Universidad Nacional de Ingeniería (UNI)
Escuela Profesional de
Ciencia de la Computación
Sílabo 2026-I

1. CURSO

CS100. Introducción a la Ciencia de la Computación (Obligatorio)

2. INFORMACIÓN GENERAL

2.1 Curso	:	CS100. Introducción a la Ciencia de la Computación
2.2 Semestre	:	2^{do} Semestre.
2.3 Créditos	:	3
2.4 Horas	:	2 HT; 2 HP;
2.5 Duración del periodo	:	16 semanas
2.6 Condición	:	Obligatorio
2.7 Modalidad de aprendizaje	:	Presencial
2.8 Prerrequisitos	:	Ninguno

3. PROFESORES

Atención previa coordinación con el profesor

4. INTRODUCCIÓN AL CURSO

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.

5. OBJETIVOS

- 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.

6. RESULTADOS DEL ESTUDIANTE

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

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

AG-C12) Usa enfoques sistémicos para seleccionar, desarrollar, aplica, integrar y administrar tecnologías seguras de computación para lograr los objetivos del usuario. (Familiarity)

7. TEMAS

Unidad 1: Pensamiento Computacional (Parte I) (4 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Explicación del sistema de evaluación. • Instrucciones generales del curso. • Definición de Computación. • La Computación como un binomio Humano-Computadora. • Distorsiones en la definición de computación. • La Computación como la automatización de la abstracción. • Computación e Ingeniería: similitudes y diferencias. • Resolución de problemas algorítmicos. • Dinámica: Comprender la ejecución de un algoritmo a velocidad humana.	• Aplicar los conceptos fundamentales de la computación en situaciones de la vida real. [Usar] • Identificar distorsiones de la Computación en situaciones de la vida real. [Usar] • Identificar claramente al menos 3 contextos de uso de la palabra "Engineer" en inglés. [Evaluar] • Identificar las limitaciones humanas para resolver problemas computacionales. [Usar]
Lecturas : [BB19]	

Unidad 2: Pensamiento Computacional (Parte II) (4 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Descomposición (Dividir un problema en piezas más pequeñas). • Abstracción (Enfocarse en lo importante). • Reconocimiento de patrones (Identificar secuencias similares basadas en problemas previos). • Diseño de algoritmos.	• Aplicar el Pensamiento Computacional a problemas del mundo real. [Usar]
Lecturas : [BB19]	

Unidad 3: Conceptos Generales (4 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Numeración binaria vs. decimal. • Representación de caracteres: la tabla ASCII. • Búsqueda binaria. • Complejidad computacional de un algoritmo.	• Aplicar varios sistemas de numeración a problemas del mundo real. [Usar] • Comprender la representación interna de caracteres en las tablas ASCII y UTF-8. [Familiarizarse] • Aplicar la estrategia algorítmica de Divide y Vencerás. [Usar] • Determinar análisis básicos de complejidad algorítmica. [Usar]
Lecturas : [BB19]	

Unidad 4: Representación de Datos a Nivel de Máquina (4 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• 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: – Punto fijo – Punto flotante • Representación de datos no numéricos • Representación de registros, arreglos y tipos de datos UTF	• Discutir por qué todo en las computadoras son datos, incluyendo instrucciones [Debatir] • Explicar cómo las representaciones numéricas de longitud fija pueden afectar la exactitud y precisión [Explicar] • Describir cómo se almacenan los enteros negativos en representaciones de magnitud con signo y complemento a dos [Describir] • Discutir cómo diferentes formatos pueden representar datos numéricos [Debatir] • Explicar la representación a nivel de bits de datos no numéricos, como caracteres, cadenas, registros y arreglos [Explicar] • Traducir datos numéricos de un formato a otro [Traducir] • 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]
Lecturas : [BB19]	

Unidad 5: Abstracción y Representación de Programas (2 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Programas que toman (otros) programas como entrada, como intérpretes, compiladores, verificadores de tipos, generadores de documentación • Componentes de un lenguaje: – Definiciones de alfabetos, delimitadores, oraciones, sintaxis y semántica – 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 • 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*). • 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 • Representación de instrucciones de bajo nivel como instrucciones de máquina virtual, lenguaje ensamblador y representación binaria	• Ilustrar el ámbito de variables y visibilidad usando programas simples [Aplicar] • Ilustrar diferentes tipos de paso de parámetros usando un lenguaje de programación pseudo simple [Aplicar] • Explicar el efecto secundario usando variables globales y no locales y cómo corregir tales programas [Explicar] • Explicar cómo los programas que procesan otros programas tratan a los otros programas como sus datos de entrada [Explicar] • Describir una gramática y un árbol de sintaxis abstracta para un lenguaje pequeño [Describir]
Lecturas : [BB19]	

