GUÍA DOCENTE DE LA ASIGNATURA

Métodos Matemáticos en Ingeniería del Software

Datos descriptivos del curso

Nombre: **Métodos Matemáticos en Ingeniería del Software**
Curso de 7.5 créditos UZ (4.5 teóricos, 3 prácticos), 6.8 créditos ECTS, optativa de segundo cuatrimestre en el segundo ciclo de la licenciatura en Matemáticas.

Prerrequisitos: Los prerrequisitos **esenciales** son haber cursado las asignaturas de **Informática I** e **Informática II**, de la licenciatura en Matemáticas (son respectivamente troncal y optativa de primer ciclo). Asimismo, es altamente **recomendable** haber cursado **Modelos Matemáticos en Bases de Datos**, optativa del primer cuatrimestre.

Profesores: Jorge Lloret Gazo (jlloret@unizar.es, tel: 2779) y José Carlos Ciria Cosculluela (jcciria@unizar.es, tel: 1131), del área de Ciencia de la Computación e Inteligencia Artificial (Departamento de Informática e Ingeniería de Sistemas).

Sentido del curso en el perfil de la titulación

Bloque formativo

La asignatura pertenece al bloque formativo en Informática, del que es responsable el área de Ciencia de la Computación e Inteligencia Artificial.

Con este bloque pretendemos, en primer lugar, que el estudiante aprenda a utilizar la Informática como herramienta para resolver distintos tipos de problemas. Queremos formar usuarios expertos, que conozcan en profundidad las nociones que subyacen a la programación y sean capaces, dado un problema, de escoger la técnica más adecuada a su naturaleza y a los recursos disponibles para su resolución.

Pretendemos, igualmente, que el estudiante se familiarice con las tecnologías, metodologías y actitudes que hoy día son necesarias para el desarrollo de aplicaciones informáticas en la empresa y en investigación.

Nuestros objetivos generales son:

- Proporcionar al estudiante una aproximación disciplinada al diseño, codificación, depuración y documentación de programas, utilizando un buen estilo de programación. Hacemos hincapié en el papel central que tiene la abstracción en la tarea de programar, profundizando en el concepto de abstracción operacional y de datos.

- Capacitar al estudiante para formular, representar y resolver problemas utilizando el ordenador.

- Que el estudiante conozca y adquiera destreza en el uso de distintos paradigmas de programación.

- Proporcionar al estudiante nociones de algoritmia: que sea capaz de evaluar la corrección, precisión y la eficiencia de un algoritmo. Dado un problema el estudiante deberá ser capaz de elegir, entre varios algoritmos que lo resuelven, aquél que da un resultado óptimo en función de la naturaleza del problema y de la disponibilidad de recursos (tiempo de CPU, memoria).

- Que el estudiante se familiarice con tecnologías, metodologías y habilidades que se le pueden exigir en el desarrollo de aplicaciones informáticas en la empresa y en investigación.

Sentido de la asignatura dentro del bloque formativo

Lo natural es cursar nuestra asignatura al final del bloque. Al llegar a ella el estudiante ya es capaz de utilizar paradigmas de programación (imperativo, orientado a objetos, declarativo), y está familiarizado con las distintas fases del proceso de construcción de un programa (análisis, diseño, implementación).

En esta asignatura construiremos aplicaciones informáticas en las que combinaremos el uso esos paradigmas. El proceso de construcción estará guiado por la exigencia de calidad Aprenderemos a aplicarles un control de calidad que mida si:

- se garantiza la seguridad de la aplicación frente a accesos no permitidos.
- se garantiza la consistencia de los datos con que se trabaja.
- la aplicación es amigable para el usuario, esto es: si su uso resulta intuitivo a las personas que harán uso de ella (un cliente, otro miembro del equipo de investigación).

