1. Datos Generales de la asignatura

Nombre de la asignatura:	Paradigmas de ingeniería de software
Clave de la asignatura:	IAD-2421
SATCA[1]:	2-3-5
Carrera:	Ingeniería en Inteligencia Artificial

2. Presentación

Caracterización de la asignatura

Aportación al perfil de egreso:
Capacita a los estudiantes para diseñar, desarrollar y mantener sistemas de software, contribuyendo al perfil de egreso al satisfacer la demanda de profesionales con habilidades técnicas y analíticas en el campo de la tecnología de la información.

Fomenta la capacidad de trabajo en equipo y el liderazgo al participar en proyectos de desarrollo de software, lo que fortalece las habilidades de comunicación, colaboración y gestión de proyectos.
Promueve la innovación y la mejora continua al aplicar metodologías y prácticas de ingeniería de software en la resolución de problemas complejos y en la creación de soluciones tecnológicas efectivas.

Contribuye al compromiso ético y social al enfatizar la importancia de la calidad, la seguridad y la ética en el desarrollo de software, formando profesionales responsables y conscientes de su impacto en la sociedad.

Importancia de la Asignatura:
La Ingeniería de Software es crucial en la era digital actual, donde la demanda de sistemas de software confiables, seguros y eficientes está en constante crecimiento. Esta asignatura prepara a los estudiantes para enfrentar los desafíos de la industria de la tecnología de la información y contribuir al desarrollo de soluciones innovadoras y de alta calidad.

Contenido de la Asignatura:
La asignatura aborda temas fundamentales y avanzados en el proceso de desarrollo de software, incluyendo análisis de requisitos, diseño de sistemas, programación, pruebas, mantenimiento y gestión de proyectos. Se enfoca en metodologías ágiles, modelos de ciclo de vida de desarrollo de software y herramientas de gestión de proyectos, así como en la aplicación de buenas prácticas de ingeniería de software.

Relación con Otras Asignaturas:
Se relaciona con "Fundamentos de Programación" y "Programación Orientada a Objetos" al enseñar los principios básicos de la programación y la construcción de software.

[1] Sistema de Asignación y Transferencia de Créditos Académicos

Tiene vínculos con "Bases de Datos" y "Arquitectura de Computadoras" al abordar la integración de sistemas y la gestión de datos en el desarrollo de software.

Complementa a "Gestión de Proyectos de Software" en la planificación y ejecución de proyectos de desarrollo de software.

Relacionada con "Fundamentos de Investigación" y "Taller de Investigación" al fomentar la investigación en áreas de ingeniería de software y la aplicación de nuevos enfoques y tecnologías.

Intención didáctica

La asignatura de Ingeniería de Software será abordada de manera teórico-práctica, integrando conceptos fundamentales con aplicaciones prácticas en el desarrollo de software. Se utilizarán ejemplos reales de proyectos de software y estudios de caso para ilustrar los conceptos teóricos y facilitar su comprensión.

Enfoque:
El enfoque principal será centrado en el desarrollo de competencias prácticas y habilidades técnicas necesarias para el diseño, desarrollo, y mantenimiento de sistemas de software. Se aplicarán metodologías ágiles y buenas prácticas de ingeniería de software para promover la calidad, la eficiencia y la innovación en el proceso de desarrollo.

Extensión y Profundidad de los Contenidos:
Los contenidos se abordarán de manera exhaustiva, cubriendo desde los conceptos básicos de ingeniería de software hasta temas más avanzados como arquitectura de software, gestión de requisitos, pruebas de software y gestión de configuración. Se profundizará en cada tema mediante ejercicios prácticos, estudios de casos y proyectos reales.

Actividades del Estudiante para el Desarrollo de Competencias Genéricas:
Se fomentará la participación activa de los estudiantes en actividades como análisis de casos, resolución de problemas, trabajos en equipo, presentaciones y debates. Estas actividades permitirán desarrollar competencias genéricas como el pensamiento crítico, la comunicación efectiva, el trabajo en equipo y el liderazgo.

