ANÁLISIS Y DISEÑO DE SISTEMAS 2

MARTES 7:10 A 8:50 (T-3 216) – JUEVES 17:20 A 19:00 (T-3 212)

PROGRAMA DEL CURSO

ING. RICARDO MORALES
EMAIL: JRMPRADO@PROTONMAIL.COM

DESCRIPCIÓN GENERAL

¿ACERCA DE QUÉ ES EL CURSO?

¿Has pensado en que existe mas de una forma de definir la estructura interna de un sistema de software? En un mundo donde se busca obtener resultados más rápido, manteniendo la calidad de un producto, los conceptos de patrones de diseño y arquitectura de software son herramientas de gran utilidad para alcanzar esos objetivos

El curso es acerca de patrones aplicables para el diseño de componentes de una aplicación, es decir patrones de diseño. El uso de dichos patrones, de patrones de arquitectura, técnicas para definir una arquitectura que satisfaga los requerimientos funcionales y no funcionales, comprenden el resto del curso. Se aplicarán los elementos anteriores en un proyecto de curso para poder contar con una experiencia real en el uso de los patrones y técnicas presentados.

¿POR QUÉ ES IMPORTANTE EL CURSO?

Dada la presencia del software en casi todos los aspectos de la sociedad, la confiabilidad, disponibilidad. seguridad y calidad en general del mismo es un factor crítico de éxito, tanto para un producto de software, como para la comunidad de usuarios que lo utilice.

Para alcanzar niveles de calidad acorde a las expectativas de los clientes, es necesario aplicar procesos en el diseño del software, de manera que los mismos consideren los diferentes tipos de requerimientos que pueden impactar en la definición de la estructura de un sistema de software.

OBJETIVOS

¿QUÉ DEBO SABER Y PODER HACER AL FINAL DEL CURSO?

Identificará y describirá los conceptos de arquitectura de software para el desarrollo de sistemas informáticos

Identificará y describirá los conceptos de atributos de calidad y tácticas aplicables para alcanzarlos

Evaluará y adaptará el diseño de un sistema de software de acuerdo con los diferentes tipos de requerimientos de un proyecto

Diseñará la arquitectura de software de un proyecto real, con base en diferentes puntos de vista y perspectivas para alcanzar los atributos de calidad de dicho proyecto.

Aplicará técnicas para medición cuantificable del rendimiento y escalabilidad de una aplicación

CONTENIDO

Conceptos generales de arquitectura de software

Elementos de diseño

- Patrones de diseño
- Principios de diseño

Arquitectura de software

- Vistas, puntos de vista y perspectivas
- Atributos de calidad
- Estilos de arquitectura
- Proceso de arquitectura
- Descripciones Estructurales (Vistas Estáticas)
- Descripciones de Comportamiento (Vistas Dinámicas)

Optimización

- Uso de perspectivas
- Pruebas cuantitativas de rendimiento y escalabilidad

METODOLOGÍA

Se impartirá clase magistral 2 veces por semana.

Así mismo se formarán grupos para el desarrollo de un proyecto donde se deben aplicar los conceptos aprendidos en el curso, de forma práctica, dando solución a un problema real, lo cual se complementará con tareas y evaluaciones.

Se desarrollarán investigaciones y exposiciones de patrones de diseño y conferencias recientes relacionadas con el curso.

EVALUACIÓN

EVALUACIONES PARCIALES

Se realizarán 2 evaluaciones parciales individuales (26 febrero y 2 de abril). Las evaluaciones se realizarán en clase, de forma escrita o electrónica y evaluarán el contenido cubierto hasta la clase anterior a la evaluación.

PROYECTO

Los estudiantes deben formar grupos para el desarrollo de un proyecto de curso que consistirá en la construcción de un sistema de software utilizando las prácticas que están en el contenido del curso.

El proyecto está dividido en 3 fases:

- Fase 1 (16 febrero), evaluar framework de desarrollo, herramientas de apoyo al proyecto, sistema a desarrollar, evaluar plataformas en la nube y arquitectura a utilizar.
- Fase 2 (20 marzo), definición de arquitectura, desarrollo, profiling y pruebas de rendimiento.
- Fase 3 (30 abril), mejora de arquitectura y aplicación de perspectivas.

