Proyecto Fin de Carrera
Ingeniería en Informática
Curso 2010/2011

Sistema para la petición de cita de tutoría usando tecnología de Portlets

Marcos Mainar Lalmolda

Director: Pedro Javier Álvarez Pérez-Aradros

Departamento de Informática e Ingeniería de Sistemas
Centro Politécnico Superior
Universidad de Zaragoza

Zaragoza, febrero de 2011

Sistema para la petición de cita de tutoría usando tecnología de Portlets

RESUMEN

Este proyecto fin de carrera comprende el análisis, diseño e implementación de un sistema accesible vía Web que facilite a profesores y alumnos la gestión de la asistencia a las tutorías.

Desde un punto de vista funcional, el sistema permite a los profesores configurar las asignaturas para las cuales quieren utilizar el sistema y establecer sus horarios de tutorías. Por otro lado, los alumnos podrán consultar disponibilidad horaria para asistir a estas tutorías y solicitarlas en base a esta disponibilidad. El sistema notificará al alumno vía correo electrónico la aceptación de su solicitud. Antes de cada tutoría, el profesor recibirá un informe por correo electrónico con el listado de las solicitudes existentes. Con cada solicitud, un alumno podrá también exponer el objeto de la tutoría, facilitando al profesor una posible preparación previa si fuese necesario.

Además de la funcionalidad básica expuesta en el párrafo anterior, el sistema también ofrece otras posibilidades a profesores y alumnos. Los profesores pueden visualizar, aplicando diferentes criterios, las tutorías pendientes de ser atendidas, consultar su histórico de solicitudes de tutorías o anotar qué alumnos acuden a las tutorías que solicitaron. También podrán configurar la frecuencia de los informes de solicitudes de tutorías (por cada nueva solicitud, diarios, semanales, etc.) y la forma en que se lleva a cabo la asignación de turnos de tutorías a los alumnos: permitiéndoles elegir hora entre las disponibles o asignándoles una hora concreta de forma consecutiva. Por otro lado, un alumno podrá consultar todas las tutorías asignadas, modificarlas o cancelarlas. Adicionalmente, el sistema detectará posibles alumnos que repetidamente no asisten a las tutorías solicitadas, realizará automáticamente copias de seguridad de su información, almacenará ficheros de *log* con todas las acciones que ocurran en el sistema para su posterior análisis e implementará un acceso seguro basado en claves privadas.

El sistema desarrollado consta de dos aplicaciones:

- Una aplicación Web de configuración de asignaturas y sus horarios de tutorías utilizada por los profesores.
- Una aplicación Portlet utilizada por los alumnos para la solicitud de cita de tutoría.

La tecnología de Portlets utilizada permite crear aplicaciones Web fácilmente integrables en un portal Web. Gracias a ella, los profesores pueden, de forma muy sencilla, poner el sistema a disposición de sus alumnos en su página Web.

Por último, el sistema desarrollado se ha instalado en el servidor Alkaid del Centro Politécnico Superior para ponerlo a disposición de profesores y alumnos.

Agradecimientos

En primer lugar, quería agradecer al director de este proyecto, Pedro Álvarez, por darme la oportunidad de realizarlo y por su ayuda a lo largo del mismo.

A mis padres por haberme apoyado y aconsejado acertadamente a lo largo de toda mi vida.

A los amigos que me preguntaban y se interesaban constantemente durante todo este tiempo. También agradezco los consejos y experiencias de los que habían realizado PFC antes así como las fuentes de LaTeX facilitadas.

A Ismael Saad por las numerosas horas que hemos compartido como compañeros de prácticas y estudio a lo largo de toda la carrera. También por proporcionarme las fuentes de LaTeX y realizar pruebas sobre el sistema.

Por último, también me gustaría nombrar a los desarrolladores que, de forma voluntaria, dedican su tiempo a programar algunas de las herramientas utilizadas, sin las cuales este proyecto habría sido mucho más costoso.

Índice general

Índice de figuras

Capítulo 1

Introducción

En este capítulo se ofrece una visión inicial del proyecto. En la sección 1.1 se pone en contexto el proyecto. En la sección 1.2 se explica la motivación del proyecto, derivada del contexto actual, y su propósito principal. En la sección 1.3 se enumeran las tecnologías y herramientas utilizadas. Finalmente, en la sección 1.4 se explica la organización del resto del documento.

1.1 Contexto del proyecto

Actualmente, por lo general, cada profesor universitario establece un horario semanal de tutorías para atender a sus alumnos y resolver sus dudas y problemas. Los profesores comunican este horario a sus alumnos durante los primeros días de clase, mediante una nota en la puerta de su despacho, por correo electrónico, a través de su página Web, o combinando varios de estos métodos.

En cualquier caso, la organización de las tutorías recae a título individual de cada profesor. Algunos profesores solicitan a sus alumnos el envío de correos electrónicos para concertar las tutorías. Sin embargo, en la mayoría de los casos, los alumnos acuden sin previo aviso. Esto impide al profesor la posibilidad de preparar la tutoría con antelación, por ejemplo, recopilando material que pueda resultar útil para el alumno. Además, conforme se acerca la época de exámenes, el número de alumnos que asisten a estas sesiones aumenta significativamente, provocando su acumulación a determinadas horas y, por tanto, largas esperas.

1.2 Motivación

Desde el punto de vista de los alumnos, resulta interesante conocer la disponibilidad horaria de un profesor para asistir a sus tutorías, solicitarlas de acuerdo a esta

disponibilidad, exponer el objeto de las mismas y poder modificarlas o cancelarlas. Por otro lado, desde el punto de vista del profesor, resulta muy útil saber si existen alumnos que pretenden acudir a sus tutorías y el motivo de su asistencia, para facilitar una preparación previa si fuera necesaria. Todo ello redunda en un mejor aprovechamiento del tiempo tanto del profesor como del alumno y facilita una atención más individual y especializada en las tutorías.

El objetivo de este proyecto fin de carrera es el diseño e implementación de un sistema accesible vía Web que facilite a profesores y alumnos la gestión de la asistencia a las tutorías. Por tanto, este proyecto resuelve un "problema" real en el contexto universitario.

Este proyecto ha sido realizado en el Departamento de Informática e Ingeniería de Sistemas de la Universidad de Zaragoza. El sistema desarrollado está siendo utilizado de manera experimental por el director del proyecto. A través de su página Web, `http://webdiis.unizar.es/~alvaper/`, los alumnos matriculados en las asignaturas que imparte pueden solicitar cita de tutoría.

1.3 Tecnologías y herramientas utilizadas

En esta sección se enumeran las tecnologías y herramientas empleadas en el proyecto, clasificándolas por categorías en función de su ámbito de aplicación. Desde el principio del proyecto, se estableció un entorno de trabajo completo, es decir, se evaluaron y eligieron las diferentes tecnologías y herramientas de soporte al proyecto. En el anexo B se incluye un análisis completo de las tecnologías principales utilizadas en el proyecto. El anexo A contiene un análisis específico de la tecnología de Portlets. En la sección 2.3 se ofrece un resumen del análisis de las tecnologías necesarias para el desarrollo del proyecto. Se explican brevemente y se justifican las razones de su elección.

Las tecnologías y herramientas utilizadas en el proyecto se muestran en la tabla 1.1.

Categoría	**Tecnología/Herramienta**
Generales de desarrollo	Java EE [16] Portlets [1] Eclipse IDE. XML.
Gestión y documentación del proyecto	Microsoft Project Subversion Maven [59] OpenOffice LaTeX con el editor Kile [60]
Soporte al análisis y diseño del sistema	Microsoft Visio ArgoUML ObjectAid UML Explorer Edge Diagrammer
Implementación de la BBDD del sistema	MySQL Community Server [28] Hibernate Framework [26] Hibernate Tools [48] Java Persistence API (JPA) [18]
Implementación de la lógica del sistema	Spring Framework [27] JavaMail [58] Quartz Job Scheduler [49] iTextPDF [50] SLF4J [51] Log4J [52] DWR (Direct Web Remoting) [53]
Implementación de la interfaz de usuario	Apache Tapestry Web Framework [24] HTML, CSS, JavaScript JavaScript Object Notation (JSON) [57] Componentes AJAX [4] Java Servlet [22] JavaServer Pages (JSP) [5] JavaServer Pages Std. Tag Library (JSTL) [54]
Pruebas del sistema	JUnit [56] Mozilla Firefox, Google Chrome Microsoft Internet Explorer y Apple Safari
Instalación del sistema	Apache Tomcat [21] OpenPortal Portlet Container [20] Liferay Portal Server [55]

Tabla 1.1. Tecnologías y herramientas utilizadas en el proyecto

1.4 Estructura de la memoria

La memoria consta de cinco capítulos que resumen el proyecto. Además del presente capítulo de introducción al proyecto, el resto de capítulos profundizan en el trabajo realizado.

El capítulo 2 describe el análisis del problema a resolver. En primer lugar, se describen los objetivos generales del proyecto. En segundo lugar, se detallan los requisitos concretos en los que se materializan los objetivos generales anteriores. Por último, se realiza un análisis de las tecnologías requeridas para el proyecto.

En el capítulo 3 se explica el diseño del sistema. Se comienza con una visión global del entorno de aplicación del sistema. Posteriormente, se describe la arquitectura general del sistema. Y, sucesivamente, se va desglosando dicha arquitectura para mostrar las partes más relevantes del diseño.

En el capítulo 4 se describe la gestión del proyecto: metodología utilizada, planificación, estimaciones iniciales y esfuerzo real dedicado.

Finalmente, en el capítulo 5 se presentan los resultados del proyecto, reflexiones personales del autor y posibles trabajos futuros relacionados con el proyecto realizado.

La memoria contiene, además, una serie de anexos que complementan la información de los capítulos anteriores. Estos anexos son los siguientes:

- El anexo A contiene un análisis detallado de la tecnología de Portlets.
- El anexo B contiene un análisis del resto de tecnologías principales utilizadas en el proyecto.
- El anexo C contiene información complementaria del diseño del sistema.
- En el anexo D se incluye el manual de usuario del sistema.
- El anexo E corresponde al manual de instalación y mantenimiento del sistema.
- El anexo F incluye las transparencias de un seminario sobre la tecnología de Portlets elaborado al principio del proyecto.

Finalmente, se presentan las referencias bibliográficas que han servido de soporte al proyecto.

Capítulo 2

Análisis del problema

En este capítulo se presenta el análisis del problema a resolver. En la sección 2.1 se enumeran los objetivos generales del proyecto derivados del análisis de las necesidades de los usuarios del sistema y de la naturaleza del mismo. Estos objetivos se fueron refinando y descomponiendo en detalle para dar lugar a los requisitos concretos del sistema que se detallan en la sección 2.2. Por último, en la sección 2.3 se incluye un análisis de las tecnologías necesarias para desarrollar el sistema.

