000027

Arquitectura de Software 1

INSTITUTOS/S: **INSTITUTO DE TECNOLOGÍA E INGENIERÍA**

CARRERA/S: **Licenciatura en Informática**

RESPONSABLE DE LA ASIGNATURA Y EQUIPO DOCENTE.
Luis Moyano

Ricardo Brea

AÑO: 2025

CRÉDITOS: 6

CARGA HORARIA DE INTERACCIÓN PEDAGÓGICA: 64

CARGA HORARIA TOTAL: 150

CÓDIGO DE LA MATERIA EN SIU:

000027

1. Fundamentación

La arquitectura de software es una disciplina que integra los conocimientos adquiridos en muchas de las asignaturas de la carrera, aportando sinergia entre los distintos componentes de un sistema.

Esta asignatura se centra en los elementos fundamentales de la arquitectura, con especial énfasis en conceptos como acoplamiento, cohesión y su evolución en los modelos arquitectónicos contemporáneos. A través de estos ejes, se abordan temas clave que permiten comprender cómo las decisiones arquitectónicas impactan en la calidad, mantenibilidad y escalabilidad de los sistemas.

El enfoque del curso combina teoría con ejemplos prácticos, integrando tecnologías actuales que favorecen el análisis crítico de distintas alternativas arquitectónicas, según los contextos y restricciones que los estudiantes podrían enfrentar en su futuro profesional.

De este modo, la asignatura contribuye al desarrollo de competencias clave en la toma de decisiones, la comunicación técnica y la evaluación de soluciones arquitectónicas, preparando a las/los futuras/os profesionales para enfrentar escenarios complejos con una base sólida y actualizada.

2. Propósitos y/u objetivos

- *Comprender el rol y las responsabilidades del arquitecto de software, y su interacción con los distintos actores del proyecto.*
- *Determinar la arquitectura más adecuada en función de los requerimientos funcionales y no funcionales, así como de las restricciones del proyecto.*
- *Identificar las mejores estrategias de definición arquitectónica de acuerdo con el tipo de sistema de información a desarrollar.*

- *Conocer y aplicar herramientas conceptuales y técnicas para asistir en el diseño de arquitecturas de software.*

- *Comunicar eficazmente la arquitectura del sistema, justificando las decisiones tomadas y su alineación con los requerimientos del sistema.*

- *Conocer y evaluar los principales patrones arquitectónicos, comprendiendo sus ventajas, desventajas y adecuación a distintos contextos.*

- *Evaluar los aspectos de seguridad del sistema como parte integral del diseño arquitectónico.*

- *Utilizar herramientas tecnológicas que apoyen la implementación y verificación de las decisiones arquitectónicas adoptadas.*

3. Contenidos mínimos

Arquitecturas en diferentes metodologías de desarrollo. Requerimientos funcionales y no funcionales, restricciones, influencias, entorno social y técnico, estándares, herramientas disponibles. Objetivos de una arquitectura. Estilos arquitectónicos. Arquitectura de dominio. Arquitectura y Diseño. Patrones. Patrones arquitecturales para la interfaz de usuario. Integración con el dominio. Internacionalización. Arquitecturas orientadas a servicios. Arquitecturas extensibles. Arquitecturas basadas en plugins. Arquitecturas de seguridad. Estrategias de verificación de arquitecturas. Arquitecturas concurrentes y distribuidas. Herramientas tecnológicas para soportar las decisiones arquitectónicas.

