Campus de la UNA
SAN LORENZO-PARAGUAY

UNIVERSIDAD NACIONAL DE ASUNCIÓN
FACULTAD POLITÉCNICA
CONSEJO DIRECTIVO

RESOLUCIÓN 24/26/72-00
ACTA 1208/16/12/2024

"POR LA CUAL SE APRUEBA EL PROGRAMA DE ESTUDIO DE LA ASIGNATURA *INGENIERÍA DE SOFTWARE II*, DE LA CARRERA LICENCIATURA EN CIENCIAS INFORMÁTICAS – PLAN 2023 DE LA FP-UNA"

VISTO: El Memorando DA/2437/2024 del Director Académico de la FP-UNA, Prof. MSc. Felipe Santiago Uzabal Escurra, con el cual remite el Memorando CCPTCC/036/2024 de la Comisión Coordinadora del Proyecto de Transformación Curricular de Carreras de Grado de la FP-UNA, en el que presenta la propuesta de Programas de Estudio de las Asignaturas de la Carrera Licenciatura en Ciencias Informáticas.

CONSIDERANDO: La Ley 4995/2013 de Educación Superior, el Estatuto de la Universidad Nacional de Asunción y las deliberaciones sobre el tema.

Que la Comisión Coordinadora del Proyecto de Transformación Curricular de Carreras de Grado, solicita la aprobación del Programa de Estudio de la asignatura **"Ingeniería de Software II"**, de la carrera Licenciatura en Ciencias Informáticas – Plan 2023, cuyo plan de estudio ya fue aprobado por el Consejo Superior Universitario.

EL CONSEJO DIRECTIVO DE LA FACULTAD POLITÉCNICA
RESUELVE:

24/26/72-01 APROBAR el Programa de Estudio de la Asignatura **"Ingeniería de Software II"**, de la carrera Licenciatura en Ciencias Informáticas – Plan 2023 de la FP-UNA, detallado en el ANEXO 64 de la presente Acta.

24/26/72-02 COMUNICAR, copiar y archivar.

Prof. Abg. Joel Arsenio Benítez Santacruz
Secretario

Prof. Ing. Silvia Teresa Leiva León, MSc.
Presidenta

TEL/FAX: +595 21 5887000 – C.C. 1130 (Asunción) – 2111 (San Lorenzo)
www.pol.una.py

Campus de la UNA
SAN LORENZO-PARAGUAY

UNIVERSIDAD NACIONAL DE ASUNCIÓN
FACULTAD POLITÉCNICA
CONSEJO DIRECTIVO

Resolución 24/26/72-00 Acta 1208/16/12/2024
ANEXO 64

DEPARTAMENTO DE ENSEÑANZA DE INFORMÁTICA
PROGRAMA DE ESTUDIO

I. IDENTIFICACIÓN

Asignatura	Ingeniería de Software II				
Carrera	**Plan**	**Sede/Filial**	**Carácter**	**Semestre**	**Prerrequisitos**
Licenciatura en Ciencias Informáticas	2023	Sede San Lorenzo/Filial Villarrica/Filial Coronel Oviedo	Obligatoria	Quinto	Ingeniería de Software I.
Horas semanales	4				
Total de horas teóricas semestral	36				
Total de horas prácticas semestral	36				
Total de horas semestral	72				
Valor en créditos académicos	La valoración en créditos académicos será comunicada en su oportunidad ajustada al Reglamento General del Sistema de Créditos Académicos de la UNA, el cual se encuentra en proceso de elaboración conforme a las disposiciones de la Resolución CONES N° 221/2024, en su artículo N° 10.				
Actualización	Al egreso de la primera cohorte.				

II. FUNDAMENTACIÓN

En la asignatura Ingeniería de Software II se reforzará la adquisición de buenas prácticas, así como el conocimiento teórico y posterior implementación de arquitecturas, frameworks y metodologías de diseño y desarrollo, que constituirán un refuerzo en el aprendizaje de los fundamentos de desarrollo de software de alta calidad.

Con un enfoque en áreas clave como el diseño de software, desarrollo, pruebas, gestión de configuración, integración continua y la interacción con sistemas legados, esta asignatura ofrece a los estudiantes la oportunidad de profundizar en su conocimiento y habilidades en el campo de la ingeniería de software. Además, se fomentará la importancia del trabajo en equipo y la colaboración efectiva, ya que estos son componentes esenciales en la creación exitosa de software en el entorno profesional actual.

En relación a la naturaleza de la asignatura, se aborda de manera teórico-práctico, se combinarán conceptos teóricos con ejercicios prácticos. La organización de la asignatura se basa en los ejes temáticos, se incluyen conceptos fundamentales como: Diseño de software. Desarrollo de software. Pruebas de software. Gestión de la configuración del software (SCM). Ambientes de desarrollo, testing y producción. Integración continua. Flujo de trabajo basado en sistemas de versionamiento de código, distribución y backups. Integración con sistemas legados.

