TALLER DE PROGRAMACIÓN ORIENTADA A OBJETOS I	**Carrera/ Plan:** *Licenciatura en Informática* Plan 2021 / Plan 2015 *Licenciatura en Sistemas* Plan 2021 / Plan 2015 *Licenciatura en Informática* Plan 2003-07/Plan 2012 *Licenciatura en Sistemas* Plan 2003-07/Plan 2012 **Área:** Algoritmos y Lenguajes **Año:** 4º o 5º año **Régimen de Cursada:**Semestral **Carácter:** Optativa **Correlativas:** Orientación a objetos 2 Proyecto De software
Año 2022	**Profesores:** *Federico Balaguer* **Hs semanales:** *6hs*

FUNDAMENTACIÓN

Esta materia les permite a los alumnos profundizar en el paradigma de orientación a objetos, poniendo en práctica el conocimiento adquirido durante la carrera en diferentes dominios, para aplicarlos en un desarrollo ágil. Se profundizan en particular los temas de patrones de diseño, frameworks, refactoring, sistemas distribuidos, desarrollo de aplicaciones Web, servicios Web, pruebas de unidad, etc. Se tocan además temas avanzados de lenguajes como es el tema de reflexión.

A través de esta materia los alumnos se exponen a una práctica profesional concreta, desarrollando un sistema real en un grupo de trabajo, con un cliente y utilizando una metodología ágil muy actual en el ámbito profesional.

Por otro lado, las actividades que se desarrollan tienen como objetivo fomentar la comunicación dentro del grupo de trabajo, con el cliente y con los demás grupos. Cada uno de los alumnos también debe hacer presentaciones para desarrollar sus habilidades de expresión oral, muy útiles hoy en día tanto para el desarrollo de software en grupos de trabajo, como para la docencia y presentación de trabajos de investigación. Con respecto a esto último los alumnos también se introducen en algunos temas de investigación al tener que leer artículos de revistas internacionales actuales en inglés para exponer en clase.

OBJETIVOS GENERALES:

Profundizar en la tecnología de orientación a objetos y diferentes temas de alto impacto en la actualidad, para desarrollar un trabajo de mediana envergadura en forma grupal siguiendo una metodología ágil. El desarrollo conlleva la utilización de frameworks orientados a objetos, patrones de diseño, y herramientas de: seguimiento de proyectos, versionamiento de código, refactoring y testing.

Dado que en el marco de un proceso de desarrollo ágil la comunicación es uno de los valores esenciales, se pretende fomentar la comunicación entre alumnos y con los docentes, tanto oral (a través de presentaciones en clase) como escrita (a través de documentación del proyecto).

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.

- LS-CE3- Dirigir el relevamiento y análisis de los procesos funcionales de una Organización, con la finalidad de dirigir proyectos de diseño de Sistemas de información asociados, así como los Sistemas de Software que hagan a su funcionamiento. Determinar, regular y administrar las pautas operativas y reglas de control que hacen al funcionamiento de las áreas informáticas de las empresas y organizaciones.

CONTENIDOS MINIMOS

- Diseño orientado a objetos avanzado con lenguajes de patrones
- Utilización y desarrollo de frameworks orientados a objetos
- Estructuras de Analisis de Programas Orientados a Objetos
- Reflexión. Potencia y ejemplos de utilización.
- Uso corriente de Conceptos de Programacion Funcional: Expresiones Lambda y “Continuaciones”
- Metodologías ágiles de desarrollo de software: XP, Scrum, TDD
- Prácticas ágiles de desarrollo de software: Refactoring, Testing,
- Repositorios de código y manejo de versiones

PROGRAMA ANALÍTICO

I- Desarrollo de software de mediana y gran envergadura

1. Configuración de un sistema. Versionamiento de código. Utilización de repositorios de versionamiento. Organización del repositorio, grupo de trabajo y versiones de código
2. Herramientas de seguimiento de proyectos. Trello. Kanban.
3. Patrones y lenguajes de patrones de analísis, diseño y dominios específicos.
4. Frameworks orientados a objetos. Documentación y utilización. Estrategias de extensión de frameworks. Instanciación de frameworks. Ejemplos.

