000027

Arquitectura de Software 2

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

CARRERA/S: **Licenciatura en Informática**

RESPONSABLE DE LA ASIGNATURA Y EQUIPO DOCENTE:
Daniel Buaón
Ricardo Brea

AÑO: 2025

CRÉDITOS: 5

CARGA HORARIA DE INTERACCIÓN PEDAGÓGICA: 64

CARGA HORARIA TOTAL: 125

CÓDIGO DE LA MATERIA EN SIU:

000027

1. Fundamentación

La arquitectura de software es una disciplina que integra conocimientos adquiridos en distintas asignaturas de la carrera, generando sinergia entre los diversos componentes de un sistema.

Uno de los ejes principales de la arquitectura de software son los requerimientos no funcionales. En esta segunda etapa de la asignatura se hace énfasis en la eficiencia, la eficacia y la confiabilidad. Estos requerimientos se abordan mediante arquitecturas distribuidas que, si bien ofrecen mayor capacidad de respuesta ante la demanda, también introducen mayor complejidad en su diseño, implementación y mantenimiento. En este contexto, la incorporación de enfoques para el ciclo de vida como DevOps y técnicas como GitOps, resultan necesarios para garantizar la evolución continua del sistema, permitiendo incorporar nuevas funcionalidades de forma segura, automatizada y sostenible en el tiempo. Se incluyen ejemplos prácticos utilizando tecnologías actuales, como microservicios y servicios en la nube, con el objetivo de ofrecer herramientas que habiliten un análisis crítico de distintas alternativas tecnológicas según los escenarios que pueden presentarse en el ejercicio profesional.

2. Propósitos y/u objetivos

- *Tomar decisiones en contextos arquitectónicos.*
- *Evaluar riesgos y costos asociados a arquitecturas distribuidas.*
- *Evaluar los requerimientos no funcionales al momento de diseñar la arquitectura.*
- *Realizar un análisis crítico de distintas alternativas tecnológicas, basado en los requerimientos y libre de sesgos.*
- *Comprender los conceptos de eficiencia y eficacia dentro del marco de normas y estándares vigentes.*
- *Identificar problemas de performance y proponer posibles soluciones en sistemas ya implementados.*

- *Conocer tecnologías actuales, sus ventajas y desventajas, para diseñar, desarrollar e implementar sistemas considerando metodologías ágiles.*

- *Conocer enfoques modernos del ciclo de vida del software como DevOps, DevSecOps, GitOps.*

3. Contenidos mínimos

Escalabilidad, eficiencia y efectividad. Dimensionamiento de los requerimientos de hardware y de las necesidades de red de un sistema de software. Técnicas para escalamiento vertical y horizontal. Clustering, balanceo de carga, afinidad, sharding. Estrategias de particionamiento de bases de datos. Tolerancia a fallos. Estrategias de cache de datos. Nociones de minería de datos. Hardware específico para sistemas de gran envergadura. Virtualización. Software y hardware como servicios. Verificación del cumplimiento de los requerimientos no funcionales: performance, tolerancia a fallos, carga. Operación y monitoreo de sistemas. Estrategias de logging para sistemas de gran envergadura. Herramientas para medición de performance. Profiling. Información caliente e información de ciclo de vida largo. Análisis de servicios en red, análisis de tráfico. Herramientas de monitoreo de fallas. Metodología DevOps.

4. Carga horaria

Créditos	*Interacción pedagógica*		*Trabajo Autónomo*	*Total*
5	*Teoría: 24*	*Práctica: 40*	*61*	*125*

4.1. Trabajo autónomo de la/el estudiante

Actividad	*Carga horaria*
Resolución de cuestionarios de autoevaluación	*8*
Réplica de los ejercicios prácticos vistos en clase	*8*
Investigación soluciones viables del trabajo práctico	*8*
Implementación manual de la solución	*16*
Implementación automatizada de la solución	*16*
Documentación de la solución final	*5*

5. Programa analítico

UNIDAD 1: *Eficiencia, Eficacia.*

Definición de eficiencia y eficacia. Requerimientos no funcionales: performance, tolerancia a fallos, confiabilidad, carga. Estándares de calidad. Verificación del cumplimiento de los requerimientos no funcionales.

UNIDAD 2: *Escalabilidad y Arquitectura Distribuida.*

Escalabilidad: definición, vertical vs horizontal (ventajas, desventajas, casos de uso). Paralelismo, uso de CPU multinúcleo. Técnicas: clustering, balanceo de carga, afinidad. Estrategias de particionamiento de bases de datos y sharding. Estrategias de cacheo de datos.

UNIDAD 3: *Confiabilidad, Tolerancia a Fallos y Alta Disponibilidad.*

Definición del requerimiento no funcional: Confiabilidad. Tolerancia a fallos (Fault Tolerance) y alta disponibilidad (High Availability). Redundancia, punto único de falla, detección de fallos. Medición de disponibilidad. Recuperación ante desastres. Buenas prácticas de despliegue para alta disponibilidad. Clustering en Kubernetes.

UNIDAD 4: *Observabilidad.*

Técnicas de profiling y análisis de performance. Dimensionamiento de recursos de hardware y red. Observabilidad como clave para mejora continua. Identificación de cuellos de botella. Logging: estructura, tecnologías, auditoría vs logging, buenas prácticas. Logs distribuidos e ID de correlación. Herramientas de monitoreo y visualización (bases de datos de series de tiempo, dashboards).