4. Carga horaria

Créditos	*Interacción pedagógica*		*Trabajo Autónomo*	*Total*
6	*Teoría: 32*	*Práctica: 32*	*86*	*150*

4.1. Trabajo autónomo de la/el estudiante

Actividad	*Carga horaria*

Resolución de cuestionarios de autoevaluación	*10*
Réplica de los ejercicios prácticos vistos en clase	*10*
Investigación soluciones viables del trabajo práctico	*10*
Implementación manual de la solución	*26*
Implementación automatizada de la solución	*20*
Documentación de la solución final	*10*

5. Programa analítico

UNIDAD 1: *Fundamentos de la Arquitectura de Software.*

Concepto y objetivos de la arquitectura de software y su diferencia con el diseño. Responsabilidades del arquitecto. Principios de arquitectura. Requerimientos funcionales y no funcionales, restricciones. Estilos arquitectónicos. Sinergia, acoplamiento, cohesión y quantum. Diagramas y modelos arquitectónicos (UML, C4). Matriz de Pugh. Influencias del entorno técnico y social. Cuestiones humanas, sociales y organizacionales en torno a la arquitectura. Trabajo en equipos multidisciplinarios.

UNIDAD 2: *Diseño e Integración de Componentes.*

Componentes: lógica de dominio, interfaz de usuario, persistencia, seguridad, entre otros. División y Extensión de las funcionalidades de un sistema. Modular-monolitico, micro-kernel / plug-in). Patrones arquitectónicos para la lógica de negocio (Domain-Driven Design, Clean Architecture). Persistencia: Active Record, ORM (Entity Framework), configuración y convención. Integración y comunicación entre componentes (AOP, transporte, transacciones). Interfaz de usuario: patrón Observer, internacionalización, accesibilidad, MVC, MVVM. Clientes livianos y pesados.

UNIDAD 3: *Integración de Aplicaciones y Arquitecturas Orientadas a Servicios.*

Mecanismos de integración. Integración sincrónica y asincrónica: colas de mensajes, callbacks. Arquitecturas concurrentes y distribuidas: Arquitecturas orientadas a servicios, microservicios, eventos, space-based, middleware y Enterprise Service Bus. Coreografía y orquestación. Manejo de transacciones y compensaciones.

UNIDAD 4: *Diseño de Interfaces y APIs.*

Tecnologías de API: REST S, SOAP, ODATA, Grpc. Diseño de APIs REST y buenas prácticas: Semántica HTTP, RFC7807 Open API, Versionado. Documentación autogenerada. Servicios de directorio / Service Discovery. Patrones de interfaz de usuario aplicados a APIs (SPAs / Aplicaciones Mobiles).

UNIDAD 5: *Arquitecturas de Seguridad.*

Autenticación, autorización, control de acceso basado en roles (RBAC). Patrones y estándares: OIDC, SAML. Perspectivas de seguridad: web, sistema operativo, base de datos, middleware. Cifrado y Seguridad de transporte (TLS). Seguridad multifactorial: OTP, WebAuthn. Seguridad máquina a máquina (M2M).

UNIDAD 6: E*valuación de Arquitecturas.*

Evaluación formal de arquitecturas. Verificación de requerimientos no funcionales. Verificación automática de una arquitectura. Herramientas para pruebas de arquitectura. Registro de decisiones arquitectónicas (ADR).

6. Ejes y enunciados Multidimensionales y transversales

Eje	*Nivel de logro de Aprendizaje*	*Acciones de enseñanza*
C1. Identificación, formulación y resolución de problemas de informática	*Alto*	*Resolución de problemas prácticos utilizando los conceptos teóricos.*
C2. Concepción, diseño y desarrollo de proyectos de informática	*Alto*	*Aprendizaje basado en proyectos.*
C3. Gestión, planificación, ejecución y control de proyectos de informática	*Medio*	*Aprendizaje basado en proyectos.*
C4. Utilización de técnicas y herramientas de aplicación en la informática	*Alto*	*Clases prácticas con herramientas de software.*
C5. Generación de desarrollos tecnológicos y/o innovaciones tecnológicas	*Medio*	*Clases expositivas dialogadas.*

C6. Fundamentos para el desempeño en equipos de trabajo	*Alto*	*Clases teóricas y prácticas.*
C7. Fundamentos para la comunicación efectiva	*Alto*	*Método de casos.*
C8. Fundamentos para la acción ética y responsable	*Bajo*	*Clases expositivas dialogadas.*
C9. Fundamentos para evaluar y actuar en relación con el impacto social de su actividad en el contexto global y local	*Medio*	*Aprendizaje orientado a proyectos.*
C10.Fundamentos para el aprendizaje continuo	*Alto*	*Aprendizaje cooperativo.*
C11.Fundamentos para la acción emprendedora	*Alto*	*Aprendizaje cooperativo.*

7. Bibliografía y recursos

7.1 Bibliografía obligatoria

Richards, Mark; Ford, Neal. (2020). "Fundamentals of Software Architecture" (4ta Edition 2021). O'Reilly Media, Inc.

