METODOS AGILES PARA APLICACIONES WEB

Año 2022

Carrera/ Plan:

Licenciatura en Informática Plan 2021/Plan 2015/Plan 2012/Plan 2003-07
Licenciatura en Sistemas Plan 2021/Plan 2015/Plan 2012/Plan 2003-07
Analista en Tecnologías de la Información y la Comunicación Plan 21/Plan 2017
Año: 3ro, 4to, 5to.
Régimen de Cursada: Semestral
Carácter (Obligatoria/Optativa): Optativa
Correlativas:
Orientación a Objetos 1
Ingeniería de Software 1
Profesor/es: Alejandra Garrido
Hs. semanales: 6 hs

FUNDAMENTACIÓN

Esta materia les permite a los alumnos profundizar los conocimientos adquiridos en el Paradigma de Orientación a Objetos y en conceptos básicos de Ingeniería de Software, en el contexto de la construcción de aplicaciones complejas como es el caso de las aplicaciones web con procesos de negocio.

En particular abarcaremos el desarrollo y el mantenimiento de aplicaciones web con metodologías ágiles, enfatizando los valores y principios de las mismas y ejercitando sus prácticas. Nos enfocaremos además en el desarrollo de software de calidad y centrado en el usuario, integrando métodos de evaluación y mejora de la usabilidad y la experiencia del usuario (UX) con prácticas ágiles.

Dentro de los métodos y técnicas que se abordarán para permitir construir y mantener la calidad del software se cubrirán temas de testing de distintos niveles y refactoring, tanto en relación a la calidad interna del código como a la calidad en cuanto a experiencia del usuario. También cubriremos conceptos y técnicas relacionadas al desarrollo ágil que han surgido con énfasis en los últimos años como el concepto de deuda técnica y las técnicas de visual management.

A través de esta materia los alumnos se expondrán al desarrollando un sistema real en un grupo de trabajo, y utilizando una metodología ágil muy actual en el ámbito profesional.

OBJETIVOS GENERALES

- Presentar a los alumnos la problemática de construir y mantener una aplicación compleja como es el caso de las aplicaciones web con procesos de negocio, teniendo en cuenta las necesidades de los usuarios, y respetando criterios de calidad interna y externa.
- Discutir los valores, principios y prácticas de las metodologías ágiles para la construcción de aplicaciones complejas en el contexto de un equipo de desarrollo, así como los principios y prácticas de métodos más específicos surgidos del movimiento ágil, como son “lean software development” y “visual management” de proyectos ágiles.
- Introducir nociones básicas de usabilidad, accesibilidad, experiencia del usuario y diseño de interacciones, asi como los problemas comunes que surgen en la interacción del usuario con interfaces web y las posibles soluciones.
- Ejercitar el diseño centrado en el usuario a través del maquetado de interfaces.
- Exponer el concepto de “technical debt” en el mantenimiento de software y presentar técnicas específicas para su abordaje a través de testing y refactoring. Profundizar estas técnicas no solo para la mejora de atributos de calidad interna, sino también para aquellos de calidad externa como la usabilidad.

COMPETENCIAS

- LI-CE4- Planificar, dirigir, realizar y/o evaluar proyectos de relevamiento de problemas del mundo real, especificación formal de los mismos, diseño, implementación, prueba, verificación, validación, mantenimiento y control de calidad de sistemas de software/sistemas de información que se ejecuten sobre equipos de procesamiento de datos, con capacidad de incorporación de tecnologías emergentes del cambio tecnológico. Capacidad de análisis, diseño y evaluación de interfases humano computador y computador-computador.
-LS-CE1- Planificar, dirigir, realizar y/o evaluar proyectos de relevamiento de problemas del mundo real. Especificación formal, diseño, implementación, prueba, verificación, validación, mantenimiento y control de calidad de sistemas de software que se ejecuten sobre sistemas de procesamiento de datos, con capacidad de incorporación de tecnologías emergentes del cambio tecnológico. Capacidad de análisis, diseño y evaluación de interfases humano computador y computador-computador.

