FACULTAD DE INFORMATICA | UNIVERSIDAD NACIONAL DE LA PLATA

Especialización en Ingeniería de Software

Tópicos de Ingeniería de Software II

Plan de estudios 2010
Año 2024

Docente Responsable: Dr. Urbieta, Matías

Docentes Tutores: Dr. Díaz Pace, Jorge A.
(Académico/Tecnológico)
Dra. Pons, Claudia
(Académico/Tecnológico)

Tutores: **Dr. Joaquin Bogado**
(Académico/Tecnológico)
Dr. Gardey, Juan Cruz
(Académico/Tecnológico)
Pizarro, María Alejandra
(Administrativo)

Duración: 108 hs. Totales.

OBJETIVOS GENERALES:

Brindar a los asistentes técnicas y conceptos avanzados del desarrollo de Software, que permiten construir software robusto, mantenible, extensible y reutilizable. Se abordará el diseño de software desde una perspectiva arquitectural y de atributos de calidad, con énfasis en la reutilización de software a partir del concepto de frameworks. Se abordarán patrones de arquitectura, con un foco en el ámbito de aplicaciones Web. Por otro lado se introducirán conceptos de inteligencia artificial y cómo puede aplicarse al campo de la ingeniería de software.

COMPETENCIAS A DESARROLLAR EN RELACION CON EL OBJETIVO DE LA CARRERA

C.1- Manejar y aplicar tecnologías actuales para el desarrollo de sistemas de software, incluyendo métodos, lenguajes, arquitecturas, frameworks y herramientas.

C.3- Gestionar, planificar y controlar proyectos de software de distinta envergadura.

C.4- Definir parámetros de calidad tanto interna como externa de un producto software, y establecer procesos de evaluación y mejora que atiendan la satisfacción de todos los involucrados (el cliente, los usuarios y su experiencia, y el equipo de desarrollo).

C.6- Tener capacidad de analizar el estado del arte en los distintos aspectos de la ingeniería de software.

CONTENIDOS MÍNIMOS:

- Nociones de arquitectura de software y atributos de calidad
- Conceptos avanzados del diseño orientado a objetos y frameworks de desarrollo.
- Arquitecturas web y diseño de aplicaciones Web.
- Conceptos de Infraestructura como código, Contenerización y despliegues.
- Introducción a la Inteligencia Artificial, y su impacto sobre la Ingeniería de Software.

PROGRAMA

Arquitectura y Atributos de Calidad

- Definición de arquitectura. Rol en el desarrollo de software.
- Atributos de calidad como conductores del diseño arquitectónico.
- Estilos arquitectónicos más comunes.
- Captura de decisiones arquitectónicas.
- Vistas de módulos, componentes y conectores, y despliegue.

Conceptos de frameworks

- Tipos de frameworks. Características.
- Principio de inversión de control. Hotspots.
- Composición vs. Herencia en frameworks. Ejemplos.
- Relación con arquitecturas de referencia.

Arquitecturas Web y Microservicios

- Frameworks tradicionales multicapa. Model-View-Controller. Principios.
- Frameworks de aplicaciones Single-Page.
- Arquitecturas basadas en Microservicios.
- Nociones de infraestructura para puesta en producción.

Infraestructura como código

- Fundamentos. Infrastructure as Code, Contenedores, infraestructura.
- Diseño de infraestructura
- Esquema de despliegue

Introducción a la Inteligencia Artificial.

- Fundamentos. IA simbólica vs. IA no simbólica.
- IA no simbólica: Aprendizaje automático o machine learning. Aprendizaje supervisado vs. Aprendizaje no supervisado. Árboles de decisión. Deep Learning (Redes Neuronales Artificiales). Tipos de redes neuronales.
- Aplicaciones de la Inteligencia Artificial en el campo de la Ingeniería de Software.
- Ciclo de vida de los modelos de Inteligencia Artificial y MLOps

MODALIDAD DE EVALUACION Y ACREDITACIÓN

La evaluación de la materia es a través de trabajos prácticos parciales y de un trabajo práctico final integrador. La calificación final del alumno es en base a los trabajos parciales y la entrega final. Los trabajos prácticos parciales son de pequeña envergadura y tienen una función de poner rápidamente en práctica los conocimientos que se van abordando. Estos son realizados y aprobados durante la cursada de la materia.

El trabajo final integrador combina los conceptos presentados durante el curso para el diseño y prototipado de una solución al enunciado provisto por los profesores. Una vez finalizado el curso, el alumno cuenta con 2 meses para presentar el trabajo.

Cualquier evidencia de plagio en los trabajos entregados será considerado causal de desaprobación.

RECURSOS Y MATERIALES DE ESTUDIO

El curso propone 15 encuentros. Se requiere 80% de asistencia a los encuentros, incluyendo el encuentro inicial de presentación de la materia, y el encuentro final de integración, ambos de asistencia obligatoria.

