PROGRAMA de Prácticas de Desarrollo de Software

Carrera: Licenciatura en Informática

Asignatura: Práctica de Desarrollo de Software

Núcleo al que pertenece: Básico

Profesor: Fernando Dodino

Asignaturas Correlativas: Construcción de Interfaces de Usuario – Estrategias de Persistencia – Elementos de Ingeniería de Software

Objetivos:

Se espera que quienes cursen la materia:

- comprendan las problemáticas de calidad transversales a todo desarrollo de software como: versionado, escenarios complejos de versionado como múltiples líneas de desarrollo, testeo continuo, automatización, trazabilidad, etc.
- incorporen herramientas y prácticas continuas para atacar dichos aspectos.
- sean capaces de atacar dichas problemáticas tanto en proyectos de gran escala como al desarrollo de pequeños programas/aplicaciones.
- sean capaces de tomar decisiones sobre la elección de herramientas/servicios para la resolución de problemas relacionados con aspectos de calidad como: métricas, SCM, versionado, etc.
- sean capaces de tomar decisiones en cuanto al flujo de trabajo y coordinación entre los diferentes participantes de un proyecto de software, referido a la interacción de código y ambientes de desarrollo de software.
- adquieran una visión amplia pero a su vez concreta de las herramientas actualmente utilizadas en la industria.

Contenidos mínimos:

- Validación y testing como un proceso continuo que se lleva a cabo durante todo el ciclo de vida del software, desde que se comienza a programar hasta que, luego de ser implementado y utilizado, el sistema se vuelve obsoleto.
- Test de integración. Problemática específica para la automatización de test de integración, persistencia e interfaz de usuario.
- Técnicas para diagnóstico de problemas: stacktraces, breakpoints, watchpoints. Manejo de excepciones. Relación con unit testing.

- Reingeniería de software. Técnicas de refactorización sobre un proyecto funcionando.
- Migraciones y actualizaciones evolutivas a los modelos de datos. Compatibilidad hacia atrás.
- Herramientas metodológicas y conceptuales para trabajo en grupo. División de tareas planeando reunión de los resultados. Aprovechamiento de los conceptos de programación en objetos, por ejemplo: interfaces como forma de coordinar la tarea de distintas personas o grupos, mock objetos para simular los objetos de otros grupos, etc.
- Versionado y compartición de programas fuente. Repositorios de código centralizados y distribuidos. Técnicas para la modificación de una misma base de código por múltiples desarrolladores en forma concurrente. Resolución de conflictos.
- Versionado y compartición de bibliotecas y ejecutables. Administración de entregables y dependencias. Repositorios de bibliotecas.
- Integración continua. Automatización de procesos en desarrollos de envergadura, como integración, compilación, verificación, versionado, despliegue, entre otros.
- Control de cambios. Trazabilidad de requerimientos, errores y cambios de funcionalidad. Herramientas para la administración integral de cambios y correcciones.
- Aplicación e integración de las técnicas, prácticas y herramientas aprendidas en un proyecto mediano de desarrollo de software.

Carga horaria semanal: 8 hs

Carga horaria total: 144 horas (Presenciales: 90 horas / Semipresenciales: 55 horas)

Programa analítico:

Unidad 1 - Introducción: Complejidades del desarrollo de Productos

Introducción a la materia. Relación con otras materias de la carrera y la tecnicatura. Introducción a las problemáticas de esquemas complejos de proyectos y productos. Multiequipos. Concepto de Proyecto y Producto. Relación entre ambos. Líneas de trabajo y versiones.

Unidad 2 - Sistemas de Versionado de Código.

Concepto de sistemas de versionado. Head/Trunk, tags y branches. Resolución de conflictos. Sistemas centralizados. Sistemas distribuidos. Pull-Requests. Hooks y customizaciones. Buenas prácticas de trabajo concurrente.

Unidad 3 - SCM

Concepto de artefacto. Manejo de dependencias. Versionado de componentes: pre/alpha, beta, RC, GA. Release y trazabilidad. Administración y publicación de binarios/paquetes entregables. Automatización de tareas.

Unidad 4 - Prácticas continuas

Concepto de Continuous Integration. Relación con Sistemas de Versionado. Resolución de problemas. Relación con SCM y deployment (nightly build, releases). Continuous Deployment. Evaluación y prueba de herramientas. Continuous Inspection: Métricas y

alarmas. Calidad del software. Análisis estático y dinámico de código. Complejidad, Cobertura, etc.

