4131 - INGENIERÍA DE SOFTWARE

I - Datos de identificación de la asignatura

Carrera:	Licenciatura en Análisis de Sistemas		
Código:	4131	**Plan:**	2024
Denominación:	Ingeniería de Software		
Área:	Tecnologías aplicadas		
Año:	Tercer		
Horas con acompañamiento docente (HTD), semanal			4
Horas de Trabajo Independiente del estudiante (HTI), semanal			5
Horas semanales (HS)			9
Cantidad de sesiones			32
Total Horas de Trabajo con el docente (THTD)			128
THD teóricas	128	**THD prácticas**	0
Total de Horas de Trabajo Independiente del estudiante (THTI)			160
Total Horas Académicas (THA)			288
Crédito académico (CA)			11,5
Pre-requisito:	Fundamentos del desarrollo de software		

II - Fundamentación

En esta asignatura se abordan las actividades técnicas y de ingeniería que se desarrollan a lo largo del ciclo de vida de un producto de software. Se exploran los problemas, métodos y tecnologías asociadas a la Ingeniería de Software, centrándose específicamente en el desarrollo de proyectos de software y haciendo hincapié en la gestión y control de calidad.

El objetivo principal de la asignatura es proporcionar a los estudiantes una comprensión integral de los conceptos fundamentales de la Ingeniería de Software, así como las habilidades prácticas necesarias para aplicar esos conceptos en la práctica. Se busca desarrollar en los estudiantes las competencias necesarias para abordar eficazmente los desafíos y complejidades inherentes al desarrollo de proyectos de software.

La naturaleza de la asignatura es teórico-práctica, lo que significa que se combina la adquisición de conocimientos teóricos con la aplicación práctica de esos conocimientos a través de ejercicios y proyectos. Se enfatiza la importancia de la resolución de problemas y la toma de decisiones informadas en el contexto de la ingeniería de software.

III - Competencias a desarrollar

Competencias genéricas

1. Liderar proyectos en el campo de la informática.
2. Integrar equipos multidisciplinarios y realizar trabajos colaborativos.
3. Analizar, delinear, gestionar, desarrollar, implementar y evaluar proyectos con innovación y creatividad.
4. Formular, gestionar, participar y ejecutar proyectos.
5. Identificar, analizar, abstraer, formular y resolver problemas relacionados con sus áreas de competencia.

Competencias específicas

1. Comprender integralmente las etapas del ciclo de vida del desarrollo de sistemas, desde la identificación de requisitos hasta el mantenimiento, para aplicar un enfoque sistemático y ordenado que garantice la calidad del software.
2. Aplicar de forma estructurada una metodología de desarrollo de software reconocida, siguiendo sus fases, actividades y artefactos, para asegurar una construcción eficiente y coherente de sistemas que respondan a los objetivos del proyecto.
3. Identificar y comprender claramente los roles del equipo de desarrollo (analista, diseñador, programador, tester, gestor), para favorecer la colaboración efectiva y la comprensión del proceso de desarrollo de software.
4. Utilizar con eficiencia herramientas de gestión de proyectos y seguimiento de tareas, para planificar actividades, asignar recursos, establecer hitos y controlar el progreso del proyecto, mejorando la organización del equipo de desarrollo.
5. Utilizar con criterio técnico plataformas colaborativas y sistemas de control de versiones, para gestionar código, tareas y documentos compartidos, promoviendo la comunicación y coordinación dentro del equipo de desarrollo.
6. Aplicar de manera efectiva herramientas de seguimiento de incidencias y solicitudes de cambio, para interactuar con clientes y partes interesadas, facilitando la gestión de requerimientos y la toma de decisiones durante el desarrollo.
7. Utilizar apropiadamente herramientas y marcos de pruebas de software (unitarias, integración, aceptación), para asegurar la calidad y el correcto funcionamiento del sistema desarrollado.
8. Aplicar con precisión técnicas de modelado utilizando UML, diagramas ERD y de clases, para representar gráficamente la estructura y el comportamiento del sistema, favoreciendo el diseño, análisis y documentación del software.
9. Comprender y aplicar con responsabilidad los procesos de entrega y mantenimiento de software, incluyendo planificación de implementación, documentación y soporte continuo, para garantizar la sostenibilidad del sistema en su ciclo de vida.
10. Aplicar prácticas y estándares de calidad en el desarrollo de software, evaluando la calidad del código, el proceso y la satisfacción del cliente, para mejorar continuamente el producto y asegurar su valor funcional.