CONTENIDOS MINIMOS

- Arquitecturas de software. Arquitecturas web desktop y móviles.
- Valores, principios y prácticas de las metodologías ágiles. Métodos ágiles: XP y Scrum.
- Origen de Lean software development. Principios y prácticas Lean.
- Principios y herramientas de Visual Management en proyectos ágiles.
- Mantenimiento de software. Introducción al concepto de Technical Debt o Deuda Técnica. Tipos de Deuda Técnica. Abordaje de la Deuda Técnica.
- Testing. Tipos de tests: de aceptación, de unidad, de integración. Integración continua. Refactoring. Concepto de bad smell.
- Usabilidad en las aplicaciones web. Accesibilidad.
- Refactoring como técnica para mejorar la usabilidad de aplicaciones web.

PROGRAMA ANALÍTICO

I - Arquitecturas de software

1. Definición de arquitectura de software. Diferentes abordajes.
2. Arquitecturas de software más reconocidas. Patrones arquitecturales.
3. Arquitecturas web.

II - Metodologías ágiles de desarrollo y prácticas relacionadas

4. Metodologías Agiles. Filosofía. Características fundamentales. Diferencias con las metodologías tradicionales de desarrollo de software.
5. Extreme Programming. Valores, principios y prácticas.
6. Scrum. Historia. Roles, reuniones y artefactos. Estimación de tareas.
7. Lean software development. Principios y prácticas Lean.
8. Visual management. Kanban. Herramientas de visual management más reconocidas. Trello.
9. Lean UX. Principios. Proceso de lean UX.

III - Diseño centrado en el usuario y experiencia del usuario

10. Características del Diseño Centrado en el Usuario (UCD). Historia. Ciclo de vida UCD.
11. Nociones básicas de usabilidad y diseño de interacciones. Concepto de Experiencia del Usuario (UX). Accesibilidad en aplicaciones web.

12. Métodos de evaluación de la usabilidad. Métodos de inspección. Heurísticas de Nielsen. Recorridos cognitivos. Guías de accesibilidad de la W3C.
13. Métodos empíricos de evaluación de la usabilidad. User Testing. Cuestionarios de satisfacción. S.U.S. A/B testing y Split testing aplicado a UX. Integración de user testing en un desarrollo ágil.

IV - Maquetado de interfaces gráficas

14. Prototipado. Su importancia y función.
15. Prototipos de distinta fidelidad. Wireframes, mockups y storyboards.
16. Estereotipos que se utilizan para la documentación de storyboards en la materia.
17. Consideraciones de usabilidad y diseño de interacciones durante el maquetado.

V - Construcción y mantenimiento de software con métodos ágiles

18. Dificultades en el diseño de software. Rol de los patrones de diseño. Diseño incremental. Etapa de mantenimiento de software. Costo del mantenimiento
19. Refactoring. Características. Proceso de refactoring. El concepto de bad smell. Límites en la preservación del comportamiento. Herramientas.
20. Refactoring en distintos lenguajes y paradigmas de programación. Refactoring de modelos y de otros artefactos de software. Refactoring de arquitecturas. Refactoring to patterns. Refactoring para la mejora de atributos de calidad internos (flexibilidad, mantenibilidad).
21. Test Driven Development. Características. Tipos de Testing: de unidad, de integración, de aceptación, de regresión. Características de cada uno. Estrategias de testing de unidad. Framework de testing de unidad: familia XUnit.
22. Technical debt o Deuda Técnica. Tipos de Deuda Técnica. Abordaje de la Deuda Técnica
23. Integración continua. Herramientas.

VI - Refactoring de UX en aplicaciones web

24. Dificultades del diseño de interacción en interfaces de usuario. Patrones de interfaz de usuario.
25. Refactoring como técnica para mejorar la experiencia del usuario, usabilidad y accesibilidad de aplicaciones web. Definiciones. Refactoring de usabilidad a nivel de modelos. Catálogo.
26. Malos olores de usabilidad y UX: “UX smells”. Definición y catálogo de UX smells.
27. Accessibility smells y Accessibility refactorings.
28. Identificación automática de usability smells a partir de eventos de interacción.
29. Refactoring de UX en el cliente o Client-Side Web Refactoring (CSWR). Catálogo de CSWR.
30. Herramienta de aplicación de CSWRs: UX-Painter. Versionado de aplicaciones web para A/B testing basado en UX.