Para la evaluación de cada fase se definirá una rúbrica que contendrá detalle los aspectos a calificar.

EXPOSICIÓN TEMAS

Cada grupo debe realizar 2 exposiciones en el curso:

- Un patrón de diseño
- Una conferencia de InfoQ

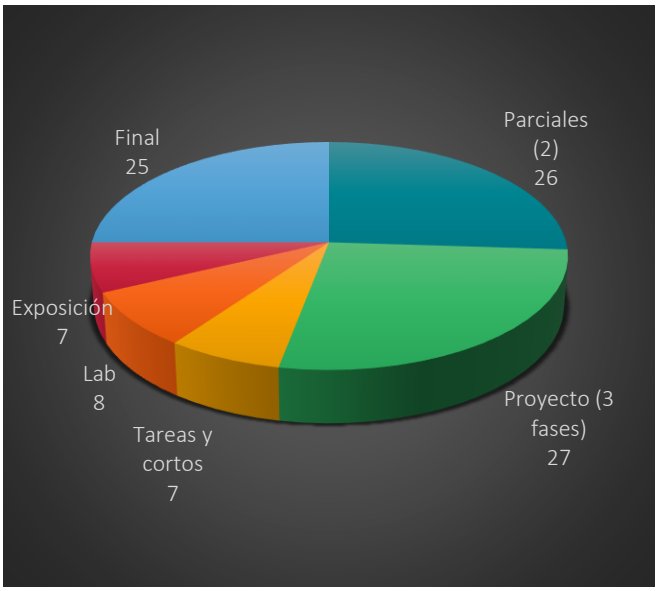
LABORATORIO

El auxiliar del curso llevará a cabo actividades de apoyo al desarrollo de las diferentes fases del proyecto y aplicación de las prácticas del curso.

TAREAS Y CORTOS

En las clases durante el semestre se asignarán lecturas para resumen y discusión, así como exámenes cortos.

DISTRIBUCIÓN DE LA EVALUACIÓN



POLÍTICAS

Las entregas fuera de fecha no son aceptadas.

No es permitido el uso de celulares durante la clase y exámenes parciales.

Debe existir respeto por las opiniones de los demás.

Como estudiantes universitarios, se espera que sepan y entiendan las guías de ética y plagio relacionadas con trabajos de otros autores.

Al final del semestre, no se asignarán trabajos extra para recuperar puntos de zona.

Se debe aprobar (61/100) el laboratorio del curso para tener derecho a examen final.

BIBLIOGRAFÍA

1. Essential software architecture, Ian Gorton
2. Just enough software architecture, George Fairbanks
3. Head first design patterns, Elisabeth Freeman, Eric Freeman, Bert Bates, Kathy Sierra
4. Software systems architecture, working with stakeholders using viewpoints and perspectives, Nick Rozanski, Eoin Woods
5. Software performance and scalability: a quantitative approach, Henry H. Liu

CALENDARIO

FECHA	EN CLASE	TAREA / ENTREGAS
Martes Ene 22	Bienvenida Revisión programa del curso **Conceptos generales de arquitectura de software** Definición Importancia de arquitectura de software Ejercicio requerimientos diferentes sistemas	Lectura capítulos 1 y 2, bibliografía 2 Ejercicio requerimientos diferentes sistemas
Jueves Ene 24	**Patrones de diseño** Conceptos generales Patrón strategy Patrón observer	Entrega ejercicio requerimientos diferentes sistemas Lectura artículo Five Things Every Developer Should Know about Software Architecture, https://www.infoq.com/articles/architecture-five-things
Martes Ene 29	**Principios de diseño** Single Responsibility Principle (SRP) Open-Closed Principle (OCP)	Exposición patrón decorator Exposición patrón Factory
Jueves Ene 31	Liskov Substitution Principle (LSP) Interface Segregation Principle (ISP)	Exposición patrón singleton Exposición patrón command
Martes Feb 05	Dependency Inversión Principle (DIP) Inversión del Control (IoC)	Exposición patrón adapter y facade Exposición patrón template method
Jueves Feb 07	**Vistas, puntos de vista y perspectivas** Vista Punto de vista Perspectiva	Exposición patrón iterator y composite Exposición patrón state
Martes Feb 12	**Atributos de calidad**	Exposición patrón proxy Exposición patrones compuestos
Jueves Feb 14	**Estilos de arquitectura** Escenarios Estilos de arquitectura	Exposición infoq, grupo 1
Martes Feb 19	Exposición fase 1 proyecto	

