PATRONES DE ARQUITECTURAS DE SOFTWARE

Año 2025

Carrera/ Plan: ***(Dejar lo que corresponda)***

Licenciatura en Informática Plan 2021/Plan 2015
Licenciatura en Sistemas Plan 2021/Plan 2015

Año: 2025
Régimen de Cursada: *Semestral (primer semestre)*
Carácter (Obligatoria/Optativa): Optativa
Correlativas: Proyecto de Software - Orientación a Objetos 2 - Ingeniería de Software 2

Profesor/es: Luis Mariano Bibbo
Hs. semanales teoría: 4 hs.
Hs. semanales práctica: 4 hs.

FUNDAMENTACIÓN

La Arquitectura de Software es el diseño de alto nivel de la estructura de un sistema. Durante la etapa inicial de un proyecto de software, existen decisiones tempranas que sientan las bases de diseños técnicos futuros y que por ende son costosas de modificar. Estas decisiones son las que conforman la arquitectura de la aplicación y es importante conocer los diferentes estilos existentes, las ventajas y desventajas de cada uno, para elegir él o los más adecuados en cada caso.

Los diseñadores de arquitecturas de software tienen que prestar especial atención a varios aspectos que intervienen en el producto a desarrollar, dado que no solo deberán atender los aspectos funcionales de las aplicaciones, sino que tendrán que trabajar en cuestiones de eficiencia, escalabilidad, performance, seguridad, entre otras.

Los arquitectos de software suelen enfrentar problemas que ocurren una y otra vez en distintos proyectos de software; como dividir a la aplicación en capas, manejar la concurrencia de acceso a recursos o vincular la aplicación con la estructura de datos. En estos casos se recurre a soluciones que resultaron exitosas en otros casos similares. Al implementar esta modalidad el arquitecto o diseñador aplicó un patrón de diseño.

OBJETIVOS GENERALES

A partir del dictado de esta materia se pretende que los alumnos:

- Aprender a identificar tempranamente las decisiones que hacen a la arquitectura que por definición son costosas de modificar en el futuro.
- Conocer los desafíos y problemas que dan lugar a la evolución de los estilos de arquitectura.
- Conocer ventajas y desventajas de cada uno de los estilos arquitectónicos vigentes.
- Aprender a poner en contexto cada estilo arquitectónico con el objetivo de elegir el más adecuado.

RESULTADOS DE APRENDIZAJE

Que los alumnos puedan elegir y utilizar modelos de procesos adecuados y entornos de programación relacionados con el diseño y desarrollo de arquitecturas de software.

Que los alumnos puedan describir y explicar diseños de sistemas donde se involucre la arquitectura de estos.

Que los alumnos puedan describir y explicar las técnicas de gestión correspondientes al diseño, implementación, análisis, uso y mantenimiento de sistemas informáticos, incluyendo gestión de proyectos, de configuración y de cambios, así como las técnicas de automatización correspondientes.

COMPETENCIAS

Concebir, diseñar y desarrollar aspectos relacionados con las arquitecturas de software dentro de los proyectos de informática.

Conocer e interpretar los conceptos, teorías y métodos relativos al diseño de arquitecturas de software.

Capacidad de organizar equipos de trabajo relacionados con el desarrollo de arquitecturas de software en proyectos de informática.

CONTENIDOS MINIMOS (de acuerdo al Plan de Estudios)

- ✓ Arquitectura
- ✓ Requerimientos no funcionales que afectan a la arquitectura
- ✓ Monolito
- ✓ Microservicios
- ✓ Interoperabilidad e Integración de sistemas
- ✓ Servicios de infraestructura para despliegue de aplicaciones

PROGRAMA ANALÍTICO

Unidad 1: Introducción a arquitectura

- Definiciones de arquitectura de software
- Alcance de la arquitectura de software
- Componentes, módulos, acoplamiento, dependencia, cohesión
- Diseño e implementación de arquitectura de software

Unidad 2: Requerimientos para definir una arquitectura

- Análisis de los principales requerimientos no funcionales que afectan decisiones de arquitectura: Performance, Seguridad, UIX, Evolución, Concurrencia, Disponibilidad, capacidad, escalabilidad, portabilidad, hot deploy, cantidad de usuarios, tiempos de respuesta, SLAs.
- Trade-off entre requerimientos no funcionales
- Evaluación para medir el cumplimiento de requerimientos no funcionales
- "Cost as a fitness function"

Unidad 3: Arquitectura de software modernas

- Desafíos técnicos
    - Diferentes formas de despliegue (local o cloud).
    - Agilidad para desarrollo y despliegue de nueva funcionalidad.
    - Distribución e interoperabilidad.
    - Requerimientos de escalabilidad.
    - Equipos de desarrollo de mediano/gran tamaño (5-200 personas).
    - Múltiples clientes (aplicaciones web, mobile, otros sistemas).
    - Grandes volúmenes de datos.
    - Seguridad. Protocolo de autenticación. Soluciones de autorización de operaciones y de datos.