VII - Integración de métodos ágiles y centrados en el usuario

31. Agile vs. UCD. Similitudes y diferencias.
32. Diferentes propuestas de integración de UCD con prácticas agiles. Diferentes dimensiones en la integración. Prácticas y herramientas.
33. Ciclo de vida del desarrollo “Agile+UX”
34. Método de mejora incremental de la UX en aplicaciones web a través de testing y refactoring.

BIBLIOGRAFÏA

- Extreme Programming Explained. Kent Beck. Addison Wesley. 2005.
- Essential Scrum: A Practical Guide to the Most Popular Agile Process. Kenneth Rubin. Addison-Wesley. 2012.
- Lean Software Development. Dasari. Ravi Kumar. The PROJECT PERFECT White Paper Collection.
- Visual Management – A General Overview. Algan Tezel, Lauri Koskela, Patricia Tzortzopoulos. 5th Int. Conference on Construction in the 21st Century (CITC-V). 2009.
- Lean UX: Designing Great Products with Agile Teams. O'Reilly Media, Inc. 2016.
- The evaluation of accessibility, usability and user experience. Helen Petrie and Nigel Bevan. The Universal Access Handbook, C Stepanidis (ed), CRC Press, 2009.
- Don't make me think. Revisited: A Common Sense Approach to Web Usability. Steve Krug. New Riders Publishing. 2013.
- Agile Design Process with User-Centered Design and User Experience in Web Interfaces: A Systematic Literature Review. J. Giacomelli Beux, E. A. Bellei, L. A. Brock, A. C. Bertoletti De Marchi, C. A. Holbig. LATIN AMERICAN JOURNAL OF COMPUTING, VOL. 5, NO. 2, NOVEMBER 2018.
- The evolution of agile UXD. Tiago Silva Da Silva, Milene Selbach Silveirab, Frank Maurerc, Fábio Fagundes Silveira. Information and Software Technology 102 (2018).
- Refactoring: Improving the Design of Existing Code. Fowler, Martin. Addison-Wesley, 1999.
- Test Driven Development. Kent Beck. Addison-Wesley. 1999.
- Automatic detection of usability smells in web applications. Julián Grigera, Alejandra Garrido, José Matías Rivero, Gustavo Rossi. Int. J. Hum.-Comput. Stud. 97: 129-148 (2017).
- Usability Improvement Through A/B Testing and Refactoring. Firmenich, Garrido, Grigera, Rivero, Rossi. Software Quality Journal 27 (1) 2019. DOI:10.1007/s11219-018-9413-y
- UX Painter: An Approach to Explore Interaction Fixes in the Browser. Gardey, Garrido, Firmenich, Grigera, Rossi. Proc. ACM Hum. Comput. Interact. 4(EICS): 89:1-89:21 (2020)

METODOLOGÍA DE ENSEÑANZA

Los conceptos teóricos son presentados en clases teóricas por el profesor, y se les ofrecerá material extra a los alumnos que permita profundizar los temas durante las clases teóricas, a partir de distintos artículos que disparen la inquietud de los alumnos y que motiven la participación en clase.

Durante las clases prácticas los alumnos irán desarrollando una aplicación web en grupo, siguiendo una metodología ágil, que constará de diferentes instancias de entrega. El proyecto que los alumnos deben desarrollar profundizará la tecnología de orientación a objetos, utilizando prácticas que aseguren la calidad interna y externa de la aplicación (como testing y refactoring). Para esto los alumnos utilizarán distintas herramientas de acceso libre o provistas por la cátedra. Se le asignará un peso importante a la UX de la aplicación resultante.

En el caso de que la materia se desarrolle en **modalidad virtual**, las clases teóricas se ofrecerán en 2 formatos: uno a través de videos y presentaciones de transparencias que quedarán disponibles en la plataforma virtual que utiliza la materia y otro a través de clases virtuales sincrónicas en la que se espera que los alumnos participen activamente. Además se proveerá un foro de discusión donde los alumnos podrán consultar asincrónicamente el material teórico. Asimismo, las clases prácticas en formato de reuniones de grupo se desarrollarán a través de reuniones sincrónicas en la plataforma virtual de la materia.

