UNIVERSIDAD RICARDO PALMA

SÍLABO

Escuela de Posgrado
Maestría en ingeniería Informática con mención en Ingeniería de Software

I. DATOS ADMINISTRATIVOS

1. Asignatura : Diseño de Software I
2. Código : MII-103
3. Naturaleza : Teórico/práctica
4. Condición : Obligatoria
5. Requisito : Ninguna
6. Número de créditos : 4
7. Número de horas semanales : 4 Horas Teóricas y 2 Horas de Práctica
8. Semestre académico : 2025-I
9. Docente : Víctor Cabrejos Yalán
10. Correo electrónico : victor.cabrejos@urp.edu.pe

:

II. SUMILLA

El curso de Diseño de Software I constituye un curso de formación específica, de carácter teórico práctico. Proporciona a los estudiantes conocimientos, habilidades y herramientas necesarias para abordar los principios y patrones de diseño de software en un nivel fundamental, preparando a los estudiantes para el diseño de software dirigido por el dominio del negocio. El desarrollo del curso comprende: Fundamentos de estilos y patrones de diseño a nivel estratégico y táctico para sistemas escalables, flexibles y de alta disponibilidad centrados en el dominio, aplicando enfoques, técnicas y herramientas para el diseño iterativo y colaborativo de soluciones de software. Durante el curso el estudiante investiga y desarrolla habilidades prácticas a través de proyectos aplicados, con un enfoque en la ética y la privacidad de datos de ser el caso. Asimismo, analiza, sintetiza y presenta un artículo de investigación relacionado al curso.

III. COMPETENCIAS GENÉRICAS A LAS QUE CONTRIBUYE LA ASIGNATURA

a) Pensamiento crítico y creativo. El curso fomenta la creación de nuevos algoritmos a partir de los ya existentes.
b) Autoaprendizaje. Los proyectos que se desarrollan en el curso inciden en la investigación y el autoaprendizaje.
c) Investigación científica y tecnológica. El curso hace uso intensivo de herramientas de software y fomenta la investigación a través del desarrollo de nuevos casos y la creación de nuevas técnicas y métodos algorítmicos.
d) Comportamiento ético. La aplicación y uso de la Ciencia de Datos debe enmarcarse según principios éticos.

IV. COMPETENCIAS ESPECÍFICAS A LAS QUE CONTRIBUYE LA ASIGNATURA

La asignatura se enfoca en desarrollar competencias de ingeniería de software aplicadas al diseño de sistemas escalables y mantenibles. Los estudiantes adquieren la capacidad de aplicar principios SOLID, patrones de diseño estratégicos y tácticos, y metodologías de trabajo colaborativas como GitFlow. A través de proyectos alineados a su tesis, serán capaces de estructurar código limpio, modular y ético, documentar soluciones de manera profesional y reflexionar sobre el impacto social del software. Esto les proporciona una base sólida para enfrentar retos reales en el desarrollo de sistemas inteligentes y adaptarse a las buenas prácticas del entorno tecnológico actual.

V. DESARROLLA EL COMPONENTE DE:

INVESTIGACION (X) RESPONSABILIDAD SOCIAL (X)

VI. LOGRO DE LA ASIGNATURA

Al terminar el ciclo académico, el estudiante será capaz de diseñar soluciones de software escalables, modulares y éticas aplicando principios de ingeniería de software, patrones de diseño estratégicos y tácticos, y buenas prácticas de programación en Python, con enfoque en proyectos alineados a su tesis.

VII. PROGRAMACIÓN DE CONTENIDOS

UNIDAD 1: Fundamentos de Estilos y Patrones de Diseño	
LOGRO DE APRENDIZAJE: El estudiante identificará los fundamentos de los estilos de diseño y aplicará patrones estratégicos básicos en el desarrollo de software orientado a dominios reales, haciendo uso de herramientas modernas de programación y control de versiones.	
Semana	**Contenido**
1	Introducción al Diseño de Software • Presentación del curso • Introducción a los principios del diseño de software • Herramientas y entornos de desarrollo
2	Fundamentos de Estilos de Diseño • Definición y clasificación de estilos de diseño • Ejemplos prácticos de estilos de diseño
3	Patrones de Diseño a Nivel Estratégico • Introducción a patrones de diseño • Patrones de diseño estratégicos: Singleton, Factory, y Prototype
4	Aplicación de Patrones Estratégicos • Ejercicios y casos prácticos sobre patrones estratégicos • Discusión de ventajas y desventajas

UNIDAD 2:

LOGRO DE APRENDIZAJE: El estudiante implementará patrones de diseño a nivel táctico y aplicará principios de escalabilidad y modularidad mediante código limpio y estructurado, empleando buenas prácticas profesionales.

