FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Planificaciones

7571 - Seminario

Docente responsable: COSSO PABLO GUSTAVO

OBJETIVOS

Que los alumnos:
1) se acerquen a tecnologías innovadoras, apliquen conceptos aprendidos en la carrera y asimilen nuevos conceptos de la frontera del conocimiento tecnológico para que puedan entender, diseñar e implementar sistemas innovadores.
2) aprendan a utilizar herramientas de software libre, para cumplir el objetivo anterior sin que sea oneroso y poder conocer la cultura del desarrollo con software libre.
3) puedan conocer y profundizar sus conocimientos de la plataforma Java, dado que las tecnologías que habilita son herramientas fundamentales del paradigma centrado en la red.
4) puedan desarrollar el prototipo de una aplicación de mediana complejidad, de acuerdo a sus inquietudes, para introducirlos en la disciplina del emprendedor tecnológico.

CONTENIDOS MÍNIMOS

-

PROGRAMA SINTÉTICO

El cambio tecnológico que se observa en la actualidad en el software es muy veloz. Desde la difusión masiva de Internet, a partir de 1995, la complejidad de las aplicaciones fue creciendo para poder cumplir con los requerimientos del paradigma centrado en la red. Esto hizo que las nuevas arquitecturas de las aplicaciones fueran más complejas ya que involucran muchas más capas de abstracción interconectadas.
Las mejores soluciones para atacar dicha complejidad implican la modularidad de las aplicaciones. Además se requiere conocer nuevas herramientas tecnológicas para el desarrollo de los productos, poseer metodologías para que el proceso de desarrollo se ejecute en tiempo y forma y observar los aspectos de la conformación de los equipos de trabajo adecuados.
La elección del lenguaje Java y/o los lenguajes que corren sobre la "Máquina Virtual Java (JVM, Java Virtual Machine"), como lenguajes para el desarrollo de la materia tienen varias ventajas, considerado desde varias perspectivas:
1) Java es uno de los lenguajes que los alumnos aprenden en la materia Algoritmos III.
2) Evoluciona constantemente para mejorar la productividad de las aplicaciones.
3) Soporta muy bien la modularidad de la aplicaciones.
4) Es un lenguaje muy rico en APIs (bibliotecas de clases) que soportan el desarrollo de todas las tecnologías que están en la frontera del conocimiento tecnológico del momento.
5) Existen infinidad de herramientas de software libre escritas en Java, que permiten dar un mejor soporte al desarrollo de las aplicaciones.
6) Hace algunos años Sun Microsystem (hoy comprada por Oracle) ha liberado al mismo lenguaje como “software libre”, lo que permite que los alumnos puedan utilizarlo sin pagar por las licencias de uso, ver el código fuente del mismo, e incluso realizar o sugerir mejoras, lo que para un ambiente académico es de suma importancia.
7) La plataforma Java permite que otros lenguajes que surgieron posteriormente y que corren sobre la JVM (por ejemplo, Groovy, Scala, Clojure, etc.) puedan utilizar muchos de los recursos disponibles para Java, por ejemplo, sus bibliotecas de clases (APIs) de modo natural. Eso permite reutilizar mucho trabajo realizado ya en Java.
8) La plataforma Java facilitó el desarrollo de frameworks de infraestructura para el desarrollo de aplicaciones. tales como Spring y Grails (que utiliza Spring), que mejora el rendimiento de los tiempos de desarrollo de aplicaciones.

