UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS

PROGRAMA DEL CURSO
NOMBRE DEL CURSO: SOFTWARE AVANZADO

CÓDIGO:	0780	**CRÉDITOS:**	6
ESCUELA:	CIENCIAS Y SISTEMAS	**ÁREA A LA QUE PERTENECE:**	DESARROLLO DE SOFTWARE
PRE REQUISITO:	785	**POST REQUISITO:**	Ninguno
CATEGORÍA:	OBLIGATORIO	**VIGENCIA:**	PRIMER SEMESTRE 2025
CATEDRÁTICO (A):	Marco Tulio Aldana Prillwitz	**AUXILIAR:**	Diego Rene Molina Roldán
EDIFICIO:		**SECCIÓN:**	B
SALÓN DEL CURSO:	CURSO A DISTANCIA	**SALON DEL LABORATORIO:**	CURSO A DISTANCIA
HORAS POR SEMANA DEL CURSO:	4	**HORAS POR SEMANA DEL LABORATORIO:**	2
DÍAS QUE SE IMPARTE EL CURSO:	Jueves y Viernes	**DIAS QUE SE IMPARTE EL LABORATORIO:**	Viernes
HORARIO DEL CURSO:	07:10 - 08:50	**HORARIO DEL LABORATORIO:**	16:30 – 18:10

DESCRIPCIÓN DEL CURSO:

Software Avanzado es un curso profesional que pertenece al área de software de la carrera de ingeniería en Ciencias y Sistemas, el cual trata sobre fundamentos de arquitectura: Explica qué es la arquitectura de software, el rol del arquitecto, metáforas arquitectónicas y principios de diseño.
Atributos de calidad: Describe atributos clave como performance, escalabilidad, disponibilidad, seguridad, modificabilidad, entre otros. Enfoques arquitectónicos: Detalla enfoques orientados a objetos, eventos, capas, microservicios, espacios de datos. Vistas arquitectónicas: Presenta vistas lógica, de procesos, física, de desarrollo y de escenarios. Patrones arquitectónicos: Explica en detalle los patrones más usados como MVC, flujo de datos, proxy, broker, entre otros. Estilos arquitectónicos: Cubre clientes-servidor, servicios, mensajería, por capas, híbridos y de integración. Decisiones de diseño: Discute principios SOLID, separación de intereses, acoplamiento, cohesión, composición.
También se desarrolla la administración de programas de Tecnología de la información y gerencia de proyectos de software, con base en mejores prácticas presentadas a través de marcos de trabajo. Durante el desarrollo del curso se hace énfasis en la importancia que tienen los modelos de referencia a manera de guía técnica de gestión de los recursos de TI de una empresa, que tiene como fin la elaboración de una adecuada planificación y seguimiento para lograr el éxito de un proyecto de software, que se traduzca en valor para la empresa

OBJETIVOS:

General

- Lograr que el estudiante comprenda los fundamentos de la arquitectura de software y adquiera los conocimientos para la gestión de tecnología de la información de cualquier institución, por medio de marcos de trabajo para gestión por procesos (COBIT e ITIL).

Específico

1. Analizar y aplicar eficazmente los conceptos de arquitectura de software como abstracción, descomposición, acoplamiento y cohesión.
2. Identificar los atributos de calidad críticos en un sistema y seleccionar las técnicas apropiadas para analizarlos y medirlos en una arquitectura.
3. Reconocer la diferencia entre estilos y patrones arquitectónicos comunes y seleccionar el más adecuado para un contexto de diseño específico.
4. Definir e implementar procesos efectivos para gestionar el ciclo de vida y la evolución de la arquitectura de acuerdo a las necesidades del negocio.
5. Desarrollar habilidades para producir representaciones y documentos arquitectónicos completos y consistentes utilizando estándares y herramientas establecidas.
6. Planificación general y específica en proyectos de software.
7. Gestión de proyectos de Software e IT para la aplicación al ámbito profesional.
8. Gestión y análisis de riesgos
9. Reconocimiento y aplicación de metodologías de CD/CI o DevOps según las necesidades del proyecto.
10. Reconocimiento de amenazas y vulnerabilidades de ítems de configuración y evaluación de riesgos para el tratamiento de estos.
11. Introducción a los procesos de soporte de la ingeniería de software.
12. Aplicación de técnicas, modelos y herramientas para la gestión de recursos de TI

METODOLOGÍA:

- Clases diarias.
- Elaboración de investigaciones y tareas.
- Práctica de exámenes cortos x clase y parciales.
- Laboratorio taller.
- Elaboración de proyectos de programación.