Unidad 4: Layers en Modelo monolito

- Evolución de las arquitecturas en capas.
- Descripción de las capas principales.
- ¿Cuándo es útil? Ventajas y desventajas.
- Que eventos dispararon la aparición de modelos alternativos al monolito.

Unidad 5: Modelo basado en Microservicios:

- Características del modelo. Ventajas y desventajas.
- Objetivos principales y problemas que resuelve.
- ¿Cuándo usar microservicios?

- Estrategias de modularización.
- Estrategias de modularización del modelo de dominio.
- Ciclo de vida
- Diseño de interfaces
  - data on the inside vs data on the outside
  - evolución de las APIs
- Desafíos y problemas frecuentes a resolver con microservicios.

Unidad 6: Interoperabilidad e integración

- Seguridad
- Invocación de servicios privados y públicos
- Invocación sincrónica y asincrónica.
- Protocolos y formatos
- Protocolos de interoperabilidad específicos de dominio.
- ESB

Unidad 7: Infraestructura

- Serverless
  - Functions as a service.
  - Container as a service.
- Estado
  - Event-sourcing.
  - Streams como otro modelo de arquitectura.
- Distribución
- Contenedores

Unidad 8: Evolución y mantenibilidad de arquitectura

- Casos típicos de evolución de un sistema
- Cómo se implementa en un monolito
- Cómo se implementa con microservicios

BIBLIOGRAFÍA

- Patterns of Enterprise Application Architecture. Martin Fowler. Addison-Wesley Professional, 2002, ISBN 0-321-12742-0.
- Software Architecture in Practice, Second Edition. Len Bass, Paul Clements, Rick Kazman. Addison Wesley, 2003, ISBN 0-321-15495-9.
- Clean Architecture: A Craftsman's Guide to Software Structure and Design, Robert C. Martin, September 2017, Publisher(s): Pearson, ISBN: 9780134494272
- Design It!: From Programmer to Software Architect; Michael Keeling, Published: October 2017- ISBN: 9781680502091
- Release It!: Design and Deploy Production-Ready Software (Pragmatic Programmers); Michael T. Nygard, Released March 2007, Publisher(s): Pragmatic Bookshelf, ISBN: 9780978739218
- Designing Data-Intensive applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems; Martin Kleppmann, Released March 2017, Publisher(s): O'Reilly Media, Inc. ISBN: 9781449373320
- Building Microservices: Designing Fine-Grained Systems, February 2015 Publisher(s): O'Reilly Media, Inc. ISBN: 9781491950357
- Microservices patterns: With examples in Java; Chris Richardson, October 2018, ISBN 9781617294549
- Vural H., Koyuncu M., Guney S. (2017) A Systematic Literature Review on Microservices. In: Gervasi O. et al. (eds) Computational Science and Its Applications – ICCSA 2017. Lecture Notes in Computer Science, vol 10409. Springer, Cham. https://doi.org/10.1007/978-3-319-62407-5_14

METODOLOGÍA DE ENSEÑANZA

Se organizan clases teóricas en formato digital en las cuales el docente, con participación de los alumnos, presentará los temas del programa. Además, se establecerán distintos ejercicios a desarrollar que guiarán a los alumnos en la concreción de los objetivos anteriormente planteados para que los resuelvan durante el transcurso de la materia. Promediando la cursada se establecerá una modalidad de seminario donde los alumnos presentarán un conjunto de patrones y se discutirán la posible utilidad de estos en distintas situaciones que se irán planteando. Los alumnos participarán activamente interactuando con los compañeros y el cuerpo docente. Se dispondrá de un entorno virtual para descargar los materiales de clase y hacer un seguimiento de las consultas de los alumnos.

EVALUACIÓN

La evaluación de la cursada comprende la aprobación de una serie de ejercicios obligatorios (2-3) o pruebas de concepto donde los alumnos implementarán algunos patrones vistos en las clases teóricas.
Para la evaluación final es requisito que los alumnos que aprobaron la cursada realicen un trabajo final donde los alumnos tengan que elegir y utilizar los modelos de arquitectura de software adecuados. Asimismo, deberán describir los diseños de arquitectura realizados y explicar las técnicas de análisis, diseño e implementación utilizados.

Como resultado de la evaluación final se les pasará una nota global que comprende los ejercicios obligatorios y el trabajo final de la materia.