Competencias Genéricas Desarrolladas:
Pensamiento Crítico: Al analizar y evaluar diferentes enfoques y soluciones en el desarrollo de software.

Comunicación Efectiva: Al presentar ideas, argumentar puntos de vista y colaborar con otros en proyectos de equipo.

Trabajo en Equipo: Al colaborar con otros estudiantes en la planificación, diseño y desarrollo de proyectos de software.
Liderazgo: Al asumir roles de liderazgo en proyectos de equipo y guiar el proceso de desarrollo de software.

Papel del Docente: El docente jugará un papel activo como facilitador del aprendizaje, proporcionando orientación, retroalimentación y apoyo a los estudiantes a lo largo del curso. Además de impartir conocimientos teóricos, el docente fomentará el pensamiento crítico y la resolución de problemas mediante el diseño de actividades y proyectos desafiantes. También promoverá un ambiente de aprendizaje colaborativo donde los estudiantes puedan compartir conocimientos y experiencias entre ellos.

3. Participantes en el diseño y seguimiento curricular del programa

Lugar y fecha de elaboración o revisión	**Participantes**	**Observaciones**
Tecnológico Nacional de México del 4 al 06 de marzo del 2024.	Representantes de los Institutos Tecnológicos de: Celaya, Chihuahua, Iztapalapa III, La Paz, Matehuala, Mérida, Minatitlán, Querétaro, Saltillo, Tijuana. Institutos Tecnológico Superior de Teziutlán. Tecnológico de Estudios Superiores de Ixtapaluca.	Propuesta sintética de la carrera de Ingeniería en Inteligencia Artificial.
Tecnológico Nacional de México del 22 al 26 de abril del 2024	Representantes de los Institutos Tecnológicos de: Celaya, Chihuahua, Iztapalapa III, La Paz, Matehuala, Mérida, Minatitlán, Querétaro, Saltillo, Tijuana. Institutos Tecnológico Superior de Teziutlán, Tecnológico de Estudios Superiores de Ixtapaluca.	Diseño y/o desarrollo curricular de la carrera de Ingeniería en Inteligencia Artificial.
Tecnológico Nacional de México del 27 al 31 de mayo del 2024.	Representantes de los Institutos Tecnológicos de: Celaya, La Paz, Matehuala, Mérida, Minatitlán.	Consolidación curricular de la carrera de Ingeniería en Inteligencia Artificial.

4. Competencia(s) a desarrollar

Competencia(s) específica(s) de la asignatura
Analizar: El estudiante deberá ser capaz de analizar los principios, metodologías y técnicas de ingeniería de software para comprender la naturaleza y los procesos involucrados en el desarrollo de sistemas de software. Diseñar: El estudiante debe ser capaz de diseñar soluciones de software utilizando métodos y herramientas adecuadas, considerando requisitos funcionales y no funcionales, arquitectura de software y buenas prácticas de diseño. Implementar: El estudiante debe ser capaz de implementar sistemas de software utilizando lenguajes de programación, frameworks y tecnologías relevantes, siguiendo estándares de codificación y mejores prácticas de desarrollo. Evaluar: El estudiante debe ser capaz de evaluar la calidad, el rendimiento y la eficacia de los sistemas de software, aplicando técnicas de prueba, depuración y análisis para identificar y corregir posibles fallos o mejoras. Gestionar: El estudiante debe ser capaz de gestionar proyectos de software, incluyendo la planificación, la asignación de recursos, el seguimiento del progreso y la gestión de riesgos, para garantizar la entrega exitosa y oportuna de los productos de software.

5. Competencias previas

Aplicar: El estudiante debe ser capaz de aplicar los fundamentos de programación y algoritmia para resolver problemas computacionales de manera eficiente y estructurada. Utilizar: El estudiante debe ser capaz de utilizar técnicas de análisis y diseño de sistemas para comprender y especificar requisitos de software de manera clara y precisa. Demostrar: El estudiante debe demostrar habilidades en el uso de herramientas y tecnologías de desarrollo de software, así como comprender los principios básicos de arquitectura de computadoras y sistemas operativos. Comprender: El estudiante debe comprender los conceptos de matemáticas discretas, álgebra lineal y estadística para aplicarlos en la resolución de problemas relacionados con la ingeniería de software. Aplicar: El estudiante debe aplicar conocimientos de gestión de proyectos y habilidades de comunicación para trabajar de manera efectiva en equipos multidisciplinarios y gestionar proyectos de software de manera exitosa.

