ARQUITECTURA Y DISEÑO DE SOFTWARE	**Carrera:** Doctorado en Ciencias Informáticas (plan2001) **Docente Responsable:** Dr.Gustavo Rossi **Docente:** Mg. Nicolas Battaglia **Duración total:** 100hs (75hs de interacción pedagógica y 25hs de trabajo autónomo del alumno) **Modalidad:** Virtual sincrónico **Créditos:** 4 (cuatro)

OBJETIVOS

El objetivo general de la asignatura es presentar las tendencias actuales de arquitectura y diseño de software, particularmente para desarrollar sistemas que cumplen los requisitos de calidad deseables hoy en día. Se analizan dichos atributos (llamados también características arquitecturales o requerimientos no funcionales) y se muestran como los diferentes estilos arquitecturales permiten satisfacerlos o no. Adicionalmente se presentan algunos conceptos avanzados de diseño de software.

Objetivos específicos

- Conocer el estado del arte de las Arquitecturas de software, los problemas más importantes y las tendencias actuales
- Discutir los lenguajes de descripción de arquitecturas más importantes
- Analizar los diferentes estilos arquitecturales, actuales e históricos
- Presentar arquitecturas monolíticas y distribuidas
- Discutir los conceptos de deuda arquitectural y de malos olores arquitecturales y de diseño
- Presentar los conceptos de refactorización de diseño y arquitectural

PRE-REQUISITOS

Dado que la mayor parte del material disponible sobre el tema es en inglés, es requisito leer fluidamente en inglés.
Se asume que los asistentes han tenido algún tipo de capacitación en ingeniería

de software y manejan lenguajes de programación modernos (aunque en principio este no es un requisito fuerte).

PROGRAMA

Unidad 1: Introducción. Elementos de arquitectura
Motivación. ¿Por qué estudiar Diseño y Arquitectura? Tipos de problemas a resolver. Atributos de calidad (interna o externa) involucrados. Relación entre el enfoque de desarrollo (monolítico, ágil) en la toma de decisiones. Ejemplo prototípico de sistema y discusión de como cada problema repercute en un posible nivel de diseño. Breve repaso de técnicas de diseño *in the small*. Polimorfismo, Doble despacho, inversión de control, métodos abstractos.

Unidad 2: Principios de diseño y buenas prácticas
Diseño Orientado a Objetos (*in the small*). Guidelines y Principios. Principios de Diseño OO. Ejemplos. Micro Arquitecturas. Introducción a Patrones de Software que facilitan la aplicación de los principios. Malos olores y refactorización Introducción a los malos olores de diseño y arquitectura. Principios de refactorización. Frameworks de aplicación. Frameworks orientados a objetos. Construcción y uso

Unidad 3: Arquitecturas monolíticas
Conceptos generales de diseño de arquitecturas OO. SOLID en la práctica. Patrones de Arquitectura (Repository, UnitOfWork, etc). Construyendo arquitecturas evolutivas.
Conceptos generales. Entropía y desgaste del software desde el punto de vista arquitectónico. Tipos y patrones de arquitecturas. Arquitecturas monolíticas Arquitectura en capas. Onion, Hexagonal, Clean. (in the small). Arquitecturas monolíticas Análisis, diseño e implementación de arquitectura tipo hexagonal. Ejercicios

Unidad 4: Arquitecturas distribuidas
SOA. Análisis y diseño orientado a servicios. SOA vs Monolitos. Conceptos generales. RPC / SOAP. RESTFul. Protocolo HTTP (in the small).REST. Buenas prácticas. Recursos. Stateless. Seguridad. Idempotencia. Autenticación y autorización libre de estado. JWT
REST. Implementación de una API Rest. Ejercicios. Microservicios. De los monolitos a los microservicios. Características de los microservicios

METODOLOGÍA
El curso busca una dinámica interactiva, combinando momentos en formato de

clase magistral con trabajos prácticos y monográficos de parte de los alumnos.

MODALIDAD DE EVALUACION

La evaluación consiste en una presentación durante el curso y el desarrollo de un trabajo final de investigación.

BIBLIOGRAFIA GENERAL

- Martin Fowler, Kent Beck, John Brant, William Opdyke, Don Roberts - Refactoring - Improving the Design of Existing Code-Addison-Wesley Professional (1999)
- Kruchten, P. 1995. The 4+1 View Model of Architecture. IEEE Software12(6).
- Shaw, Mary, and David Garlan. Software Architecture—Perspectives on an Emerging Discipline. Upper Saddle River, NJ: Prentice Hall, 1996.
- Clements, Paul, Rick Kazman, and Mark Klein. Evaluating Software Architectures. Boston, MA: Addison-Wesley, 2002
- Bass, L., Clements, P., and Kazman, R. 1998. Software Architecture in Practice. Reading, MA.: Addison-Wesley.
- ma E, Helm R, Johnson R, Vlissides J. Design patterns: elements of reusable object-oriented software. Addison-Wesley Longman Publishing Co.; 1995.
- M. Richards, Software architecture patterns, O'Reilly Media, Incorporated, 2015.
- Sam Newman - Monolith to Microservices_ Evolutionary Patterns to Transform Your Monolith (2019, O'Reilly Media)
- Sam Newman - Building Microservices (2015, O'Reilly Media)
- D. C. Schmidt, M. Stal, H. Rohnert, F. Buschmann, Pattern-Oriented Software Architecture, Patterns for Concurrent and Networked Objects, volume 2, John Wiley & Sons, 2013
- Rozanski, E. Woods, Software systems architecture: working with stakeholders using viewpoints and perspectives, Addison-Wesley, 2012
- Neal Ford_ Mark Richards - Fundamentals of Software Architecture_ A Comprehensive Guide to Patterns, Characteristics, and Best Practices-O'Reilly Media (2020)