Interés de la materia para el futuro profesional

Una opinión generalizada entre licenciados y empleadores es que en los estudios de Matemáticas existe un déficit en cuanto a conocimientos de Informática[1]. Esos estudios son, precisamente, los más valorados por la empresa (que los considera muy relevantes, por ejemplo, en el proceso de selección de personal). El bloque formativo en Informática tiende a cubrir esa necesidad de formación.

Por otro lado, hay toda una serie de competencias cuya relevancia en la vida profesional está unánimemente aceptada, y que no se adquieren durante la carrera. Dichas competencias son instrumentales (toma de decisiones, capacidad de comunicación), personales (integración efectiva en un equipo) , sistémicas (iniciativa, motivación por la calidad), profesionales (aplicación de los conocimientos a la práctica)... Competencias que marcan el salto entre un estudiante y un profesional (de la empresa, la investigación o la docencia). Es deseable acortar este salto para facilitar la empleabilidad de los estudiantes[2]. Una posible estrategia para ello es incorporar la práctica profesional al currículo universitario[3]. En nuestra asignatura simulamos un entorno en que puedan desarrollar actividades propias de la profesión de informático. Por supuesto, la simulación es limitada y no comparable con una experiencia real de trabajo; pero es útil para que el estudiante tome conciencia de necesidades que van a ser fundamentales en su vida profesional: seguir una metodología de trabajo, documentar adecuadamente el propio trabajo, apreciar un entorno de trabajo colaborativo (los miembros de un equipo colaboran para obtener un producto; todo equipo utilizará un producto elaborado por otro, o tendrá presente que su producto será esencial para el trabajo de otros equipos).

---

[1] Agencia Nacional de Evaluación de la Calidad y Acreditación, *Título de Grado en Matemáticas*, marzo 2004. http://www.aneca.es/modal_eval/docs/conver_matematicas.pdf

Agencia Nacional de Evaluación de la Calidad y Acreditación, *Encuesta de Inserción Laboral,* marzo 2004. http://www.aneca.es/docs_trabajo/doc_trabajo.html

[2] Esta idea es recurrente en los textos relacionados con la convergencia de Bolonia. Ver, por ejemplo, http://www.crue.org/mensajeconvESP.htm

[3] ACM/IEEE CS Joint Task Force, *Computing Curricula 2001*, Final Report, December 2001, http://www.acm.org/sigcse/cc2001/15-12-2001.html.

Objetivos – Competencias

Nuestro propósito es que, a lo largo del curso, el estudiante aprenda a construir aplicaciones informáticas sencillas que accedan a bases de datos. Para ello debe:

- **Conocer distintos modelos de arquitectura** de una aplicación (cliente-servidor, multicapa) y adquirir criterios para decidir cuándo usarlas.

- **Aprender a seguir la metodología** que se sigue en la comunidad de software para el desarrollo de aplicaciones, y a utilizar **tecnologías** de bases de datos y programación en la WEB.

- **Adquirir destreza en el uso de distintos paradigmas de programación** que están presentes en las distintas capas de una aplicación: declarativo, imperativo, orientado a objeto.

- **Aprender a someter su trabajo a un control de calidad**, referido a garantizar la **seguridad y consistencia de los datos** almacenados, la **usabilidad** de la aplicación y la **ergonomía** en el diseño de la interfaz.

- **Desarrollar la capacidad de trabajo en equipo,** necesario para construir aplicaciones con un cierto nivel de complejidad.

- **Acostumbrarse a comunicarse de modo eficaz oralmente** (con el cliente que encarga la aplicación, con el asesor al que se consultan dudas tecnológicas y metodológicas, con los demás equipos en seminarios conjuntos) **y a documentar el propio trabajo**. Este último punto es absolutamente necesario ya que, en el mundo real, los distintos equipos que participan en la construcción de una aplicación a menudo reutilizan el trabajo de otros equipos.