Unidad 6: Fundamentos de Criptografía (2 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Basic concepts on Cryptography • Public-keys, Private-keys	• Explicar el papel de la criptografía en el soporte de la seguridad y la privacidad [Explicar] • Discutir los riesgos de inventar los propios métodos criptográficos [Debatir] • Discutir la importancia de los números primos en criptografía y explicar su uso en algoritmos criptográficos [Debatir] • Implementar y criptoanalizar cifrados clásicos [Implementar]
Lecturas : [BB19]	

Unidad 7: Gestión de Memoria (4 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Revisión de memoria física, traducción de direcciones y hardware de gestión de memoria. Ver también: Arquitectura y Organización (AR)-MemoryHierarchy, Fundamentos Matemáticos y Estadísticos (MSF)-Discrete • 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: Arquitectura y Organización (AR)-MemoryHierarchy, Fundamentos de Sistemas (SF)-Performance • Direccionamiento lógico y físico, virtualización de espacio de direcciones. Ver también: Arquitectura y Organización (AR)-MemoryHierarchy, Fundamentos Matemáticos y Estadísticos (MSF)-Discrete • 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 – 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: Arquitectura y Organización (AR)-FunctionalOrganization, Arquitectura y Organización (AR)-MemoryHierarchy, Fundamentos de Sistemas (SF)-Performance) • Memoria virtual: aprovechando el hardware de memoria virtual para servicios del SO y eficiencia	• Explicar jerarquía de memoria y compensaciones costo-rendimiento [Explicar] • Resumir los principios de memoria virtual aplicados a almacenamiento en caché y paginación [Resumir] • Evaluar las compensaciones en términos de tamaño de memoria (memoria principal, memoria caché, memoria auxiliar) y velocidad del procesador [Evaluar] • 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]
Lecturas : [BB19]	

Unidad 8: Rol y Propósito de los Sistemas Operativos (2 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• 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) • Influencias de seguridad, redes, multimedia, computación paralela y distribuida	• Comprender los objetivos y funciones de los sistemas operativos modernos [Explicar] • Evaluar los problemas de diseño en diferentes escenarios de uso (por ejemplo, sistema operativo en tiempo real, móvil, servidor) [Evaluar] • Comprender cómo la evolución y la estabilidad son deseables y mutuamente antagónicas en las funciones del sistema operativo [Explicar]
Lecturas : [BB19]	

Unidad 9: Principios del Sistema Operativo (2 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Diseño y enfoques de software del sistema operativo (por ejemplo, monolítico, en capas, modular, microkernel, unikernel) • Abstracciones, procesos y recursos • La evolución del vínculo entre la arquitectura del hardware y las funciones del sistema operativo • Aprovechamiento de interrupciones desde el nivel de hardware: rutinas de servicio e implementaciones. Ver también: Arquitectura y Organización (AR)-AssemblyLevelMachineOrganization – Interrupciones de temporizador para implementar segmentos de tiempo – Interrupciones de E/S para poner hilos bloqueados en espera sin sondeo	• 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] • Categorizar llamadas al sistema por propósito [Categorizar] • 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]
Lecturas : [BB19]	

Unidad 10: Fundamentos de Redes y Comunicaciones (4 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Importancia de las redes en la informática contemporánea, y desafíos asociados. Ver también: Sociedad, ética y la Profesión (SEP)-Context, Sociedad, ética y la Profesión (SEP)-Privacy • Organización de Internet (por ejemplo, usuarios, Proveedores de Servicios de Internet, sistemas autónomos, proveedores de contenido, redes de entrega de contenido) • Capas y sus roles (aplicación, transporte, red, enlace de datos y física) • 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.)	• Articular la organización de Internet [Articular] • Listar y definir la terminología de red apropiada [Listar/Enumerar] • Describir la estructura en capas de una arquitectura de red típica [Describir]
Lecturas : [BB19]	