IV - Cuerpo de conocimientos

Unidad 1: Ciclo de desarrollo de sistemas (SDLC)
Contenidos:

- Fases del SDLC. Resumen de las principales preocupaciones del desarrollo de sistemas de software.
- La naturaleza secuencial implícita del SDLC, en términos de predicados y flujo.
- Variedad de modelos que describen el flujo y la evolución del SDLC en la práctica.

Unidad 2: Metodología de desarrollo de sistemas/software
Contenidos:

- Modelo Waterfall. Conceptos, los supuestos, ventajas y desventajas.

- Modelo iterativo en cascada como un conjunto de procesos secuenciales en cascada. Conceptos, supuestos, ventajas y desventajas.
- Modelo Espiral como una abstracción por fases de los modelos Iterativo y Cascada. Conceptos, supuestos, ventajas y desventajas.
- Modelo Agile como un enfoque iterativo e incremental que utiliza la validación empírica frecuente y la colaboración del cliente. Conceptos, supuestos, ventajas y desventajas.

Unidad 3: El rol y las responsabilidades de los participantes en el SDLC
Contenidos:

- Funciones y responsabilidades
- Plan de gestión de partes interesadas
- Ingeniería de requerimientos

Unidad 4: Herramientas para la gestión de procesos de software
Contenidos:

- Herramientas que rastrean y administran proyectos de software con respecto a recursos, características, plazos, conflictos e impedimentos.
- Estimación, validación y velocidad en lo que respecta a la productividad.
- Herramientas: Jira, Azure DevOps para equipos, TeamCity, etc.

Unidad 5: Herramientas para la colaboración y comunicación en equipo
Contenidos:

- Comunicación en tiempo real que también está vinculada a la gestión de funciones, la gestión de códigos y el seguimiento de defectos
- Enlaces a confirmaciones de control de versiones y desarrollo de código colaborativo
- Exposición de al menos una herramienta, como Jira, Trello, TeamCity, BandCamp, etc.

Unidad 6: Herramientas para la colaboración y comunicación con el cliente
Contenidos:

- Seguimiento de las pruebas de aceptación interactivas con el usuario
- Características, personas, casos de uso o historias de usuarios en relación con los módulos del sistema, los flujos de información y los flujos de experiencia del usuario.
- Disposiciones para implementaciones de prueba y validación.

Unidad 7: Herramientas para pruebas (unidad, integración, aceptación)
Contenidos:

- Sistema para desarrolladores y usuarios finales para identificar defectos de software.
- Sistema para desarrolladores y usuarios finales para rastrear el estado de los defectos.
- Integración con sistemas de control de versiones y gestión de funciones.
- Una "pirámide" de pruebas vinculada e integradora que va en cascada desde las pruebas manuales hasta las pruebas de integración y las pruebas unitarias.
- Pruebas de regresión para protegerse contra la regresión de características y funciones
- Uso y familiaridad con al menos una biblioteca o arnés de pruebas unitarias.