CRONOGRAMA DE CLASES Y EVALUACIONES

Clase	Fechas	Contenidos/Actividades
1	11/3/2025	Exposición teórica de los siguientes temas: *Introducción a arquitectura* • Definiciones de arquitectura de software • Alcance de la arquitectura de software • Componentes, módulos, acoplamiento, dependencia, cohesión • Diseño e implementación de arquitectura de software
2		Exposición teórica de los siguientes temas: *Requerimientos para definir una arquitectura* • Análisis de los principales requerimientos no funcionales que afectan decisiones de arquitectura: Performance, Seguridad, UIX, Evolución, Concurrencia, Disponibilidad, capacidad, escalabilidad, portabilidad, hot deploy, cantidad de usuarios, tiempos de respuesta, SLAs. • Trade-off entre requerimientos no funcionales • Evaluación para medir el cumplimiento de requerimientos no funcionales • "Cost as a fitness function" • Definición de Componentes, módulos, acoplamiento, dependencia, cohesión • Diseño e implementación de arquitectura de software Presentación del TP1: Requerimientos de una Arquitectura
3		Revisión/Consulta del avance del ejercicio planteado.
4		Exposición teórica de los siguientes temas: *Arquitectura de software modernas* • Desafíos técnicos
5	15/4/2025	Entrega del TP1. Exposición teórica de los siguientes temas: Requerimientos de una Arquitectura • Desafíos Técnicos
6		Exposición teórica de los siguientes temas: *Layers en Modelo monolito* • Evolución de las arquitecturas en capas. • Descripción de las capas principales. • ¿Cuándo es útil? Ventajas y desventajas. • Que eventos dispararon la aparición de modelos alternativos al monolito. Presentación de TP2: Diseño de prueba de concepto de arquitectura monolito
7		Consulta/Revisión del avance del TP2.
8	20/5/2025	Seminario:

		• Presentación de los grupos de alumnos sobre el TP2.
9		Exposición teórica de los siguientes temas: Modelo basado en Microservicios: • Características del modelo. Ventajas y desventajas. • Objetivos principales y problemas que resuelve. • ¿Cuándo usar microservicios? • Estrategias de modularización. • Estrategias de modularización del modelo de dominio. • Ciclo de vida • Diseño de interfaces • Desafíos y problemas frecuentes a resolver con microservicios.
10		Exposición teórica de los siguientes temas: *Interoperabilidad e integración* • Seguridad • Invocación de servicios privados y públicos • Invocación sincrónica y asincrónica. • Protocolos y formatos • Protocolos de interoperabilidad específicos de dominio. • ESB
11		Exposición teórica de los siguientes temas: *Infraestructura* • Serverless • Estado • Distribución • Contenedores Presentación TP3. Diseño e implementación de arquitectura de Microservicio.
12		Revisión/Consulta del avance del ejercicio planteado TP3.
13	19/06/2025	Seminario: • Presentación de los grupos de alumnos sobre el TP3.
14		Exposición teórica de los siguientes temas: *Evolución y mantenibilidad de arquitectura* • Casos típicos de evolución de un sistema • Cómo se implementa en un monolito • Cómo se implementa con microservicios
15		Presentación / Discusión del trabajo final de la materia que involucra todos los conceptos tratados durante las clases teóricas, la bibliografía sugerida y en los trabajos prácticos realizados
16		Revisión/Consulta del avance del ejercicio planteado como trabajo final.
17		Revisión/Consulta del avance del ejercicio planteado como trabajo final.
18		Revisión/Consulta del avance del ejercicio planteado como trabajo final.
	08/07/2025	

Evaluaciones Previstas	Fechas
Evaluación del TP1 entregado	15/4/2025
Evaluación del TP2 entregado	20/5/2025
Evaluación del TP3 entregado	19/06/2025
Evaluación de los trabajos finales presentados	12/07/2025

BIBLIOGRAFÍA COMPLEMENTARIA

- Software Architecture, Perspectives on an Emerging Discipline. M. Shaw and D. Garlan. Prentice-Hall. 1996.
- Evaluating Software Architectures: Methods and Case Studies. Len Bass, Paul Clements, Rick Kazman. Addison Wesley
- The Art of Software Architecture: Design Methods and Techniques. Stephen T. Albin. John Wiley & Sons, 2002, ISBN 0-471-22886-9.
- Software System Architecture: Working With Stakeholders Using Viewpoints and Perspectives. Nick Rozanski, Eoin Woods. Addison Wesley, 2005, ISBN 0-321-11229-6
- Documenting Software Architectures: Views and Beyond. Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judith Stafford. Addison Wesley, 2002, ISBN 0-201-70482-X.

- Design Patterns. Elements of Reusable Object-Oriented Software - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - Addison Wesley (GoF- Gang of Four)
- A System of Patterns - Buschmann, Meunier, Rohnert, Sommerlad, Stal - Wiley
- UML y Patrones. Introducción al análisis y diseño orientado a objetos - Larman - Prentice Hall
- Pattern-Oriented Software Architecture: A System of Patterns. Buschmann, F.,Meunier, R.,Rohnert, H.,Sommerlad, P.,Stal, M. 1996. Chichester: John Wiley and Sons, 1996, ISBN 0-47195889-7

Contactos de la cátedra:

- **Mail (obligatorio):** lmbibbo@lifia.info.unlp.edu.ar
- **Sitio WEB:**
- **Plataforma virtual:** https://catedras.linti.unlp.edu.ar/
- **Otros:**

Firma del/los profesor/es

Luis Mariano Bibbo