Unidad 11: Aplicaciones en Red (4 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• 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: Computación Paralela y Distribuida (PDC)-Communication, Computación Paralela y Distribuida (PDC)-Coordination • Diversidad de demandas de aplicaciones en red (por ejemplo, latencia, ancho de banda y tolerancia a pérdidas). Ver también: Computación Paralela y Distribuida (PDC)-Communication, Sociedad, ética y la Profesión (SEP)-Sustainability, Sociedad, ética y la Profesión (SEP)-Context • Cobertura de protocolos de capa de aplicación (por ejemplo, HTTP) • Interacciones con APIs de TCP, UDP y Socket. Ver también: Computación Paralela y Distribuida (PDC)-Programs	• Definir los principios de denominación, direccionamiento y localización de recursos [Definir] • Analizar las necesidades de demandas específicas de aplicaciones en red [Analizar]
Lecturas : [BB19]	

Unidad 12: Marco de Análisis de Complejidad (2 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Marco de Análisis de Complejidad: – Rendimiento de un algoritmo en el mejor caso, caso promedio y peor caso – Mediciones empíricas y relativas (Orden de Crecimiento) – Tamaño de entrada y operaciones primitivas – Eficiencia de tiempo y espacio • Mediciones empíricas de rendimiento • Compromisos tiempo-espacio en algoritmos	• 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] • Para cada algoritmo en la unidad Fundamentos Algorítmicos (AL)-FoundationalDataStructuresAlgorithms, explicar su clase de complejidad de tiempo de ejecución y por qué pertenece a esta clase [Explicar] • Evaluar informalmente la clase de complejidad fundamental de algoritmos simples [Evaluar] • 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] • Explicar ejemplos que ilustren los compromisos tiempo-espacio de algoritmos [Explicar] • Explicar cómo el balance del árbol afecta la eficiencia de las operaciones de árbol de búsqueda binaria [Explicar]
Lecturas : [BB19]	

Unidad 13: Notación Asintótica y Clases de Complejidad (1 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Análisis de complejidad asintótica (cotas promedio y peor caso): – Notaciones formales Big-O, Big-Omega y Big-Theta – Clases de Complejidad Fundamentales y Ejemplos/Problemas Representativos: ∗ $O(1)$ Constante (por ejemplo, acceso a arreglo) ∗ $O(\log_2 n)$ Logarítmica (por ejemplo, búsqueda binaria) ∗ $O(n)$ Lineal (por ejemplo, búsqueda lineal) ∗ $O(n \log_2 n)$ Log Lineal (por ejemplo, mergesort) ∗ $O(n^2)$ Cuadrática (por ejemplo, ordenamiento por selección) ∗ $O(n^c)$ Polinomial (por ejemplo, $O(n^3)$ eliminación gaussiana) ∗ $O(2^n)$ Exponencial (por ejemplo, Mochila, Satisfactibilidad (SAT), Viajante de Comercio (TSP), todos los subconjuntos) ∗ $O(n!)$ Factorial (por ejemplo, circuito hamiltoniano, todas las permutaciones)	• Usando ejemplos, explicar cada una de las clases de complejidad fundamentales en esta unidad [Explicar] • Para cada clase de complejidad fundamental en esta unidad, explicar un algoritmo que demuestre la complejidad de tiempo de ejecución asociada [Explicar] • Explicar a una audiencia no técnica la importancia de los algoritmos tratables versus intratables usando una explicación intuitiva de la complejidad Big-O [Explicar]
Lecturas : [BB19]	

Unidad 14: Análisis de Complejidad II: Recursión, Amortización y Cotas Ajustadas (1 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Notaciones Little-o, Little-Omega y Little Theta	• 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]
Lecturas : [BB19]	

Unidad 15: Estructuras de Datos Fundamentales (3 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Tipo de Dato Abstracto (ADT) y operaciones sobre un ADT: – Operaciones de diccionario (insertar, eliminar, encontrar) • Arreglos: – Numéricos vs no numéricos, cadenas de caracteres – Unidimensionales (vector) vs multidimensionales (matriz) • Registros/Estructuras/Tuplas y Objetos • Listas enlazadas (por razones históricas): – Simples vs Dobles y Lineales vs Circulares • Pilas • Colas y deques: – Cola de prioridad basada en montículo • Tablas/mapas hash: – Resolución de colisiones y complejidad (por ejemplo, sondeo, encadenamiento, rehash) • árboles: – Binarios, n-arios y árboles de búsqueda – Balanceados (por ejemplo, AVL, Rojo-Negro, Montículo)	• Para cada ADT/Estructura de Datos en esta unidad: – Explicar su definición, propiedades, representación(es) y operaciones de ADT asociadas. – Explicar paso a paso cómo las operaciones de ADT asociadas con la estructura de datos la transforman. [Explicar] • Explicar cómo se maneja la evitación de colisiones y la resolución de colisiones en tablas hash [Explicar] • Explicar la propiedad de montículo y el uso de montículos como una implementación de una cola de prioridad [Explicar]
Lecturas : [BB19]	