PROGRAMA ANALÍTICO

Trayectorias tecnológicas que fundamentan la plataforma Java y su evolución.
• El Sistema Operativo UNIX
• Los lenguajes de programación C y C++
• La evolución de Internet
• La evolución de los lenguajes que corren sobre la JVM

Arquitecturas para las aplicaciones de empresa.
• La definición de J2EE. EJBs. Sus problemas.
• Nuevas arquitecturas de software libre. J2EE sin EJBs.
• Tendencias en arquitecturas: microservices, devOps

Los lenguajes de programación Java y Groovy
• Objetivos del diseño

• Historia
• Comparación de los aspectos principales de los lenguajes
  o Tipos de datos primitivos vs objetos
  o Operadores aritméticos y lógicos
  o Arreglos.
  o Strings.
  o Corte de secuencia de varios niveles
  o Recolector de basura
  o Aspectos eliminados de C y C++ en Java y diferencias con Groovy.
• Soporte para la programación orientada a objetos: abstracción, encapsulación, ocultamiento de la información, modularidad, herencia polimorfismo.
• Otras características
• representaciones en tiempo de ejecución.
• Seguridad
• Excepciones
• Multithreading
• Paquetes de los lenguajes: JDK y GDK

La complejidad inherente del software
• Ejemplos de sistemas complejos
• Atributos de los sistemas complejos. La forma canónica de un sistema complejo
• Jerarquía estructural “parte de” y jerarquía “es un”.
• Descomposición algorítmica y orientada a objetos.
• El modelo de objetos

Las generaciones de lenguajes de programación.
• Lenguajes basados en objetos y lenguajes orientados a objetos.
• Topologías de las distintas generaciones de lenguajes de programación.
• Programación orientada a objetos.

Elementos esenciales del modelo de objetos
• Abstracción
• Encapsulamiento
• Modularidad
• Jerarquía.

Clases y objetos en Java y Groovy
• Campos
• Métodos
• Sobrecarga de métodos
• Constructores
• Sobrecarga de constructores
• Constructor this
• Control de acceso de miembros
• Especificación de accesos
• Miembros públicos y privados
• Campos static
• Bloques de inicialización static
• Métodos estáticos
• El método main
• El método toString

Herencia
• La cláusula extends.
• Variables ocultas
• Constructores
• Control de accesos en la herencia
• Solapamiento
• super
• Métodos y clases final
• Métodos y clases abstractos
• Clase Object
• Clonando objetos

• Interfaces
• La cláusula implements.
• Uso de interfaces

Excepciones
• Concepto
• Creando excepciones
• Cláusulas: throw, throws, try, catch y finally
• Tendencia hacia el manejo de exepciones "no probadas".

Operadores y expresiones
• Comentarios
• “doc comments”
• Identificadores
• Tipos primitivos versus objetos
• Literales: enteros, de punto flotante
• Declaración de variables
• Significado de nombres
• Arreglos
• Asociación de los operadores
• Orden de evaluación
• Tipos de una expresión
• Conversiones implícitas de referencias
• Conversiones explícitas (cast)
• Conversiones para objetos
• Conversiones a String
• Acceso a miembros
• Operadores aritméticos
• Aritmética entera
• Aritmética de punto flotante
• Operadores lógicos y de relación
• Expresiones condicionales
• Operadores de asignación

Control de Flujo
• Sentencias y bloques.
• if – else. else - if
• switch
• while
• do while
• for
• Rótulos. break y continue
• return

Modularidad. Paquetes.
• Concepto.
• Las cláusulas package e import.

La evolución de Java a partir de Java 5
• Tipos genéricos.
• Tipos enumerativos (enum)
• Anotaciones.
• Static import.
• Unboxing y autoboxing.
• La sentencia foreach.
• Métodos con argumentos variables.
• Tipos de retorno covariantes.
• Java 8: Streams y Lambdas. Java 9: Modularidad

Características principales de Groovy que se distinguen de Java
• Clausuras
• Listas y Maps como elementos del lenguaje
• Metaobject protocol

Principios de diseño
• Principios de "La banda de los 4" (GOF)
• Principio abierto-cerrado.
• Principio de sustitución de Liskov.
• Principio de segregación de interfaces.