EVALUACIÓN

Para aprobar la **cursada** de la materia se requiere aprobar todas y cada una de las instancias de entrega del proyecto donde se desarrollará una aplicación web en forma grupal. Además se requiere asistencia en algunos encuentros obligatorios. Las entregas son progresivas e incrementales, de manera que se hará un seguimiento puntual de cada grupo, intentando asegurar además una carga balanceada de trabajo entre los miembros del grupo. Los grupos tendrán la oportunidad de corregir o completar las entregas que tengan algún comentario antes de la siguiente instancia. En el caso de que la materia se realice en **modalidad virtual** se mantienen las mismas condiciones, y los encuentros obligatorios se realizarán por videoconferencia a través de la plataforma virtual que provee la materia. Además se utilizarán diferentes herramientas de trabajo online colaborativo.

Para aprobar el **final** de la materia se requiere entregar un documento que describe el proyecto desarrollado, su diseño, implementación, instalación y conclusiones. Además los alumnos deberán presentarse a un coloquio en el que deberán responder preguntas de integración de los temas abordados en la materia con su aplicación práctica en el proyecto realizado en grupo. En la nota final se considerarán estas 2 instancias, además del rendimiento individual durante la cursada. El rendimiento se puede mejorar con una presentación en clase profundizando los tópicos de la materia.

CRONOGRAMA DE CLASES Y EVALUACIONES

Clase	Contenidos/Actividades	Evaluaciones previstas
1	Presentación de la materia. Requisitos de la materia. Forma de aprobación. Fechas importantes. Presentación de proyectos posibles. Arquitecturas de software	
2	Metodologías agiles. Manifiesto, XP, Lean, Scrum. Herramientas de visual management: Trello. Herramienta de manejo de código: GitHub	Presentación del proyecto seleccionado
3	Diseño Centrado en el Usuario. Conceptos de usabilidad, UX y accesibilidad.	Presentación del “product backlog” y planificación del 1er sprint
4	Maquetado de interfaces web: storyboards y mockups. Integración de mockups con métodos agiles. Mockup-Driven Development.	
5	Métodos de evaluación de la usabilidad. Heurísticas y recorridos cognitivos. Métodos de evaluación empírica	Presentación de mockups del 1er sprint
6	Testing de usuario. Integración de tests de usuario en un desarrollo ágil. Técnica de A/B testing.	
7	Desafíos del desarrollo ágil y el mantenimiento de software. Refactoring de código. Code smells.	Presentación de user tests
8	Concepto de deuda técnica. Tipos de Deuda Técnica. Abordaje de la Deuda Técnica	Presentación de MVP 1 en review meeting. Planificación del 2do sprint

9	Testing. Test Driven Development. Framework XUnit. Testing de aplicaciones web. Herramienta para crear tests de usuario.	Presentación de mockups del 2do sprint
10	Refactoring para mejorar usabilidad y UX. UX smells. Accessibility smells. Refactorings en el cliente. UX-Painter.	Presentación de propuesta de Split testing del MVP 1.
11	Accesibilidad. W3C Guidelines: WCAG	
12	Identificación de usability smells a partir de eventos de interacción	Presentación del MVP 2 y planificación del 3er sprint
13	Proceso agil de mejora incremental de la usabilidad y UX durante el desarrollo ágil. Herramientas.	Presentación de propuesta de Split testing del MVP 2.
14	Clases especiales de alumnos	
15	Clases especiales de alumnos	
16	Integración final de temas	Presentación del producto final

Contacto de la cátedra (mail, sitio WEB, plataforma virtual de gestión de cursos):

- Docentes:
  - Profesora: Alejandra Garrido: garrido@lifia.info.unlp.edu.ar
  - Ayudante: Juan Cruz Gardey: jcgardey@lifia.info.unlp.edu.ar
- Plataforma virtual:
  - https://catedras.info.unlp.edu.ar . Curso: “Metodos Ágiles para Aplicaciones Web”

Firma del/los profesor/es