Bibliografía básica

"Ingeniería del Software (6ª edición)", Roger S. Pressman -Editorial Mc Graw-Hill 2005
ISBN: 9701054733

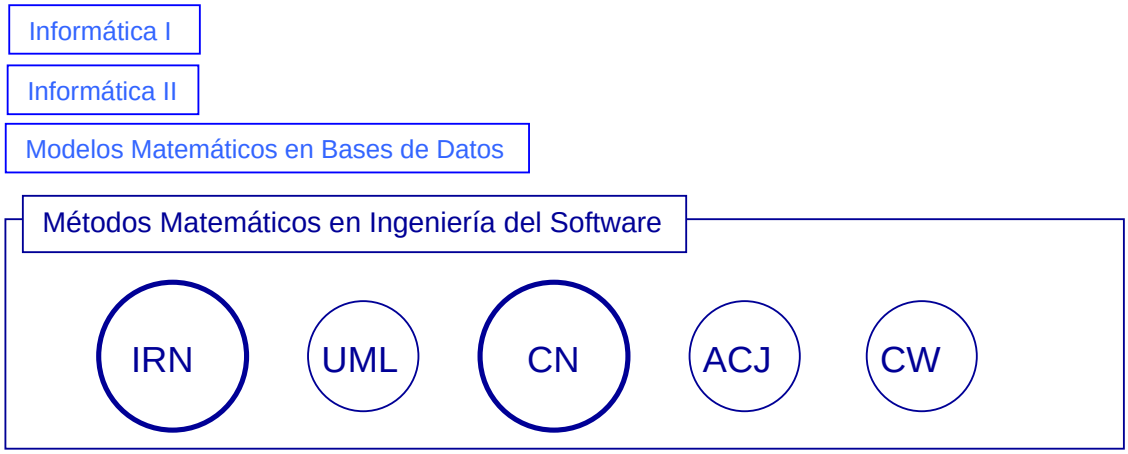
"Software Engineering" (8ª edición), Ian Sommerville . Addison Wesley 2006

"Fundamentals of Software Engineering" (2ª edición) Prentice-Hall, 2002
Carlo Ghezzi , Mehdi Jazayeri , Dino Mandrioli
ISBN 0-13-305699-6

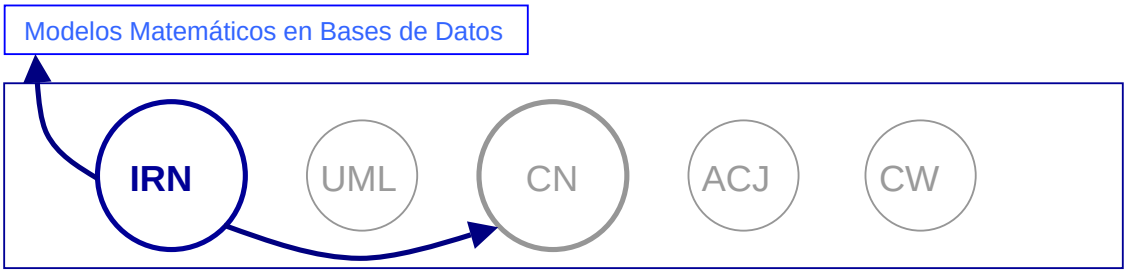
Contenidos del curso

Tema 1: Implementación de Reglas de Negocio (IRN)
Tema 2: Nociones de UML: diagramas de clases y de actividad (UML)
Tema 3: Capa de Negocios (CN)
Tema 4: Aplicaciones Cliente con Java (ACJ)
Tema 5: Capa WEB (CW)

En las siguientes páginas se detallarán el sentido, contenidos, competencias trabajadas y bibliografía de cada tema, así como la relación entre éste , otros temas y el resto de asignaturas del bloque. Para representar los temas y asignaturas utilizaremos los siguientes iconos:



Tema 1: Implementación de Reglas de Negocio



Sentido del tema: Este tema sirve de enlace entre la asignatura anterior del bloque de Informática (Modelos Matemáticos en Bases de Datos) y la presente. En aquélla el estudiante conoce el paradigma de programación declarativo (con SQL) y aprende a diseñar bases de datos que modelen una parte del mundo que nos interesa como problema (el Universo de Discurso) utilizando distintos modelos (entidad-asociación, relacional). Al final del curso se le hace ver que existen muchas características del mundo real que no pueden representarse con esos modelos.