2.1 Objetivos generales

El objetivo de este proyecto es el diseño e implementación de un sistema accesible vía Web que facilite a profesores y alumnos la gestión de la asistencia a las tutorías. Los objetivos generales derivados son los siguientes:

- Estudiar y analizar el sistema utilizado actualmente por los profesores universitarios para la gestión de las tutorías.
- Diseñar y desarrollar dos aplicaciones: una aplicación Web utilizada por los profesores y una aplicación Portlet utilizada por los alumnos.
- Los objetivos de la aplicación Web son:
  - Permitirá a los profesores, de forma muy sencilla, configurar sus asignaturas, dar de alta a sus alumnos y establecer sus horarios de tutorías.
  - Permitirá visualizar, aplicando diferentes criterios, las tutorías pendientes de ser atendidas y las históricas, anotar qué alumnos acuden a las tutorías que solicitaron y cancelar sesiones de tutorías.

  - Permitirá al profesor recibir informes con las solicitudes de tutorías por correo electrónico. El profesor podrá configurar si desea recibir informes diarios con las tutorías del día, informes al principio de la semana con las tutorías de toda la semana o una notificación individual por cada nueva solicitud de tutoría.
  - El profesor podrá configurar si desea que los alumnos puedan elegir hora de tutoría entre las disponibles o si la aplicación asignará automáticamente horas de forma consecutiva.
  - La aplicación deberá ser accesible a una amplia comunidad de profesores sin que tengan que realizar una instalación previa.

- Los objetivos de la aplicación Portlet son:
  - Permitirá a los alumnos consultar la disponibilidad horaria para asistir a las tutorías y solicitar una tutoría en base a esta disponibilidad.
  - Permitirá a los alumnos consultar las tutorías asignadas, modificarlas o cancelarlas.
  - La aplicación no requerirá que los alumnos se registren previamente.
  - La aplicación se integrará fácilmente en portales Web de forma que los profesores puedan ponerla a disposición de sus alumnos, para que éstos reserven cita de tutoría.

Ambas aplicaciones deberán facilitar especialmente la mantenibilidad, tanto desde el punto de vista de la mejora de la funcionalidad existente, como para la adición de nueva funcionalidad en el futuro.

2.2 Especificación de requisitos

Los objetivos generales se materializan en una serie de requisitos que se han clasificado en funcionales y no funcionales. Los requisitos funcionales son aquellos que describen el comportamiento del sistema. Los requisitos no funcionales se refieren a otras cuestiones como la tecnología concreta a utilizar, dónde se llevará a cabo la instalación y operación del sistema, etc.

2.2.1 Requisitos funcionales

Los requisitos funcionales se han dividido en cuatro grupos: requisitos desde el punto de vista del profesor, requisitos desde el punto de vista del alumno, requisitos desde el punto de vista del administrador y requisitos generales del sistema. Estas cuatro visiones del sistema permiten capturar adecuadamente todas las necesidades según el tipo de usuario, así como necesidades estructurales del sistema.

Requisitos desde el punto de vista del profesor:

1. Un profesor podrá registrarse en el sistema para poder gestionar de manera automática sus tutorías.
2. Un profesor podrá configurar las asignaturas para las que quiere utilizar el sistema. Para cada asignatura, importará un fichero con un formato preestablecido que contenga el listado de alumnos matriculados.
3. Un profesor podrá configurar para cada asignatura el día y la hora de sus tutorías (es decir, sus sesiones de tutorías) y el tiempo asignado a cada alumno.
4. Un profesor podrá visualizar las solicitudes de tutoría para cada sesión. Podrán ser aplicados diferentes criterios: temporales, por asignatura, etc.
5. Un profesor podrá recibir por correo electrónico información puntual sobre las solicitudes de tutoría o una planificación completa de sus sesiones.
6. Un profesor podrá consultar su histórico de solicitudes de tutorías aplicando criterios temporales y por asignatura.
7. Un profesor podrá consultar las incidencias de los alumnos. Se entiende por incidencia la modificación del horario de una tutoría previamente reservada o la cancelación de la misma.
8. Un profesor podrá cancelar sus sesiones de tutorías en un rango de fechas determinado. El sistema lo notificará por correo electrónico a los alumnos que hubieran reservado tutorías para esas sesiones.
9. Un profesor podrá cancelar una determinada reserva de tutoría. El sistema notificará por correo electrónico al alumno al que corresponda la reserva cancelada.
10. Un profesor podrá imprimir los listados con las solicitudes de tutorías recibidas, los históricos de tutorías y las incidencias de alumnos, utilizando un fichero en formato PDF que contenga el listado correspondiente.
11. Un profesor podrá configurar si prefiere que el sistema asigne hora de tutoría a los alumnos de forma automática o si éstos pueden elegir hora entre las disponibles.
12. Un profesor podrá registrar si un alumno acude a la tutoría que había solicitado.
13. Un profesor podrá comprobar qué alumnos no acuden a las tutorías asignadas.

Requisitos desde el punto de vista del alumno:

1. Un alumno accederá al sistema utilizando su NIP y su clave de acceso, que será válida para todas las asignaturas.
2. Un alumno podrá solicitar una tutoría indicando: la asignatura, el nombre del profesor, la fecha y hora a la que desea acudir (siempre que ésta esté libre). También podrá realizar comentarios que considere de interés para el profesor.
3. Una vez comprobado por el sistema que la solicitud de un alumno es correcta y factible, el alumno recibirá una notificación de aceptación por correo electrónico.
4. Un alumno podrá modificar o cancelar una tutoría previamente asignada (con al menos un día de antelación).
5. Un alumno podrá consultar las tutorías que tiene asignadas.

Requisitos desde el punto de vista del administrador:

1. El administrador podrá consultar las estadísticas generales del sistema (profesores, asignaturas, alumnos, horarios, grupos, titulaciones, etc).
2. El administrador podrá introducir las fechas relevantes del curso académico actual (inicio y fin del curso, inicio y fin de cada cuatrimestre).
3. El administrador podrá introducir un fichero que contenga los días festivos del curso académico actual.
4. El administrador podrá realizar y restaurar copias de seguridad del sistema.

Requisitos generales del sistema:

1. Para cada solicitud de tutoría, el sistema comprobará la disponibilidad del profesor para atender al alumno.
2. El sistema almacenará ficheros de *log* donde se registren las acciones más relevantes que realicen los usuarios del sistema. El formato del fichero deberá ser apropiado para futuros análisis de su contenido.
3. El sistema deberá ser capaz de realizar copias de seguridad de su información de manera periódica.
4. El sistema generará una clave de acceso concreta para cada nuevo alumno dado de alta. La clave será enviada a la cuenta de correo electrónico del alumno en la Universidad de Zaragoza.

2.2.2 Requisitos no funcionales

Los requisitos no funcionales del sistema son los siguientes:

1. El sistema de solicitud de tutoría será desarrollado con tecnología de Portlets.
2. El sistema de solicitud de tutoría deberá integrarse en un portal Web con un esfuerzo mínimo (cualquier profesor sin conocimientos técnicos deberá ser capaz de hacerlo en un breve espacio de tiempo).
3. El sistema dispondrá de una aplicación Web para que cada profesor configure sus asignaturas y su perfil y consulte la planificación de sus sesiones.
4. El sistema deberá estar instalado y operativo en un servidor que favorezca el uso por parte de una amplia comunidad de profesores y alumnos en el futuro.
5. Al finalizar el proyecto, deberá estar operativo y siendo utilizado al menos por el director del proyecto.

2.3 Análisis de tecnologías

En esta sección se presenta un análisis resumido de las tecnologías principales necesarias para el desarrollo del proyecto. El análisis completo se puede consultar en el anexo B. El anexo A es específico para la tecnología de Portlets.

2.3.1 Requisitos tecnológicos

Este proyecto plantea una serie de necesidades tecnológicas para resolver diferentes cuestiones del mismo. A continuación se enumeran y justifican brevemente estas necesidades.

1. *Plataforma de desarrollo y framework de soporte.* La plataforma debe permitir desarrollar aplicaciones Web modulares formadas por diferentes capas y componentes. El framework de soporte para la plataforma permitirá simplificar la configuración del sistema y, en general, su implementación, coordinando las diferentes capas y componentes.
2. *Tecnologías de soporte al almacenamiento de información.* Se requiere un repositorio permanente de información así como tecnologías para acceder al mismo con el fin de gestionar la información relativa al dominio del sistema.
3. *Tecnología Web orientada a servicios que facilite la integración en portales Web.* Gracias a esta tecnología, los profesores podrán poner el sistema a disposición de sus alumnos de forma sencilla.

4. *Tecnologías que permitan desarrollar interfaces Web de usuario dinámicas.* Se necesitan tecnologías abiertas y basadas en estándares que estén soportadas por cualquier navegador Web para maximizar la accesibilidad del sistema.

5. *Tecnologías que proporcionen el entorno de ejecución del sistema.* Dado que el sistema a construir es accesible vía Web, se necesita un servidor Web en el que residirán las aplicaciones que forman el sistema.

2.3.2 Tecnologías elegidas

Los requisitos tecnológicos anteriores llevan a elegir una serie de tecnologías. A continuación, se exponen dichas tecnologías explicando brevemente en qué consisten y justificando su elección.

Plataforma de desarrollo y framework de soporte

Se ha optado por la plataforma Java EE [16]. De esta forma, se aprovecha la experiencia previa del alumno con el lenguaje de programación Java y además el proyecto se beneficia de su compatibilidad con otras tecnologías empleadas. En particular, la tecnología de Portlets es un estándar de Java EE, por tanto, requería el uso de esta plataforma para la aplicación de reserva de cita de tutoría utilizada por los alumnos. Por ello, se decidió utilizar Java para el sistema completo.

Como framework de soporte se decidió utilizar Spring [27]. Spring es un framework de desarrollo de aplicaciones para la plataforma Java que favorece la adopción de buenas prácticas de programación. Se eligió fundamentalmente dado que permite construir sistemas a partir de componentes con bajo acoplamiento entre sí, gracias al principio de inversión de control (*Inversion of Control*, IoC) o inyección de dependencias (*Dependency Injection*, DI) [10] [11]. También, debido a que es estable, muy utilizado, compatible con numerosos frameworks y librerías, cuenta con excelente documentación y se adapta perfectamente al tipo de sistema a desarrollar.