Introducción al diseño de dominios
• La potencia de Java y Groovy para la construcción de soluciones: POJOs y POGOs
• Domain-Driven Design. La construcción de conocimiento. Aprendizaje continuo. El lenguaje ubicuo. Un modelo expresado en software. Entities. Value Objects. Services. Modules
• El ciclo de vida de un objeto del dominio. Aggregates. Factories. Repositories. Diseñando objetos para bases de datos relacionales.
• Diseño estratégico. Contexto limitado. Mapa de contexto.

Inyección de Dependencias
• El problema de las dependencias. Inversión de control. Inyección de dependencias. Formas de inyección de dependencias: inyección por constructor y por setter. El patrón Factory. El patrón Decorator.

Incumbencias transversales
• Entendiendo las incumbencias transversales. Código enredado. Código disperso. Alternativas a AOP. El patrón Observer. El patrón Chain Of Responsibility. Los patrones Decorator y Proxy. Conceptos básicos de AOP.

Introducción al desarrollo con Grails
• Las falencias de J2EE con EJBs.
• Características principales de Grails
• Grails y su relación con el framework Spring
• Arquitectura de capas
  o Modelo de dominio con POGOs (Plain Old Groovy Objects)
  o Capa de persistencia: GORM
  o Capa de presentación. GrailsMVC. Controladores y vistas (gsp).
  o Servicios
• Grails como Framework de desarrollo de aplicaciones productivo. Experiencias reales.
• Evolución de Grails. Su relación con Spring Boot y con Micronaut.

BIBLIOGRAFÍA

• Object Oriented Analysis and Design with applications. Grady Booch
• The Java Programming Language. Ken Arnold – James Gosling – David Holmes
• POJOs in Action. Chris Richardson.
• Pro Spring 2.5. Chakraborty, Ditt, Vukotic, Machacek.
• Domain Driven Design. Eric Evans.
• Program Development in Java – Abstraction, Specification and Object-Oriented Design. Barbara Liskov with John Guttag.
• Java in a Nutshell 5th Edition. David Flanagan.
• Agile Software Development Robert C. Martin.
• Design Patterns. Gamma, Vlissides, Helm, Johnson.
• Programming in Scala. Martin Odersky, Lex Spoon, Bill Venners.
• Groovy in Action. Koenig, Glover, King, Laforge, Skeet
• Programming Groovy 2. Venkat Subramaniam
• Grails in Action. Glen Smith and Peter Ledbrook
• Programming Grails. Burt Beckwith
• The Definitive Guide to Grails 2. Graeme Rocher, Jeff Scott Brown
• Web 2.0: A Strategy Guide. Amy Shuen

Documentación online:
• Inversion of Control Containers and the Dependency Injection pattern. Martin Fowler (en wwww.martinfowler.com).

• Páginas oficiales de Lenguajes:
  Java : www.javasoft.com
  Groovy: www.groovy-lang.org
  Scala : www.scala-lang.org

• Otras páginas sobre Java:
www.javaworld.com;
www.ibm.com/developerworks/java
• Otras páginas de interés:
www.infoq.com
www.theserverside.com
www.dzone.com
domaindrivendesign.org
www.springframework.org
www.grails.org
www.martinfowler.com

RÉGIMEN DE CURSADA

Metodología de enseñanza

Las clases son teórico-prácticas. Se brinda una revisión histórica de los conceptos fundamentales de programación, para establecer el contexto de porqué surgen determinados lenguajes y cómo fueron evolucionando para atacar la complejidad de la construcción de software.
Se juega constantemente con los conceptos de orientación a objetos, con los conceptos del lenguaje Java y los nuevos lenguajes que corren sobre la JVM y con los aspectos que hacen a la utilización en la industria de tales lenguajes. De esta manera, el alumno va observando que los conceptos académicos sólidos que se le imparten tienen una utilización en la vida profesional y están en el estado del arte, en el ámbito académico y en la industria.
Existen clases de consulta en donde los alumnos presentan sus propuestas de trabajos prácticos y se los guía para el mejor desarrollo del mismo.