6. Temario

No.	Temas	Subtemas
1	Fundamentos de ingeniería de software moderna	1.1. Evolución de la ingeniería de software: de los enfoques tradicionales a los métodos ágiles. 1.2. Principios y valores ágiles: Manifiesto Ágil y Principios del Desarrollo Ágil. 1.3. Enfoques modernos de desarrollo de software: Scrum, Kanban, XP, DevOps. 1.4. Herramientas y prácticas para la colaboración y gestión de proyectos: Git, CI/CD, JIRA, Trello. 1.5. Integración de DevOps y prácticas de entrega continua.
2	Desarrollo ágil y prácticas de ingeniería de software	2.1. Desarrollo de software basado en pruebas (TDD) y desarrollo dirigido por comportamiento (BDD). 2.2. Diseño ágil y arquitecturas emergentes. 2.3. Refactoring y mejora continua del código. 2.4. Automatización de pruebas y control de calidad. 2.5. Gestión de la deuda técnica y la calidad del código.
3	Desarrollo de software escalable y distribuido	3.1. Arquitecturas de microservicios y contenedores. 3.2. Diseño de sistemas distribuidos resilientes y tolerantes a fallos. 3.3. Escalabilidad y rendimiento en sistemas distribuidos. 3.4. Seguridad en sistemas distribuidos y API. 3.5. Herramientas y plataformas para el desarrollo de software escalable. (Como Docker, Kubernetes, AWS, Azure,GCP).
4	Innovaciones y tendencias en ingeniería de software	4.1. Inteligencia artificial y aprendizaje automático en desarrollo de software. 4.2. Desarrollo impulsado por datos y análisis de datos en el ciclo de vida del software. 4.3. Computación cuántica y su impacto potencial en la ingeniería de software. 4.4. Desarrollo ético de software y consideraciones sociales.

		4.5. Exploración de tendencias emergentes y futuras en la ingeniería de software.

7. Actividades de aprendizaje de los temas

1. Fundamentos de ingeniería de software moderna	
Competencias	Actividades de aprendizaje
Específica(s): • Diseña y aplica metodologías ágiles para el desarrollo de software. • Utiliza herramientas de gestión de proyectos ágiles como JIRA o Trello. • Comprende y aplica los principios del Manifiesto Ágil en proyectos reales. *Genéricas:* • Comunicación efectiva en equipos ágiles. • Trabajo en equipo y colaboración en un entorno ágil. • Pensamiento crítico para evaluar y adaptar prácticas ágiles a situaciones específicas.	• Participación en un proyecto práctico utilizando metodologías ágiles como Scrum o Kanban. • Elaboración de un plan de proyecto ágil utilizando herramientas como JIRA. • Presentación y discusión de casos de estudio sobre implementaciones exitosas de metodologías ágiles en la industria.
2. Desarrollo ágil y prácticas de ingeniería de software	
Competencias	Actividades de aprendizaje
Específica(s): • Aplica técnicas de desarrollo dirigido por pruebas (TDD) y desarrollo dirigido por comportamiento (BDD). • Identifica y remediar deuda técnica utilizando técnicas de refactorización. • Automatiza pruebas y procesos de integración continua. *Genéricas:* • Pensamiento analítico para identificar y resolver problemas de código. • Creatividad y adaptabilidad en la implementación de prácticas de desarrollo ágil.	• Desarrollo de pequeños proyectos utilizando TDD y BDD para garantizar la calidad del código. • Sesiones prácticas de refactorización de código para mejorar su mantenibilidad y legibilidad. • Configuración y ejecución de pruebas automatizadas utilizando herramientas como JUnit o Selenium.