Semana	Contenido
5	Patrones de Diseño a Nivel Táctico • Patrones de diseño tácticos: Adapter, Decorator, y Facade • Ejemplos y aplicaciones prácticas
6	Diseño de Sistemas Escalables y Flexibles • Principios de escalabilidad y flexibilidad en el diseño de software • Estrategias y técnicas para sistemas escalables
7	Revisión y Preparación para el Examen • Revisión de conceptos clave • Resolución de dudas y práctica
8	Examen Parcial

UNIDAD 3:

LOGRO DE APRENDIZAJE: El estudiante aplicará el enfoque de diseño dirigido por el dominio (DDD) e implementará soluciones colaborativas e iterativas que respeten principios éticos y técnicos del desarrollo moderno de software.

Semana	Contenido
9	Diseño Centrado en el Dominio • Introducción al diseño dirigido por el dominio (DDD) • Conceptos clave y terminología en DDD
10	Técnicas de Diseño Iterativo y Colaborativo • Enfoques iterativos en el diseño de software • Herramientas y técnicas para la colaboración en equipo
11	Proyectos Aplicados en Diseño de Software • Introducción a proyectos aplicados
12	Ética y Privacidad de Datos en el Diseño de Software • Principios de ética en el desarrollo de software • Consideraciones de privacidad y protección de datos

UNIDAD 4:

LOGRO DE APRENDIZAJE: El estudiante desarrollará y presentará un proyecto funcional alineado a su tesis y sustentará un artículo de investigación sobre diseño de software, demostrando capacidad analítica, técnica y comunicativa.

Semana	Contenido
13	Análisis y Síntesis de Artículos de Investigación • Métodos para analizar y sintetizar artículos de investigación • Presentación de artículos relacionados con el curso
14	Revisión y Preparación para el Examen Final • Revisión de conceptos clave • Resolución de dudas y práctica
15	Exposición de Trabajos
16	Examen Final
17	Entrega de Notas

VIII. ESTRATEGIAS DIDÁCTICAS

Aprendizaje basado en Proyectos, Aula invertida, Aprendizaje Colaborativo, Disertación

IX. EVALUACIÓN

TIPOS DE EVALUACIÓN	PESOS
Examen Parcial – Evaluación 1	30 %
Evaluación Continua – Evaluación 2	40 %
Examen Final – Evaluación 3	30 %

Nota mínima aprobatoria: Para aprobar el curso, la nota mínima es trece (13.00).

Desde el semestre 2025-I, solo se considerará aprobado el curso si el estudiante obtiene una calificación igual o mayor a 13.00.

X. RECURSOS

- Equipos: computadora, laptop, tablet, celular
- Materiales: apuntes de clase del Docente, separatas de problemas, lecturas, videos.
- Software y Herramientas: Visual Studio Code (VS Code), Git y GitHub, Python 3.11

XI. REFERENCIAS BIBLIOGRÁFICAS

BÁSICAS

- Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Evans, E. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley.
- Freeman, E., Robson, E., Bates, B., & Sierra, K. (2004). *Head First Design Patterns*. O'Reilly Media.

- Larman, C. (2002). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall.

COMPLEMENTARIAS

- Slatkin, B. (2015). *Effective Python: 59 Specific Ways to Write Better Python*. Addison-Wesley.
- Hamel, S. (2020). *Machine Learning for Developers: Practical Algorithms for Building AI Systems in Python*. Apress.
- Tiago, M. (2021). *Clean Architectures in Python: A practical approach to better software design*. Leanpub.
- FastAPI. (2024). *FastAPI Documentation*. https://fastapi.tiangolo.com/
- Pedregosa, F., et al. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825-2830.

Santiago de Surco, Abril 2025