Tecnologías de soporte al almacenamiento de información

Se ha elegido utilizar como repositorio una base de datos relacional y el gestor de bases de datos relacional MySQL [28]. La razón principal es su contrastado funcionamiento en numerosas aplicaciones Web disponibles en el mercado. También, para servir de aprendizaje al alumno, ya que anteriormente había utilizado el sistema gestor de bases de datos Oracle.

A pesar de elegir MySQL, se deseaba que el sistema diseñado fuera independiente del gestor de base de datos relacional concreto utilizado. Por ello, se ha utilizado el framework Hibernate. Hibernate [26] es el mapeador objeto-relacional (*Object-Relational Mapping*, ORM) por excelencia de la comunidad Java. Un mapeador objeto-relacional es una herramienta que facilita la conversión entre las entidades de una base de datos relacional y las clases del dominio de la aplicación. Además,

Hibernate se utiliza como tecnología de acceso a la base de datos relacional ofreciendo una capa de abstracción por encima de JDBC [79]. Por último, para facilitar el mantenimiento del sistema y permitir el cambio de Hibernate a otro proveedor de persistencia, se ha configurado Hibernate para que utilice la API estándar de Java para persistencia, conocida como JPA [18].

Tecnología Web orientada a servicios que facilite la integración en portales Web

Para este requisito se ha elegido la tecnología de Portlets [1]. Intuitivamente, los Portlets se pueden considerar como la siguiente generación de servicios Web. Un servicio Web tradicional devuelve los datos con la respuesta a la invocación de un método en un documento con formato XML, JSON, etc. Sin embargo, la representación visual de esa información depende de la aplicación concreta que invoca el servicio Web. En cambio, los Portlets, además de devolver la respuesta a la invocación, también devuelven información de interfaz de usuario, es decir, de cómo representar visualmente la información devuelta. Para ello, devuelven directamente código de *markup*. Por tanto, podrían considerarse como servicios Web orientados a presentación o visuales.

A nivel técnico, un Portlet es una mini aplicación Web interactiva gestionada y visualizada a través de un portal Web. La tecnología de Portlets es una API estándar de Java. La versión actual del estándar es la 2.0 [37]. La elección de la tecnología de Portlets se basó en dos razones fundamentales. Primero, en la necesidad de tener un sistema altamente modular e integrable que los profesores pudieran poner a disposición de sus alumnos de forma sencilla. En segundo lugar, por interés en aprender esta tecnología emergente, conocer sus posibilidades y, en definitiva, ampliar la formación del alumno. Durante la fase de formación del proyecto se realizó un análisis de esta tecnología donde se describe su contexto, estándares que intervienen, herramientas para desarrollar con ella, etc. Este análisis se puede consultar en el Anexo A.

Como consecuencia de la utilización de la tecnología de Portlets, se requiere un contenedor de Portlets o un servidor de portales Web (que integra el contenedor de Portlets) para proveer al Portlet de su entorno de ejecución. En este proyecto se ha utilizado el contenedor de Portlets OpenPortal Portlet Container [20]. Se eligió este contenedor concreto ya que puede ser instalado de forma muy sencilla en numerosos contenedores Web como Tomcat, JBoss, Jetty, Oracle WebLogic y GlassFish. También se ha utilizado el servidor de portales Liferay Portal Server [55] para realizar pruebas con Portlets y adquirir formación relativa a esta tecnología.

Tecnologías que permitan desarrollar interfaces Web de usuario dinámicas

Para implementar la interfaz de usuario Web de la aplicación de los profesores, se ha utilizado Tapestry. Tapestry es un framework Web orientado a componentes, de forma que modela cada página Web como un componente que puede reaccionar a diversos eventos. Se ha elegido Tapestry debido a que facilita el desarrollo de interfaces Web de usuario, reduce la configuración necesaria en el sistema

utilizando convenciones [14], se integra fácilmente con Hibernate y Spring y requiere escribir menos código fuente que otros frameworks Web similares como por ejemplo Struts [23]. Tapestry utiliza como tecnología subyacente la API estándar de Java Servlets [22]. Por tanto, el entorno de ejecución del sistema debe soportar esta tecnología. Sin embargo, el framework Tapestry en su versión actual (versión 5) no tiene soporte para desarrollo de Portlets. Por ello, para el desarrollo de la interfaz de usuario Web de la aplicación Portlet de los alumnos se ha utilizado la tecnología JavaServer Pages (JSP) [5]. De nuevo, el entorno de ejecución del sistema debe soportar esta tecnología.

Tecnologías que proporcionen el entorno de ejecución del sistema

Como tecnología que sirva para proveer al sistema de su entorno de ejecución se ha utilizado el servidor Web Apache Tomcat [21]. Tomcat es un servidor Web multiplataforma con soporte para Servlets y JavaServer Pages (JSP). Es un contenedor Web ligero ya que, al contrario que los servidores de aplicaciones de la plataforma Java EE como JBoss, GlassFish, etc., no requiere tantos recursos para su ejecución. La elección de Tomcat se basó en que utiliza pocos recursos y se integra muy fácilmente con el contenedor de Portlets OpenPortal Portlet Container. Se consideró también la posibilidad de usar Jetty, otro servidor Web con soporte para Servlets. Sin embargo, la disponibilidad de mejor documentación para Tomcat hizo que la decisión se decantara a favor de éste.

Capítulo 3

Diseño de la solución

En este capítulo se describe el diseño del sistema realizado. En la sección 3.1 se presenta el entorno de aplicación del sistema. En la sección 3.2 se explica su arquitectura. En las sucesivas secciones se explica y justifica el papel de cada una de las capas que conforman la arquitectura del sistema, sus aspectos de diseño de alto nivel y las tecnologías utilizadas. La sección 3.3 describe la capa de acceso a datos. La sección 3.4 describe la capa de dominio. La sección 3.5 describe la capa de lógica de negocio. Por último, la sección 3.6 corresponde a la capa de presentación.

En el anexo C se ofrece información complementaria del diseño del sistema desde una perspectiva cercana a la implementación.

3.1 Entorno de aplicación del sistema

El sistema diseñado se divide en dos aplicaciones tal y como se puede observar en la figura 3.1. Por un lado, una aplicación Portlet utilizada por los alumnos para la reserva de cita de tutorías, consulta de tutorías asignadas y modificación y cancelación de tutorías previamente asignadas. Por otro lado, una aplicación Web utilizada por los profesores para configurar su perfil, asignaturas y horarios de tutorías, gestionar la asistencia de los alumnos, consultar listados de solicitudes de tutorías, históricos e incidencias, etc. Esta aplicación también es utilizada por el administrador para configurar su perfil, las fechas del curso académico actual, visualizar estadísticas generales y realizar copias de seguridad de los datos. Ambas aplicaciones son accedidas por los usuarios utilizando un navegador Web. El sistema utiliza como mecanismo de almacenamiento de datos persistentes una base de datos que es compartida por ambas aplicaciones.

3.2 Arquitectura general del sistema

La arquitectura del sistema se basa en un diseño multicapa [2]. Los diseños multicapa son muy utilizados en las aplicaciones de gestión ya que permiten separar claramente

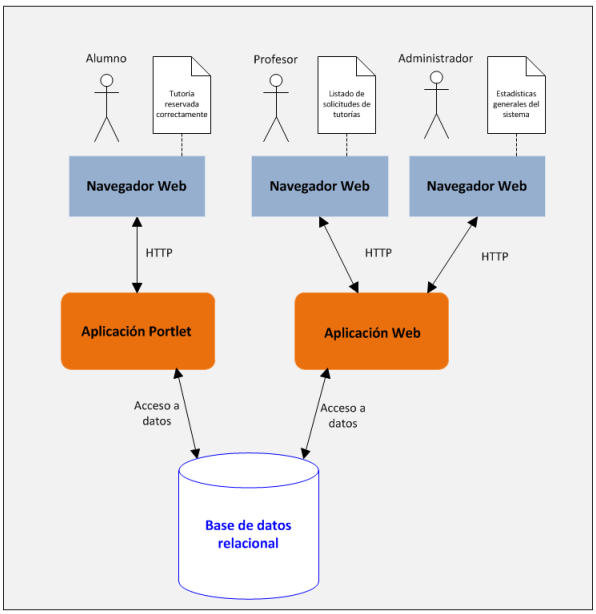


Figura 3.1. Entorno de aplicación del sistema

las tres funciones principales presentes en la mayor parte de este tipo de aplicaciones: tratamiento de los datos persistentes, lógica de negocio e interfaz del usuario. Esto permite reducir las dependencias entre capas facilitando el mantenimiento del sistema, dado que los cambios en una parte del mismo sólo afectarán a la capa concreta, sin necesidad de tener que modificar el resto de capas.

El sistema diseñado consta de cuatro capas tal y como se puede observar en la figura 3.2. Por un lado, existen tres capas denominadas *capa de acceso a datos*, *capa de lógica de negocio* y *capa de presentación Web*, que se corresponden con las funciones principales antes mencionadas. Estas tres capas, se construyen cada una en términos de la inmediatamente inferior. Por otro lado, hay una capa transversal al sistema, es decir, utilizada por las otras tres capas, denominada *capa de dominio*. Esta capa encapsula las denominadas clases persistentes, que surgen de utilizar un mapeador objeto-relacional (Hibernate en este caso) y que se corresponden con las entidades de la base de datos. Los objetos de esta capa modelan el dominio de la aplicación (Asignatura, Profesor, Horario, Alumno, etc.) y son utilizados e intercambiados por las otras tres capas. Estas clases siguen la filosofía POJO [30] de Java.

