



Fecha: Febrero de 2026

PROGRAMA ACADÉMICO: INGENIERIA DE SISTEMAS Y COMPUTACIÓN

SEMESTRE: VII

ASIGNATURA: INGENIERÍA DE SOFTWARE II

CÓDIGO: 8108267

NÚMERO DE CRÉDITOS: 4

PRESENTACIÓN

Este curso, junto con Ingeniería de Requisitos e Ingeniería de Software I, ofrece una introducción amplia a los fundamentos de la Ingeniería de Software.

En Ingeniería de Software II se abordan los conocimientos posteriores a los adquiridos en asignaturas previas, que incluyen: técnicas de recolección de información, especificación, análisis y validación de requisitos, procesos de la ingeniería de software, paradigmas y metodologías de desarrollo, fundamentos de diseño, modelos de calidad, gestión de proyectos de software, modelos de persistencia orientados a bases de datos relacionales, así como su diseño. También se espera que el estudiante haya completado su ciclo de formación en programación, incluyendo modelado, buenas prácticas, principios fundamentales de diseño (encapsulamiento, cohesión, acoplamiento, DRY, SoC, Ley de Demeter, KISS, SOLID, entre otros) y patrones de diseño (creacionales, de comportamiento y estructurales).

En este curso se trabajará en temas de arquitectura y construcción de software, así como en tendencias actuales, recomendando a los estudiantes profundizar en ellas posteriormente.

JUSTIFICACIÓN

Los ingenieros de sistemas y computación de la UPTC deben ser capaces de aplicar sus conocimientos para realizar tareas y resolver problemas en diversas áreas de las Tecnologías de la Información y las Comunicaciones. Por lo tanto, es fundamental formar profesionales competentes en ingeniería de software y en el manejo de tecnologías actuales. Además, deberán desempeñarse eficazmente en la sociedad del conocimiento, en la que el aprendizaje continuo a lo largo de la vida profesional es una necesidad ineludible.

Las principales razones para ofrecer este curso son las siguientes:

- ✓ Relevancia en el mercado laboral: La industria tecnológica experimenta un rápido crecimiento y demanda ingenieros de software en múltiples sectores, tales como empresas de desarrollo, consultorías de TI, compañías de tecnología emergente y organizaciones que requieren soluciones tecnológicas avanzadas.
- ✓ Adaptación a las necesidades de la sociedad: La transformación digital permea todos los aspectos de la vida cotidiana, desde la educación hasta los servicios públicos. Los ingenieros de software cumplen un papel clave en el diseño, desarrollo y mantenimiento de sistemas y aplicaciones que impulsan esta transformación.
- ✓ Desarrollo de habilidades fundamentales: El curso permite comprender los principios de diseño arquitectónico de software y capacita para analizar y resolver problemas complejos.
- ✓ Fomento del pensamiento crítico y la creatividad: A través de desafíos de diseño, se promueve la innovación y la resolución efectiva de problemas desde diversas perspectivas.
- ✓ Integración teoría-práctica: Combina fundamentos teóricos sólidos con proyectos prácticos que preparan para los retos del mundo laboral.

- ✓ Oportunidad para investigación y desarrollo: Proporciona una base sólida para quienes busquen especialización mediante asignaturas electivas o programas de posgrado.

COMPETENCIAS

Algunas de las competencias clave que deben ser cubiertas en el curso son las siguientes:

- ✓ **C1. Pensamiento analítico y crítico:** Analiza problemas de software y plantea soluciones fundamentadas en principios de diseño.
- ✓ **C2. Innovación y resolución de problemas:** Fomenta la creatividad y la capacidad de proponer soluciones innovadoras a desafíos tecnológicos.
- ✓ **C3. Diseño de software:** Desarrolla habilidades para crear soluciones efectivas y eficientes mediante el uso de estilos y patrones arquitectónicos.
- ✓ **C4. Arquitectura de software:** Formula soluciones a partir de la definición del diseño arquitectónico y detallado que cubra los requisitos del negocio y su contexto.

