Programa de
TENDENCIAS APLICADAS DE INGENIERÍA DE SOFTWARE

1. NOMBRE DE LA UNIDAD CURRICULAR

Tendencias aplicadas de Ingeniería de Software

2. CRÉDITOS

8 créditos

3. OBJETIVOS DE LA UNIDAD CURRICULAR

- Desarrollar habilidades para aplicar técnicas concretas de Arquitectura e Ingeniería de Software en contextos modernos de desarrollo.
- Conocer tendencias específicas y ganar experiencia con herramientas que permiten su aplicación.
- Desarrollar capacidades para evaluar y aplicar componentes *Open Source* que aporten en distintas dimensiones del desarrollo.
- Integrar conocimientos teóricos previos con habilidades prácticas para llevar adelante tácticas de diseño y desarrollo modernas.
- Conocer las oportunidades y riesgos de aplicar Inteligencia Artificial Generativa en distintos aspectos de la disciplina.

4. METODOLOGÍA DE ENSEÑANZA

Se dictarán 4 horas semanales de clase, incluyendo presentaciones teóricas y trabajo en laboratorio. Además, cada alumno deberá dedicar un promedio de 4 horas semanales para trabajo domiciliario.

Los estudiantes participarán en actividades prácticas de laboratorio continuas a lo largo del curso, integrando las tácticas estudiadas en cada tema. Se incluyen presentaciones, análisis de casos y desarrollo de trabajo práctico en laboratorio.

5. TEMARIO

1. Introducción a las tácticas de diseño en Ingeniería de Software

- ¿Qué son las tácticas de Diseño en Ingeniería de Software?
- La Arquitectura de Software como balance de características.
- Repaso de estilos arquitectónicos y patrones.
- Tácticas de diseño más comunes.

2. Contenerización para Pruebas y Despliegue automatizado

- Profundización en aspectos de nivel intermedio/avanzado de contenerización.
- Orquestación de despliegues, alternativas para manejar secretos.
- Diferencias entre Plataformas: Docker, Kubernetes (on-premises y cloud).
- Caso de análisis: Docker on-premises.
- Caso práctico: Orquestación de despliegue y escalabilidad.

3. Persistencia de Datos No-Relacional

- Motivación y ventajas frente al modelo relacional.
- Bases de datos no relacionales más comunes: características, ventajas y desventajas.
- Tácticas posibles con estos modelos de persistencia.
- Análisis de un caso: MongoDB / SolR.
- Comparación práctica entre PostgreSQL y MongoDB (o PostgreSQL y Solr).

4. Cache de Datos / Bases de Datos en Memoria

- Tácticas que permiten llevar a la práctica las bases de datos en memoria o cache.
- Relación entre las capas de persistencia y las de cache.
- Análisis de un caso: Redis o Hazelcast.

5. Uso avanzado del IDE

- Técnicas de Ingeniería de Software apoyadas en IDEs modernos (análisis de código, integración con gestión de código, cubrimiento de tests, etc.).
- Evaluación de un caso: IntelliJ para desarrollo Java.

6. Programación Asincrónica

- Roles que juega el asincronismo en la arquitectura de software.
- Patrones asincrónicos (canales, publish/subscribe).
- Frameworks para aplicar Asincronismo.
- Evaluación de un caso: Mosquitto (MQTT) / ActiveMQ (JMS) / Kafka.

7. Integración de GenAI en productos de Software

- Fundamentos de LLMs para desarrolladores.
- Capacidades y limitaciones técnicas. Consideraciones éticas y de privacidad.
- Uso de RAG / MCP como alternativas de integración.
- Caso Práctico: MCP Server con Spring AI para extender un sistema de información.

Temas opcionales:

A. APIs REST

- Introducción a REST y las tácticas de diseño asociadas.
- Rol técnico y estratégico de las APIs.
- Importancia de la documentación y las plataformas de gestión de APIs.
- Desarrollo de APIs a partir de documentación (ej. Open APIs).
- Generación de documentación desde el código.
- Modelo de Información.
- Tácticas que dependen de la construcción del API (seguridad, interoperabilidad, facilidad de diagnóstico, etc.).