Vlad, Khononov. (2021). "Learning Domain Driven Design" (1ra Edición 2021). O'Reilly Media, Inc.
Disponible en: https://pdfhost.io/v/WNX5ob0vd_Learning_DomainDriven_Design

Bass, Len; Clements, Paul; Kazman, Rick. (2021). "Software Architecture in Practice". 4ta Edition. Addison-Wesley Professional.

Sánchez Suarez, Jose Manuel. (2003). "Diagramas de arquitectura con C4 model" https://adictosaltrabajo.com/2023/05/06/diagramas-de-arquitectura-con-c4-model/

Martin, Robert Cecil. (2018). "Clean Architecture: A CRAFTSMAN'S GUIDE TO SOFTWARE STRUCTURE AND DESIGN".

7.2 Bibliografía optativa

Fowler, Martin; Rice, David; Foemmel, Matthew; Hieatt, Edward; Mee, Robert; Stafford.

Randy. (2002). "Patterns of Enterprise Application Architecture". Addison-Wesley.

Hohpe, Gregor; Woolf, Bobby. (2003). "Enterprise Integration Patterns: Designing, Building and Deploying Messaging Solutions". Addison-Wesley.

Richards, Mark; Ford, Neal; Sadalage, Pramod y Dehghani, Zhamak. (2021). "Software Architecture: The Hard Parts". O'Reilly Media, Inc.

Fowler, Martin. (2002). "Who Needs an Architect?". Martin Fowler's web. https://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf

Richards, Mark. (2016). "Microservices vs. Service-Oriented Architecture". O'reilly web. https://www.oreilly.com/radar/microservices-vs-service-oriented-architecture/

7.3 Recursos

En el aula virtual se propondrá material educativo; Presentaciones que el/la docente emplea durante las clases. Vinculo al repositorio de código (abierto) con los ejemplos prácticos presentados en clase. Videos y textos introductorios o ampliatorios de los temas tratados. Además, se encuentran cuestionarios de autoevaluación.

Adicionalmente se utilizan las siguientes herramientas de software:

***Visual Studio Code:** IDE de programación código multiplataforma y de código abierto desarrollado por Microsoft, utilizado principalmente para desarrollo de aplicaciones y programación en múltiples lenguajes. (https://github.com/microsoft/vscode)*

***GIT:** Sistema de control de versiones distribuido, multiplataforma y de código abierto, utilizado para gestionar cambios en el código y colaborar en proyectos de software. (https://git-scm.com/downloads/guis).*

***Docker Desktop:** Entorno de desarrollo multiplataforma de Docker que proporciona una interfaz visual para trabajar con contenedores Docker. (https://www.docker.com/products/docker-desktop/).*

Podman Desktop: *Entorno de desarrollo de contenedores con interfaz visual, basado en el motor de contenedores Podman y de código abierto. Es una alternativa a Docker Desktop. (https://github.com/containers/podman-desktop).*

8. Metodología de enseñanza

8.1 Modalidades u opciones pedagógicas

Para cada unidad temática, la diagramación es la siguiente:
Introducción del tema: Se introduce a la problemática detrás de los temas expuestos
Exposición y demostración de contenidos teóricos: Se presentará la idea principal indicando cuál es el objetivo y el camino a seguir. Se expresarán los fundamentos, estándares y/o tendencias tecnológicas actuales.
Diálogo con los alumnos para determinar el grado de interpretación de los conceptos enseñados: La finalidad de esto es reforzar aquellos conceptos que no fueron comprendidos en su totalidad. El diálogo posterior sobre el desarrollo logrado permitirá la extracción de conclusiones y una síntesis de lo explicado. Ejemplos de aplicación lograrán consolidar los conocimientos impartidos.

