UNIVERSIDAD NACIONAL DE INGENIERÍA

SÍLABO

INFORMACIÓN GENERAL

ASIGNATURA	:INGENIERÍA DE SOFTWARE
DOCENTE	:GERSON GARRIDO
HORAS POR SEMANA	:06 (TEORÍA: 04, LABORATORIO: 02)

SUMILLA

El curso es de naturaleza teórico-práctica y estudia cómo se construye software que dura: software que cambia con los requisitos, que lo desarrolla un equipo y que usan personas reales. Se organiza en cuatro partes: **proceso y requisitos** (ingeniería de software frente a programación, procesos ágiles, obtención de requisitos con un cliente, especificaciones); **diseño y arquitectura** (complejidad, módulos e interfaces, APIs, patrones de diseño, arquitectura y registro de decisiones); **calidad** (pruebas unitarias, basadas en propiedades y de mutación, revisión de código, análisis estático, integración continua y gestión de dependencias); y **producción y evolución** (despliegue, operación y respuesta a incidentes, desarrollo con herramientas de inteligencia artificial y productos que usan modelos de lenguaje, seguridad, privacidad, ética y mantenimiento). Las herramientas de inteligencia artificial para programar se estudian como parte del tema: cómo especificarles el trabajo, cómo verificar lo que producen y cuáles son sus riesgos. Se usa TypeScript con Node.js 26, Git y GitHub. Los estudiantes desarrollan en equipo, durante todo el semestre, un producto para un cliente real.

COMPETENCIAS

- Explica en qué se diferencia la ingeniería de software de la programación y elige un proceso de desarrollo adecuado para un equipo y un cliente.
- Obtiene requisitos de un cliente mediante entrevistas y los expresa como historias de usuario (*user stories*) con criterios de aceptación verificables.
- Escribe especificaciones y documentos de diseño, diseña módulos e interfaces que reducen la complejidad y documenta decisiones de arquitectura con sus alternativas.
- Verifica software con pruebas unitarias, basadas en propiedades y de mutación, revisión de código (*code review*) y análisis estático.
- Construye un pipeline de integración y entrega continuas, despliega una aplicación en contenedores, la observa en producción y responde a un incidente con un análisis posterior (*postmortem*).
- Usa herramientas de inteligencia artificial para desarrollar software con criterio: les da especificaciones y contexto, verifica lo que producen y reconoce sus riesgos.
- Trabaja en equipo con control de versiones, se comunica con un cliente y considera la seguridad, la privacidad, la accesibilidad y la ética de lo que construye.

UNIDADES DE APRENDIZAJE

Unidad	Contenido	Semanas
I	Proceso y requisitos	1–4
II	Diseño y arquitectura	5–8
III	Calidad	9–12
IV	Producción y evolución	13–16

El texto principal, *Software Engineering at Google*, es de libre acceso en línea. Cada semana indica sus capítulos y lecturas complementarias, también gratuitas en su mayoría.

PROGRAMACIÓN SEMANAL

Cada semana comprende dos sesiones de teoría de 2 horas (A y B) y una sesión de laboratorio de 2 horas. Cada capítulo de los apuntes indica qué secciones se leen antes de clase (unos 20 minutos), cuáles se explican en clase y cuáles quedan para el repaso, junto con las lecturas de la guía de la semana. Las sesiones marcadas con (*) pueden pasar a lectura autónoma si se pierden clases por feriados.

Sem.	Sesión A	Sesión B	Laboratorio
1	Presentación del curso. ¿Qué es la ingeniería de software?	La inteligencia artificial en el ciclo de desarrollo: usos, límites y uso responsable	Entorno de trabajo y Git en equipo. *P0 publicado*
2	Procesos: cascada, iterativo, Scrum y Kanban	Planificación y estimación. Estudio de caso	Issues, tablero y estimación
3	Requisitos: obtención y entrevistas	Historias de usuario, criterios de aceptación y priorización	Entrevista simulada y backlog. *Entrega P0, publicado P1*
4	Especificaciones: contratos, invariantes y tipos	Documentos de diseño y desarrollo guiado por especificaciones	De la especificación al código
5	Diseño: complejidad, módulos profundos y ocultamiento de información	Conceptos, acoplamiento y cohesión	Refactorizar un módulo. *Entrega P1, publicado P2*
6	Interfaces y APIs: REST, errores, versionado e idempotencia	Patrones de diseño	Diseñar y documentar una API
7	Arquitectura: capas, hexagonal, monolito y microservicios	Atributos de calidad y registro de decisiones de arquitectura. Diagramas C4 (*)	Registro de una decisión y diagrama de contenedores
8	Repaso	**Examen parcial**	— *Entrega P2, publicado P3*
9	Pruebas unitarias, dobles de prueba y cobertura	Desarrollo guiado por pruebas (TDD) y por comportamiento (BDD)	Pruebas unitarias
10	Pruebas basadas en propiedades y fuzzing	Pruebas de mutación y oráculos para código generado	Encontrar errores con propiedades y mutantes
11	Revisión de código	Análisis estático, tipos y linters. Revisión de código escrito con IA	Revisión cruzada entre equipos. *Entrega P3, publicado P4*
12	Integración continua y build	Dependencias y cadena de suministro	Un pipeline de integración continua