La capa de acceso a datos encapsula los componentes de acceso a los datos almacenados en la base de datos. Hay un componente de acceso a datos (denominado DAO) para cada clase persistente. La capa de lógica de negocio integra los componentes funcionales del sistema. La capa de presentación Web encapsula todos los componentes de la interfaz de usuario: las pantallas de cada aplicación y la lógica

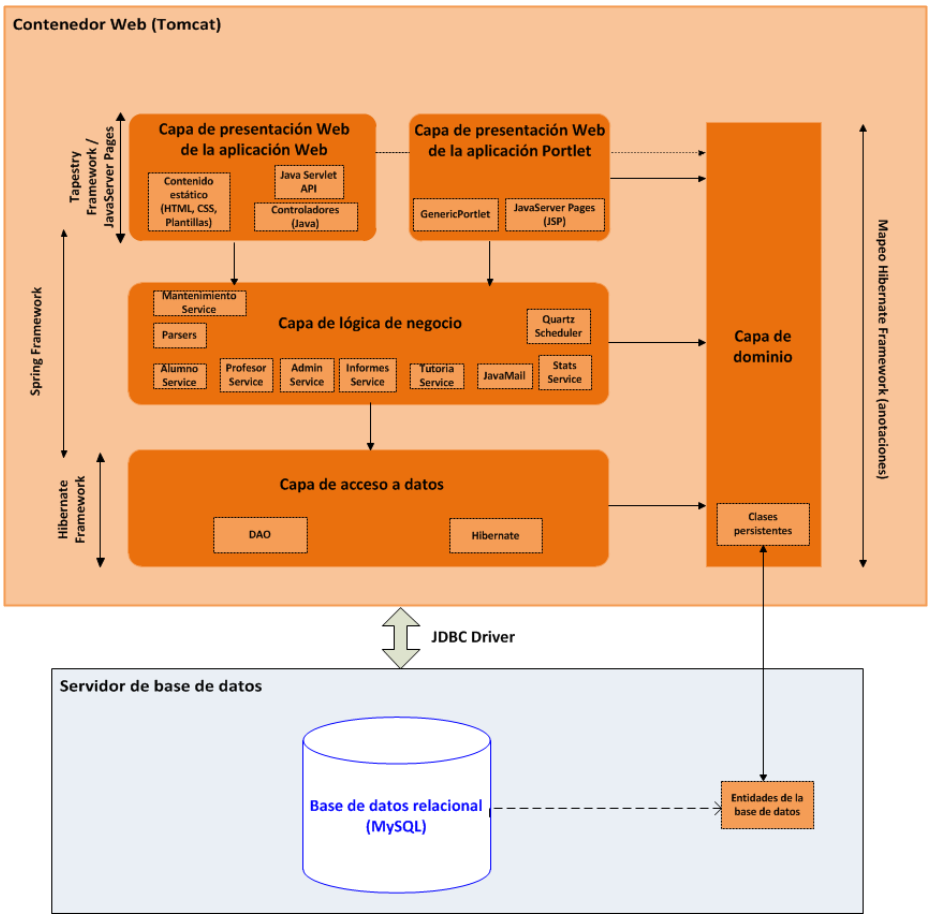


Figura 3.2. Arquitectura general del sistema

para el tratamiento de los eventos que puede generar el usuario en las diferentes pantallas.

Las capas de dominio, acceso a datos y lógica de negocio, son comunes a las dos aplicaciones del sistema: la *aplicación Web* utilizada por los profesores y el administrador y la *aplicación Portlet* utilizada por los alumnos. Sin embargo, la capa de presentación es específica a cada aplicación, dado que la interfaz de usuario es distinta tanto desde el punto de vista gráfico como desde el punto de vista del tratamiento de las peticiones enviadas por el navegador Web.

Desde un punto de vista tecnológico, cada capa se apoya en un framework específico. La capa de dominio utiliza Hibernate para realizar el mapeo o correspondencia entre entidades de la base de datos y clases del domino. La capa de acceso a datos utiliza también Hibernate para implementar sus operaciones. La capa de lógica de negocio utiliza el framework Spring para instanciar y crear los diferentes componentes. Además, Spring también se encarga de integrar esta capa con el resto.

En el caso de la aplicación Web, la capa de presentación utiliza el framework Web Tapestry. Esta herramienta simplifica el desarrollo de la interfaz Web. Sin embargo, en el caso del Portlet, al no haber aún frameworks Web lo suficientemente maduros para trabajar con esta tecnología, se utilizó directamente la tecnología JavaServer Pages (JSP), sin utilizar ningún framework Web concreto.

3.2.1 Interacción entre las capas del sistema

En esta sección se ilustra, de forma simplificada, el procesamiento de un caso de uso representativo del sistema para entender cómo trabajan las diferentes capas de forma conjunta y mostrar el orden de las invocaciones entre las clases de las capas que intervienen en la ejecución. En la figura 3.3, se muestra el procesamiento de una reserva de tutoría realizada por un alumno desde la aplicación Portlet, por medio de un diagrama de secuencia. Se ha elegido este caso de uso puesto que es uno de los fundamentales en el sistema.

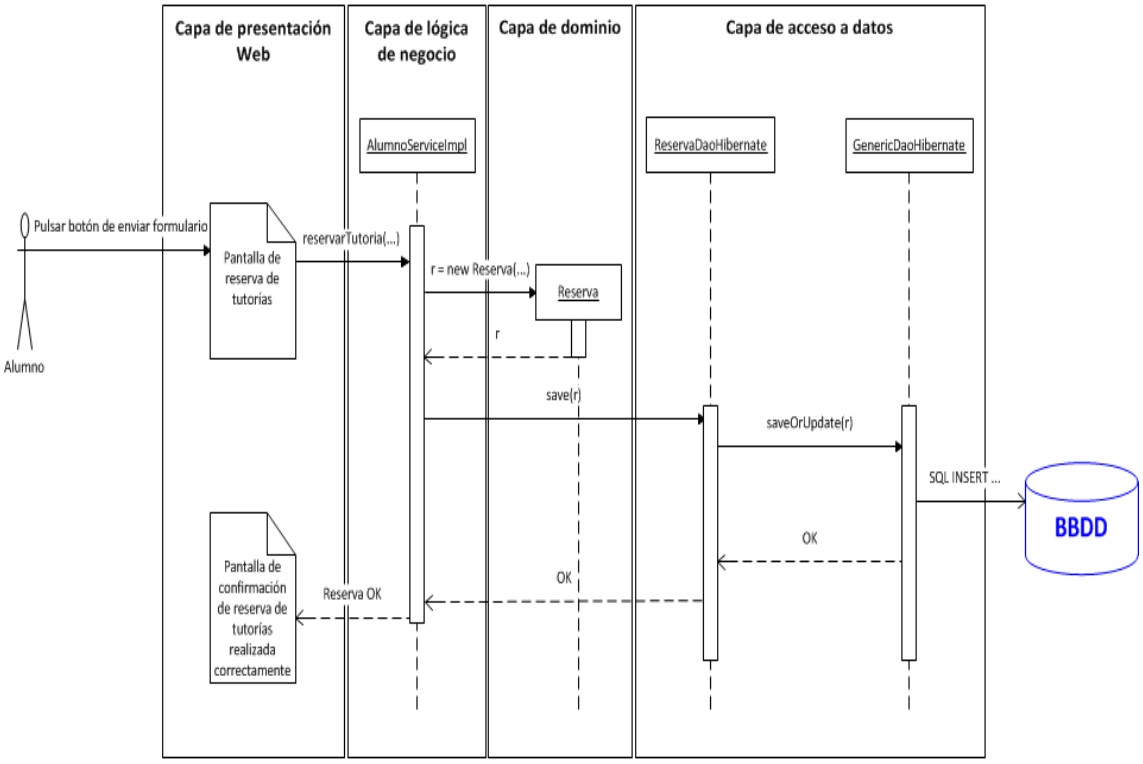


Figura 3.3. Diagrama de secuencia del caso de uso reservar tutoría

Como se puede ver, al realizar una reserva de tutoría intervienen las 4 capas del sistema. Al pulsar el botón de *Enviar* en el formulario de reserva de tutoría, desde la capa de presentación Web se invoca el método *reservarTutoria* del componente de

los alumnos de la capa de lógica de negocio. Este método, en primer lugar, crea un nuevo objeto persistente de la clase *Reserva* de la capa de dominio. Posteriormente, invoca el método *save* del objeto *ReservaDaoHibernate* de la capa de acceso a datos, que se encarga de almacenar la información de la nueva reserva en la base de datos del sistema. Finalmente, cuando las invocaciones terminan de forma correcta, la capa de presentación se encarga de mostrar al usuario la pantalla de confirmación de la reserva.

Aunque en el diagrama no se ilustra, este caso de uso se realiza dentro de una transacción. El framework Spring es el que se encarga de gestionar las transacciones desde la capa de lógica de negocio, utilizando para ello la técnica de programación orientada a aspectos (AOP) [69]. El programador utiliza anotaciones en el código para declarar los métodos transaccionales, y Spring se encarga de interceptar la invocación a estos métodos. Antes de que se ejecute el método, Spring realiza operaciones para iniciar la transacción, y al finalizar su ejecución realiza las correspondientes para el cierre de la transacción. Por tanto, no es necesario tener código relacionado con transacciones en la capa de acceso a datos, ya que esto se realiza en un nivel superior, a nivel de la capa de lógica de negocio.

3.3 Capa de acceso a datos

El objetivo de esta capa es gestionar las operaciones de tratamiento de los datos del sistema almacenados en la base de datos. Esta capa sólo tiene conocimiento de la existencia de la base de datos del sistema y de la capa de dominio. A su vez, ofrece sus operaciones a la capa de lógica de negocio. Para cada clase persistente de la capa de dominio, se crea en esta capa una abstracción que permita gestionar su persistencia. Cada una de estas abstracciones se denomina DAO. Los diferentes componentes, tecnologías de la capa de acceso a datos y la base de datos relacional utilizada se muestran en la figura 3.4. Se han utilizado puntos suspensivos en el diagrama para ilustrar que hay más componentes de los representados. La relación concreta entre los componentes se explica en la sección 3.3.2.

El diseño realizado en esta capa parte de tres objetivos fundamentales:

1. Se debe ocultar al resto del sistema la tecnología de almacenamiento de datos utilizada (ficheros, base de datos relacional, base de datos orientada a objetos, directorio, etc).

2. Se debe ocultar al resto del sistema la tecnología de base de datos concreta utilizada (MySQL, Oracle, Derby, PostgreSQL, etc).

3. Se debe ocultar al resto del sistema la tecnología utilizada para acceder a la base de datos desde el sistema (JDBC, Hibernate, etc).

Estos objetivos facilitan cambios en el tipo de repositorio de datos utilizado, en la tecnología de base de datos concreta y en la tecnología utilizada para acceder a la base de datos, consiguiendo un sistema final mucho más mantenible. Para alcanzar estos objetivos, se ha aplicado el patrón de diseño *Data Access Object* (DAO) [3]. Este patrón se utiliza de forma recurrente en las aplicaciones Web de gestión. Se encarga de separar la lógica de acceso a datos de las demás partes del sistema (lógica de negocio y presentación). Para ello, abstrae y encapsula el acceso a la fuente de datos de un sistema de información. Por tanto, permite modificar fácilmente el mecanismo de persistencia que utiliza la aplicación. Por ejemplo, facilita el cambio desde un mecanismo de persistencia basado en ficheros (de texto, XML, etc.), a otro basado en una base de datos relacional.

Dado que el sistema usa una base de datos relacional, esta capa también utiliza Hibernate como mapeador objeto-relacional (*Object Relational Mapping*, ORM) para convertir entre los datos almacenados en la base de datos relacional y los datos almacenados en las clases Java de la capa de dominio. Esto permite utilizar objetos persistentes en lugar de manipular directamente las entidades de la base de datos. También simplifica la programación de la capa de persistencia de las aplicaciones, dado que la utilización directa de JDBC (*Java Database Connectivity*) [79] resulta tediosa para el programador ya que el código que hay que escribir es muy repetitivo.