III. COMPETENCIAS DEL PERFIL DE EGRESO ASOCIADAS

1. Comunicarse en las lenguas oficiales del país y en una lengua extranjera.

2. Liderar y trabajar en equipo con eficacia y responsabilidad tomando decisiones basadas en evidencias.
3. Aplicar en la práctica profesional los valores humanos, la ética y los mecanismos de seguridad laboral.
4. Actuar proactivamente frente a los problemas sociales y ambientales.
5. Adaptarse respetuosamente a contextos nuevos o adversos, así como a diversidades personales, disciplinares y culturales.
6. Actualizarse permanentemente mediante la obtención y gestión autónoma de información de calidad, utilizando tecnología de la información y comunicación.
7. Evaluar el comportamiento de diversos fenómenos disciplinares e interdisciplinares relacionados con las ciencias informáticas con una visión de sistema, mediante modelos teóricos validados y actualizados, capaces de abarcarlos integralmente en un contexto de incertidumbre.
8. Seleccionar, utilizar y construir instrumentos innovadores asociados al ejercicio de las ciencias informáticas.
9. Aplicar, producir y difundir conocimientos técnicos y científicos en el área de las ciencias informáticas.
10. Planificar, proyectar, diseñar y ejecutar proyectos sostenibles e integrales para la resolución de problemas, la mejora y la innovación en el ámbito de las ciencias informáticas.
11. Interpretar, modelar y comunicar información referida a la ingeniería en informática en forma gráfica.

IV. ORGANIZACIÓN DE LA ASIGNATURA

Unidades	Contenidos	Resultados de aprendizaje
1. Diseño y Arquitectura de Software.	1.1. Principios de diseño de software (Abstracción, Cohesión, Acoplamiento, Modularidad, etc). 1.2. Arquitectura de software y sus componentes. 1.2.1. Patrones arquitectónicos (cliente/servidor, servicios, microservicios, otros). 1.2.2. Diseño de Diagramas (Clases y Objetos, Secuencias, Despliegue, etc.) 1.2.3. Patrones de Diseño (Singleton, Factory, Observer, Strategy, MVC, Composite, Prototype, Facade, Abstract Factory, Mediator, Interpreter, Iterator, otros) 1.3. Interfaz de Usuario y Experiencia de Usuario (UI/UX). 1.4. Consideraciones tecnológicas y toma de decisiones: escalabilidad, seguridad, integración, restricciones, limitaciones, documentación.	1. Aplica la abstracción, la cohesión, el acoplamiento y la modularidad en contextos reales o simulaciones. 2. Evalúa la elección y aplicación de patrones arquitectónicos en un proyecto de desarrollo de software. 3. Diseña diagramas para representar de manera efectiva la arquitectura estructural y el flujo de datos en una aplicación. 4. Selecciona y aplica patrones de diseño apropiados a situaciones de desarrollo específicas. 5. Aplica principios de diseño de experiencia de usuario e interfaces (UX/UI) para crear aplicaciones que se alineen con las necesidades y expectativas de los usuarios. 6. Analiza y toma decisiones fundamentadas en relación con las consideraciones tecnológicas en proyectos de desarrollo de software.

2. Procesos de desarrollo de software.	2.1. Comparación de modelos de desarrollo. 2.2. Ciclos de vida del software. 2.3. Selección de enfoques de desarrollo en función de los proyectos 2.4. Desarrollo ágil vs. en cascada: 2.4.1. Metodologías ágiles (Scrum, Kanban, XP). 2.4.2. Ventajas y desventajas de enfoques ágiles y en cascada. 2.5. Gestión de proyectos ágiles. 2.6. Programación avanzada y buenas prácticas: 2.7. 6.1 Diseño y codificación eficiente. 2.8. 6.2 Pruebas unitarias y depuración.	1. Analiza los ciclos de vida del software y seleccionar el más adecuado para un proyecto en particular, considerando sus requisitos y restricciones. 2. Explica las ventajas y desventajas de diferentes enfoques de desarrollo de software. 3. Selecciona entre diferentes enfoques de desarrollo en función de las necesidades del proyecto. 4. Utiliza estrategias de gestión de proyectos para coordinar equipos, asegurando la entrega oportuna de resultados mediante la planificación. 5. Aplica técnicas de programación avanzada y buenas prácticas de diseño y codificación eficiente en problemas de desarrollo para obtener productos de software de calidad.
3. Pruebas de Software	3.1. Estrategias de pruebas de software: 3.1.1. Pruebas funcionales y no funcionales. 3.1.2. Pruebas manuales vs. Pruebas automatizadas. 3.1.3. Pruebas de caja negra y caja blanca. 3.1.4. Pruebas de regresión y pruebas de aceptación. 3.1.5. Elaboración de un Plan de Prueba. 3.2. Tipos de pruebas: 3.2.1. Pruebas unitarias. 3.2.2. Pruebas de integración. 3.2.3. Pruebas de sistema. 3.2.4. Pruebas de rendimiento. 3.2.5. Redacción de Casos de Prueba 3.3. Evaluación de la calidad del software: 3.3.1. Métricas de calidad del software. 3.3.2. Normas y estándares de calidad. 3.3.3. Pruebas de usabilidad y accesibilidad. 3.3.4. Pruebas de seguridad y pruebas de penetración.	1. Analiza las necesidades de pruebas de software en función de los requisitos del proyecto y los riesgos potenciales. 2. Planifica y diseña estrategias de pruebas de acuerdo a los requisitos y la planificación del desarrollo de software. 3. Determina las pruebas necesarias para evaluar la funcionalidad y la estructura interna del software. 4. Evalúa la calidad del software mediante métricas específicas y asegurarse de cumplir con normas y estándares de calidad establecidos.