Los materiales de estudio son:

- Textos digitales: textos de lectura de referencia en en las temáticas tratadas durante el curso. Los textos de referencia son recuperados de la biblioteca del postgrado, revistas y/o de repositorios.
- Material preparado por los docentes del curso.
- Presentaciones digitales y materiales multimediales sobre el tema (de producción propia)

ACTIVIDADES EXPERIMENTALES Y DE INVESTIGACION PLANIFICADAS PARA LA APROPIACIÓN DE LOS SABERES Y LA EVALUACIÓN

Actividades prácticas:

Desarrollo de trabajos prácticos parciales luego de cada eje temático de la materia. Estos trabajos están pensados para ser iniciados en clases prácticas estilo taller, apuntando a una puesta en común, que luego continuará individualmente cada alumno.

Actividad 1: luego del dictado de los primeros temas, se trabaja sobre un caso de estudio para que los alumnos puedan realizar un análisis de sus principales atributos de calidad y un (re-)diseño arquitectural del mismo, para luego implementarlo sobre alguna de las tecnologías a utilizar en el curso (por ej., Java). Además, se trabajará sobre la generación de diagramas de arquitectura basados en UML.

Actividad 2: se presentarán conceptos de microservicios que permitan exponer funcionalidades del sistema. Se introducen conceptos de contenerización e Infrastructrure as code para el empaquetado y despliegue de los servicios en diferentes configuraciones de infraestructura. Se analizarán problemas de balanceo de carga, alta disponibilidad y diagnóstico de problemas.

Actividad 3: mediante esta actividad el alumno pondrá en práctica los conceptos teóricos aprendidos acerca de cómo implementar y entrenar un modelo de Inteligencia Artificial a través de un framework. Se busca que los estudiantes comprendan los conceptos de "modelo", y las diferencias entre "entrenamiento" e "inferencia". Los estudiantes aprenderán a ajustar los parámetros e hiperparámetros del modelo para realizar el entrenamiento y utilizarán diferentes métricas para medir su performance.

Trabajo Final: Consiste en la aplicación de los conceptos expuestos en las diferentes actividades para resolver un problema presentado por la cátedra.

BIBLIOGRAFÍA BÁSICA

- Cervantes, H., Kazman, R. Designing Software Architectures: A Practical Approach. Addison-Wesley. 2016
- Fayad, M. E., Schmidt, D. C., & Johnson, R. E. Implementing application frameworks: object-oriented frameworks at work. John Wiley & Sons, Inc..
- Fayad, M. E., Schmidt, D. C., & Johnson, R. E. (1999). Building application frameworks: object-oriented foundations of framework design. John Wiley & Sons, Inc.
- Antani, V., & Stefanov, S. Object-Oriented JavaScript. Packt Publishing Ltd. 2017.
- Design Patterns: Elements of Reusable Object-Oriented Software. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Addison Wesley, 1994.
- S. Russell y P. Norvig. Artificial Intelligence. A Modern Approach.Prentice Hall, 3ra edición. 2010.
- Nielsen, Jakob. Designing web usability: The practice of simplicity. New riders publishing, 1999.
- Infrastructure as Code , MANAGING SERVERS IN THE CLOUD, Kief Morris, SBN: 978-1-491-92435-8

BIBLIOGRAFÍA COMPLEMENTARIA

- Firmenich S., Bosetti G., Rossi G., Winckler M., Barbieri T. (2016) Abstracting and Structuring Web Contents for Supporting Personal Web Experiences. In: Bozzon A., Cudre-Maroux P., Pautasso C. (eds) Web Engineering. ICWE 2016. Lecture Notes in Computer Science, vol 9671. Springer,
- Martin Fowler: Domain-Specific Languages. The Addison-Wesley signature series, Addison-Wesley 2011, ISBN 978-0-321-71294-3, pp. I-XXVIII, 1-597
- Wolfgang Ertel:Introduction to Artificial Intelligence.Springer. ISBN 978-3-319-58486-7
- Matias Urbieta, Nahime Torres, José Matías Rivero, Gustavo Rossi, Francisco José Domínguez Mayo: Improving Mockup-Based Requirement Specification with End-User Annotations. XP 2018: 19-34
- Angular 6 by Example: Get Up and Running with Angular by Building Modern Real-world Web Apps, 3rd Edition Kevin D. Hennessy
- Goodfellow, Ian; Bengio, Yoshua y Courville, Aaron (2016). Deeplearning. MIT Press.
- Keras: the Python deep learning API https://keras.io/
- Shai Shalev-Shwartz y Shai Ben-David. (Ed.). (2014).Understanding Machine Learning: From Theory to Algorithms. New York. UnitedState: Cambridge University Press.
- Exploring architecture blueprints for prioritizing critical code anomalies: Experiences and tool support. E Guimaraes, S Vidal, A Garcia, JA Diaz Pace, C Marcos. Software: Practice and Experience, 48 (5), 1077-1106. Wiley. (2018)
- High-Level Design Stories in Architecture-Centric Agile Development. J. A Diaz-Pace, A. J. Bianchi. 2019 IEEE International Conference on Software Architecture Companion (ICSA-C),

137-144, IEEE. (2019)

- Assisting requirements analysts to find latent concerns with REAssistant. A Rago, C Marcos, J. A. Diaz-Pace. Automated Software Engineering, 23 (2), 219-252. Springer. (2016)
- Deep Learning. Ian Goodfellow and Yoshua Bengio and Aaron Courville. MIT Press. 2016. https://www.deeplearningbook.org/
- Understanding Deep Learning. Simon J.D. Prince. MIT Press. 2023. https://udlbook.github.io/udlbook/