Al principio de este curso se recuerda esa impotencia de los modelos y se formaliza la noción de regla de negocio (comportamiento del mundo real que queremos modelar). La identificación e implementación de las reglas de negocio garantiza que los datos almacenados en la base de datos sean consistentes. Esto es, que la base de datos mimetice el comportamiento del mundo real y no permita situaciones imposibles en éste.

Para implementar las reglas de negocio es necesario completar el lenguaje declarativo con estructuras procedurales, típicas de la programación imperativa (sentencias estructuras, funciones y procedimientos).

Epígrafes:

1.1 Presentación del lenguaje PL/SQL
1.2 Nociones básicas: variables, tipos de datos, sentencias condicionales, bloque
1.3 Inclusión de sentencias SQL en bloques PL/SQL
1.4 Cursores
1.5 Procedimientos, funciones, packages
1.6 Tratamiento de excepciones
1.7 Triggers

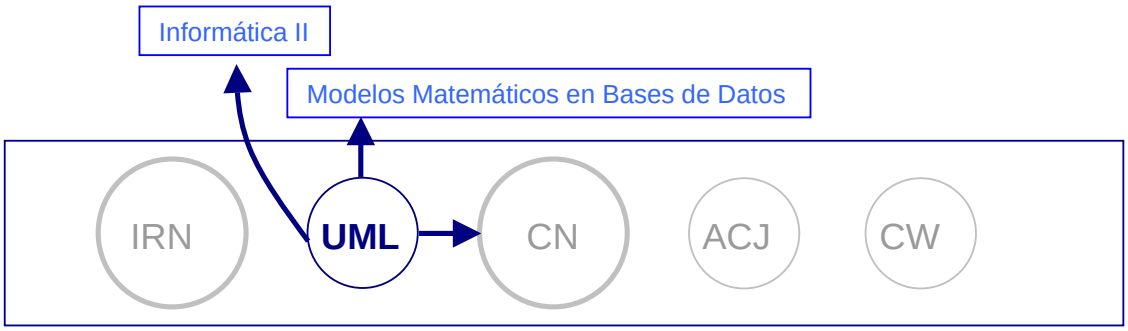
Competencias trabajadas:

En este tema se refuerza la **destreza en** el uso de **lenguajes declarativos y procedurales**. Desde el mismo inicio del curso se insiste en la necesidad de **garantizar la consistencia de los datos** almacenados, necesidad que queda cubierta mediante la identificación e implementación de las reglas de negocio.

Material de estudio:

"Reglas de Negocio", apuntes de J. Borque y J. Lloret.
"Oracle PL/SQL Language", S. Feuerstein, B. Pribyl y C. Dawes, edit O'Reilly.

Tema 2: Nociones de UML: diagramas de clases y de actividad



Sentido del tema: En este tema volvemos a recalcar la importancia del diseño en el proceso de construcción de una aplicación informática. Ésta ha sido una de las ideas recurrentes en las asignaturas previas. En Informática I el objetivo de la fase de diseño era determinar el algoritmo a seguir para resolver un problema, como paso previo a la implementación ("cuidado con el síndrome del teclado flojo: antes de empezar a teclear, asegúrate de que sabes qué quieres teclear"). En Informática II profundizamos en esta idea, y empezamos a representar gráficamente las relaciones de herencia, agregación e inclusión entre clases. En Modelos Matemáticos en Bases de Datos dimos un salto cualitativo: dedicamos la mitad del cuatrimestre al diseño de bases de datos a distintos niveles (conceptual, siguiendo el modelo entidad-asociación; lógico, con el modelo relacional; físico).