4. Gestión de Configuración del Software (SCM)	4.1. Conceptos fundamentales de SCM: 4.1.1. Definición de Gestión de Configuración del Software. 4.1.2. Importancia de la trazabilidad y control de cambios. 4.1.3. Identificación de elementos de configuración. 4.1.4. Control de versiones y herramientas. 4.2. Trazabilidad y control de cambios: 4.2.1. Registro de cambios y seguimiento de versiones. 4.2.2. Gestión de problemas y solicitudes de cambios. 4.2.3. Auditorías y revisiones de cambios.	1. Valora la importancia de la trazabilidad en el desarrollo de software y aplicarla mediante herramientas apropiadas. 2. Aplica conceptos de Gestión de Configuración del Software para mantener un registro sistemático de los elementos del software. 3. Utiliza herramientas de Control de versiones para rastrear y gestionar cambios en el código y la documentación a lo largo del ciclo de vida del software.
5. Gestión de Desarrollo, Integración Continua y Sistemas Legados	5.1. Creación de entornos de desarrollo. 5.2. Integración Continua y Despliegue Continuo. 5.3. Control de versiones y flujo de trabajo. 5.4. Integración efectiva con sistemas legados.	1. Diseña y configura entornos de desarrollo que sean óptimos para proyectos de software, considerando herramientas y recursos necesarios. 2. Identifica prácticas de Integración Continua (CI) y Despliegue Continuo (CD) para automatizar pruebas y despliegues de software de manera eficiente. 3. Realiza ajustes y mejoras en los entornos de desarrollo en función de las necesidades cambiantes de los proyectos y la infraestructura existente.

V. ESTRATEGIAS DIDÁCTICAS

En el desarrollo del programa se aplicarán estrategias didácticas conducentes a la apropiación teórica y la ejecución práctica de procesos y procedimientos, a saber:

- **Clase invertida:** con materiales didácticos dispuestos en el aula virtual previamente y aplicar en clases presenciales, analizando y respondiendo a planteamientos con estudio de casos a través de trabajos grupales, orientadas especialmente al desarrollo de métodos para la organización, tales como: la definición de la misión, la visión, la elaboración de funciones, objetivos, organigramas y planificación de trabajos colaborativos, para asimilar los contenidos de la asignatura.

- **Aprendizaje basado en proyectos:** metodología donde el estudiante participa activamente en su aprendizaje, desarrollando diferentes habilidades para solucionar un problema a través de un proyecto, y que pueda implementarse para la mejora del contexto.

- **Aprendizaje cooperativo colaborativo**: conjunto de métodos de instrucción para la aplicación en grupos pequeños, de entrenamiento y desarrollo de habilidades mixtas (aprendizaje, desarrollo personal y social) donde cada componente del grupo es responsable tanto de su aprendizaje como del de los restantes.

- **Debate:** exposición por parte del docente de los conceptos básicos por unidad, con materiales de lectura y ejemplos orientados a la enseñanza de las competencias específicas de la asignatura. El docente asume el rol de expositor y buscará generar el debate a través de preguntas sobre lo expuesto y desde la participación de los estudiantes.

La elección particular de la estrategia didáctica aplicada será explícita en el plan de clases, de acuerdo con el perfil de los estudiantes, los recursos disponibles y el contexto educativo.

VI. ESTRATEGIAS EVALUATIVAS

Procesos de producción grupales e individuales, pruebas individuales orales y/o escritas durante el desarrollo de las unidades con diálogos e interpretaciones que los estudiantes realicen sobre los contenidos, debates, retroalimentación en casos necesarios y actividades que amplíen el conocimiento, que serán valorados y que en su conjunto aportarán para la calificación y promoción, las que serán aplicadas según normativas institucionales.

VII. MEDIOS AUXILIARES

Aula virtual, pizarrón, proyector, marcadores, equipo de audio, ordenadores, wifi, celulares, plataformas de videoconferencia, salas de chats, herramientas de gamificación.

VIII. BIBLIOGRAFÍA

- Pressman, R. S. (2010). Ingeniería del software: un enfoque práctico. (7° ed.). México: McGraw-Hill.
- Sommerville, I. (2011). Ingeniería de software. (9° ed.). México: Addison Wesley.
- Institute of Electrical and Electronics Engineers Computer Society. (2014). Software Engineering Body of Knowledge (SWEBOK).
- International Software Testing Qualifications Board (2023). Certified Tester Foundation Level Syllabus v4.0.
- Sommerville, I. (2005). Ingeniería de software. (7° ed.). Madrid: Addison Wesley.