgebra Lineal	1er Sem	4
BMA102	Cálculo Diferencial	5	BMA01	Cálculo Diferencial	1er Sem	5
BCH101	Química I	5	BQU01	Química I	1er Sem	5
BEI101	Inglés I	2	CL002	Inglés 1	2do Sem	2

Segundo Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS100	Introducción a la Ciencia de la Computación	3			1er Sem	
CS112	Programación Orientada a Objetos I	4	CC112	Fundamentos de programación	2do Sem	4
CS1D1	Estructuras Discretas	4	CM2H1	Matemática Discreta	3er Sem	4
BMA103	Cálculo Integral	5	BMA02	Cálculo Integral	2do Sem	5
BFI101	Física I	5	BFI01	Física I	1er Sem	5
BEI102	Inglés II	2	BIE01	Idioma Extranjero Avanzado	4to Sem	2

Tercer Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS113	Programación Orientada a Objetos II	4	CC211	Programación Orientada a Objetos	3er Sem	4
CS2B1	Desarrollo Basado en Plataformas	4			3er Sem	
AI161	IA Aplicada	4			3er Sem	
STA251	Estadística y Probabilidades	3	CM1H2	Cálculo de probabilidades	2do Sem	3

Cuarto Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS210	Algoritmos y Estructuras de Datos	4	CC232	Algoritmo y Estructura de Datos	4to Sem	4
CS211	Teoría de la Computación	4	CC321	Teoría de Autómatas Lenguaje y Computación	5to Sem	4
CS221	Arquitectura de Computadores	4	CC221	Arquitectura de Computadores	3er Sem	4
CS271	Bases de Datos I	4	CC202	Base de Datos	4to Sem	4
CS401	Metodología de la Investigación	2	CC4H1	Metodología de la Investigación	7mo Sem	2
MA202	Métodos Numéricos	3	CM4F1	Análisis y Modelamiento Numérico I	5to Sem	5

Quinto Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS212	Análisis y Diseño de Algoritmos	4	CC371	Análisis y Diseño de Algoritmos	5to Sem	4
CS261	Inteligencia Artificial	4	CC421	Inteligencia Artificial	7mo Sem	4
CS272	Bases de Datos II	3			5to Sem	
CS291	Ingeniería de Software I	4	CC341	Ingeniería de Software	5to Sem	4
CS2S1	Sistemas Operativos	4	CC222	Sistemas Operativos	4to Sem	4

Sexto Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS231	Redes y Comunicación	3	CC311	Computación Centrada en Redes	5to Sem	4
CS311	Algoritmos para Problemas Complejos	3			6to Sem	
CS312	Estructuras de Datos Avanzadas	4			6to Sem	
CS342	Compiladores	4			5to Sem	
AI263	Introducción al Aprendizaje de Máquina	4			6to Sem	4
FI201	Física Computacional	3			2do Sem	

Séptimo Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS251	Computación Gráfica	4	CC431	Computación Gráfica	7mo Sem	4
CS292	Ingeniería de Software II	4	CC3S2	Desarrollo de Software	6to Sem	4
CS2H1	Experiencia de Usuario (UX)	4	CC451	Interacción Humano-Computador	7mo Sem	4
AI264	Aprendizaje Profundo	4			7mo Sem	
FG211	Ética Profesional	2	BEF01	Ética y Filosofía Política	6to Sem	2

Octavo Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS281	Computación en la Sociedad	2			8vo Sem	
CS3I1	Seguridad en Computación	3	CC411	Seguridad en Sistemas Informáticos	7mo Sem	4
CS3P1	Computación Paralela y Distribuída	4	CC4P1	Programación Concurrente y Distribuida	7mo Sem	4
CS402	Proyecto de tesis 1	4	CC412	Proyecto de Tesis I	8vo Sem	4
EX301	Actividades Extracurriculares	2	BAE01	Actividades Extracurriculares	10mo Sem	1
CS391	Ingeniería de Software III	3			8vo Sem	
CS393	Sistemas de Información	3			7mo Sem	