II- Metodologías ágiles y prácticas relacionadas

5. Metodologías Agiles. Filosofía. Características fundamentales. Diferencias con las metodologías tradicionales de desarrollo de software.
6. Extreme Programming. Valores, principios y prácticas.
7. Scrum. Roles, reuniones y artefactos.
8. Refactoring. Características. Proceso de refactoring. Límites en la preservación del comportamiento. Herramientas. Refactoring en distintos lenguajes y paradigmas de programación. Refactoring de modelos y de otros artefactos de software. Refactoring de arquitecturas. Refactoring para la mejora de atributos de calidad internos (flexibilidad, mantenibilidad) así como externos (performance, usabilidad, accesibilidad). Refactoring y control de versiones. Líneas actuales de investigación en refactoring.
9. Test Driven Development. Características. Tipos de Testing: de unidad, de integración, de aceptación. Características de cada uno. Estrategias de testing de unidad. Framework de testing de unidad: familia XUnit.
10. El valor de la comunicación. Cómo hacer una buena presentación oral; consejos y prácticas. Buen estilo de escritura. Reglas simples para una composición clara y efectiva.

III-Temas avanzados

11. Reflexión. Introducción. Características. Implementación en distintos lenguajes. Reflexión en Lisp. Reflexión en Smalltalk. Introspección en Java. Herramientas desarrolladas con reflexión: debuggers, profilers, tracers. Implementación de patrones de diseño utilizando reflexión: Proxy, Decorator. Potencia del la aplicación de la reflexión en frameworks: Inyección de dependencias; Traits.
12. Expresiones Lambda y “Continuaciones”. Introducción. Caracteristicas. Impacto en diferentes lenguajes: C#, Java, Smalltalk.
13. Distintos artículos sobre temas relacionados a cada proyecto grupal:
    a. Desarrollo de sistemas distribuidos; brokers and proxies.
    b. Arquitecturas peer-to-peer
    c. Arquitecturas orientadas a servicios y micro-servicios.
    d. Crowdsourcing

e. Usabilidad, experiencia del usuario, y accesibilidad en aplicaciones web.
f. Test Driven Web Application Development. Herramientas actuales.
g. Desarrollo de software de código abierto.
h. Juegos aplicados a la educación.
i. Manejo visual de proyecto y Lean software development
j. Cloud computing y fog computing
k. Cloud manufacturing
l. Internet of things

METODOLOGÍA DE ENSEÑANZA

El proyecto que los alumnos deben desarrollar profundizará la tecnología de orientación a objetos, utilizando frameworks orientados a objetos y patrones de diseño. Para profundizar en estos temas es importante la reflexión durante las clases teóricas a partir de distintos artículos que disparen la inquietud de los alumnos y que motiven la participación en clase. En estas clases es el profesor quien debe introducir cada tema y moderar la participación de los alumnos.

Por otro lado, como también se pretende trabajar la capacidad de comunicación oral de los alumnos, habrá algunas clases teóricas a cargo de cada grupo de proyecto, donde además tendrán que ponerse de acuerdo en como presentar el tema que el profesor les designa, realizando transparencias en forma grupal. Se requiere asistencia a estas clases.

En cuando a las clases prácticas o de laboratorio, se proponen un conjunto de proyectos interesantes para los alumnos, útiles en su formación profesional y realizables en el término de la materia. Los alumnos tienen la posibilidad de elegir a sus compañeros de grupo y elegir el proyecto que les interesará desarrollar. Durante las clases prácticas, los docentes actúan por un lado como clientes, generando los requerimientos del sistema, y por otro lado como mentor o “Scrum master”, realizando el seguimiento pormenorizado de cada grupo y cada proyecto.

La modalidad semi-presencial establece que a los alumnos que cursen en esta modalidad no se les requerirá asistencia a las clases teóricas; en cambio, se les proporcionará con suficiente antelación el material sobre el tema que se presentará en cada clase, y deberán enviar por email hasta el día antes de la clase, un resumen del tema y al menos 3 preguntas para que respondan quienes realicen la exposición.

Ante la eventual imposibilidad de realizar la cursada de manera presencial, la materia se adapta para proveer tanto las clases teoricas, las presentaciones de alumnos y las reuniones quincenales.