Unidad 8: UML, ERD y diseño de clase/objeto
Contenidos:

- Diagramas estructurales UML para realizar y validar el diseño de bibliotecas y módulos orientados a objetos
- Diagramas de comportamiento UML para realizar y validar el diseño lógico

Unidad 9: Principios de entrega y mantenimiento de sistemas
Contenidos:

- Estrategias de Integración Continua y Entrega Continua
- Desarrollo basado en pruebas
- Entornos de desarrollo por etapas
- Plataforma e Infraestructura como Servicio

Unidad 10: Calidad de sistemas

Contenidos:

- Conceptos de Calidad y Calidad Total.
- Calidad del proceso y del producto.
- Aseguramiento y estándares de calidad.
- Planeamiento de la calidad.
- Control de la calidad.
- Revisiones Técnicas formales. Inspecciones.
- Modelos de madurez para las organizaciones de desarrollo de software (CMM-CMMI).
- Normas ISO 9000.

V - Estrategias didácticas a ser implementadas en el proceso de enseñanza aprendizaje. (abarcando actividades de formación e investigación)

El curso consta de clases teóricas, explicaciones de práctica y clases prácticas.

Las explicaciones de práctica apuntan a brindar las herramientas necesarias para la realización de los trabajos prácticos.

Las actividades prácticas comprenden la realización de ejercicios de modelización de requerimientos aplicando diferentes herramientas explicadas en la teoría.

En la cátedra se pone énfasis en el proceso de identificación de problemas del mundo real, especificación de los mismos como problemas resolubles desde la informática y en el desarrollo de soluciones verificables para los mismos utilizando las técnicas de especificación de requerimientos establecidas en el programa de la materia

Los ejercicios de la práctica son evaluados teniendo en cuenta el contenido técnico, pero también la estructura, organización, sintaxis, claridad conceptual.

Además, los alumnos deben realizar una actividad grupal de elicitacion de requerimientos que es evaluada de manera oral por el ayudante a cargo.

Todas las evaluaciones orales realizadas se reflejan en planillas escritas que conforman documentación de evaluación del trabajo.

VI - Estrategias de evaluación.

La evaluación será formativa y procesual, se realizará a través de pruebas (exámenes) que podrán ser escritas, orales o de ejecución que a su vez podrá ser mediante trabajos individuales o grupales. La materia consta de dos pruebas parciales, con un recuperatorio y tres oportunidades para la prueba final.

En estos parciales, así como en el examen final, se evaluarán las competencias alcanzadas a través de actividades de contenido teórico y práctico que permitan dar cuenta del avance conceptual en los temas que se han desarrollado, se incorporan preguntas específicas tipo sobre “donde cree Ud. que es aplicable este conocimiento/método” y se refleja en la corrección de las pruebas del alumno.

En algunos temas se trabaja también con ejercitaciones de aplicación en clase, que requieren de un ejercicio de integración de conceptos y que complementan la evaluación a través de los parciales.

Para la obtención de calificaciones parciales y finales se tendrá en cuenta el Reglamento Académico de la universidad.

VII - Actividades de extensión y de responsabilidad social universitaria.

Rige de acuerdo al reglamento de la Universidad y el reglamento interno de la facultad.

VIII - Fuentes bibliográficas

Básica

- Ingeniería de Software. 9na Edición. Ian Sommerville. Pearson. 2011.
- Software Engineering: Theory and Practice. 4th Edition. Shari Pfleeger. Prentice Hall. 2009. (Edición en castellano: Ingeniería de Software. Teoría y Práctica. Shari Pfleeger. Pearson Education. 2002)
- Ingeniería de Software. Un enfoque práctico. 7ma Edición. Roger Pressman. McGraw-Hill. 2010.
- Sistemas de Información Administrativa. Murdick R. Prentice Hall. 1988
- Systems Analysis and Design, 9/E. Kendall & Kendall. Pearson. 2013. (Edición en castellano: Análisis y diseño de sistemas. 8va Edición. Kendall & Kendall. Pearson. 2011)

Complementaria

- Coad, P., & Yourdon, E. (1991). Análisis basado en objetos. Prentice-Hall.
- Martin, J. (1992). Análisis y Diseño Orientado a Objetos. Prentice Hall.
- Yourdon, E. (1989). Análisis Estructurado Moderno. Prentice Hall.
- Structured Systems Analysis: Tools and Technics, Chris Gane/Trish Sarson