Noveno Semestre – Plan2027			Plan 2018		
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS370	Big Data	3	CC442	Minería de Datos	8vo Sem	4
CS400	Prácticas Pre-profesionales	2	XP200	Prácticas Pre-Profesionales II	10mo Sem	2
CS403	Proyecto de tesis 2	3	CC511	Proyecto de Tesis II	9no Sem	4
AI365	Modelos Generativos Avanzados en IA	4			9no Sem	
CB309	Bioinformática	4			9no Sem	
CS351	Tópicos en Computación Gráfica	3			9no Sem	
CS353	Computación Cuántica	3			10mo Sem	
CS392	Tópicos en Ingeniería de Software	3			9no Sem	
CS3P3	Internet de las Cosas	3			10mo Sem	

Décimo Semestre – Plan2027			Plan 2018			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CS3P2	Cloud Computing	3	CC422	Infraestructuras de Computación	8vo Sem	4
CS404	Taller de Investigación	4	CC512	Proyecto de Tesis III	10mo Sem	4

Equivalencia del Plan 2018 al 2027

Primer Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
BIC01	Introducción a la Computación	2	BIC101	Introducción a la Computación	1er Sem	4
BMA03	Álgebra Lineal	4	BMA101	Algebra Lineal	1er Sem	4
BMA01	Cálculo Diferencial	5	BMA102	Cálculo Diferencial	1er Sem	5
BQU01	Química I	5	BCH101	Química I	1er Sem	5
			CS100	Introducción a la Ciencia de la Computación	2do Sem	3
BFI01	Física I	5	BFI101	Física I	2do Sem	5

Segundo Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CL002	Inglés 1	2	BEI101	Inglés I	1er Sem	2
CC112	Fundamentos de programación	4	CS112	Programación Orientada a Objetos I	2do Sem	4
BMA02	Cálculo Integral	5	BMA103	Cálculo Integral	2do Sem	5
CM1H2	Cálculo de probabilidades	3	STA251	Estadística y Probabilidades	3er Sem	3
			FI201	Física Computacional	6to Sem	3

Tercer Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CM2H1	Matemática Discreta	4	CS1D1	Estructuras Discretas	2do Sem	4
CC211	Programación Orientada a Objetos	4	CS113	Programación Orientada a Objetos II	3er Sem	4
CC221	Arquitectura de Computadores	4	CS221	Arquitectura de Computadores	4to Sem	4

Cuarto Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
BIE01	Idioma Extranjero Avanzado	2	BEI102	Inglés II	2do Sem	2
CC232	Algoritmo y Estructura de Datos	4	CS210	Algoritmos y Estructuras de Datos	4to Sem	4
CC202	Base de Datos	4	CS271	Bases de Datos I	4to Sem	4
CC222	Sistemas Operativos	4	CS2S1	Sistemas Operativos	5to Sem	4
BEG01	Economía General					

Quinto Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CC321	Teoría de Autómatas Lenguaje y Computación	4	CS211	Teoría de la Computación	4to Sem	4
CM4F1	Análisis y Modelamiento Numérico I	5	MA202	Métodos Numéricos	4to Sem	3
CC371	Análisis y Diseño de Algoritmos	4	CS212	Análisis y Diseño de Algoritmos	5to Sem	4
CC341	Ingeniería de Software	4	CS291	Ingeniería de Software I	5to Sem	4
CC311	Computación Centrada en Redes	4	CS231	Redes y Comunicación	6to Sem	3

Sexto Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
		4	AI263	Introducción al Aprendizaje de Máquina	6to Sem	4
CC3S2	Desarrollo de Software	4	CS292	Ingeniería de Software II	7mo Sem	4
BEF01	Ética y Filosofía Política	2	FG211	Ética Profesional	7mo Sem	2
BRN01	Realidad Nacional, Constitución y Derechos Humanos					
CC3M2	Matemática Computacional					

Séptimo Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CC4H1	Metodología de la Investigación	2	CS401	Metodología de la Investigación	4to Sem	2
CC421	Inteligencia Artificial	4	CS261	Inteligencia Artificial	5to Sem	4
CC431	Computación Gráfica	4	CS251	Computación Gráfica	7mo Sem	4
CC451	Interacción Humano-Computador	4	CS2H1	Experiencia de Usuario (UX)	7mo Sem	4
CC411	Seguridad en Sistemas Informáticos	4	CS3I1	Seguridad en Computación	8vo Sem	3
CC4P1	Programación Concurrente y Distribuida	4	CS3P1	Computación Paralela y Distribuída	8vo Sem	4
			CS393	Sistemas de Información	8vo Sem	3