Clases Teoricas: haciendo que las clases (presentadas por la catedra) se realizan de dos modalidades: Streaming en vivo y Recurso Off-line. Streaming en vivo convoca a la participacion de los alumnos y el docente. Se organiza una llamada grupal en donde el docente presenta un tema y los alumnos tienen la posibilidad de preguntar. Recursos Off-line requiere la preparacion de una presentacion basada en diapositivas y el audio explicativo. En ambos casos el material utilizado incluye ejercicios relacionados con el tema presentado.

Presentacion de Alumnos: las presentaciones se realizan en modalidad Steaming en vivo con la presencia obligatoria de todos los alumnos que cursan la materia.

Reuniones Quincenales: las reuniones semanales se pactan por grupo y se realizan con herramientas para llamadas grupales.

EVALUACIÓN

Para **aprobar la cursada** de la materia se requiere:

- Presentación oral: haber hecho una presentación oral del tema establecido, en una de las clases teóricas de la materia.

- Asistencia a las presentaciones grupales: se exige un mínimo de 2/3 de asistencia para incentivar la participación de los alumnos en las presentaciones dictadas por sus compañeros.

- Seguimiento quincenal del proyecto: los alumnos de cada grupo tendrán que asistir a las presentaciones de avance del proyecto con el docente que le corresponda. Hay suficientes horarios disponibles en 2 bandas horarias, de manera que aún los alumnos que trabajan puedan asistir. Para aprobar la cursada se requiere no haber fallado 2 veces consecutivas a las reuniones o 3 veces en total (donde "falla" significa falta o no haber realizado ningún avance).

- Entrega del código del proyecto tres semanas antes de la finalizacion de la semestre correspondiente según el calendario academico. La entrega del codigo se realiza en el repositorio que la materia ha organizado en GitHub.

- Aquellos alumnos que no cumplan con los requisitos antes descriptos para la aprobacion de la cursada tienen la chance de rendir un examen sobre los temas presentados en la materia

Por tratarse de un taller donde se pretende exponer a los alumnos al trabajo grupal, en un proyecto ágil con fechas estipuladas, la cátedra considera que rendir un examen escrito no cumpliría con los objetivos de la materia. Por lo tanto, para **aprobar la materia** se requiere la entrega y aprobación de la documentación del proyecto, que se presenta en un coloquio grupal que debe realizarse 3 semanas antes al comienzo del siguiente ciclo lectivo de la materia.

Ante la eventual imposibilidad de realizar la cursada de manera presencial, los parametros de evaluacion se mantienen y el coloquio grupas se realiza mediante una reunion grupal remota.

BIBLIOGRAFÍA OBLIGATORIA

- Design Patterns. Elements of Reusable Objects Oriented Software. Garnma, Helm, Johnson, Vlissides, Addison-Wesley, Professional Computing Series.
- Refactoring: lmproving the Design of Existing Code. Fowler, Martin. Addison-Wesley, 1999.
- Simple Smalltalk Testing: With Patterns. Kent Beck. http://www.xprogramming.com/testfram.htm
- SUnit Explained. Stephane Ducasse. http://www.iam.unibe.ch/~ducasse/
- Cloud Computing: The Business Perspective. Marston, Sean R. and Li, Zhi and Bandyopadhyay, Subhajyoti and Ghalsasi, Anand and Zhang, Juheng, Decision Support Systems 51 (2011). pp 176-189.
- Reflective Facilities in Smalltalk-80. Brian Foote y Ralph Johnson. OOPSLA'89.

