Análisis y Diseño II

2 créditos teóricos y 2 créditos prácticos

A. Información del profesor

Nombre del profesor

Inga. Hilda Ruth Flores Muñoz

Correo electrónico

hrflores@correo.url.edu.gt

Campus o sede

Central

Horario

Martes y jueves de 5:30 a 7:00 p.m. y viernes de 5:30 a 9:00 p.m.

B. Información general

Descripción

Para la formación del ingeniero de software de nuestros días, la recomendación curricular de ACM e IEEE enfatiza la importancia de las áreas de conocimiento de Modelación y Análisis de Software (MAA) y Diseño de Software (DES). Este es el segundo de dos cursos de especialización en la disciplina de Ingeniería de Software y se asume que el estudiante ya cuenta con conocimientos sólidos de programación orientada a objetos y bases de datos. El primer curso se enfoca en las técnicas básicas de análisis y diseño de software, mientras que el segundo profundiza en las actividades de diseño de componentes, de arquitecturas de software y de la interfaz de usuario.

Modalidad

Mixta (Blended). Se combinarán momentos de aprendizaje autónomo, de parte del estudiante y guiado en la plataforma de aprendizaje de la Universidad; así como conferencias virtuales con los profesores, donde se favorecerá la metodología activa.

C. Malla curricular

COMPETENCIAS GENÉRICAS

El egresado landivariano se identifica por:

Pensamiento lógico, reflexivo y analógico	Pensamiento crítico	Resolución de problemas
Habilidades de investigación	Uso de TIC y gestión de la información	Comunicación efectiva, escrita y oral
Comprensión lectora	Compromiso ético y ciudadanía	Liderazgo constructivo
Aprecio y respeto por la diversidad e interculturalidad	Creatividad	

COMPETENCIAS ESPECÍFICAS (propias del curso)

Competencia 1

Construye diseños óptimos para soluciones de software que responden a las necesidades del negocio y el contexto del mercado, buscando constantemente la optimización de recursos, la consideración de atributos de calidad y la aplicación de patrones.

METODOLOGÍA

Este curso se desarrollará a través de los siguientes métodos de aprendizaje-enseñanza: Aprendizaje invertido, gamificación, aprendizaje basado en retos, aprendizaje por indagación, aprendizaje basado en equipos, análisis de casos.

PROGRAMACIÓN

COMPETENCIA 1

Construye diseños óptimos para soluciones de software que responden a las necesidades del negocio y el contexto del mercado, buscando constantemente la optimización de recursos, la consideración de atributos de calidad y la aplicación de patrones.

Saber conceptual (contenido temático)

FUNDAMENTOS

Introducción: definición de arquitectura de software, perfil del arquitecto, prácticas de ingeniería, operaciones y devops, fundamentos.
Pensamiento arquitectónico: arquitectura y diseño, equilibrio de arquitectura y codificación.
Modularidad: definición, cohesión, acoplamiento, métricas, transición a componentes.
Características arquitectónicas: operativas, estructurales, transversales. Extracción del dominio, de los requerimientos, explícitas e implícitas. Caso de Sand. de Silicón.
Medición y gestión de arquitectura.
Pensamiento basado en componentes.

ESTILOS ARQUITECTÓNICOS

Fundamentos: patrones, unitary, cliente servidor, desktop + database server, browser + webserver, three-tier, monolithic vs. distributed, falacias de diseño.
Estilos de arquitectura por capas.
Estilo de arquitectura basada en servicios.
Estilo de arquitectura basada en microservicios.
Cómo elegir el estilo de arquitectura adecuada. Caso de estudio.

HABILIDADES TÉCNICAS Y BLANDAS

Decisiones de arquitectura: anti-patrones de diseño, registros de decisiones de arquitectura, estructura básica (ADR), analizando riesgos de la arquitectura con modelos ágiles, diagramación y presentación de arquitecturas (UML, C4, ArchiMate).
Equipos de arquitectos, negociación y liderazgo.
Validación y verificación.

Saber procedimental (habilidades y destrezas)

Resolución de problemas con software.
Aplicación del modelo idóneo a un problema que se resuelve con software.

Saber actitudinal (conductas observables)

Lectura comprensiva y preparación antes de la clase.
Participación activa.
Trabajo colaborativo.
Desarrollo de competencias del perfil de egreso.
Integridad, puntualidad y compromiso con la calidad.

Indicador de logro 1 (resultado): Construye diseños óptimos para soluciones de software que responden a las necesidades del negocio y el contexto del mercado, buscando constantemente la optimización de recursos, la consideración de atributos de calidad y la aplicación de patrones.