Octavo Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CC412	Proyecto de Tesis I	4	CS402	Proyecto de tesis 1	8vo Sem	4
			CS391	Ingeniería de Software III	8vo Sem	3
CC442	Minería de Datos	4	CS370	Big Data	9no Sem	3
CC422	Infraestructuras de Computación	4	CS3P2	Cloud Computing	10mo Sem	3

Noveno Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
CC511	Proyecto de Tesis II	4	CS403	Proyecto de tesis 2	9no Sem	3
CC521	Programación Evolutiva					

Décimo Semestre – Plan 2018			Plan2027			
Código	**Curso**	**Cr**	**Código**	**Curso**	**Sem**	**Cr**
BAE01	Actividades Extracurriculares	1	EX301	Actividades Extracurriculares	8vo Sem	2
XP200	Prácticas Pre-Profesionales II	2	CS400	Prácticas Pre-profesionales	9no Sem	2
			CS3P3	Internet de las Cosas	9no Sem	3
CC512	Proyecto de Tesis III	4	CS404	Taller de Investigación	10mo Sem	4

Capítulo 8

Laboratorios

BIC101. Introducción a la Computación (Obligatorio) 1er Sem, Lab: 4 hr(s)

Laboratorio Básico de Computación:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

BCH101. Química I (Obligatorio) 1er Sem, Lab: 2 hr(s)

Laboratorio de Química:

- **Equipos**: Balanzas analíticas, microscopios, espectrofotómetros, kits de vidriería.
- **Seguridad**: Campanas extractoras, duchas de emergencia, kits para derrames.
- **Reactivos**: Ácidos/bases, soluciones tampón, solventes orgánicos, indicadores de pH.

CS112. Programación Orientada a Objetos I (Obligatorio) 2do Sem, Lab: 4 hr(s)

Laboratorio Básico de Computación:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

BFI101. Física I (Obligatorio) 2do Sem, Lab: 4 hr(s)

Laboratorio de Física:

- **Equipos y Herramientas**: Bancos ópticos, mesas de fuerzas, péndulos, carros dinámicos, espectrómetros, sensores PASCO y módulos de medición (multímetros, fuentes, vernier).
- **Hardware y Tecnología**: Laptops de uso académico, sensores y módulos Arduino para adquisición de datos.
- **Software**: Arduino IDE, MATLAB para análisis, Microsoft Office y software especializado en adquisición de datos experimentales.
- **Servicios y Experimentos**: Prácticas de Física I y II (Mecánica, Termodinámica, Electromagnetismo, Ondas), capacitación en instrumentación y apoyo a proyectos de investigación.

CS113. Programación Orientada a Objetos II (Obligatorio) 3er Sem, Lab: 2 hr(s)

Laboratorio Básico de Computación:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

CS2B1. Desarrollo Basado en Plataformas (Obligatorio) 3er Sem, Lab: 4 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

AI161. IA Aplicada (Obligatorio) 3er Sem, Lab: 4 hr(s)

Laboratorio Básico de Computación:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

CS210. Algoritmos y Estructuras de Datos (Obligatorio) 4to Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS211. Teoría de la Computación (Obligatorio) 4to Sem, Lab: 2 hr(s)

Laboratorio Básico de Computación:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.

- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

CS221. Arquitectura de Computadores (Obligatorio) 4to Sem, Lab: 4 hr(s)

Laboratorio de Redes y Ciberseguridad:
- **Hardware**: 30 PCs + racks de red (switches/routers Cisco, clusters Raspberry Pi).
- **Sistemas Operativos**: Linux (Kali, CentOS) / Windows Server.
- **Software**: Wireshark, GNS3, VMware ESXi, emuladores 8086.
- **Enfoque**: Simulación de redes, pruebas de penetración, protocolos IoT.

CS271. Bases de Datos I (Obligatorio) 4to Sem, Lab: 4 hr(s)

Laboratorio Avanzado de Computación:
- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS212. Análisis y Diseño de Algoritmos (Obligatorio) 5to Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:
- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS261. Inteligencia Artificial (Obligatorio) 5to Sem, Lab: 2 hr(s)

Laboratorio Básico de Computación:
- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

CS272. Bases de Datos II (Obligatorio) 5to Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:
- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).

- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS291. Ingeniería de Software I (Obligatorio) 5to Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:
- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS2S1. Sistemas Operativos (Obligatorio) 5to Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:
- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS231. Redes y Comunicación (Obligatorio) 6to Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:
- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS311. Algoritmos para Problemas Complejos (Obligatorio) 6to Sem, Lab: 2 hr(s)

Laboratorio Básico de Computación:
- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

CS312. Estructuras de Datos Avanzadas (Obligatorio) 6to Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS342. Compiladores (Obligatorio) 6to Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

AI263. Introducción al Aprendizaje de Máquina (Obligatorio) 6to Sem, Lab: 4 hr(s)

Laboratorio Básico de Computación:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

FI201. Física Computacional (Obligatorio) 6to Sem, Lab: 2 hr(s)

Laboratorio de Física:

- **Equipos y Herramientas**: Bancos ópticos, mesas de fuerzas, péndulos, carros dinámicos, espectrómetros, sensores PASCO y módulos de medición (multímetros, fuentes, vernier).
- **Hardware y Tecnología**: Laptops de uso académico, sensores y módulos Arduino para adquisición de datos.
- **Software**: Arduino IDE, MATLAB para análisis, Microsoft Office y software especializado en adquisición de datos experimentales.
- **Servicios y Experimentos**: Prácticas de Física I y II (Mecánica, Termodinámica, Electromagnetismo, Ondas), capacitación en instrumentación y apoyo a proyectos de investigación.

CS251. Computación Gráfica (Obligatorio) 7mo Sem, Lab: 4 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.

- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS292. Ingeniería de Software II (Obligatorio) 7mo Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS2H1. Experiencia de Usuario (UX) (Obligatorio) 7mo Sem, Lab: 4 hr(s)

Laboratorio Básico de Computación:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

AI264. Aprendizaje Profundo (Obligatorio) 7mo Sem, Lab: 4 hr(s)

Laboratorio Básico de Computación:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

CS3I1. Seguridad en Computación (Obligatorio) 8vo Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS3P1. Computación Paralela y Distribuída (Obligatorio) 8vo Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS402. Proyecto de tesis 1 (Obligatorio) 8vo Sem, Lab: 4 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

AI268. Visión Computacional (Electivo) 8vo Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS393. Sistemas de Información (Electivo) 8vo Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS370. Big Data (Obligatorio) 9no Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS403. Proyecto de tesis 2 (Obligatorio) 9no Sem, Lab: 4 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

AI365. Modelos Generativos Avanzados en IA (Obligatorio) 9no Sem, Lab: 4 hr(s)

Laboratorio Básico de Computación:

- **Hardware**: 30 equipos (Intel Core i5, 8GB RAM, SSD 256GB, gráficos/sonido integrados, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Ubuntu LTS) / Windows 11 Pro (arranque dual).
- **Lenguajes**: Python, Java, C++ (C++11/14/17/20), C#, Prolog, Lex/Yacc.
- **Herramientas**: Git, VS Code, Eclipse, Jupyter Notebook.

AI369. Robótica (Electivo) 9no Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS351. Tópicos en Computación Gráfica (Electivo) 9no Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.

- *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
- *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS353. Computación Cuántica (Electivo) 9no Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS392. Tópicos en Ingeniería de Software (Electivo) 9no Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS3P3. Internet de las Cosas (Electivo) 9no Sem, Lab: 4 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS3P2. Cloud Computing (Obligatorio) 10mo Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

CS404. Taller de Investigación (Obligatorio) 10mo Sem, Lab: 4 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

AI367. Tópicos en Inteligencia Artificial (Obligatorio) 10mo Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.

AI368. Computación Evolutiva (Obligatorio) 10mo Sem, Lab: 2 hr(s)

Laboratorio Avanzado de Computación:

- **Hardware**: 30 PCs de alto rendimiento (Intel Core i7, 16GB RAM, SSD NVMe 512GB, NVIDIA GTX 1660, Ethernet Gigabit).
- **Sistemas Operativos**: Linux (Fedora) / Windows 11 Pro.
- **Lenguajes**: C++17/20, Python (con librerías científicas).
- **Software**:
  - *Desarrollo*: J2EE, .NET 6, MATLAB, GCC/Clang.
  - *Bases de datos/Cloud*: PostgreSQL, MongoDB, Docker.
  - *Especializado*: Rational Rose, VTK, LEDA, emuladores 8086.