Sem.	Sesión A	Sesión B	Laboratorio
13	Despliegue: contenedores, entornos y feature flags	Operación: observabilidad, objetivos de nivel de servicio, incidentes	Desplegar, observar y analizar un incidente
14	Desarrollo con agentes: herramientas, protocolos e ingeniería de contexto	Productos con modelos de lenguaje: evaluaciones, inyección de prompts y costo	Evaluaciones de una funcionalidad con LLM. *Entrega P4, publicado P5*
15	Seguridad y privacidad desde el diseño	Ética, accesibilidad y licencias. Procedencia del código generado (*)	Auditoría de seguridad y accesibilidad
16	Mantenimiento: código heredado, deuda técnica y deprecación	Presentaciones finales	Defensa oral de los proyectos. *Entrega P5*
—	**Examen final**, en el período de exámenes		

METODOLOGÍA

- **Teoría:** clases expositivas con discusión de casos reales, ejercicios en clase (requisitos, diseño, pruebas, revisión de código) y demostraciones en vivo sobre la aplicación del laboratorio. El estudiante lee antes de cada semana las secciones de los apuntes que indica su ruta de lectura, y después las lecturas de la guía semanal.
- **Laboratorio:** prácticas guiadas sobre Kallpa, una aplicación de reservas escrita para el curso, en un contenedor común para todos los estudiantes, con Node.js 26 y las herramientas del curso.
- **Proyecto:** cada equipo construye un producto para un cliente real en tres sprints, con seis entregas que tienen revisión automática y revisión del docente y del cliente.
- **Inteligencia artificial:** se permite en el proyecto y en los laboratorios, con un registro de uso y explicación oral; los exámenes se rinden sin ella (ver *Uso de herramientas e inteligencia artificial*).
- Los términos técnicos se usan en inglés cuando así se dicen y así aparecen en la bibliografía y en la documentación (pull request, backlog, sprint, pipeline). Los que de verdad se usan en español (requisito, prueba, revisión de código, despliegue) se presentan en español, con su equivalente en inglés la primera vez.

Proyecto: un producto para un cliente real

N.º	Entrega	Lo que agrega	Semanas
P0	Primer aporte	Un cambio a la aplicación del laboratorio en una rama, con pruebas y revisión (individual)	1–3
P1	Requisitos	Cliente, entrevistas, historias de usuario, backlog priorizado y prototipo	3–5
P2	Diseño y primer sprint	Documento de diseño, decisiones de arquitectura, especificación de la API y primer incremento	5–8
P3	Segundo sprint: calidad	Pruebas unitarias y de propiedades, revisión de código y análisis estático	8–11
P4	Tercer sprint: entrega continua	Pipeline de integración y entrega, despliegue, observabilidad y análisis de un incidente	11–14
P5	Producto final	Revisión de seguridad, entrega al cliente, documentación de mantenimiento, retrospectiva y defensa oral	14–16

USO DE HERRAMIENTAS E INTELIGENCIA ARTIFICIAL

Las herramientas de inteligencia artificial son parte del tema del curso. Se permiten en el proyecto y en los laboratorios, con una condición: el equipo responde por todo lo que entrega como si lo hubiera escrito a mano. Los exámenes se rinden sin ellas.

Herramienta	Proyecto y laboratorios	Exámenes
Diagnósticos del compilador y del intérprete, y herramientas que revisan el programa mientras corre	Sí	Solo si el examen se rinde en el contenedor del curso
Documentación oficial y búsqueda en la web	Sí	No
Autocompletado del editor que completa nombres y líneas, sin generar soluciones enteras	Sí	No
Corrector ortográfico, traductor y herramientas de accesibilidad	Sí	Solo las de accesibilidad que el estudiante necesite
Formateadores de código y linters	Sí	Solo si el examen se rinde en el contenedor del curso
Inteligencia artificial generativa para explicar conceptos, errores o código, sin que escriba lo entregado	Sí, con registro	No
Inteligencia artificial generativa que escribe código, pruebas o documentos	Sí, con registro	No

Declaración. Cada equipo lleva en su repositorio un registro de uso de inteligencia artificial: fecha, herramienta, qué se pidió, qué se aceptó, qué se rechazó y por qué. Un cambio grande hecho con una de estas herramientas y sin registro se considera no entregado. Las herramientas de las cinco primeras filas no se declaran.

Explicación oral. A cualquier estudiante se le puede pedir que explique de forma oral cualquier parte de lo que entregó su equipo: qué hace, por qué es así y cómo se verificó, lo haya escrito a mano o con ayuda de una herramienta.

BIBLIOGRAFÍA

Texto principal

1. Winters, T., Manshreck, T. y Wright, H. (2020). *Software Engineering at Google: Lessons Learned from Programming Over Time*. O'Reilly. Disponible gratuitamente en línea: `https://abseil.io/resources/swe-book`

Textos complementarios

2. Ousterhout, J. (2021). *A Philosophy of Software Design* (2.ª ed.). Yaknyam Press.
3. Jackson, D. (2021). *The Essence of Software: Why Concepts Matter for Great Design*. Princeton University Press.
4. Fox, A. y Patterson, D. *Engineering Software as a Service: An Agile Approach Using Cloud Computing* (2.ª ed.). Strawberry Canyon. `https://www.saasbook.info/`
5. Beyer, B., Jones, C., Petoff, J. y Murphy, N. R. (2016). *Site Reliability Engineering*. O'Reilly. Disponible gratuitamente en línea: `https://sre.google/sre-book/table-of-contents/`

Recursos en línea

- Reportes de investigación de DORA: `https://dora.dev/research/`
- OWASP Top 10: `https://owasp.org/www-project-top-ten/`
- Documentación de Node.js: `https://nodejs.org/docs/latest-v26.x/api/`
- Curso 17-313, Carnegie Mellon University: `https://cmu-313.github.io/`
- Curso CS146S, Stanford University: `https://themodernsoftware.dev/`
- Sitio del curso: guías semanales, apuntes, diapositivas, código y enunciados.