EVALUACION DEL RENDIMIENTO ACADÉMICO:

Clase teórica (75 puntos)		**Laboratorio**	
Descripción	**Pts.**	**Descripción**	**Pts.**
Cortos y Asistencia	13	Diez Prácticas	10
Primer parcial	14	Fase 1 Proyecto	10
Segundo parcial	14	Fase 2 Proyecto	10
Tercer parcial	14	Fase 3 Proyecto	20
Laboratorio (Practicas y Proyecto)	20	Entrega Final Proyecto	40
Tarea de clase en cada parcial		Exámenes cortos	10
Zona total	75		
Examen Final	25		
Total	100	Total	100

El curso se gana con 61 pts. de 100. Y el laboratorio gana con 61 pts. de 100.
Es necesario tener un 80%, como mínimo, de asistencia a clase o laboratorio para aprobar el curso. El trabajo de las prácticas y fases de proyecto se hará en parejas.

CONTENIDO:

1. **Arquitectura de Software**
    - 1.1. Arquitectura Empresarial
        - 1.1.1. Arquitectura Empresarial como estrategia
        - 1.1.2. Arquitectura Zachman
        - 1.1.3. Diagramas de Arquitectura Empresarial
            - 1.1.3.1. Canvas Empresarial
            - 1.1.3.2. Diagrama de Cadena de Valor
            - 1.1.3.3. Business Process Management
            - 1.1.3.4. Archimate
    - 1.2. Fundamentos de Arquitectura
        - 1.2.1. Qué es arquitectura de software
        - 1.2.2. Rol del Arquitecto
        - 1.2.3. Qué son Características de Arquitectura
        - 1.2.4. Qué son Decisiones de Arquitectura
        - 1.2.5. Qué son Estilos de Arquitectura
        - 1.2.6. Qué son principios de diseño
        - 1.2.7. Etapas de la Arquitectura de Software
        - 1.2.8. Minimum Viable Product
    - 1.3. Gestión de Requerimientos
        - 1.3.1. Requerimientos Estratégicos
        - 1.3.2. Riesgos
        - 1.3.3. Requerimientos Funcionales
        - 1.3.4. Atributos de Calidad
            - 1.3.4.1. Definición de Calidad
            - 1.3.4.2. Performance
            - 1.3.4.3. Escalabilidad
            - 1.3.4.4. Disponibilidad
            - 1.3.4.5. Seguridad
            - 1.3.4.6. Modificabilidad
            - 1.3.4.7. Disponibilidad
            - 1.3.4.8. Tolerante a Fallas
            - 1.3.4.9. Agilidad
    - 1.4. Enfoques Arquitectónicos
        - 1.4.1. Orientado a objetos
        - 1.4.2. Eventos
        - 1.4.3. Capas
        - 1.4.4. Orientado a Servicios
        - 1.4.5. Microservicios
        - 1.4.6. Basado en Tuberías y Filtros
        - 1.4.7. De integración
    - 1.5. Vistas Arquitectónicas
        - 1.5.1. Vista Lógica
        - 1.5.2. Vista de Procesos
        - 1.5.3. Vista Física
        - 1.5.4. Vista de Desarrollo
        - 1.5.5. Vistas de Escenarios
    - 1.6. Patrones Arquitectónicos
        - 1.6.1. Model View Controller
        - 1.6.2. Flujo de Datos
        - 1.6.3. Proxy
        - 1.6.4. Broker
        - 1.6.5. Publish And Subscribe
    - 1.7. Decisiones de Diseño
        - 1.7.1. Características de Diseño
        - 1.7.2. Principios de Diseño
        - 1.7.3. Paradigmas de Diseño
        - 1.7.4. Patrones de Diseño