Se ha configurado Hibernate para que utilice la API de persistencia de la plataforma Java EE denominada *Java Persistence API* (JPA) [18]. De esta forma, tampoco se depende de la herramienta Hibernate si no que se utiliza el estándar de Java para persistencia.

En la siguiente sección se describe el modelo de datos del sistema que representa la información almacenada en la base de datos relacional utilizada.

3.3.1 Modelo de datos del sistema

El modelo de datos surge a partir del análisis de la información persistente necesaria para el funcionamiento del sistema. Se puede observar el esquema entidad-relación de la base de datos del sistema en la figura 3.5. En este esquema no se han representado los atributos de las entidades ni de las interrelaciones para facilitar su lectura.

En primer lugar, se ha modelado el concepto de horario de tutorías. Para ello, se utilizan las entidades *Horario*, *Franja Horaria* y la entidad agregada *Tutoría*. La entidad *Horario* consiste en una o varias franjas horarias. Cada franja horaria guarda la información de un día determinado de la semana (lunes-domingo), una hora de inicio y una hora de fin. Por ejemplo, una franja horaria sería: *lunes de 10:30 a 12:30*. Se ha utilizado una entidad agregada, *Tutoría* para representar la información completa de un horario de tutorías. Esta entidad agregada representa, conceptualmente, la nota que habitualmente coloca un profesor en la puerta de su despacho con sus horarios semanales de tutorías.

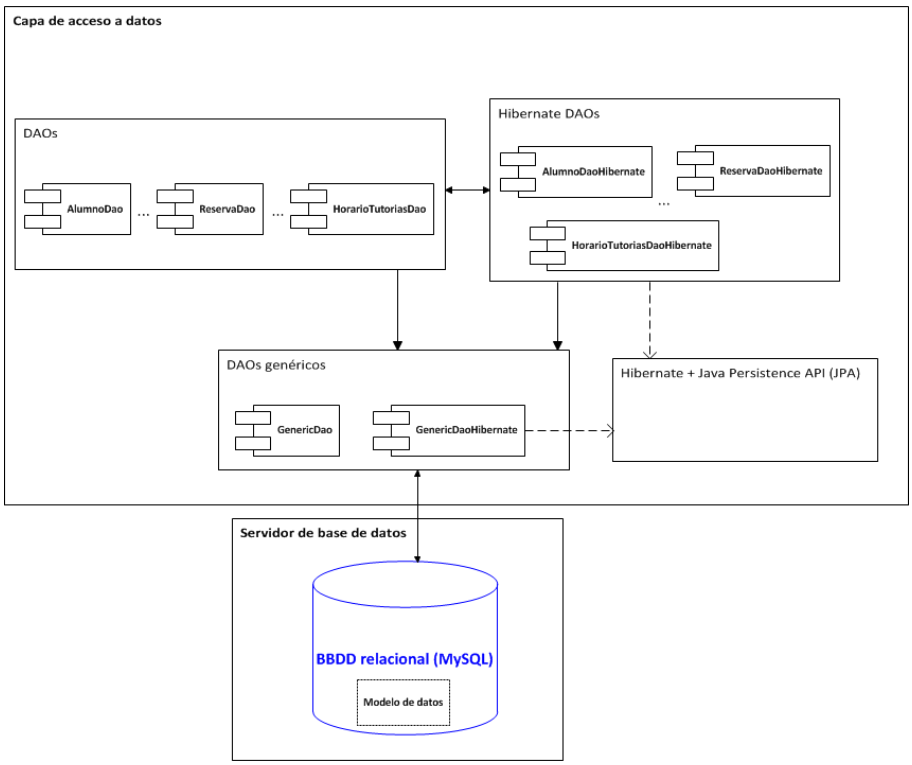


Figura 3.4. Diagrama de la capa de acceso a datos

Con el fin de posibilitar la reserva de tutorías para un día y hora concretos, un alumno debe poder tener una visión del calendario de sesiones de tutorías. Este calendario se almacena en la entidad *Sesión de Tutorías*. Cada sesión de tutorías surge a partir de una determinada franja horaria de un horario de tutorías y del tiempo asignado a cada alumno. El tiempo asignado a cada alumno se representa a través de un atributo de la interrelación *consistir en* (no se muestra en el diagrama). De esta forma, el sistema permite que el profesor asigne un tiempo por alumno para cada franja horaria de sus horarios de tutorías. Un ejemplo de sesión de tutorías asociada a la franja horaria de ejemplo anterior y considerando un tiempo asignado a cada alumno de 30 minutos sería: *lunes 7 de marzo de 2011 de 10:30 a 11:00*.

Por cuestiones de eficiencia, se decidió almacenar el calendario de sesiones de tutorías ya pre-calculado en lugar de tener que calcularlo dinámicamente en tiempo de ejecución cada vez que un alumno quiera solicitar una tutoría. Es decir, se prefiere utilizar más espacio de almacenamiento y disminuir el tiempo de ejecución del sistema con el fin de mejorar la interactividad y experiencia del usuario.

También se han modelado las asignaturas y grupos de docencia. Cada asignatura puede tener uno o varios grupos de docencia. A su vez, cada grupo de docencia puede estar asociado a una o varias asignaturas. Además, cada asignatura forma

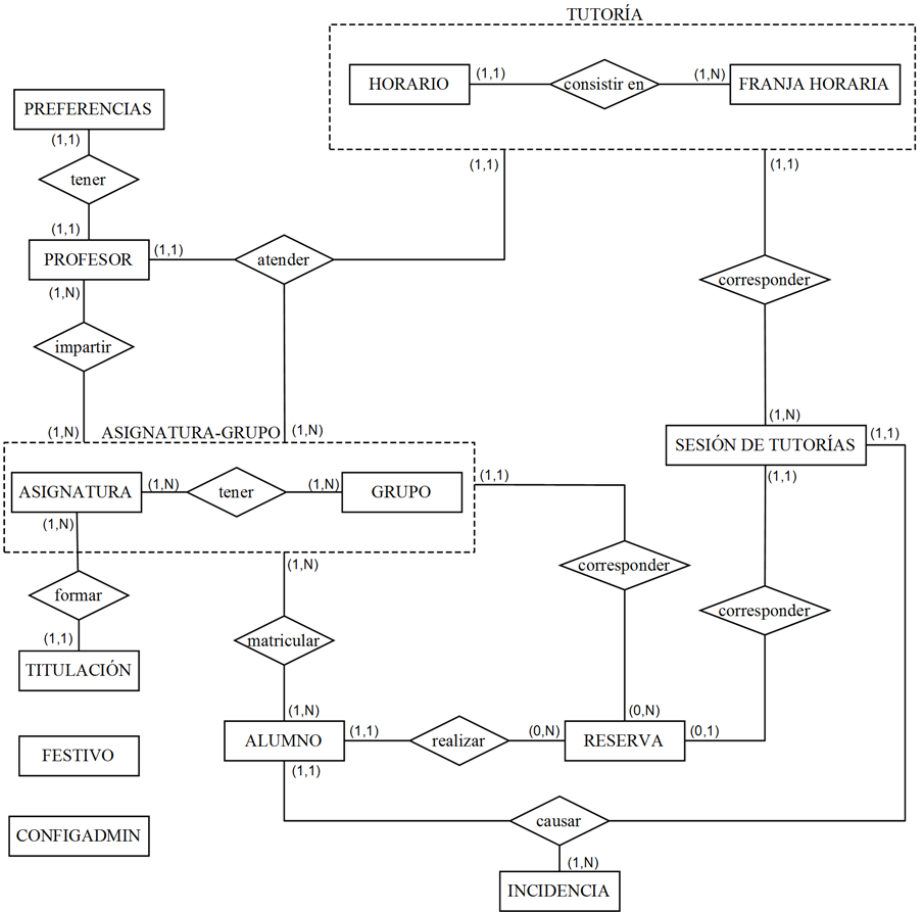


Figura 3.5. Diagrama entidad-relación de la base de datos del sistema

parte de una única titulación. Dado que un profesor imparte uno o varios grupos de una determinada asignatura, se ha utilizado una entidad agregada, denominada *Asignatura-Grupo*, para representar las asignaturas y los grupos de docencia. Cada profesor también tiene unas preferencias de configuración en el sistema.

Tradicionalmente, cada profesor tiene un único horario de tutorías semanal en el que atiende dudas y consultas de los alumnos relativas a todas sus asignaturas. El sistema es más flexible dado que ofrece la posibilidad de tener horarios de tutorías diferentes para cada asignatura individual o para conjuntos de las asignaturas que imparte (por ejemplo, un horario para las asignaturas del primer cuatrimestre y otro para las del segundo). Por supuesto, también permite conservar la visión tradicional y utilizar el mismo horario de tutorías para todas las asignaturas que imparte. Esto es posible gracias a la interrelación ternaria *atender* entre las entidades *Profesor*, *Asignatura-Grupo* y *Tutoría*.

Un alumno puede estar matriculado en varias *Asignaturas-Grupos* y puede

realizar el número de reservas de tutoría que desee. Cada reserva se corresponde con una determinada sesión de tutorías y también con una *Asignatura-Grupo*. Además, un alumno sólo puede realizar reservas de *Asignaturas-Grupos* en los que se encuentre matriculado durante el año académico. También se decidió almacenar información de las modificaciones o cancelaciones de reservas de tutorías realizadas por los alumnos. Se utiliza para ello la entidad *Incidencia*. Una incidencia la causa un alumno y está asociada una sesión de tutorías, aquella que tenía previamente reservada y que ha cancelado o modificado.

Por último, también se muestran las entidades *ConfigAdmin*, donde se almacenan los datos del administrador del sistema y *Festivo*, donde se almacenan las fechas festivas del curso académico actual introducidas por el administrador.

3.3.2 Diseño de un DAO

Un DAO, a nivel intuitivo, es un objeto que se encarga de acceder a un repositorio de información, ya sea un fichero, una base de datos relacional, etc.

A nivel técnico, un DAO define una interfaz para las operaciones de persistencia (métodos CRUD, *Create Read Update Delete*, y de búsqueda) de una clase del dominio o persistente concreta.

La aplicación concreta del patrón utilizada en el proyecto es posible gracias al uso de clases Java genéricas. Está basada en las explicaciones encontradas en [6] y [9].