RESULTADOS DE APRENDIZAJE

Los resultados de aprendizaje esperados en el curso de Ingeniería de Software II, están orientados a garantizar que los estudiantes adquieran las competencias y habilidades necesarias para desenvolverse de manera efectiva en el campo del desarrollo de software. Al completar el curso, se espera que el estudiante:

- ✓ **RA1.** Explica los principios y conceptos fundamentales de la arquitectura de software, así como los diferentes patrones y estilos arquitectónicos.
- ✓ **RA2.** Analiza las características, ventajas, limitaciones y contextos de aplicación de diversos estilos y patrones arquitectónicos, incluyendo Cliente-Servidor, Peer-to-Peer, Microkernel y SOA.
- ✓ **RA3.** Diseña arquitecturas de software utilizando modelos, diagramas y estándares adecuados para garantizar claridad y mantenibilidad.
- ✓ **RA5.** Implementa arquitecturas modernas como microservicios, EDA, CQRS, DDD y arquitectura hexagonal, seleccionando la más adecuada según las necesidades del proyecto.

METODOLOGÍA

Se propone una metodología de enseñanza que combine tanto aspectos teóricos como prácticos, enfocándose en el aprendizaje activo y la participación de los estudiantes. Una metodología adecuada podría ser la siguiente:

- ✓ **Clases expositivas:** Presentación de conceptos fundamentales de arquitectura de software y otros temas clave.
- ✓ **Proyectos prácticos:** Trabajo en equipo para diseñar, desarrollar y probar aplicaciones de software.
- ✓ **Aprendizaje colaborativo:** Fomento del trabajo en equipo y habilidades de comunicación.
- ✓ **Sesiones de revisión y retroalimentación:** Evaluación periódica de avances con observaciones constructivas.
- ✓ **Evaluación continua:** Uso de exámenes, mini-proyectos, participación en clase y otras actividades para identificar áreas de mejora.

INVESTIGACIÓN

Incorporar actividades de investigación en un curso de Ingeniería de Software II enriquece significativamente la experiencia educativa y fomentar el pensamiento crítico y la creatividad. Algunas actividades de investigación que podrían incluirse son las siguientes:

- ✓ Clean architecture
- ✓ Soft skills
- ✓ Estilos arquitectónicos
- ✓ Patrones arquitectónicos
- ✓ Arquitecturas modernas

Estas actividades de investigación pueden llevarse a cabo individualmente o en grupos, y los estudiantes pueden presentar sus hallazgos a través de informes escritos, presentaciones orales, posters o proyectos prácticos.

A partir de las actividades de investigación se proponen las siguientes líneas de investigación:

1. Fundamentos y evolución de la arquitectura de software

- Evolución histórica de la arquitectura de software: de monolitos a microservicios.
- Principios de diseño sostenible en arquitectura de software.
- Factores que influyen en la elección de un estilo arquitectónico.

2. Patrones arquitectónicos

- Comparación de patrones arquitectónicos clásicos y modernos.
- Aplicación de patrones arquitectónicos para mejorar escalabilidad y mantenibilidad.
- Análisis de patrones arquitectónicos en sistemas distribuidos.

3. Estilos arquitectónicos

- Cliente-Servidor vs. Peer-to-Peer: ventajas, limitaciones y casos de uso.
- Microkernel: aplicaciones en sistemas modulares y de alto rendimiento.
- Service-Oriented Architecture (SOA) frente a microservicios: similitudes, diferencias y tendencias.

4. Arquitecturas modernas y emergentes

- Microservicios: retos en su implementación y estrategias para la comunicación efectiva.
- Arquitectura orientada a eventos (EDA) en sistemas de alta concurrencia.
- Aplicación de CQRS en sistemas con alto volumen de operaciones de lectura y escritura.
- Arquitecturas Service-Mesh y Data-Mesh
- Domain-Driven Design (DDD) como base para arquitecturas complejas.
- Arquitectura hexagonal: enfoque para mejorar la independencia de la infraestructura.

