Universidad Tecnológica Nacional
Facultad Regional Buenos Aires

ANEXO I

RESOLUCIÓN Nº...../21

ASIGNATURA:	TÉCNICAS AVANZADAS DE PROGRAMACIÓN	**CODIGO:**	
DEPARTAMENTO:	INGENIERÍA EN SISTEMAS DE INFORMACIÓN	**CLASE:**	Cuatrimestral
ÁREA:	INGENIERÍA EN SISTEMAS DE INFORMACIÓN	**HORAS SEM.:**	4hs.
BLOQUE:	ELECTIVAS	**HORAS / AÑO:**	Reloj 48hs./ Cátedra 64hs

Fundamentación:

Actualmente es necesario que el ingeniero conozca las técnicas de desarrollo de *software* más avanzadas para su correcta construcción. A su vez, debe reconocer sus posibilidades tecnológicas para que pueda implementar las soluciones más adecuadas según cada situación. En este contexto, las empresas y las consultoras valoran profesionales con estos conocimientos, por lo que es de capital importancia que pueda integrar de manera exitosa estas técnicas de programación a su conocimiento. con el fin de satisfacer la gran demanda de trabajo que se presenta hoy en el área de datos

Objetivos:

Seleccionar herramientas conceptuales y estrategias de programación orientadas a objetos y funcional para resolver problemas de desarrollo de *software* de complejidad creciente.
Identificar ventajas y desventajas (alcances y límites) de cada escenario para resolver problemas de desarrollo de software de complejidad creciente
Reconocer la complejidad real de los sistemas, donde el *software* construido debe adaptarse a las interacciones con un entorno que presente restricciones o que no maneje las mismas abstracciones conceptuales del *software* en construcción.
Identificar cómo las tecnologías de implementación afectan al proceso de construcción.
Reconocer las herramientas que permiten mantener una estrategia conceptual según cada contexto.
Realizar prácticas de desarrollo que simplifiquen la construcción de software de alta complejidad, frameworks y extensiones a lenguajes de programación existentes.
Desarrollar una visión técnica de más alto nivel para comprender las prácticas en las que se basan los proyectos desarrollados y/o sistemas utilizados.

Programa analítico:

Unidad Temática 1 – Extensiones modernas al paradigma orientado a objetos
Mixins y traits; comparación, ventajas y desventajas de cada uno. Contraste con el modelo clásico de Herencia Simple. Herencia Múltiple. Linearización y Aplanado de definiciones. Method lookups avanzados; resolución de conflictos y álgebras combinatorias. Impacto de los diferentes esquemas de definición de comportamiento en el diseño con objetos. Patrones de diseño aplicables al nuevo esquema e impacto en la arquitectura del software.

Unidad Temática 2 – Metaprogramación
Metaprogramación, programación reflexiva, introspección, self-modification. Comparación entre definiciones estáticas declarativas y modelos auto-descriptivos de runtime; open classes. Construcción de herramientas y frameworks, manipulación de artefactos abstractos. Diferencia entre meta-modelo y modelo, arquitectura de lenguajes complejos. Bloques, expresiones lambda, y contextos de ejecución. Scope de evaluación y binding dinámico.

Unidad Temática 3 – Sistemas de tipos y esquemas de binding
Clasificación de los sistemas de tipos: tipos nominales y estructurales, tipado explícito e implícito, dinámico y estático. Chequeo e inferencia de tipos. Concepto de binding, implicancias del uso de late binding, polimorfismo. Varianza de tipos: Invarianza, Covarianza, Contravarianza. Smart-casting. Tipado contextual.

Unidad Temática 4 – Programación híbrida Objetos-Funcional
Inmutabilidad, efecto, transparencia referencial. Diferencia entre mutable y reasignable. Aplicación de diversas formas de polimorfismo en contextos con efecto: Polimorfismo paramétrico vs. Ad-hoc, Pattern Matching, Polimorfismo de objetos; aplicaciones prácticas de la rotura del polimorfismo y encapsulamiento. Desarrollo orientado al comportamiento vs. desarrollo orientado a la estructura. Aplicación de teoría de categorías al modelo jerárquico orientado a objetos; monoides, mónadas, functores. Manejo de conjuntos inmutables. Maybe vs. Null. Manejo de errores con valores.
.

Distribución de carga horaria entre actividades teóricas y prácticas:

Tipo de actividad	Carga horaria total en hs. reloj	Carga horaria total en hs. cátedra
Teórica	30	50
Formación practica	18	14
Formación experimental	0	0
Resolución de problemas	0	0
Proyectos de diseño	0	0
Practica de supervisada	0	0
Total	48	64

Articulación Horizontal y vertical con otras materias

La asignatura Técnicas Avanzadas de programación se articula en forma vertical con dos (2) asignaturas que la preceden en el plan de estudio, específicamente Sintaxis y Semántica de Lenguajes y Paradigmas de programación.
Cada estudiante deberá tener regularizada cada una de estas asignaturas al momento de comenzar la cursada. El estudio de las técnicas que se verán en esta asignatura, requiere conocimientos de paradigmas de programación y de la sintaxis y semántica utilizada en la confección de las herramientas y lenguajes de programación. De esta forma, se podrá abordar e incorporar eficientemente el contenido impartido en la asignatura Técnicas Avanzadas de programación.
Además, los conocimientos adquiridos en esta asignatura serán de gran utilidad para extender los alcances de otras materias, ya que imparte conocimientos y criterios abarcativos que permiten unificar los contenidos incorporados individualmente en las asignaturas previas. Además, será de gran utilidad para afrontar el Proyecto Final de la carrera, en el caso de que los estudiantes optarán por el desarrollo de una aplicación de *software*.
En cuanto a la articulación horizontal, Técnicas Avanzadas de Programación brinda conocimientos que son compatibles y complementarios con conceptos y contenidos de otras asignaturas, fomentando así la interdisciplinariedad.
El equipo docente participa de reuniones inter-cátedras convocadas por el Departamento, a fin de generar acuerdos temáticos y de metodologías que faciliten la articulación horizontal y vertical entre las distintas asignaturas.

Unidad temática	Duración en horas cátedra
1	10
2	20
3	10
4	24

Bibliografía:

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. – (1995). Design Patterns: Elements of Reusable Object-Oriented Software.
Martin Odersky, Lex Spoon, and Bill Venners (2008)
Nathanael Schärli, Stateful Traits - Alexandre Bergel, Stephane Ducasse, Oscar Nierstrasz, Roel Wuyts (2007). Traits: Composable Units of Behaviour .
Paolo Perrotta (2014) Metaprogramming Ruby 2 .
Paul Chiusano, Runar Bjarnason (2014). Programming in Scala - Functional Programming in Scala.
Stéphane Ducasse, Oscar Nierstrasz, and Andrew P. Black (2003). Mixin-based Inheritance-

PÁGINAS WEB DE INTERÉS

Sitio de la materia: https://tadp-utn-frba.github.io/

Correlativas:

PARA CURSAR:

Cursadas: Análisis de sistemas
Sintaxis y semántica de los lenguajes
Paradigmas de Programación

PARA RENDIR:

Aprobadas: Análisis de sistemas
Sintaxis y semántica de los lenguajes
Paradigmas de Programación