Jueves Feb 21	Estilos de arquitectura	Exposición infoq, grupo 2
Martes Feb 26	Primer examen parcial	
Jueves Feb 28	**Proceso de arquitectura** Definir contexto y alcance inicial Involucrar stakeholders Capturar primera versión de temas de interés Definir la arquitectura	Exposición infoq, grupo 3
Martes Mar 05	**Catálogo de puntos de vista** Punto de vista de contexto Punto de vista funcional Punto de vista de información	Exposición infoq, grupo 4
Jueves Mar 07	Punto de vista de concurrencia punto de vista de desarrollo	Exposición infoq, grupo 5
Martes Mar 12	Prácticas de desarrollo	Entrega Fase 2 proyecto
Jueves Mar 14	Punto de vista de deployment Punto de vista operacional	Exposición infoq, grupo 6
Jueves Mar 19	**Catálogo de perspectivas** Perspectiva de seguridad Perspectiva de rendimiento y escalabilidad	Exposición infoq, grupo 7
Jueves Mar 21	Calificación fase 2 proyecto	
Jueves Mar 26	Perspectiva de rendimiento y escalabilidad Perspectiva de disponibilidad y resistencia	Exposición infoq, grupo 8
Jueves Mar 28	**Pruebas cuantitativas de rendimiento y escalabilidad** Introducción Stack de software	
Martes Abr 02	Probando rendimiento y escalabilidad de software	

Jueves Abr 04	Segundo examen parcial	
Martes Abr 24	Probando rendimiento y escalabilidad de software	
Jueves Abr 26	Probando rendimiento y escalabilidad de software Teoría de colas	
Martes Abr 30	Teoría de colas	Entrega fase 3 proyecto
Jueves May 02	Calificación fase 3 proyecto	

LISTADO CONFERENCIAS INFOQ

Nombre	Link	Descripción
The Great Migration: from Monolith to Service-Oriented	https://www.infoq.com/presentations/airbnb-soa-migration	Jessica Tai provides an overview of trade-offs and motivation for the SOA migration. She discusses Airbnb's architectural tenets around service building and dives deep into lessons learned and best practices developed when undertaking the massive SOA challenge.
With Great Scalability Comes Great Responsibility	https://www.infoq.com/presentations/scalability-architectural-patterns	Dana Engebretson covers the contextual pros and cons of a number of architectural patterns given real world scalability constraints; from orchestrating Lambdas with AWS step functions to multiprocessing with S3 triggers to rate limiting with queues like SQS.
Deep Learning for Application Performance Optimization	https://www.infoq.com/presentations/dl-performance-optimization	Zoran Sevarac presents his experience and best practice for autonomous, continuous application performance tuning using deep learning. He explains how to build deep learning models in order to model the application performance for various configuration settings.
Seven Strategies for Scaling Product Security	, https://www.infoq.com/presentations/security-2018	Angelo Prado discusses seven strategies for scaling and embedding security in a product.

Full Stack Web Performance	https://www.infoq.com/presentations/web-app-performance	Nik Molnar discusses how to use client and server side profiling tools to improve the performance of a web application, providing solutions to the most common performance problems.
Java Performance Engineer's Survival Guide	https://www.infoq.com/presentations/java-performance-guide	Monica Beckwith provides a step-by-step approach to finding the root cause of any performance problem in a Java app, showcasing through an example a few performance tools and the performance process.
Containers - What Are They Good For?,	https://www.infoq.com/presentations/containers-net	Jimmy Bogard takes a look at containerizing all aspects of a developer's toolkit, from local dependencies, development, build, test, and finally, production, to see where containers fit.
The Cloud Challenge	https://www.infoq.com/presentations/fidelity-investments-cloud	Richard Moran discusses how Fidelity Investments is leveraging people, process and technology to tackle cultural and technological problems and take on the cloud challenge.
Improving the Design of Existing Software	https://www.infoq.com/presentations/design-app-improve	Steve Smith looks at some common places for signs of app design degradation, showing steps to take to improve the code. Examples use C#/.NET but are generally applicable regardless of platform.