1.7.5. Lenguaje de patrones de Diseño
1.7.6. Estándar de Diseño
1.7.7. Mejores prácticas
1.7.8. Marco de referencia de Diseño
1.8. Principios de Diseño
1.8.1. Contrato de Servicios
1.8.2. Acoplamiento de Servicios
1.8.3. Abstracción de Servicios
1.8.4. Reusabilidad de Servicios
1.8.5. Autonomía de Servicios
1.8.6. Manejo de Estado de Servicios
1.8.7. Descubrimiento de Servicios
1.8.8. Composición de Servicios

2. ITIL 4

2.1. Fundamentos de ITIL 4
2.2. Conceptos de Gestión de Servicios
2.2.1. Definición de Valor y co-creación de Valor
2.2.2. Definición de Organización
2.2.3. Proveedores de Servicio
2.2.4. Consumidores de Servicios
2.2.5. Interesados
2.2.6. Productos y Servicios
2.2.7. Oferta de Servicios
2.2.8. Relaciones en Servicios
2.2.9. Garantía y Utilidad
2.3. Sistema de Valor de Servicio de ITIL
2.3.1. Las entradas al sistema
2.3.2. Los elementos del Sistema
2.3.3. Las Salidas del Sistema
2.3.4. Cadena de Valor de Servicio de ITIL
2.3.4.1. Definición de las actividades del SVC
2.3.4.1.1. Planeación
2.3.4.1.2. Mejora
2.3.4.1.3. Compromiso
2.3.4.1.4. Diseño y Transición
2.3.4.1.5. Obtención o Compra
2.3.4.1.6. Entrega y soporte
2.3.5. Prácticas de ITIL
2.3.6. Principios Guías de ITIL
2.3.7. Conceptos de Gobernanza
2.3.7.1. Gobernanza de la organización
2.3.7.2. Cuerpo de Gobernanza
2.3.8. Conceptos de Mejora Continua
2.3.8.1. Modelo de Mejora Continua
2.3.8.1.1. Visión
2.3.8.1.2. Situación Actual
2.3.8.1.3. ¿A dónde queremos llegar?
2.3.8.1.4. ¿Cómo llegar ahí?
2.3.8.1.5. Acciones
2.3.8.1.6. Revisión de resultados
2.3.8.1.7. Mantener el momentum
2.4. Modelo de cuatro dimensiones
2.4.1. Personas y Organización
2.4.2. Información y Tecnología
2.4.3. Proveedores y aliados
2.4.4. Flujo de Valor y Procesos
2.5. Prácticas de Gestión de ITIL
2.5.1. Prácticas de Gestión Generales

2.5.2. Prácticas de Gestión de Servicio
2.5.3. Prácticas de Gestión Técnicas

3. COBIT 2019
3.1. Concepto de Objetivos de Gobierno y Gestión
3.2. COBIT como marco de Gobierno de la información y la tecnología
3.3. Visión General
3.4. Terminología y conceptos clave del marco de referencia COBIT 2019
3.4.1. Objetivos de Gobierno y Gestión
3.4.2. Componentes del Sistema de Gobierno
3.4.3. Áreas prioritarias
3.5. Cascada de metas
3.6. Componentes de Objetivos de Gobierno y Gestión
3.6.1. Definición de proceso
3.6.2. Estructuras organizativas
3.6.3. Flujos y elementos de información
3.6.4. Personas, habilidades y competencias
3.6.5. Políticas y procedimientos
3.6.6. Cultura, ética y comportamiento
3.6.7. Servicios, infraestructura y aplicaciones
3.7. Modelo fundamental de COBIT
3.7.1. Evaluar, Dirigir y Monitorizar
3.7.2. Alinear, Planificar y Organizar
3.7.3. Construir, Adquirir e Implementar
3.7.4. Entregar, Dar Servicio y Soporte
3.7.5. Monitorizar, Evaluar y Valorar