Unidad 16: Algoritmos Fundamentales (3 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Grafos (por ejemplo, [no]dirigidos, [a]cíclicos, [no]conexos y [no]ponderados): – Representación de grafos: lista de adyacencia vs matriz • Algoritmos de búsqueda: – Complejidad $O(n)$ (por ejemplo, búsqueda lineal/secuencial en arreglo/lista) – Complejidad $O(\log_2 n)$ (por ejemplo, búsqueda binaria) – Complejidad $O(\log_b n)$ (por ejemplo, búsqueda en árbol no informada en profundidad/amplitud) • Algoritmos de ordenamiento (por ejemplo, estables, inestables): – Complejidad $O(n^2)$ (por ejemplo, inserción, selección) – Complejidad $O(n \log n)$ (por ejemplo, quicksort, merge, timesort) • Algoritmos de grafos: – Camino más corto (por ejemplo, Dijkstra, Floyd) – árbol de expansión mínima (por ejemplo, Prim, Kruskal) • Algoritmos de ordenamiento: – Complejidad $O(n \log n)$ heapsort – Pseudo $O(n)$ complejidad (por ejemplo, bucket, counting, radix) • Algoritmos de grafos: – Clausura transitiva (por ejemplo, Warshall) – Ordenamiento topológico	• Para cada algoritmo en esta unidad explicar paso a paso cómo opera el algoritmo [Explicar] • Para cada enfoque algorítmico (por ejemplo, ordenamiento) en esta unidad aplicar un ejemplo prototípico del enfoque (por ejemplo, ordenamiento por mezcla) [Aplicar] • 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] • 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]
Lecturas : [BB19]	

Unidad 17: Algoritmos Avanzados (2 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Algoritmos de criptografía (por ejemplo, SHA-256) • Algoritmos paralelos	• Una apreciación de la computación cuántica y su aplicación a ciertos problemas [Explicar]
Lecturas : [BB19]	

Unidad 18: Conceptos Fundamentales de Sistemas de Bases de Datos (4 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• 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) • Gestión de transacciones • Normalización • Enfoques para gestionar grandes volúmenes de datos (por ejemplo, sistemas de bases de datos NoSQL, uso de MapReduce) • Bases de datos distribuidas/sistemas basados en la nube • Uso de un lenguaje de consulta declarativo	• Identificar al menos cuatro ventajas que proporciona el uso de un sistema de base de datos [Analizar] • Enumerar los componentes de un sistema de base de datos (relacional) [Enumerar] • Defender el valor de la independencia de datos [Defender] • Componer una consulta simple de selección-proyección-unión en SQL [Componer] • Describir las ventajas de eliminar datos duplicados repetidos [Describir] • Esbozar cómo MapReduce usa paralelismo para procesar datos eficientemente [Esquematizar/Esbozar]
Lecturas : [BB19]	

Unidad 19: Programación Orientada a Objetos I: Fundamentos (4 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Programación imperativa como subconjunto de la programación orientada a objetos. • Diseño orientado a objetos: – Descomposición en objetos que poseen estado y tienen comportamiento. – 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. • Idiomas orientados a objetos para encapsulación: – Privacidad, ocultación de datos y visibilidad de miembros de clase. – Interfaces que revelan solo firmas de métodos. – Clases base abstractas, rasgos (*traits*) y mixins.	• Enumerar las diferencias entre los paradigmas de programación imperativa y orientada a objetos [Enumerar] • Componer una clase a través del diseño, implementación y prueba para cumplir con los requisitos de comportamiento [Componer] • Usar mecanismos de encapsulación orientados a objetos como interfaces y miembros privados [Usar]
Lecturas : [BB19]	