UNIDAD 5: *Cloud Computing y Arquitecturas Contenerizadas.*

Concepto de Cloud Computing. Modelos de implementación: IaaS, PaaS, SaaS. Almacenamiento en la nube: caliente y de ciclo de vida largo. Costeo de infraestructura en la nube (AWS, Azure). Comparativa Virtualización vs Contenedores. Consideraciones en el Diseño de Software como Servicio: configuración, concurrencia, estado, puertos, recursos, seguridad y homogeneidad de ambientes. Protocolos: SSL, TLS, HTTPS, reverse proxy, modelo OSI.

UNIDAD 6: *Ciclo de vida y automatización en sistemas distribuidos.*

DevOps: historia, fundamentos, relación con metodologías ágiles. Integración continua (CI), entrega continua (CD), infraestructura como código. Control de versiones (GIT), estrategias de branching. DevSecOps, seguridad desde el inicio, análisis estático, OWASP. Implementación continua, criterios de aceptación, testing automatizado.

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 ejercicios y problemas.*
C2. Concepción, diseño y desarrollo de proyectos de informática	*Alto*	*Aprendizaje basado en problemas.*
C3. Gestión, planificación, ejecución y control de proyectos de informática	*Alto*	*Resolución de ejercicios y problemas.*
C4. Utilización de técnicas y herramientas de aplicación en la informática	*Alto*	*Resolución de ejercicios y problemas aprendizaje basado en proyectos.*
C5. Generación de desarrollos tecnológicos y/o innovaciones tecnológicas	*Medio*	*Clases teóricas, Resolución de ejercicios y problemas.*
C6. Fundamentos para el desempeño en equipos de trabajo	*Alto*	*Clases teóricas y clases prácticas.*
C7. Fundamentos para la comunicación efectiva	*Alto*	*Aprendizaje orientado a proyectos.*
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	*Bajo*	*Resolución de ejercicios y problemas aprendizaje basado en proyectos.*
C10.Fundamentos para el aprendizaje continuo	*Alto*	*Debates.*

C11.Fundamentos para la acción emprendedora	*Alto*	*Aprendizaje cooperativo.*

7. Bibliografía y recursos

7.1 Bibliografía obligatoria

Len Bass, Paul Clements, Rick Kazma. (2021). "Software Architecture in Practice". 4ta Edición. Addison-Wesley Professional. EEUU

David Farley, Jez Humble. (2010). "Continuous delivery: reliable software releases through build, test, and deployment automation". Addison-Wesley Professional. EEUU.

Richards, Mark; Ford, Neal. (2020). "Fundamentals of Software Architecture" (4ta Edición 2021). O'Reilly Media, Inc.
Disponible en:
https://dokumen.pub/fundamentals-of-software-architecture-an-engineering-approach-1nbsped-1492043451-9781492043454.html

7.2 Bibliografía optativa

Martin Fowler, David Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, Randy Stafford. (2002). "Patterns of Enterprise Application Architecture". Addison-Wesley. EEUU.

C.J. Date. (2001). "Introducción a los sistemas de base de datos". PEARSON EDUCACIÓN, México.

Betsy Beyer, Chris Jones, Jennifer Petoff and Niall Richard Murphy. (2016). "Site Reliability Engineering". O'Reilly Media, Inc.

Betsy Beyer, Niall Richard Murphy, David K. Rensin, Kent Kawahara, Stephen Thorne. (2016). "The Site Reliability Workbook" . O'Reilly Media, Inc.

Peter Mell, Timothy Grance. (Septiembre 2011). "The NIST Definition of Cloud Computing". NIST Technical Series Publications. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Andrej Dyck; Ralf Penners; Horst Lichter. (2015). "Towards Definitions for Release Engineering and DevOps". (Mayo 2015). Institute of Electrical and Electronics Engineers. https://www2.swc.rwth-aachen.de/docs/2015_RELENG_Dyck.pdf

7.3 Recursos

Documentación de Kubernetes. Sitio oficial de Kubernetes. https://kubernetes.io/es/docs/home/

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.

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).*

***Rancher Desktop**: Plataforma de administración de contenedores y Kubernetes de código abierto y multiplataforma. Permite desarrollar, probar y administrar entornos Kubernetes localmente, con soporte para Docker y otros motores de contenedores. (https://github.com/rancher-sandbox/rancher-desktop).*

8. Metodología de enseñanza

8.1 Modalidades u opciones pedagógicas

Para cada unidad temática la diagramación es la siguiente:
Parte Teórica:
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 por 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

En las instancias prácticas se presentan ejemplos aplicados de los contenidos teóricos, los cuales deberán ser implementados posteriormente en el trabajo práctico final.

- *Práctica-Unidad 2: Escalabilidad en Kubernetes (Pods, Deployments, Services, ConfigMaps, secrets).*
- *Práctica – Unidad 3: Probes en Kubernetes (Liveness, Readiness, Volumes).*
- *Práctica – Unidad 4: Integración y agregación de logs, herramientas de monitoreo.*
- *Práctica – Unidad 5: Ingress Controller en Kubernetes, certificados, Let's Encrypt.*
- *Práctica – Unidad 6: CI/CD con microservicios, certificados, despliegue automatizado.*

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. Los mismos se realizarán en las fechas, que, a tal efecto, se establezcan en el cronograma correspondiente que se encuentra en el campus virtual.

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. 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.