En resumen, el diseño ha ido adquiriendo una importancia creciente. De las reticencias iniciales de los estudiantes en primero (que, impacientes por teclear programas, percibían el diseño en seudocódigo como la obligación de aprenderse un lenguaje de programación añadido) hemos asentado la idea de que el éxito de una aplicación (esto es, su capacidad para resolver problemas y para adaptarse a la evolución de las necesidades) depende en buena medida de su capacidad para modelar correctamente el trozo de mundo que nos interesa como problema y, por consiguiente, del diseño.

En este tema formalizamos las nociones de diseño de clases siguiendo el estándar UML, recalcando las semejanzas y diferencias con el modelo entidad-asociación. Es básico saber diseñar clases, ya que las aplicaciones que vamos a construir están pensadas para la WEB , y la WEB "habla" un lenguaje orientado a objetos (sea en Java o en la plataforma .NET). Presentaremos igualmente los diagramas de actividad, que utilizaremos para modelar la navegación a través de la aplicación.

Epígrafes:

2.1 Diagramas de clases
2.2 Diagramas de actividades

Competencias trabajadas:

En este tema se induce al estudiante a **seguir una metodología** para el desarrollo de aplicaciones, haciendo hincapié en la fase de diseño. Igualmente **se profundiza en** las nociones de los lenguajes que siguen **el paradigma orientado a objetos**.

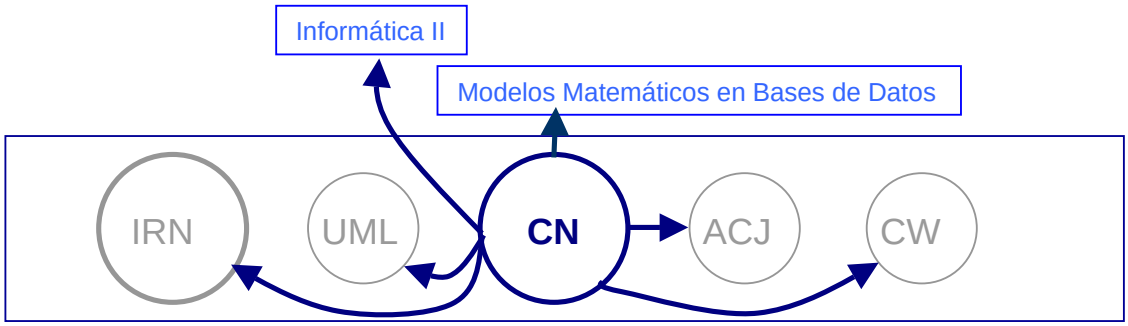
A través del uso de diagramas de actividad se pretende que el estudiante **aprenda a programar una aplicación pensando en** el usuario, siguiendo criterios de **amigabilidad y usabilidad**. La aplicación debe guiar al usuario de modo que su uso sea intuitivo.

Material de estudio:

" *The Unified Modeling Language User Guide* ", Grady Booch, Jim Rumbaugh, Ivar Jacobson, Addison-Wesley

"UML 2", J. Arlow e I. Neustadt, Anaya.

Tema 3: Capa de Negocios



Sentido del tema: En este tema abordamos el segundo nivel de una arquitectura multicapa. A estas alturas del curso ya sabemos diseñar y construir bases de datos siguiendo el modelo relacional; sabemos igualmente programar en Java, lenguaje orientado a objetos ampliamente extendido en la WEB. La capa de negocios es un puente entre estos dos mundos: traduce los elementos del relacional al orientado a objetos, y realiza las operaciones necesarias para garantizar la persistencia de los objetos, almacenándolos en la base de datos relacional.

En la capa de negocios, además, se puede programar la lógica de validación de las reglas de negocios. Se garantiza que sus objetos modelen fielmente el comportamiento de los entes del mundo real que representan. La capa actúa, pues, como un filtro: asegura que los datos que se envían al servidor de base de datos están libres de error.