En la figura 3.6 se puede ver el ejemplo del diseño de un DAO. Se ha tomado como ejemplo el *AlumnoDao*, utilizado para gestionar la clase persistente *Alumno*. Esta clase contendrá los datos de la entidad *Alumno* de la base de datos. El diseño del resto de DAOs se ha realizado de forma análoga.

La aplicación del patrón DAO realizada utiliza una interfaz y una clase Java genéricas. En primer lugar, la interfaz genérica, *GenericDao*, recibe 2 argumentos:

- E, la clase persistente para la que se implementará el DAO.
- PK, define el tipo de identificador (clave primaria) de la clase persistente.

Esta interfaz genérica declara las 4 operaciones de persistencia comunes a todas las entidades:

- *save*: Guarda o actualiza la entidad E recibida como argumento en el repositorio de datos utilizado.
- *find*: Si existe, devuelve la entidad cuya clave primaria es la recibida como argumento (PK). Si no existe, lanzará una excepción de tipo *InstanceNotFoundException*.

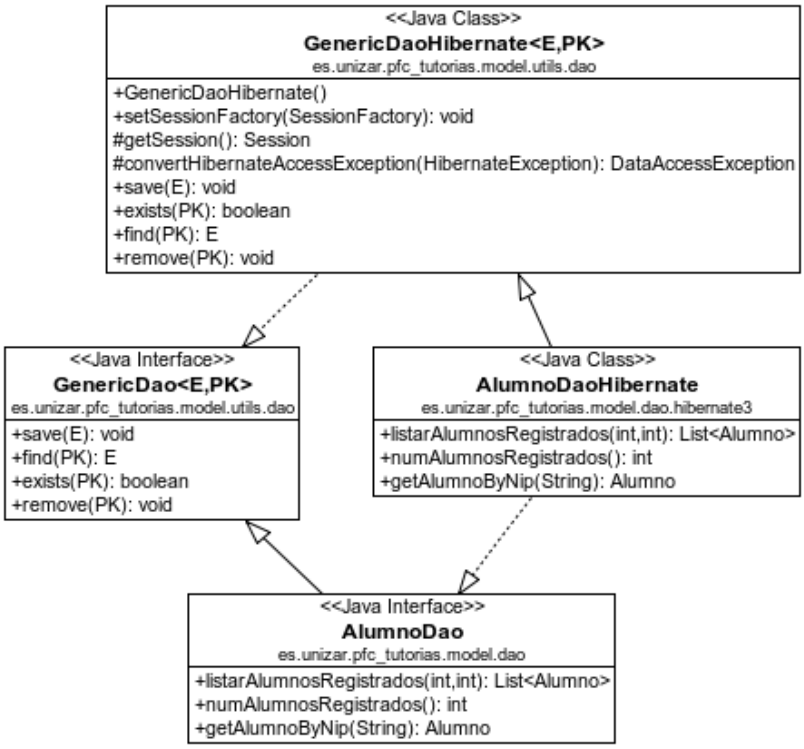


Figura 3.6. Diagrama de clases de un DAO

- *exists*: Devuelve cierto, si existe una entidad cuya clave primaria es la recibida como argumento (PK) y falso, en caso contrario.
- *remove*: Elimina, si existe, la entidad cuya clave primaria es la recibida como argumento (PK). Si no existe, lanzará una excepción de tipo *InstanceNotFoundException*.

Con estas 4 operaciones anteriores declaradas en la interfaz genérica *GenericDao* es posible crear, leer, actualizar y borrar (CRUD) cualquier dato de tipo genérico en la base de datos.

En segundo lugar, la clase genérica *GenericDaoHibernate* implementa la interfaz *GenericDao* y por tanto, también tiene los 2 argumentos mencionados anteriormente. Esta clase implementa los métodos *save*, *find*, *exists* y *remove* utilizando para ello los métodos de la API de Hibernate.

Para cada clase persistente:

- Se utiliza una interfaz (*AlumnoDao* en el ejemplo) que hereda del DAO genérico (*GenericDao*) proporcionando como argumentos el tipo de la clase

persistente (*Alumno* en el ejemplo aunque no se muestra en el diagrama) y el tipo de la clave primaria de la clase persistente (*Long* en el ejemplo, tampoco se muestra en el diagrama). Esta interfaz también define métodos adicionales de búsqueda (*listarAlumnosRegistrados*, *numAlumnosRegistrados*, *getAlumnoByNip*) específicos de la clase persistente.

- Se utiliza una clase (*AlumnoDaoHibernate* en el ejemplo) que implementa la interfaz anterior (*AlumnoDao* en el ejemplo) y hereda de la clase genérica (*GenericDaoHibernate*).

Con este diseño se consigue no tener que replicar las operaciones básicas de acceso a datos en cada clase persistente al estar definidas en la interfaz genérica *GenericDao*. También es posible cambiar de mapeador objeto-relacional y dejar de usar Hibernate. Para ello, bastaría con crear una clase genérica que implementara la interfaz *GenericDao* utilizando los métodos del nuevo mapeador. Después, se proporcionaría para cada entidad una clase que herede de dicha clase genérica. De esta forma, el diseño tampoco depende del mapeador objeto-relacional concreto utilizado.

3.4 Capa de dominio

En esta capa se encuentran las denominadas clases persistentes o del dominio. Estas clases se corresponden con entidades del modelo de datos existente en la base de datos del sistema. Cada clase se corresponde con una entidad o interrelación del modelo de datos. Cada objeto o instancia de una determinada clase se corresponde con una tupla de la entidad correspondiente. La función de esta capa es, por tanto, ofrecer una visión del modelo de datos utilizando para ello objetos de un lenguaje de programación orientado a objetos (en este caso Java).

Los objetos de esta capa pueden ser requeridos desde cualquiera de las otras 3 capas. La capa de acceso a datos necesita estos objetos para almacenarlos en la base de datos. La capa de lógica de negocio utiliza estos objetos para implementar la funcionalidad del sistema. Por ejemplo, cuando un alumno solicita una cita de tutoría, se requiere crear un nuevo objeto de la clase *Reserva* utilizando los datos introducidos por el alumno en la pantalla correspondiente. La capa de presentación Web puede también utilizar las clases del dominio a la hora de implementar las acciones que se ejecutan en respuesta a eventos generados por la interfaz de usuario. Por ejemplo, cuando se pulsa el botón de enviar formulario en la pantalla de registro de un profesor, el método que se ejecuta en respuesta a este evento crea ya el objeto *Profesor* con los datos introducidos en el formulario.

A nivel tecnológico, estas clases han sido generadas automáticamente utilizando la herramienta Hibernate Tools. Con ella, a partir de una base de datos preexistente y del fichero de configuración de Hibernate, se pueden obtener directamente estas

clases con sus atributos, métodos de acceso a los atributos y las anotaciones de Hibernate insertadas en el código de las clases que especifican la correspondencia con las entidades de la base de datos. Sin embargo, al tratarse de una herramienta genérica de generación de código, el código fuente generado junto con las anotaciones no siempre se adapta a las necesidades específicas del sistema. Por ejemplo, Hibernate Tools considera por defecto todas las relaciones como bidireccionales. Por ello, hay que realizar modificaciones en las anotaciones y añadir nuevas anotaciones con el fin de adaptar las clases a nuestro modelo de dominio.

3.4.1 Modelo de dominio

En esta sección se muestran las clases del problema y las asociaciones que existen entre ellas. Se puede observar el diagrama de clases del dominio en la figura 3.7. Como se puede apreciar, el modelo de dominio es muy similar al modelo de datos del sistema, dado que se ha utilizado un mapeador objeto relacional. Una de las diferencias principales radica en que los objetos pueden contener referencias a otros objetos o incluso listas o conjuntos de otros objetos derivados de las asociaciones representadas. Sin embargo, el modelo relacional y la teoría de normalización de bases de datos impide que haya listas de datos en las entidades de la base de datos. Por ejemplo, se puede observar como un horario contiene una o varias franjas horarias. En la sección C.4.2 del anexo de diseño se explican una serie de patrones objeto-relacional que se utilizan para realizar la correspondencia.

3.5 Capa de lógica de negocio

El propósito de esta capa es integrar los componentes que implementan la funcionalidad del sistema. Para ello, se ha agrupado la funcionalidad del sistema de forma lógica. Es decir, funcionalidades relacionadas forman parte del mismo componente. Estos componentes implementan la lógica principal de la aplicación. Para implementar las operaciones necesarias, esta capa utiliza las operaciones ofrecidas por las interfaces de la capa de acceso a datos y las clases de la capa de dominio. También los componentes de esta capa pueden utilizar operaciones ofrecidas por otros componentes de la propia capa. Esta capa ofrece sus operaciones a la capa de presentación.

A nivel tecnológico, la implementación de esta capa se apoya en el contenedor de inversión de control (*IoC container*) de Spring para gestionar los diferentes componentes. Como consecuencia, las clases (*beans* en la terminología de Spring) son instanciadas e inicializadas por Spring. Además, las dependencias de cada clase (otras clases que utiliza) son inyectadas por Spring en tiempo de ejecución.

Spring también se encarga de gestionar aspectos transversales del sistema (*cross-cutting concerns*) [68], es decir, aquellos que afectan a todo el sistema. Algunos

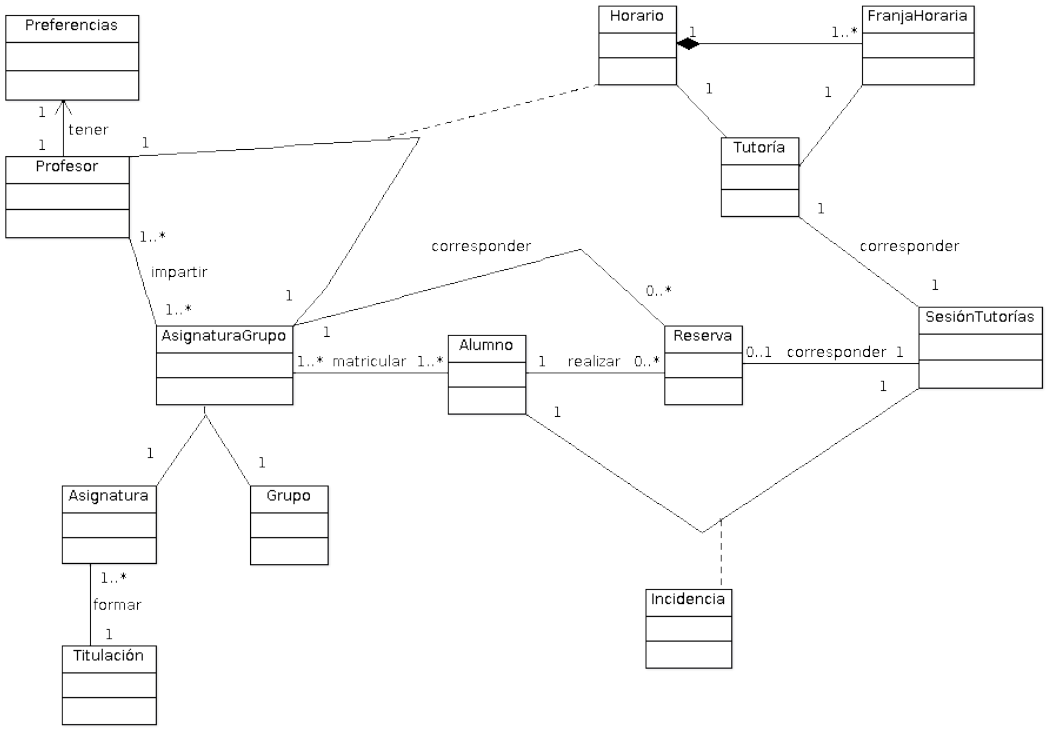


Figura 3.7. Diagrama de clases de la capa de dominio

ejemplos de estos aspectos son el *logging*, la gestión de excepciones y la gestión de transacciones. Para ello, Spring utiliza la programación orientada a aspectos [69]. Esta técnica de programación permite aislar estos aspectos y aplicarlos a los componentes que los precisen utilizando anotaciones en el código. De esta forma, se evita la pérdida modularidad que se produce al tener que repetir código en diferentes capas del sistema para gestionar adecuadamente estos aspectos.