• Gestión del tiempo y organización para cumplir con los objetivos de desarrollo en ciclos cortos.	
3. Desarrollo de software escalable y distribuido	
Competencias	Actividades de aprendizaje
Específica(s): • Diseña arquitecturas de microservicios para sistemas distribuidos. • Implementar estrategias de escalabilidad y tolerancia a fallos en sistemas distribuidos. • Utiliza herramientas de contenerización como Docker para desplegar aplicaciones escalables. *Genéricas:* • Resolución de problemas en entornos distribuidos y complejos. • Colaboración efectiva en equipos de desarrollo distribuido. • Gestión del cambio y adaptabilidad en entornos de desarrollo escalables.	• Diseño y desarrollo de un sistema distribuido utilizando arquitecturas de microservicios. • Simulación de escenarios de fallo y recuperación en sistemas distribuidos. • Despliegue y monitoreo de aplicaciones en contenedores utilizando Docker y Kubernetes. • como sugerencia usar aplicaciones como:Herramientas y plataformas para el desarrollo de software escalable: Docker, Kubernetes, AWS, Azure o en su defecto herramientas similares
4. Innovaciones y tendencias en ingeniería de software	
Competencias	Actividades de aprendizaje
Específica(s): • Aplica técnicas de inteligencia artificial y aprendizaje automático en el desarrollo de software. • Analiza y utilizar grandes volúmenes de datos en el ciclo de vida del software. • Explorar nuevas tendencias y tecnologías emergentes en la ingeniería de software. *Genéricas:* • Habilidades de investigación y análisis de nuevas tecnologías. • Creatividad y pensamiento innovador en la aplicación de tecnologías emergentes.	• Desarrollo de un proyecto de inteligencia artificial o aprendizaje automático aplicado a un problema de software. • Análisis de conjuntos de datos y aplicación de técnicas de análisis de datos en el desarrollo de software. • Investigación y presentación de nuevas tendencias y tecnologías emergentes en la ingeniería de software.

• Conciencia ética y social en el desarrollo y aplicación de tecnologías de software.	

8. Práctica(s)

Práctica 1: Configuración de un proyecto ágil
Configura un proyecto utilizando una herramienta de gestión ágil como JIRA o Trello.
Define las historias de usuario y tareas iniciales del proyecto.

Práctica 2: Sprint de desarrollo ágil
Trabaja en un sprint de desarrollo ágil para implementar funcionalidades definidas en la práctica anterior.
Utiliza reuniones diarias de stand-up y técnicas ágiles para gestionar el progreso.

Práctica 3: Desarrollo Dirigido por Pruebas (TDD)
Implementa una funcionalidad utilizando la metodología TDD.
Escribe pruebas automatizadas antes de escribir el código de la funcionalidad.

Práctica 4: Refactorización y mejora continua
Identifica y refactoriza partes del código para mejorar su mantenibilidad y legibilidad.
Utiliza herramientas como SonarQube o Checkstyle para analizar la calidad del código.

Práctica 5: Diseño de arquitectura de microservicios
Diseña la arquitectura de un sistema utilizando el enfoque de microservicios.
Define los límites de los servicios y las interfaces de comunicación.

Práctica 6: Implementación de escalabilidad y tolerancia a fallos
Implementa estrategias de escalabilidad y tolerancia a fallos en un sistema distribuido.
Utiliza técnicas como el circuit breaker pattern y la replicación de datos.

Práctica 7: Aplicación de inteligencia artificial
Desarrolla un modelo de aprendizaje automático para resolver un problema de software específico.
Utiliza bibliotecas como TensorFlow o scikit-learn para entrenar y evaluar el modelo.

Práctica 8: Investigación de tendencias emergentes
Investiga y presenta una nueva tendencia o tecnología emergente en el campo de la ingeniería de software.
Analiza su relevancia y aplicabilidad en proyectos futuros de desarrollo de software.

9. Proyecto de asignatura

Proyecto de asignatura: Desarrollo de una aplicación Web de gestión de proyectos

Fundamentación:
En esta fase, los estudiantes realizarán un análisis teórico y conceptual sobre la gestión de proyectos de software y las metodologías ágiles. Se revisarán diferentes marcos de trabajo como Scrum, Kanban o Xtreme Programming, así como conceptos fundamentales de ingeniería de software. Además, se analizarán casos de estudio y tendencias actuales en el desarrollo de aplicaciones web.