BIBLIOGRAFÍA COMPLEMENTARIA

- Design Patterns Smalltalk Companion. Alpert, Brown, Wolf. Addison Wesley.
- Pattern Languages of Program Design Series. Addison-Wesley.
- Pattern Oriented Software Architecture. A system of patterns. Buschmann, Meunier, Rohnert, Sommerlad and Stal. Wiley. 1996.
- Refactoring to Patterns. Joshua Kerievsky. Addison Wesley, 2004.
- Building Application Frameworks: Object-Oriented Foundations of Framework Design. Mohamed E. Fayad (Editor), Ralph E. Johnson (Author), Douglas C. Schmidt (Editor).
- Extreme Programming Explained. Kent Beck. Addison Wesley. 2005.
- Software Testing. A Craftsman's approach, 4th ed. Paul Jorgensen. CRC Press, 2013.
- "Documenting Frameworks Using Patterns". Ralph Johnson. OOPSLA'92. Vancouver, Canada.
- Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks, D. Roberts and R. Johnson. Proceedings of Pattern Languages of Programs, Allerton Park, Illinois, September 1996.
- Wrappers to the Rescue. John Brant, Brian Foote, Ralph Johnson y Don Roberts. ECOOP'98.
- "Hydra: a massively-multiplayer peer-to-peer architecture for the game developer". Chan, Yong, Bai, Leong, Tan. Netgames 2007.
- "Seaside: A Flexible Environment for Building Dynamic Web Applications –". Ducasse, Lienhard and Renggli. IEEE Software 2007.
- "Service Oriented Architecture". Perrey and Lycett. Proceedings of the Symposium on Applications and the Internet Workshops, 2003.
- "Patterns for SOA", capitulo 2 de "Patterns: Service-Oriented Architecture and Web Services". Endrei et al. IBM. ibm.com/redbooks
- "SOA Test Methodology". Torry Harris. www.thbs.com/soa

- "Crowdsourcing Systems on the WWW". Doan, Ramakrishnan and Halevy. Communications of the ACM. 2011.
- "Test-Driven Web Application Development in Java". Pipka. Lecture Notes in Computer Science, 2003, Volume 2591/2003, 378-393.
- "The Cathedral and the Bazaar". Eric Raymond. 2000.
- The New Methodology. Martin Fowler. http://www.martinfowler.com/articles/newMethodology.html
- "Style. Towards clarity and grace". Joseph Williams. University of Chicago Press (June 15, 1995)
- " The evaluation of accessibility, usability and user experience ". Helen Petrie and Nigel Bevan. The Universal Access Handbook, C Stepanidis (ed), CRC Press, 2009.
- "Computer Games and Learning: Digital Game-based Learning". Marc Prensky. 2006.
- "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.
- "Cloud Manufacturing: Drivers, Current Status, And Future Trends". Dazhong Wu, Matthew J. Greer, David W. Rosen, Dirk Schaefer. Proceedings of the ASME 2013 International Manufacturing Science and Engineering Conference. MSEC2013. 2013, Madison, Wisconsin, USA
- "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions". Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, Marimuthu Palaniswami. Future Generation Comp. Syst. 29(7): 1645-1660 (2013)

CRONOGRAMA DE CLASES Y EVALUACIONES

Inicio de clases: 12 de Marzo 2020

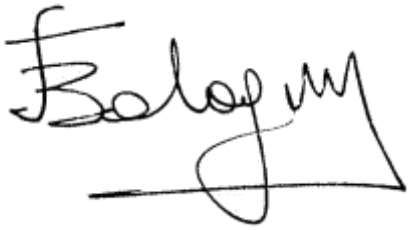
Clase	Tema
1	Introducción. Modalidad de la materia. Presentación de listado de proyectos posibles a desarrollar en grupo.
2	Manejadores de código y versiones de código (software configuration management). Herramientas de seguimiento de proyectos.
3	Base de datos no relacionales
4	Metodologías ágiles. XP y Scrum.
5	Patrones y Lenguajes de patrones en diferentes dominios
6	Refactoring. Refactoring to patterns.
7	Testing. Test Driven Development. Familia XUnit de frameworks de unit testing.
8	Reflexión. Características. Introspección e intercesión. Inyección de dependencias. Traits
9	Expresiones Lambda. Continuaciones. Introduccion. Impacto en diferentes lenguajes OO.
10	Traits
11	Imagenes minimas (Boottrap)
12	Presentaciones orales a cargo de cada grupo
13	Presentaciones orales a cargo de cada grupo
14	Presentaciones orales a cargo de cada grupo

Evaluaciones previstas	Fecha

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

- Docente: Dr. Federico Balaguer: **federico.balaguer@gmail.com**
- Sitio web: **https://sites.google.com/site/tpoounlp/home**
- Sitio web de cursada: <moodle de la facultad>
- Foro de discusión: <moodle de la facultad>
- Repositorio de la materia: https://github.com/TPOOUNLP

Firmas del/los profesores responsables:



Dr. Federico Balaguer