Modalidad de Evaluación Parcial

Se requiere una asistencia del 75 % de las clases. La aprobación de la misma consiste en la presentación de un trabajo práctico de mediana complejidad, que puede desarrollarse de forma individual o en grupos de no más de tres personas. Para la elección del tamaño del equipo se observa la complejidad del trabajo práctico elegido.

La evaluación del trabajo práctico es grupal e individual, en aspectos que tienen que ver, entre los temas más importantes, con:
• Aspectos fundamentales del lenguaje de Programación y el framework elegido (Groovy + Grails) , en relación al lenguaje y a las APIs más importantes.
• Calidad del código.
• Arquitectura de la aplicación.
• Diseño de la aplicación.
• Metodología de desarrollo.
• División de tareas entre los miembros del equipo (si el trabajo es grupal)

También se exige que los alumnos entreguen un diseño de la aplicación, basado en el modelo de dominio elegido. Esto significa tener las clases más relevantes del diseño, realizado como POJOs (Plain Old Java Objects) o POGOs (Plain Old groovy Objects).

Para poder alcanzar los objetivos de la materia, la metodología de trabajo se basa en un contacto fluido con los alumnos en donde, a partir del desarrollo de un trabajo práctico de mediana complejidad, se los puede guiar en la elección de tecnologías innovadoras para la resolución de los problemas que se les presentan.

CALENDARIO DE CLASES

Semana	Temas de teoría	Resolución de problemas	Laboratorio	Otro tipo	Fecha entrega Informe TP	Bibliografía básica
<1> 09/03 al 14/03	Trayectorias tecnológicas que fundamentan la plataforma Java. Otras trayectorias tecnológicas. Plataformas tecnológicas.			Explicación de los requerimientos del Trabajo Práctico		
<2> 16/03 al 21/03	La complejidad inherente del software. Generaciones de lenguajes de programación. Evolución hasta la actualidad.					
<3> 23/03 al 28/03	Repaso de los elementos esenciales del modelo de objetos. Diferencias entre las implementaciones en Java y Groovy.		Modelo de objetos en Java y Groovy			
<4> 30/03 al 04/04	Introducción a Groovy. Características relevantes del lenguaje. Sus diferencias con Java		Ejemplos en Java y/o Groovy			
<5> 06/04 al 11/04	Arquitecturas de las aplicaciones de empresas. Introducción a Grails.		Ejemplos en Groovy y Grails.	Elección del trabajo práctico		
<6> 13/04 al 18/04	Arquitectura de múltiples capas. Otras características de Grails.		Ejemplos en Groovy y Grails.			
<7> 20/04 al 25/04	Diseño de dominio de aplicaciones de empresas.		Ejemplos en Groovy y Grails.			
<8> 27/04 al 02/05	Diseño de dominio de aplicaciones de empresas			Seguimiento del trabajo práctico		
<9> 04/05 al 09/05	El lenguaje de programación Groovy. Aspectos más avanzados del lenguaje.		Ejemplos en Groovy			

Semana	Temas de teoría	Resolución de problemas	Laboratorio	Otro tipo	Fecha entrega Informe TP	Bibliografía básica
<10> 11/05 al 16/05	Principios de Diseño					
<11> 18/05 al 23/05	Mapeo Objeto Relacional		Ejemplos en Grails			
<12> 25/05 al 30/05	Inyección de dependencia			Primera entrega del trabajo práctico		
<13> 01/06 al 06/06	El framework Grails: aspectos avanzados					
<14> 08/06 al 13/06	Introducción a html5, CSS y JavaScript					
<15> 15/06 al 20/06	Introducción al framework SpringBoot					
<16> 22/06 al 27/06	Nuevas tecnologías y tendencias en el desarrollo de software			Entrega final del trabajo práctico		

CALENDARIO DE EVALUACIONES

Evaluación Parcial

Oportunidad	Semana	Fecha	Hora	Aula
1º				
2º				
3º				
4º				