Planeación:
Basándose en el análisis realizado en la fase de fundamentación, los estudiantes diseñarán un proyecto para desarrollar una aplicación web de gestión de proyectos utilizando metodologías ágiles. Se establecerán los objetivos del proyecto, el alcance, los requisitos funcionales y no funcionales, así como el plan de trabajo detallado, incluyendo las iteraciones y las entregas parciales.

Ejecución:
Durante esta fase, los estudiantes llevarán a cabo el desarrollo de la aplicación web de gestión de proyectos de acuerdo con el plan establecido en la fase de planeación. Se pondrán en práctica las metodologías ágiles aprendidas en la asignatura, realizando iteraciones cortas y entregas incrementales. Se trabajará en equipo para implementar las funcionalidades de la aplicación, realizar pruebas y asegurar la calidad del software desarrollado.

Evaluación:
Al finalizar la ejecución del proyecto, se llevará a cabo una evaluación integral del trabajo realizado. Se analizarán los resultados obtenidos en comparación con los objetivos establecidos y se identificarán los logros alcanzados y los aspectos a mejorar. Además, se fomentará la reflexión sobre el proceso de desarrollo del proyecto, promoviendo la mejora continua y el desarrollo del pensamiento crítico en los estudiantes. Se valorará tanto el producto final desarrollado como el proceso seguido para su elaboración.

10. Evaluación por competencias

Competencia 1: Diseño de Software Escalable (25%)

- Examen Teórico (10%)
- Evaluación escrita sobre los principios y conceptos de diseño de software escalable.
- Diseño de la Arquitectura del Sistema (10%)
- Rúbrica de Práctica detallada como se presentó anteriormente.
- Participación en Clases (5%)
- Contribución activa en discusiones, preguntas y debates en clase relacionados con el diseño de software escalable.

Competencia 2: Desarrollo Ágil de Software (30%)

- Examen Práctico (15%)
- Evaluación práctica en la que los estudiantes resuelven problemas relacionados con el desarrollo ágil de software.
- Implementación de Scrum en un Proyecto (10%)
- Rúbrica de Práctica detallada como se presentó anteriormente.
- Participación en Clases (5%)
- Contribución activa en discusiones, ejercicios y simulaciones de prácticas ágiles en clase.

Competencia 3: Gestión de Proyectos de Software (20%)

- Examen Escrito (10%)
- Evaluación escrita sobre los conceptos y metodologías de gestión de proyectos de software.
- Planificación de Proyecto con Gantt (5%)
- Rúbrica de Práctica detallada como se presentó anteriormente.
- Participación en Clases (5%)
- Participación en ejercicios de planificación y discusiones sobre gestión de proyectos en clase.

Competencia 4: Utilización de Tecnologías Emergentes (25%)

- Examen Mixto (15%)
- Evaluación que incluye preguntas teóricas sobre tecnologías emergentes y problemas prácticos de implementación.
- Integración de Tecnologías en un Proyecto (10%)
- Rúbrica de Práctica detallada como se presentó anteriormente.
- Participación en Clases (5%)
- Participación activa en discusiones y presentaciones sobre tecnologías emergentes.
- Evaluación Integral (10%)
- Evaluación Integral del Desempeño (10%)
- Evaluación global que considera el desempeño del estudiante en todas las actividades, su progreso a lo largo del curso y su capacidad para aplicar los conocimientos en situaciones prácticas.

11. Fuentes de información

1. Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill Education.
2. Sommerville, I. (2016). Software Engineering (10th ed.). Pearson Education Limited.
3. Sommerville, I. (2011). Software Engineering (9th ed.). Addison-Wesley.
4. Pfleeger, S. L., & Atlee, J. M. (2015). Software Engineering: Theory and Practice (4th ed.). Pearson.
5. Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice (3rd ed.). Addison-Wesley.