Contenido:

3.1 Transformaciones entre los modelos relacional y orientado a objetos.
3.2 Programación de reglas de negocio en un lenguaje orientado a objetos.

Competencias trabajadas

En este tema el estudiante **confronta** los modelos relacional y orientado a objetos: lo que ha programado en **un lenguaje declarativo** (SQL) **y** lo que sabe programar en **un lenguaje orientado a objetos**. Aprende a identificar sus rasgos comunes y a traducir aquellos elementos de un mundo que no tienen correlato directo en el otro. De este modo desarrolla su destreza en programación con lenguajes de ambos paradigmas y profundiza su conocimiento de las nociones de cada paradigma.

Igualmente, el estudiante agudiza su conciencia de que una de las responsabilidades del programador de aplicaciones es **garantizar la consistencia de los datos** con que trabaja. Aprende a hacerlo **en distintas capas de una aplicación**, **y a identificar las ventajas e inconvenientes que esta programación plantea en cada capa**.

En este tema abordamos el segundo nivel de una arquitectura multicapa. A estas alturas del curso ya sabemos diseñar y construir bases de datos siguiendo

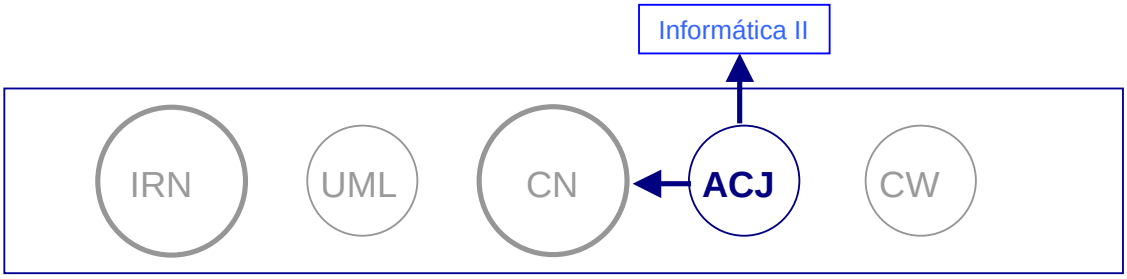
Material de estudio:

"Piensa en Java", Ecker, Prentice-Hall, 2002

"The Java Tutorial", M. Campione, K. Walrath, A. Huml Addison-Wesley, 2001

"Oracle Jdeveloper 10g Handbook", P. Koletzke, P. Dorsey y A. Faderman, McGraw-Hill, 2006

Tema 4: Aplicaciones Cliente con Java



Sentido del tema: En temas anteriores hemos aprendido a construir clases que traduzcan las estructuras de datos las restricciones de una base de datos relacional, y que modelen las características y comportamientos de los entes del mundo real. En este tema aprenderemos a mostrar los objetos de esas clases a los usuarios de la aplicación utilizando los componentes de la biblioteca gráfica Swing. Analizaremos cuál es el componente Swing más adecuado para representar cada uno de los atributos de los objetos, en función de su semántica.

Contenidos:

4.1 Interfaz de usuario: nociones básicas sobre usabilidad y ergonomía.
4.2 Componentes de la biblioteca Swing
4.3 Pautas generales para asociar un componente a un atributo en función de las reglas de negocio que le atañen.

Competencias trabajadas

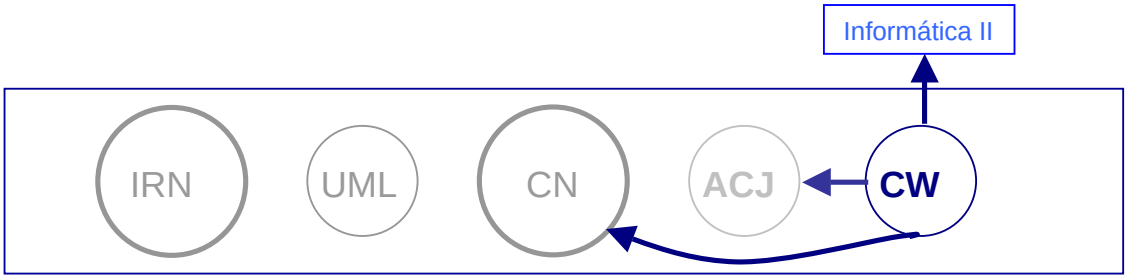
En este tema el estudiante afronta la tarea de programar la interfaz de una aplicación siguiendo criterios de **amigabilidad y usabilidad**. El usuario debe encontrar la aplicación fácil e intuitiva de uso, tanto en el modo de consultar, modificar y rellenar datos como en la **navegación** de una pantalla a otra.