EVALUACIÓN

a. Estrategias de evaluación sumativa

Estrategias	Puntaje
Pruebas parciales	20
Exámenes cortos y otras actividades en clase	20
Proyectos y prácticas de laboratorio	40
Examen final	20
TOTAL	**100**

b. Estrategias de evaluación formativa

Técnicas formativas	Procedimiento
Retroalimentación	Comentarios pertinentes en la entrega de los laboratorios y proyectos.
Diálogo socrático	Preguntas y respuestas orales a ejemplos y problemas que se realizarán a lo largo de la secuencia de aprendizaje.
One minute paper	Textos cortos sobre los temas ya vistos comprobando sus saberes.
Trabajos en pequeños grupos para resolver dudas	Dinámicas de grupo en clase como 1-2-3-all.
Citas individuales	Tutorías de retroalimentación solicitadas por el estudiante, por medios electrónicos.

CALENDARIO DE REFERENCIA POR TEMAS

Fecha	Tema	Actividad de evaluación
Semana 1 15 – 19 mayo	Introducción: definición de arquitectura de software, perfil del arquitecto, prácticas de ingeniería, operaciones y devops, fundamentos. Pensamiento arquitectónico: arquitectura y diseño, equilibrio de arquitectura y codificación. Modularidad: definición, cohesión, acoplamiento, métricas, transición a componentes.	**Períodos teóricos**: Aprendizaje invertido, gamificación, análisis de casos. **Períodos prácticos**: aprendizaje basado en retos, aprendizaje basado en proyectos, aprendizaje por indagación, aprendizaje basado en equipos.
Semana 2 22 - 26 mayo	Características arquitectónicas: operativas, estructurales, transversales. Extracción del dominio, de los requerimientos, explícitas e implícitas. Caso de Sand. de Silicón.	

	Medición y gestión de arquitectura. Pensamiento basado en componentes.	
Semana 3 29 mayo – 2-junio	Fundamentos: patrones, unitary, cliente servidor, desktop + database server, browser + webserver, three-tier, monolithic vs. distributed, falacias de diseño.	**Períodos teóricos**: Aprendizaje invertido, gamificación, análisis de casos. **Períodos prácticos**: aprendizaje basado en retos, aprendizaje basado en proyectos, aprendizaje por indagación, aprendizaje basado en equipos **Parcial I**
Semana 4 5 – 9 junio	Estilos de arquitectura por capas.	**Períodos teóricos**: Aprendizaje invertido, gamificación, análisis de casos. **Períodos prácticos**: aprendizaje basado en retos, aprendizaje basado en proyectos, aprendizaje por indagación, aprendizaje basado en equipos
Semana 5 12 – 16 junio	Estilo de arquitectura basada en servicios.	
Semana 6 19 – 23 junio	Estilo de arquitectura basada en microservicios. Cómo elegir el estilo de arquitectura adecuada. Caso de estudio.	**Períodos teóricos**: Aprendizaje invertido, gamificación, análisis de casos. **Períodos prácticos**: aprendizaje basado en retos, aprendizaje basado en proyectos, aprendizaje por indagación, aprendizaje basado en equipos **Parcial II**
Semana 7 26 – 30 junio	Decisiones de arquitectura: anti-patrones de diseño, registros de decisiones de arquitectura, estructura básica (ADR), analizando riesgos de la arquitectura con modelos ágiles, diagramación y presentación de arquitecturas (UML, C4, ArchiMate).	**Períodos teóricos**: Aprendizaje invertido, gamificación, análisis de casos. **Períodos prácticos**: aprendizaje basado en retos, aprendizaje basado en proyectos, aprendizaje por indagación, aprendizaje basado en equipos **Examen final.**
Semana 8 3 – 7 julio	Equipos de arquitectos, negociación y liderazgo. Validación y verificación	

REFERENCIAS BIBLIOGRÁFICAS

IEEE (s.f.). Software Engineering Body of Knowledge – SWEBOK. Recuperado de: https://www.computer.org/education/bodies-of-knowledge/software-engineering

IEEE SA (s.f.). Standards. Institute of Electrical and Electronics Engineers, Inc – IEEE. Recuperado de: https://standards.ieee.org/

ISO (s.f.) International Standards. International Organization for Standardization – ISO. Recuperado de: https://www.iso.org/home.html

Jacobson, I., Lawson, H., Ng, P.W., McMahon, P.E., Goedicke, M. (2019). The Essentials of Modern Software Engineering. Association for Computing Machinery and Morgan & Claypool Publishers.

Kendall, K. & Kendall, J. (2011). Análisis y diseño de sistemas. Octava edición. Pearson Educación.

Pressman, R.S., Maxim, B.R. (2010) Software Engineering. A practitioner's Approach. 8th. Edition: McGrawHill Education.

Richards, M. & Ford, N. (2020). Fundamentals of Software Architecture. O'Reilly Media, Inc.

S.f. (2022). Azure application architecture fundamentals. Microsoft. https://learn.microsoft.com/en-us/azure/architecture/guide/

Sommerville, I. (2016). Software Engineering. Pearson Education Limited.

Stephens, R. (2015). Beginning Software Engineering.

Williams, L. (2013). An Introduction to Software Engineering. Edition one.

Winters, T., Manshreck, T., Wright, H. (2020) Software Engineering at Google. Lessons Learned from Programming Over Time. O'Reilly.