4. Arquitectura empresarial con TOGAF
4.1. Introducción y conceptos básicos
4.1.1. Fundamentos y arquitectura empresarial
4.1.2. Objetivos y alcance de la metodología
4.1.3. Glosario de términos de TOGAF
4.2. Arquitectura de empresa
4.2.1. Arquitectura empresarial y la importancia de su estudio
4.2.2. Principios de la arquitectura empresarial según TOGAF
4.2.3. Alineamiento de objetivos empresariales con TOGAF
4.3. Proceso de desarrollo de la arquitectura
4.3.1. Ciclo de vida del desarrollo de la arquitectura ADM
4.3.2. Fases del ADM
4.4. Guía de arquitectura y técnicas de desarrollo
4.4.1. Guías de implementación de las fases ADM
4.4.2. Técnicas y mejores prácticas para el desarrollo de la arquitectura empresarial

5. TAREAS DE CLASE
5.1. Code Review
5.1.1. Calidad
5.1.2. Código Limpio
5.2. RIsk Management
5.2.1. Vulnerabilidades/Amenazas
5.2.2. Matriz de Riesgo
5.2.3. Gestión del Riesgo

CLÁUSULAS RESTRICTIVAS:

El perfil del estudiante de la facultad de Ingeniería de la Universidad de San Carlos de Guatemala exige una alta calidad en la excelencia académica y ética profesional. Se establecen en este curso los siguientes lineamientos que regulan el comportamiento del estudiante:

- Copias en exámenes, cortos, proyectos, tareas e investigaciones tienen cero de nota.

- No se permite entregar como práctica o proyecto el código fuente disponible en cualquiera de los repositorios de código fuente (GitLab, GitHub).
- El uso no autorizado de cualquier herramienta de Inteligencia Artificial en las actividades del curso tendrá cero de nota.
- Exámenes parciales y examen final NO tienen reposición.
- No hay prórrogas.
- No hay reposición de proyectos.
- Cualquier proyecto, tarea o investigación que se entregue después de la fecha calendarizada tiene 30 puntos menos, cada día de atraso.
- Los exámenes resueltos a lápiz no tienen derecho a revisión.
- Es obligatorio ganar el laboratorio para tener derecho a evaluación total del curso.
- Para poder optar a la revisión de la zona final es obligatorio haber asistido a los exámenes parciales y al examen final.
- El catedrático podrá remitir cualquier regla si el estudiante demuestra atenuantes.
- Los videos de las prácticas deben mostrar y narrar claramente los pasos realizados para realizar la prácticas y para ser tomado en cuenta como entrega válida, debe cumplirse al menos el 50% de la práctica.

BIBLIOGRAFÍA:

- Software Architecture in Practice. Len Bass, Paul Clements y Rick Kazman.
- Software Systems Architecture. Nick Rozanski y Eoin Woods.
- Fundamentals of Software Architecture. Mark Richards y Neal Ford.
- Building Evolutionary Architectures. Neal Ford, Rebecca Parsons y Patrick Kua.
- Kubernetes Microservices With Docker. Deepak Vohra.Apress. 2015.
- Building Microservices. Sam Newman. O´Reylly. 2015.
- Kontonya, Gerald. Sommerville Ian. Requirements Engineering. Process and Techniques. John Wiley And Sons. 1998.
- Erl, Thomas. SOA Principles of Service Design. Prentice Hall Service-Oriented Computing Series. 2008.
- Jensen, Claus T. SOA Design principles for Dummies. IBM Limited Edition. John Wiley And Sons. 2013.
- Varios Autores. ITIL Foundation 4 Edition Axelos Limited. 2019.
- DevOps. Derek Rangel. 2015
- Cualquier otro material (escrito o digital) entregado en clase o producido por el catedrático o auxiliar. .