B. Integración Continua (CI/CD)

- Repaso de CI/CD y su rol en el proceso de ingeniería de software.
- Nivel de integración con distintas herramientas: Incidentes, gestión de código.
- Técnicas para desarrollar Mocks y realizar pruebas unitarias y de integración en forma automatizada.
- Caso práctico: Instalación y uso de Jenkins con contenedores para la integración de proyectos.

6. BIBLIOGRAFÍA

Tema	Básica	Complementaria
1. Introducción a las tácticas de diseño	(1)	(2)
2. Contenerización para Pruebas y Despliegue	(1)	
3. Persistencia de Datos No-Relacional	(2)	
4. Cache de Datos / BD en Memoria	(2)	
5. Uso avanzado del IDE	(1)	
6. Programación Asincrónica	(1)(2)	
7. Integración de GenAI en Software	(3)	
A API REST		(4)

6.1 Básica

1. Sommerville, Ian (2019). Engineering Software Products. Pearson.
2. Kleppmann, Martin (2026). Designing Data-Intensive Applications, 2nd Edition. O'Reilly.
3. Chip Huyen (2025). AI Engineering: Building Applications with Foundation Models. O'Reilly.
4. Newman, Sam (2021). Building Microservices, 2nd Edition. O'Reilly.

7. CONOCIMIENTOS PREVIOS EXIGIDOS Y RECOMENDADOS

7.1 Conocimientos Previos Exigidos: Conceptos fundamentales de Ingeniería de Software, programación orientada a objetos, bases de datos relacionales.

7.2 Conocimientos Previos Recomendados: Experiencia básica con contenedores (Docker), nociones de arquitectura de software, familiaridad con entornos de desarrollo integrados (IDE) y nociones de gestión de código con Git.

ANEXO A
Para todas las Carreras

A1) INSTITUTO

Comisión nacional de carrera del Tecnólogo en Informática (UTU-UTEC-UDELAR)

A2) CRONOGRAMA TENTATIVO

Semana 1	Tema 1: Introducción a las tácticas de diseño (4 hs)
Semana 2	Tema 1: Introducción a las tácticas de diseño (4 hs)
Semana 3	Tema 2: Contenerización para Pruebas y Despliegue (4 hs)
Semana 4	Tema 2: Contenerización - Caso práctico (4 hs)
Semana 5	Tema 3: Persistencia No-Relacional (4 hs)
Semana 6	Tema 3: Persistencia - Comparación práctica (4 hs)
Semana 7	Tema 4: Cache de Datos / BD en Memoria (4 hs)
Semana 8	Tema 4: Cache - Caso práctico (4 hs)
Semana 9	Tema 5: Uso avanzado del IDE (4 hs)
Semana 10	Tema 6: Programación Asincrónica (4 hs)
Semana 11	Tema 6: Programación Asincrónica - Caso práctico (4 hs)
Semana 12	Tema 7: Integración de GenAI en Software (4 hs)
Semana 13	Tema 7: GenAI - Caso práctico MCP Server (4 hs)
Semana 14	Repaso e integración de laboratorio (4 hs)
Semana 15	Prueba final (4 hs)

A3) MODALIDAD DEL CURSO Y PROCEDIMIENTO DE EVALUACIÓN

La evaluación consistirá en:

1. Trabajo de laboratorio: Desarrollo continuo durante el curso, integrando las tácticas estudiadas.

2. Prueba final: Evaluación de conocimientos teóricos dictados y reforzados por la aplicación en el laboratorio.

Distribución de puntajes según instancias de evaluación:

- Taller: 50 puntos. Se evaluarán entregas realizadas por los estudiantes durante el taller.
- Prueba Final: 50 puntos.

Para aprobar el curso, se deberá tener un mínimo del 60% del puntaje total.

A4) CALIDAD DE LIBRE

La unidad curricular no adhiere a la Calidad de libre.

A5) CUPOS DE LA UNIDAD CURRICULAR

Cupos mínimos: no tiene
Cupos máximos: no tiene

ANEXO B para la carrera Tecnólogo en Informática

B1) ÁREA DE FORMACIÓN

Desarrollo de Software

B2) UNIDADES CURRICULARES PREVIAS

- Curso aprobado de Ingeniería de Software
- Curso aprobado de Programación Avanzada

Examen: No aplica