3.5.1 Componentes de alto nivel del sistema

En esta sección se presentan los principales componentes de la capa de lógica de negocio. Estos componentes se clasifican en: *componentes funcionales* y *componentes de soporte*. Los componentes funcionales se encargan de implementar la funcionalidad principal del sistema. Los componentes de soporte se ocupan de

funciones auxiliares o de utilidad genérica en el sistema y son utilizados por los componentes funcionales para llevar a cabo sus tareas.

El diagrama de componentes del sistema se puede observar en la figura 3.8. Todos los componentes que se muestran han sido desarrollados en este proyecto a excepción de las librerías JavaMail y el planificador Quartz. Ambas librerías son utilizadas por algunos componentes, tal y como se representa a través de flechas punteadas. Para representar el resto de las relaciones entre los componentes, se ha utilizado la idea de *bus*, representado a través de la flecha horizontal gruesa. De esta forma, se evita que el diagrama esté plagado de líneas que se cruzan facilitando así su lectura. Además, dado que todos los componentes están desplegados en el mismo servidor Web, no se muestra su disposición física en nodos.

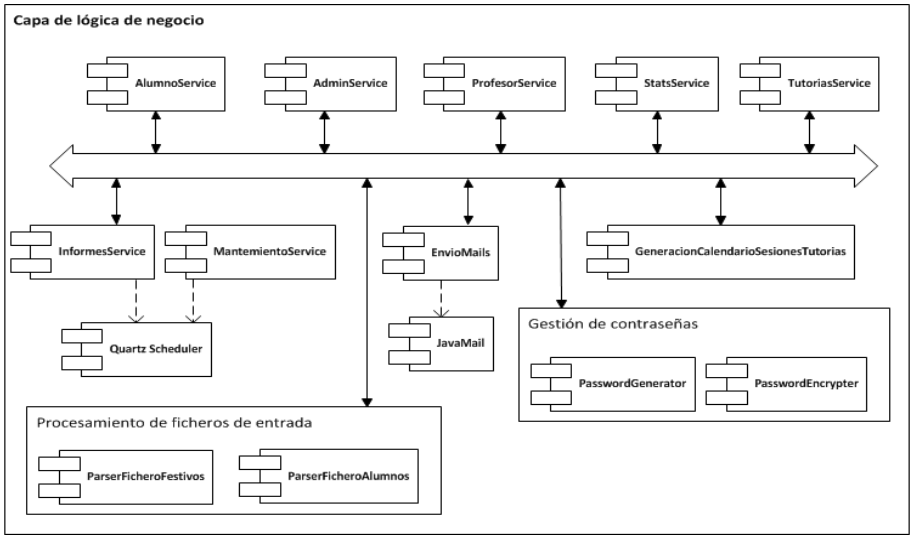


Figura 3.8. Diagrama de componentes del sistema

A continuación se describen los componentes que se muestran en el diagrama de componentes.

1. Funcionales:

   - Componentes de configuración y administración: *AdminService* para la administración del sistema y *TutoriasService* para la configuración de tutorías.
   - Componentes estadísticos: hay un único componente estadístico en el sistema, *StatsService*. Este componente se encarga de recopilar estadísticas generales de utilización.
   - Componente del profesor (*ProfesorService*): se encarga de implementar la funcionalidad del profesor. Por ejemplo, el registro, alta de asignaturas, gestión del perfil, etc.

- Componente del alumno (*AlumnoService*): se encarga de implementar la funcionalidad del alumno. Por ejemplo: reservar tutoría, consultar tutorías reservadas, modificar o cancelar reservas previas, etc.

2. De soporte:

   - Componente de mantenimiento (*MantenimientoService*). Se encarga de realizar periódicamente las copias de seguridad de los datos del sistema, de las operaciones de cambio de curso académico, etc. Este componente utiliza la librería del planificador Quartz.
   - Componente de informes (*InformesService*). Se encarga de generar los informes diarios y semanales de solicitudes de tutorías, utilizando Quartz para planificar las tareas que se han de ejecutar en segundo plano.
   - Componente para el envío de correos electrónicos (*EnvioMails*). Se utiliza para enviar los diferentes informes y notificaciones a los usuarios del sistema. Este componente se apoya en la librería JavaMail.
   - Componentes de generación y cifrado de contraseñas (*PasswordGenerator* y *PasswordEncrypter*). Se encargan de generar las contraseñas de los alumnos y cifrar las de los profesores y alumnos.
   - Analizadores sintácticos para procesar los ficheros de alumnos matriculados y días festivos (*ParserFicherosAlumnos* y *ParserFicheroFestivos*).
   - Componente de generación del calendario de sesiones de tutorías (*GeneracionCalendarioSesionesTutorias*). Se encarga de calcular las sesiones de tutorías de un determinado horario.

Algunos de estos componentes son utilizados exclusivamente por la aplicación Web (por ejemplo, el componente *ProfesorService*) o por la aplicación Portlet (por ejemplo, el componente *AlumnoService*) mientras que otros son utilizados por ambas aplicaciones (por ejemplo, el componente *EnvioMails*).

3.5.2 Diseño de un componente

En esta sección se explica el diseño de un componente de la capa de lógica de negocio y se ilustra por medio de un ejemplo. El diseño de todos los componentes del sistema sigue la misma estrategia que la explicada a través del ejemplo.

Los componentes de la capa de lógica de negocio están diseñados utilizando el mecanismo de abstracción de interfaces de Java y la técnica de inyección de dependencias [10] [11] soportada por el framework Spring gracias a su contenedor de inversión de control. La técnica de inyección de dependencias es una implementación sofisticada del patrón Factoría [25]. Se basa en la idea de que los módulos o componentes de alto nivel de un sistema no deben depender de componentes de bajo

nivel. En su lugar, ambos deben depender de abstracciones. Esta técnica se utiliza para indicar a una parte de un programa (un componente, un módulo, etc.) qué otras partes puede utilizar, proporcionándole para ello una dependencia externa (a través de una referencia). Esta técnica también se conoce con el nombre de inversión de control ya que, tradicionalmente, cada objeto es responsable de obtener sus propias referencias a los objetos con los que colabora. Sin embargo, con esta técnica, se invierte el control, ya que el objeto se limita a declarar los objetos que utiliza (sus dependencias) y un contenedor de inversión de control (el de Spring en este caso) inyecta las dependencias en tiempo de ejecución.

Ambas estrategias de diseño benefician el mantenimiento del sistema ya que reducen el código repetitivo de instanciación e inicialización de los objetos y facilitan la sustitución de implementaciones concretas y la realización de pruebas. Para ello, se pueden crear nuevas clases que implementen la interfaz, o incluso clases *mock* o *stubs*, cuya implementación de los métodos de la interfaz sea vacía y que son utilizadas para realizar pruebas, sustituir implementaciones en tiempo de ejecución, etc.

Por tanto, para el diseño de cada componente, se crea en primer lugar una interfaz Java donde se definen los métodos que ofrece. Posteriormente, se crea una clase que implementa dicha interfaz. Cada componente utilizará operaciones proporcionadas por los DAO de la capa de acceso a datos para almacenar datos en la base de datos. Los componentes también pueden utilizar operaciones proporcionadas por otros componentes de la capa de lógica de negocio, ya sean componentes creados específicamente para el proyecto o librerías genéricas empleadas (como Quartz, JavaMail, etc).

Cada componente declara, utilizando anotaciones en el código de Spring, los componentes que requiere para implementar sus operaciones. Spring se encarga, en tiempo de ejecución, de instanciar los componentes necesarios y proporcionar las dependencias a los componentes que lo requieran. De esta forma, se reduce el acoplamiento entre los objetos del sistema. Los objetos conocen sus dependencias a través de interfaces con lo que es posible cambiar la implementación de estas dependencias de forma transparente para el objeto que las contiene.

En la figura 3.9 se puede ver un diagrama de clases simplificado del diseño de un componente, en este caso el componente *AlumnoService* utilizado para implementar la funcionalidad de los alumnos. Como se puede observar, hay una interfaz Java, *AlumnoService*, donde se declaran los métodos del componente. Esta interfaz es implementada por la clase *AlumnoServiceImpl*, que provee de una implementación concreta a los métodos declarados en *AlumnoService*. Para proporcionar la implementación de los diferentes métodos, *AlumnoServiceImpl* utiliza métodos o servicios ofrecidos por otros componentes que están declarados como atributos (no se muestran en el diagrama por motivos de espacio). Se han representado sólo dos de los componentes de los que depende para facilitar la interpretación del diagrama. Estos dos componentes son: *ProfesorService*, componente de la capa de lógica de negocio

que implementa la funcionalidad de los profesores y *AlumnoDao*, componente de la capa de acceso a datos utilizado para acceder a la entidad *Alumno* de la base de datos. Como se puede apreciar, la clase *AlumnoServiceImpl* depende de los otros componentes a través de abstracciones (interfaces), no a través de implementaciones concretas. Se puede ver cómo depende del componente *ProfesorService* y que este componente tiene una clase, *ProfesorServiceImpl*, que implementa sus métodos. También se puede observar la dependencia con respecto al componente *AlumnoDao*, que es implementado por la clase *AlumnoDaoHibernate*. De esta forma, se reduce el acoplamiento de la aplicación, consiguiendo que esté formada por componentes poco acoplados entre sí.