La versatilidad de una aplicación cliente Java da gran libertad al programador para diseñar el aspecto de la interfaz. El estudiante aprende, a través del ensayo–crítica de sus propuestas y del análisis de interfaces de diversas aplicaciones encontradas en la WEB, pautas básicas para **diseñar una interfaz ergonómica**.

Además, la programación usando componentes Swing permite al estudiante **desarrollar** su **destreza en la programación orientada a objetos** y **asentar nociones características de ella**, como las de herencia y polimorfismo.

Material de estudio:
"Java Swing", M. Loy, R. Eckstein, D. Wood, J. Elliott, B, Cole, edit. O'Reilly, 2002

Tema 5: Capa WEB



Sentido del tema: Empezamos el tema reflexionando sobre las limitaciones de una aplicación cliente Java que evoluciona a través de sucesivas versiones, cada vez más completas, cuando se incrementa el número de usuarios: es difícil garantizar que todos estén utilizando la versión más reciente (un usuario no tiene por qué actualizar periódicamente la versión que utiliza, puede darse incluso que los recursos de su ordenador no permitan siquiera ejecutar las versiones más avanzadas..). Como solución se plantea un nuevo modelo de arquitectura: se añade una nueva capa (capa WEB) responsable de construir las páginas que enviará al ordenador del usuario.

Contenidos:

5.1 Patrón Modelo-Vista-Controlador
5.2 Tecnología jsp
5.3 Tecnología struts.

Competencias trabajadas

En este tema el estudiante **adquiere una perspectiva completa de** la estructura de **una arquitectura multicapa**. Igualmente, **se familiariza con algunas** de las **tecnologías** utilizadas actualmente en las aplicaciones **WEB**.

Material de estudio:

"Oracle Jdeveloper 10g Handbook", P. Koletzke, P. Dorsey y A. Faderman, McGraw-Hill, 2006

Metodología

A principios de curso planteamos a los estudiantes que la asignatura está orientada a cumplir un objetivo: la construcción de una aplicación informática que permita consultar, añadir, modificar y borrar información almacenada en una base de datos. Ésta ha sido previamente diseñada y creada por estudiantes de la asignatura Modelos Matemáticos en Bases de Datos. Los estudiantes se organizan en equipos, cada uno de los cuales es responsable de un módulo de la aplicación.

Cada equipo mantiene con el profesor una serie de reuniones de proyecto. En ocasiones, el profesor actúa con el rol de cliente: explica qué necesita de la aplicación, así como su grado de satisfacción con las soluciones que el equipo le va aportando (con la funcionalidad, facilidad de uso y ergonomía del módulo de la aplicación que desarrolla el equipo). En otras, actúa con el rol de asesor, discutiendo los problemas que al equipo le surgen durante el desarrollo de su módulo.

A cada equipo se le pide que planifique su trabajo, y que dé cuenta de la evolución del mismo en los plazos acordados. Igualmente, se le pide que compute las horas dedicadas al proyecto, utilizando una hoja de trabajo.

A lo largo del cuatrimestre desarrollamos:

- sesiones magistrales, en que se proporcionan los conocimientos necesarios para el desarrollo de la aplicación.

- prácticas guiadas, a través de las cuales los estudiantes aprenden a utilizar las tecnologías con que se construirá la aplicación.