Unidad 20: Control de Versiones y CI/CD (2 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Gestión de configuración de software y control de versiones: Fundamentos de Desarrollo de Software (SDF)-Practices – Configuración en control de versiones, builds/configuración reproducibles. – Estrategias de ramificación en control de versiones. Ramas de desarrollo vs ramas de lanzamiento. Desarrollo basado en tronco (*trunk-based development*). – Estrategias de fusión/rebase, cuando sean relevantes. • Gestión de lanzamientos (*release management*). • Automatización de procesos de software: – Sistemas de construcción (*build systems*): el valor de builds rápidos, herméticos y reproducibles, comparar/contrastar enfoques para construir un proyecto. – 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). – 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. – Gestión de dependencias: actualización de dependencias externas/aguas arriba, gestión de paquetes, SemVer. • Conceptos y mecanismos de integración de herramientas. Fundamentos de Desarrollo de Software (SDF)-Practices • Uso de las facilidades de un IDE moderno: depuración, refactorización, búsqueda/indexación, asistentes de código basados en ML, etc. Fundamentos de Desarrollo de Software (SDF)-Practices	• Describir la diferencia entre la gestión de configuración de software centralizada y distribuida [Describir] • Describir cómo el control de versiones puede usarse para ayudar en la gestión de lanzamientos de software [Describir] • Identificar elementos de configuración y usar una herramienta de control de código fuente en un proyecto pequeño basado en equipo [Analizar] • 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] • Demostrar la capacidad de usar herramientas de software para apoyar el desarrollo de un producto de software de tamaño mediano [Demostrar]
Lecturas : [BB19]	

Unidad 21: Ingeniería de Requisitos (2 horas)

Resultados esperados: AG-C02,AG-C08,AG-C12

Temas	**Aprendizaje esperado (*Learning Outcomes*)**

• Describir requisitos funcionales usando, por ejemplo, casos de uso o historias de usuario.
  – Usar al menos un método para documentar y estructurar requisitos funcionales.
  – Comprender cómo el método apoya el diseño y la implementación.
  – Fortalezas y debilidades de usar un enfoque específico.
• Obtención (*elicitation*) de requisitos.
  – Fuentes de requisitos, por ejemplo, usuarios, administradores o personal de soporte.
  – 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. Sociedad, ética y la Profesión (SEP)-Sustainability
• Identificación y gestión de riesgos, incluyendo consideraciones éticas en torno al producto propuesto. Sociedad, ética y la Profesión (SEP)-ProfessionalEthics
• Comunicar y/o formalizar especificaciones de requisitos.
• 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:
  – Aprender técnicas para estimar mejor el esfuerzo requerido para completar una tarea;
  – Practicar la estimación y compararla con cuánto tiempo toman las tareas;
  – 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.
	• Comparar diferentes métodos de obtención de requisitos a lo largo de múltiples ejes [Comparar]
	• 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]
	• Identificar qué comportamientos son requeridos, permitidos o prohibidos a partir de un conjunto dado de requisitos y una lista de comportamientos candidatos [Analizar]
	• Recopilar un conjunto de requisitos para un sistema de software simple [Analizar]
	• Identificar áreas de un sistema de software que deben cambiarse, dada una descripción del sistema y un conjunto de nuevos requisitos a implementar [Analizar]
	• Identificar los requisitos funcionales y no funcionales en un conjunto de requisitos [Analizar]
	• Estimar el tiempo para completar un conjunto de tareas, luego comparar las estimaciones con el tiempo real tomado [Estimar]
	• 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]
	• Escribir una especificación de requisitos para un sistema de software simple [Escribir]

Lecturas : [BB19]

Unidad 22: Aprendizaje Automático I: Fundamentos (1 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Definición y ejemplos de una amplia variedad de tareas de aprendizaje automático: – Aprendizaje supervisado: ∗ Clasificación ∗ Regresión – Aprendizaje por refuerzo – Aprendizaje no supervisado: ∗ Agrupamiento • Ideas fundamentales: – Teorema de no hay almuerzo gratis: ningún aprendiz puede resolver todos los problemas; las decisiones de diseño de representación tienen consecuencias. – Fuentes de error e indecidibilidad en el aprendizaje automático • Un aprendizaje supervisado simple basado en estadísticas como regresión lineal o árboles de decisión: – Enfocarse en cómo funcionan sin entrar en detalles matemáticos o de optimización; suficiente para entender y usar implementaciones existentes correctamente	• 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] • Diferenciar los términos: IA, aprendizaje automático y aprendizaje profundo [Evaluar] • Explicar cómo funciona el aprendizaje automático como un proceso de optimización/búsqueda [Explicar]
Lecturas : [BB19]	