5. Aplicaciones prácticas

- Estudio de casos de éxito y fracaso en la implementación de microservicios.
- Migración de sistemas monolíticos a arquitecturas modernas: retos y estrategias.
- Uso combinado de arquitecturas (híbridas) para resolver problemas específicos.

MEDIOS AUDIOVISUALES

Para el mejor desarrollo de las actividades propuestas, se hará uso de: herramientas de comunicación síncrona como videoconferencias y asíncronas como foros; se acudirá al uso de la plataforma institucional Moodle con el fin de socializar material de estudio y lectura, efectuar la entrega de trabajos y tareas, además para la realización de algunas evaluaciones.

EVALUACIÓN

EVALUACIÓN COLECTIVA

Se incentiva el trabajo colaborativo, para que los equipos en sinergia puedan contar con los recursos tecnológicos necesarios para el desarrollo de la asignatura. Para la evaluación se propone actividades que pueden ser realizadas de manera colectiva, se encuentran las siguientes:

Ítem	Porcentaje
Participación en actividades o talleres grupales	40%
Proyecto en cada corte	40%

EVALUACIÓN INDIVIDUAL

Se realizan actividades que permiten medir el avance en el conocimiento y consolidación de habilidades.

Ítem	Porcentaje
Evaluaciones	30%

CONTENIDOS TEMÁTICOS CENTRALES

- ✓ Arquitectura de software: principios y conceptos fundamentales.
- ✓ Patrones arquitectónicos.
- ✓ Diseño y documentación de la arquitectura de software.
- ✓ Estilos arquitectónicos de software
 - Cliente-Servidor
 - Peer-To-Peer
 - Microkernel
 - Service-Oriented Architecture
- ✓ Patrones arquitectónicos
- ✓ Arquitectura orientada a microservicios.
- ✓ Implementación de microservicios y comunicación entre ellos.
- ✓ Arquitectura orientada a eventos (EDA)
- ✓ Arquitectura CQRS
- ✓ Arquitectura orientada al Dominio
- ✓ Arquitectura hexagonal

LECTURAS MÍNIMAS

Documentos y artículos relacionados con la asignatura, proporcionados por el docente:

- ✓ Clean architecture.
- ✓ Building Evolutionary Architectures_ Support Constant Change.
- ✓ Soft Skills – The software developer's life manual

BIBLIOGRAFÍA

- ✓ Agile Software Development, principles, patterns and practices. Robert Cecil Martin, Prentice Hall, New Jersey 2003.
- ✓ Análisis de Sistemas y Diseño de Métodos. Jeffrey L. Witten & Lonnie D. Bentley, McGrawHill, México 2008.
- ✓ Building Evolutionary Architectures, Support Constant Change. Neal Ford, Rebecca Parson & Patrk Kua. O'Reilly Media, Inc, United States of America 2017.
- ✓ Clean Architecture, a craftsman's guide to software structure and design. Robert Cecil Martin, Prentice Hall, Boston 2018.
- ✓ Ingeniería de Software. Ian Sommerville, Pearson, México 2011.
- ✓ Ingeniería de Software, un enfoque práctico. Roger S. Pressman, McGrawHill México 2010.
- ✓ Patrones de diseño, elementos de software orientado a objetos reutilizables. Erich Gamma, Richard Helm, Ralph Johnson & John Vlissides. Addison Wesley, Madrid 2003.
- ✓ Software Architecture for Developers, Technical leadership by coding, coaching, collaboration, architecture sketching and just enough up front design. Simon Brown, Leanpub, Australia 2014.
- ✓ Software Development From A to Z A Deep Dive into all the Roles Involved in the Creation of Software. OLGA Filipova & Rui Vilão. Apress, Berlin 2018.
- ✓ Software Engineering, Architecture-Driven Software Development. Richard F. Schmidt. Elsevier & Morgan Kufman, United States of America 2013.
- ✓ Software Architecture Patterns, Understanding Common Architecture Patterns and When to Use Them. Mark Richards. O'Reilly Media, Inc. United States of America 2015.
- ✓ Swebok, IEEE Computer Society. Los Alamitos, California 2004.

Nombre del docente responsable: Docentes del área de Software del programa