- sesiones de discusión, en que un equipo plantea a la clase un problema particular que ha encontrado y propone una solución al mismo, o bien presenta su módulo en distintas etapas de desarrollo.

Las sesiones de discusión se desarrollan del siguiente modo: los integrantes del grupo exponen su proyecto durante 30-40 minutos. A continuación hay un turno de preguntas, tras el cual se pide a los demás estudiantes que en 5-10 minutos rellenen un formulario rápido indicando:

- Cuál ha sido el objetivo de la charla
- Cuál es el resumen de la charla
- Puntos destacados
- Valoración de la exposición.

A los integrantes del grupo se les pide igualmente que se autoevalúen: qué valoración global hacen de su presentación y qué es lo que menos les ha gustado.

El objetivo de la sesión es doble: por un lado, cada grupo informa al resto de sus avances. Por otro, se evalúa el modo en que nos expresamos en público. Insistimos en la necesidad de recalcar cuál es el objetivo principal de la charla (la idea que todo el mundo tiene que tener clara al salir), de articular la presentación de modo que sea fácil seguir el hilo argumental, de evitar vicios de principiantes (como obsesionarse con los detalles técnicos de la charla).

A través de este método **el estudiante aprende a tratar con un cliente no experto** en Informática: a analizar un problema y proponer soluciones tecnológicas a partir de la descripción (no técnica) que éste hace de su problema y de sus críticas. **Adquiere el hábito de documentar** adecuadamente su módulo para que otros equipos puedan utilizarlo o engarzar su módulo con aquél **y a defenderse oralmente** en las sesiones de discusión.

Carga ECTS

La asignatura tiene asignados 7.5 créditos UZ (6.8 créditos ECTS). Calculamos pues un total de 6.8 * 25 = 170 horas de dedicación del estudiante.

Las horas se distribuyen del siguiente modo:

Actividades	Horas Presenciales	Factor	Horas de trabajo autónomo	Total
Clases teóricas	24	1,5	36	60
Prácticas guiadas	20	1	20	40
Sesiones de discusión (1)	8			8
Desarrollo del proyecto(2)	4	10	40	44
Pruebas	6	2	12	18
Total	62		108	170

(1) El tiempo que cada equipo dedica a preparar las sesiones de discusión que protagoniza se computa en las horas de desarrollo de su proyecto.

(2) Las 4 horas presenciales de desarrollo de proyecto corresponden a las sesiones de asesoría y reuniones con el cliente.

Evaluación

Se proponen al estudiante dos modalidades de evaluación:

- La "formal", presentándose a las convocatorias oficiales a que tienen derecho por estatutos de la universidad.

- Una evaluación continua, que se basa en dos pruebas a lo largo del curso y en la evaluación del proyecto. Para evaluar el proyecto se exigirá la entrega de la aplicación desarrollada en equipo junto con toda la documentación generada, y se tendrán en cuenta la regularidad en la asistencia a las reuniones de seguimiento de proyecto y la calidad de las exposiciones públicas.

  Cada una de las pruebas tiene un peso del 35% en la nota final, y se plantean de modo que su preparación suponga poco tiempo extra de estudio para el estudiante que siga las clases al día y realice regularmente las prácticas guiadas.

  La evaluación del proyecto supone un 30% de la nota final. Se evaluará:
  - El correcto funcionamiento del módulo (1 punto).
  - La calidad de la documentación (1 punto).
  - La calidad del trabajo en equipo (1 punto): se tendrán en cuenta la claridad y eficiencia en los seminarios, la exigibilidad individual (cada miembro del equipo debe demostrar en las entrevistas con el profesor que domina no sólo la parte del trabajo que le ha sido asignada, sino también la parte de sus compañeros) y la reflexión sobre el trabajo realizado (la autocrítica en la entrevista final: qué consideras que has aprendido del trabajo en equipo, qué errores habéis detectado en vuestro modo de trabajar...)