Unidad 5 - Herramientas Colaborativas

Concepto de Issue Tracker. Relación con versionado y releases. Trazabilidad de cambios. Code Reviews: concepto, herramientas, utilidad. Relación con otras prácticas. Comunicación: listas de desarrollo, vínculo con SCM, suite Forges.

Unidad 6 - Estrategias de Testing

Mocks. Test de integración: diferentes estrategias. Test de componentes arquitectónicos. Emulación de componentes remotos (ej: web service). BDD. Test de Aceptación. Test de Bases de Datos. Test de carga y concurrencia. Herramientas para generación de test (fuerza bruta)

Bibliografía obligatoria:

- Andrew Hunt, David Thomas. The Pragmatic Programmer: From Journeyman to Master. 1999, Addison-Wesley, ISBN-13: 978-0201616224.
- John Ferguson Smart. BDD in Action: Behavior-Driven Development for the whole software lifecycle. 2014, Manning Publications, ISBN: 9781617291654.
- Kent Beck. Extreme Programming Explained: Embrace Change. 2004, Addison-Wesley, ISBN-13: 978-0321278654.
- Lisa Crispin. Agile Testing: A Practical Guide for Testers and Agile Teams. 2009, Addison-Wesley, ISBN-13: 978-0321534460.
- Michael T. Nygard. Release It!: Design and Deploy Production-Ready Software. 2007, Pragmatic Bookshelf, ISBN-13: 978-0978739218.

Bibliografía de consulta:

- Jon Loeliger, Matthew McCullough. Version Control with GIT: Powerful tools and techniques for collaborative software development. 2012, O'Reilly, ISBN-13: 978-1449316389.
- Brian R. Jackson, Maven: The Definitive Guide. 2015, O'Reilly, ISBN-13: 978-1449362805.
- Jaime Pillora, Getting Started with Grunt: The Javascript Task Runner. 2014, Packt Publishing, ISBN-13: 978-1783980628.
- Jezz Humble, David Farley, Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, 2010, Addison-Wesley, ISBN-13: 978-0321601919.

Organización de las clases:

La cursada de la materia se organizará en una combinación de clases teóricas y prácticas. Por un lado se explicarán conceptos en forma teórica y aplicada, utilizando pizarra pero además el soporte tecnológico actual como un proyector y una computadora de forma de poder visualizar los conceptos aplicados a un proyecto ejemplo.

Por otro lado, quien curse la materia deberá aplicar los conceptos aprendidos a un proyecto a modo de trabajo práctico. Este trabajo práctico durará toda la cursada, de modo de poder abarcar y así aplicar todos los conceptos de la materia.

Para seguimiento y apoyo durante el desarrollo del trabajo práctico la cursada tendrá clases prácticas de tipo Laboratorio con PCs disponibles para quienes cursen la materia.

Respecto las actividades extra-áulicas obligatorias, se espera que quien curse la materia dedique tiempo fuera del aula en el avance del Trabajo Práctico. Esto es en la aplicación de lo visto en clase para el desarrollo de su sistema de software. E incluye tanto los conocimientos de arquitectura y de programación, como también el uso de las herramientas de software y conceptuales vistas en clase para el desarrollo mismo. Por ejemplo: coordinar el desarrollo del equipo mediante un sistema de versionado distribuido, trabajando con feature-branches, y realizando revisiones de código cruzadas. Establecer un flujo de trabajo e integración de los features. Configurar un esquema de despliegue automático de la aplicación en servidores productivos, etc.

El objetivo de esto es que quien curse la materia aplique en forma concreta las prácticas y conocimientos a un proyecto que tranquilamente podría ser de la vida real en la industria.

Este trabajo fuera del aula es considerado como parte de la evaluación del Trabajo Práctico, que establecerá la nota de cursada de quien curse la materia.

Trabajos Prácticos

Los trabajos prácticos se dividen en dos partes.
Práctica Nro. 1: Durante la prácticas del primer mes, quienes cursen la materia desarrollan una aplicación con mínimas funcionalidades para refrescar/aprender los conceptos que se precisan para la Práctica Nro. 2.
Práctica Nro. 2: Durante las prácticas del resto de los meses del cuatrimestre, quienes cursen la materia se separan en grupos y deberán aplicar los conceptos aprendidos a un proyecto de software de mayor envergadura, con aplicación a un dominio real del mercado laboral. Este proyecto se divide en 5 fases:

- **1ra Fase:** Cada grupo deberá generar un documento de especificaciones del problema a resolver con un detalle mínimo de las funcionalidades a implementar. El cuerpo docente actuará como cliente del proyecto a desarrollar (la misma puede ser una aplicación móvil o de desktop o la combinación de ambas).
- **2da Fase:** Cada grupo desarrollará un prototipo de la aplicación para que el cliente pueda dar un feedback de las primeras interfaces, y las funcionalidades especificadas en la primera fase.
- **3ra Fase:** Cada grupo incrementará las funcionalidades dentro de la aplicación, incorporando las técnicas de versionado de código, sobretodo usando conceptos de branches y trabajo colaborativo a nivel código.
- **4ta Fase:** Cada grupo incorporará el uso de Bases de Datos con técnicas de Persistencia como así también la integración del back y el front end de la aplicación.
- **5ta Fase:** Cada grupo trabajará con técnicas de Tests End-to-End y de Integración Continua para testear las funcionalidades implementadas previamente.
- **6ta Fase**: Cada grupo incorporará el uso de Docker para generar la portabilidad de la aplicación en diferentes plataformas.

Estas 6 fases son iterativas pues se utilizarán desarrollo ágil para la construcción de aplicaciones.

Parte de la carga horaria de esta asignatura forma parte del grupo de espacios curriculares de la carrera que implementan horas de Instancias de Formación de Prácticas Profesionales Supervisadas (IFPPS), que se regulan a través del Reglamento aprobado por Res. (CDCyT) N.º 034/21, o cualquier otra resolución que la modifique o la reemplace. En estas instancias quienes cursen deben cumplir con la evaluación por la aplicación de los conceptos teóricos/prácticos aprendidos en relación con los objetivos de aprendizaje. Así mismo se formulan otros objetivos vinculados a la solución de un problema del mercado laboral -en el cual se insertarán como profesionales. De esta manera, en lo que respecta a las horas totales de la materia, se dedican 86 horas totales para las IFPPS y las 58 horas restantes para clases teóricas y prácticas (trabajos prácticos usuales) necesarias para el aprendizaje de los conceptos.

Esta asignatura forma parte del grupo de las asignaturas con horas presenciales y semipresenciales que están reguladas a través de la Resolución del CS 052/21, o cualquier otra resolución que la modifique o la reemplace.

Las asignaturas con esta modalidad se desarrollan con horas presenciales y encuentros virtuales en base a las necesidades de las asignaturas, sin alterar la metodología de evaluación de las materias en modalidad presencial. Los encuentros virtuales proveen contenidos, materiales y ejercicios de distintos formatos que incorporen las ventajas de la plataforma virtual indicada por la Universidad.

Modalidad de evaluación:

Los mecanismos de evaluación en modalidades libre y presencial de esta asignatura están reglamentados según los siguientes artículos del Régimen de estudios de la UNQ (Res. CS 201/18)
En la modalidad de libre, se evaluarán los contenidos de la asignatura con un examen escrito, un examen oral e instancias de evaluación similares a las realizadas en la modalidad presencial.

CRONOGRAMA TENTATIVO

Semana	Tema/unidad	Actividad*		
		Teórico	Práctico	
			Res Prob.	Lab.
1	Introducción a la materia. Conceptos preliminares. Diagnostico. Presentación del trabajo práctico.	X		
2	Sistemas de versionado: centralizados / distribuidos. Introducción a git, mejores prácticas.	X		X
3	Sistemas de versionado distribuidos. Branches, workflows, pull-requests, squashing. Resolución de conflictos.	X		X
4	Manejo de dependencias. Automatización de tareas. Task runners y hooks.	X		X
5	Corrección TP1			
6	Herramientas colaborativas. Issue tracking. Control de cambios.	X		X
7	Estrategias de testing. Tests de integración.	X		X
8	Corrección TP2			
9	Integración continua: introducción a jscript y travis	X		X
10	Integración continua: construcción de artefactos complejos. Jenkins.	X		X
11	Correccion TP3			
12	Trabajo en equipo. Revisiones de código.	X		X
13	Despliegue de aplicaciones. Contenedores y ambientes. Docker y grunt.	X		X
14	Correccion TP4			
15	Estrategias de testing no funcionales: carga, concurrencia, performance y otros.	X		X
16	Trazabilidad de cambios al modelo de datos. Mitigación de impacto y compatilidad hacia atrás.	X		X
17	Correccion TP5			
18	Recuperatorios y cierre			x

***INDIQUE CON UNA CRUZ LA MODALIDAD**

Firmado digitalmente por: SUAREZ Mariana Alejandra
Directora del Departamento de Ciencia y Tecnologla
UNIVERSIDAD NACIONAL DE QUILMES