8.2 Formación Práctica

Durante las porciones práctica se muestran ejemplos prácticos de la teoría que luego deberán ser aplicados en el trabajo práctico final.
Se realizan trabajos de: Diagramas de sistemas con el método C4. Matriz de Pugh.
Patrones. Contenedores, Docker. Implementación declarativa de servicios en Docker y docker-compose. Implementación de Identity Provider OIDC con Docker-Compose. Ejemplo práctico de escaneo estático de código.

9. Evaluación y régimen de aprobación

9.1 Modalidad de evaluación

El sistema normal de evaluación consistirá en 1 exámenes parcial y un trabajo práctico con un recuperatorio por cada uno de ellos, según el cronograma previsto. Tanto en los parciales como en los recuperatorios, los contenidos a evaluar representan la totalidad de la materia descripta en el programa propuesto en el campus. Los mismos se realizarán en las fechas, que, a tal efecto, se establezcan en el cronograma correspondiente.

Para la evaluación de los contenidos prácticos del programa, el/la estudiante deberá acordar previamente con el/la docente un trabajo práctico a presentar el día del examen final. De no hacerlo, deberá realizar ejercicios prácticos el mismo día de la evaluación.

9.2 Aprobación de la cursada

Para aprobar la cursada y obtener la condición de regular, el régimen académico establece que debe obtenerse una nota no inferior a cuatro (4) puntos. Todas las instancias evaluativas deberán tener una instancia de recuperatorio. Podrán acceder a la administración de esta modalidad solo aquellos y aquellas estudiantes que hayan obtenido una nota inferior o igual a 6 (seis) puntos en el examen parcial.

Siempre que se realice una evaluación de carácter recuperatorio, la calificación que los/as estudiantes obtengan reemplazará la calificación obtenida en el examen que se ha recuperado y será la considerada definitiva a los efectos de la aprobación.

9.3 Acreditación de la materia

La materia puede aprobarse por promoción, evaluación integradora, examen final o libre.

Promoción directa: *tal como lo establece el art°17 del Régimen Académico, para acceder a esta modalidad, el/la estudiante deberá aprobar la cursada de la materia con una nota no inferior a siete (7) puntos, no obteniendo en ninguna de las instancias de evaluación parcial menos de seis (6) puntos, sean evaluaciones parciales o recuperatorios. El promedio estricto resultante deberá ser una nota igual o superior a siete (7) sin mediar ningún redondeo.*

Evaluación integradora: *tal como lo establece el art°18 del Régimen Académico, podrán acceder a esta evaluación aquellos estudiantes que hayan aprobado la cursado con una nota de entre cuatro (4) y seis (6) puntos.*
La evaluación integradora tendrá lugar por única vez en el primer llamado a exámenes finales posterior al término de la cursada. Deberá tener lugar en el mismo día y horario de la cursada y será administrado, preferentemente, por el/la docente a cargo de la comisión. Se aprobará tal instancia con una nota igual o superior a cuatro (4) puntos, significando la aprobación de la materia. La nota obtenida se promediará con la nota de la cursada. En caso de que el/la estudiante no haya aprobado el trabajo práctico, deberá acordar

previamente con el/la docente un trabajo práctico complementario, que deberá presentarse el día de la evaluación integradora. De no hacerlo, deberá realizar ejercicios prácticos el mismo día de la evaluación integradora.

***Examen final**: Instancia destinada a quienes opten por no rendir la evaluación integradora o hayan regularizado la materia en cuatrimestres anteriores. Se evalúa la totalidad de los contenidos del programa de la materia y se aprueba con una calificación igual o superior a cuatro (4) puntos. Esta nota no se promedia con la cursada.*