Unidad 23: Aprendizaje Automático III: Redes Neuronales y Ética (1 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Redes neuronales básicas: – Fundamentos de comprensión de cómo funcionan las redes neuronales y su proceso de entrenamiento, sin detalles de los cálculos – Introducción básica a redes neuronales generativas (por ejemplo, modelos de lenguaje grandes) • ética para el Aprendizaje Automático: – Enfocarse en datos reales, escenarios reales y estudios de caso – Sesgo de conjunto de datos/algorítmico/de evaluación y consecuencias no deseadas • Aprendizaje profundo: – Redes feed-forward profundas (solo intuición, sin matemáticas) – Redes neuronales convolucionales (solo intuición, sin matemáticas) – Visualización de representaciones de características aprendidas de redes profundas – Otras arquitecturas (NN generativas, NN recurrentes, transformers, etc.) • ética para el Aprendizaje Automático: – Continuar enfocándose en datos reales, escenarios reales y estudios de caso – Privacidad – Equidad – Propiedad intelectual – Explicabilidad	• Describir el proceso de entrenamiento de redes neuronales y las representaciones aprendidas resultantes [Explicar] • 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] • 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]
Lecturas : [BB19]	

Unidad 24: Conceptos Fundamentales de Gráficos y Técnicas Interactivas (2 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• 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	• Explicar las etapas básicas del pipeline de gráficos por computadora [Explicar] • Describir técnicas básicas de renderizado como rasterización y trazado de rayos [Describir] • Aplicar teoría del color para diseñar gráficos visualmente efectivos [Aplicar] • Implementar transformaciones de coordenadas para gráficos 2D y 3D [Implementar] • Diseñar aplicaciones interactivas básicas utilizando dispositivos de entrada comunes y manejo de eventos [Diseñar] • Comparar técnicas avanzadas de renderizado como iluminación global y renderizado basado en física [Comparar] • Analizar el rol del hardware gráfico en acelerar tareas de renderizado [Analizar] • Explicar cómo la percepción humana influye en el diseño de gráficos e interacción [Explicar]
Lecturas : [BB19]	

Unidad 25: Renderizado Aplicado y Técnicas (2 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Trazado de rayos (*ray tracing*): estructuras de aceleración, muestreo (*sampling*), sombreado • Mapeo de texturas (*texture mapping*) y filtrado	• Implementar algoritmos de rasterización para renderizado de primitivas [Implementar] • Implementar trazado de rayos básico con estructuras de aceleración [Implementar]
Lecturas : [BB19]	

Unidad 26: Clase de cierre: ¿Cómo funciona un motor de búsqueda como Google? (2 horas)	
Resultados esperados: AG-C02,AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Análisis del problema. • El índice no crece linealmente con el tamaño de la información indexada. • El tiempo de respuesta no depende del tamaño de la ”base de datos”. • El tiempo de respuesta no depende del número de ocurrencias encontradas. • Combinar varias estructuras de datos para llegar a una solución. • Analizar la escalabilidad de la solución.	• Comprender los principios bajo los cuales se crea un motor de búsqueda [Usar] • Aplicar correctamente estructuras de datos para resolver el problema [Usar] • Aplicar conceptos relacionados con la complejidad algorítmica en un motor de búsqueda [Usar].
Lecturas : [BB19]	

8. PLAN DE TRABAJO

8.1 Metodología

Se fomenta la participación individual y en equipo para exponer sus ideas, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso.

8.2 Sesiones Teóricas

Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

8.3 Sesiones Prácticas

Las sesiones prácticas se llevan en clase donde se desarrollan una serie de ejercicios y/o conceptos prácticos mediante planteamiento de problemas, la resolución de problemas, ejercicios puntuales y/o en contextos aplicativos.

9. SISTEMA DE EVALUACIÓN

********* EVALUATION MISSING ********

10. BIBLIOGRAFÍA BÁSICA

[BB19] J. Glenn Brookshear and Dennis Brylow. *Computer Science: An Overview*. Global Edition. Pearson, 2019. URL: http://www.pearsonhighered.com/brookshear.