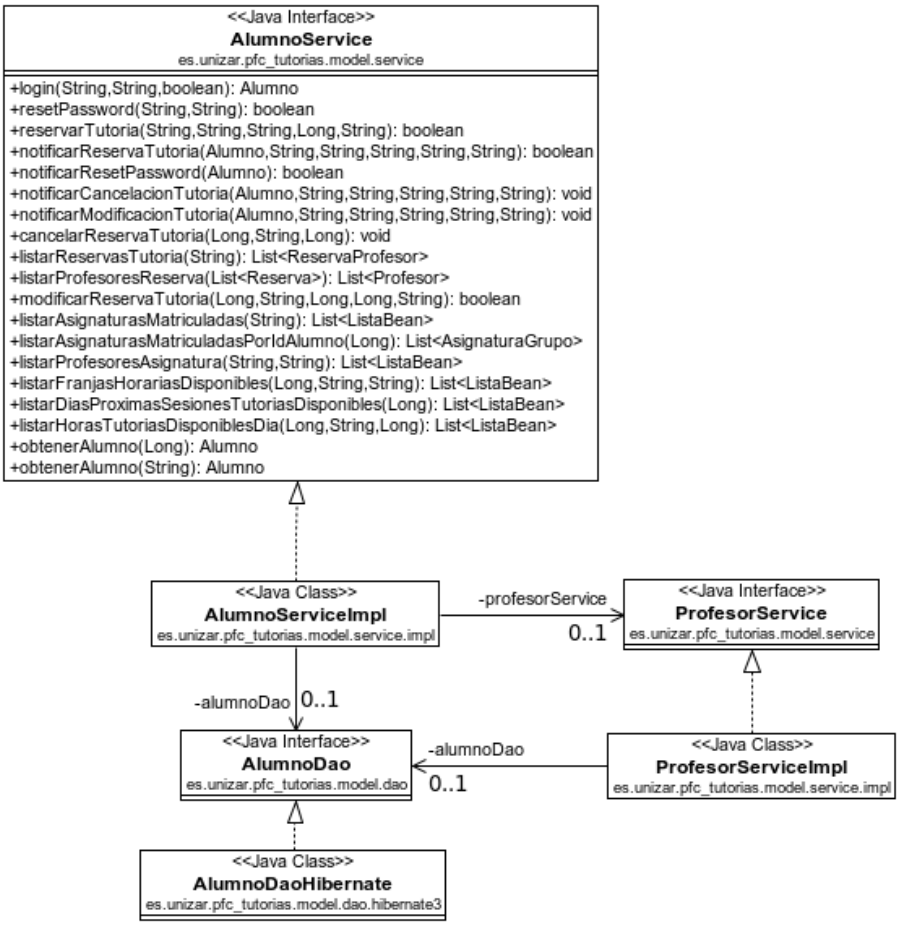


Figura 3.9. Diagrama de clases de un componente

3.6 Capa de presentación

El objetivo de esta capa es la implementación de la interfaz de usuario. Los usuarios del sistema interactúan con el mismo a través de esta capa, que es la encargada de invocar la funcionalidad de la capa de lógica de negocio.

Esta capa es diferente para la aplicación Web de los profesores (y el administrador) y la aplicación Portlet de los alumnos. Esto se debe a que el modelo de interacción de los Portlets es diferente al de las aplicaciones Web tradicionales. Los Portlets llevan a cabo una interacción en dos fases: fase de acción y fase de renderización. En el anexo A se explica la tecnología de Portlets en mayor detalle. También se debe a que la interfaz, desde el punto de vista estético, es diferente para ambas aplicaciones, dado que las opciones de menú ofrecidas varían según el tipo de usuario.

A continuación se explica el diseño de la interfaz de usuario. Posteriormente, se explican los detalles más relevantes de la implementación de esta capa.

3.6.1 Modelado de la interfaz de usuario

En esta sección se ilustra el diseño de la interfaz de usuario. El diseño consta de dos partes: los diagramas de navegación de la aplicación Web y la aplicación Portlet y los prototipos de las diferentes pantallas de ambas aplicaciones. En primer lugar, se muestran los diagramas de navegación de la aplicación Web y la aplicación Portlet. A continuación, se muestra un prototipo de ejemplo de una pantalla de la aplicación Web y otro de una pantalla de la aplicación Portlet. Finalmente, se incluye una captura de pantalla de ambas aplicaciones para ver la correspondencia con los prototipos realizados.

Para modelar la interfaz de usuario, se realizaron en primer lugar diagramas de navegación. El diagrama de navegación para la aplicación Portlet se puede observar en la figura 3.10. El correspondiente a la aplicación Web (simplificado) se puede ver en la figura 3.11. En ambos diagramas se representa cada pantalla como un recuadro y las flechas que salen de cada pantalla indican a qué otras pantallas se puede navegar.

Después, se realizaron prototipos de las diferentes páginas Web del sistema. Este proceso se realizó a mano mediante bocetos en papel. Los bocetos se utilizaron en primer lugar para comentar en las reuniones con el director del proyecto diferentes aspectos de la interfaz como la estructura, organización de los diferentes elementos, pantallas reutilizables, etc. Posteriormente, estos prototipos sirvieron de base a la hora de diseñar las plantillas y escribir el código HTML de las páginas JSP. El prototipo de la pantalla de menú principal (del profesor) de la aplicación Web se puede observar en la figura 3.12. El prototipo de la pantalla de reserva de tutoría de la aplicación Portlet se puede ver en la figura 3.13.

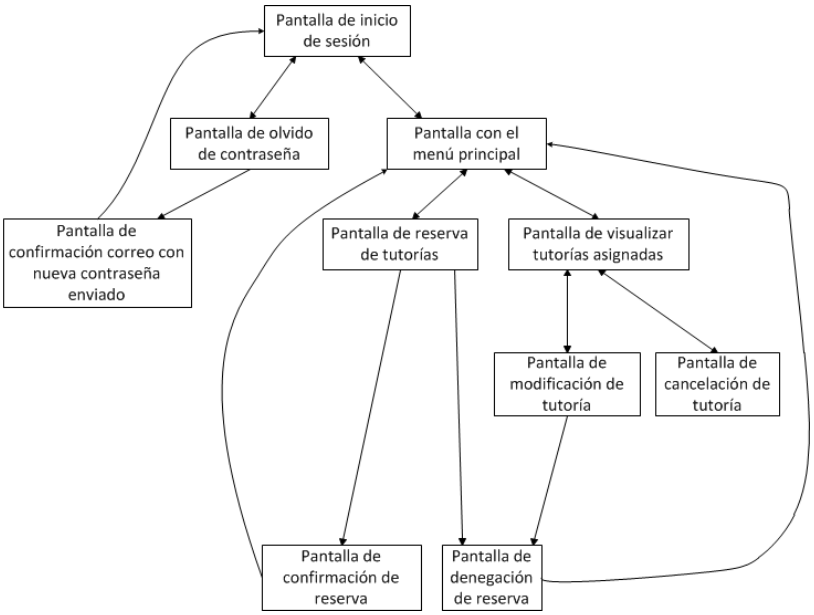


Figura 3.10. Diagrama de navegación de la aplicación Portlet

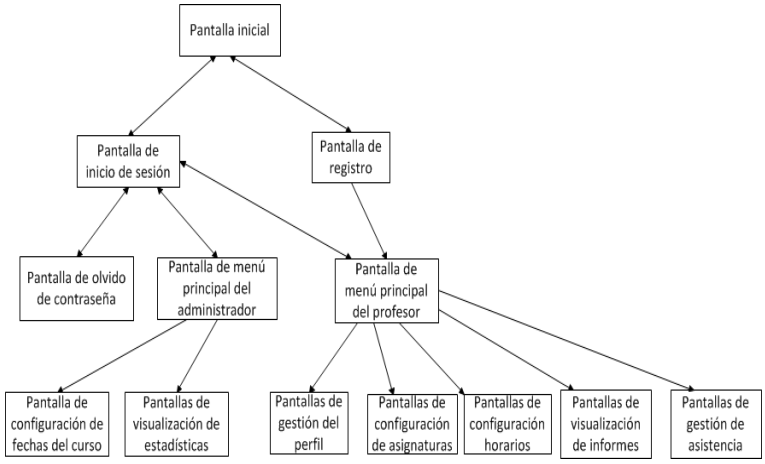


Figura 3.11. Diagrama de navegación de la aplicación Web

Por último, se incluyen 2 capturas de pantalla de las aplicaciones que forman el sistema. En la figura 3.14 se muestra una captura de pantalla de la aplicación Web. En la figura 3.15 se muestra una captura de pantalla de la aplicación Portlet. Se puede ver la correspondencia entre estas capturas de pantalla y los prototipos de las figuras 3.12 y 3.13.

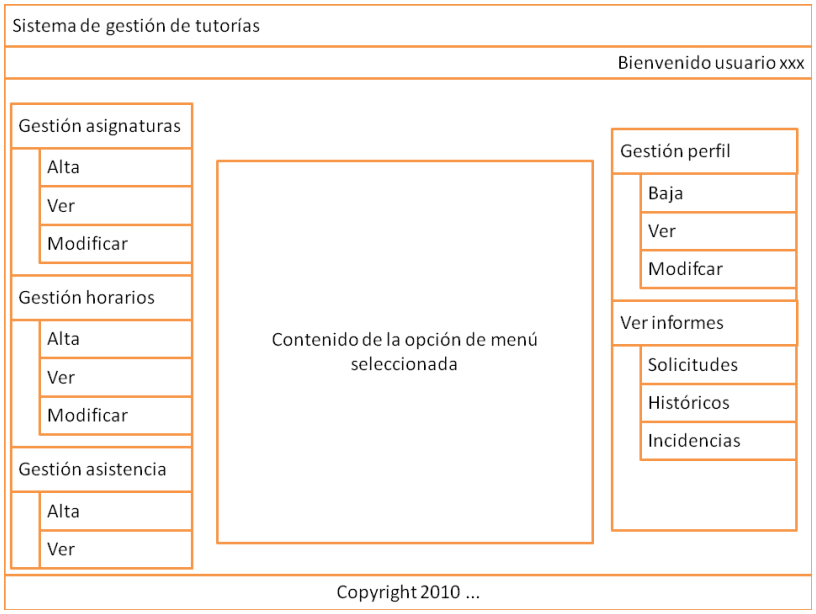


Figura 3.12. Prototipo de la pantalla con el menú principal del profesor

Figura 3.13. Prototipo de la pantalla de reserva de tutorías

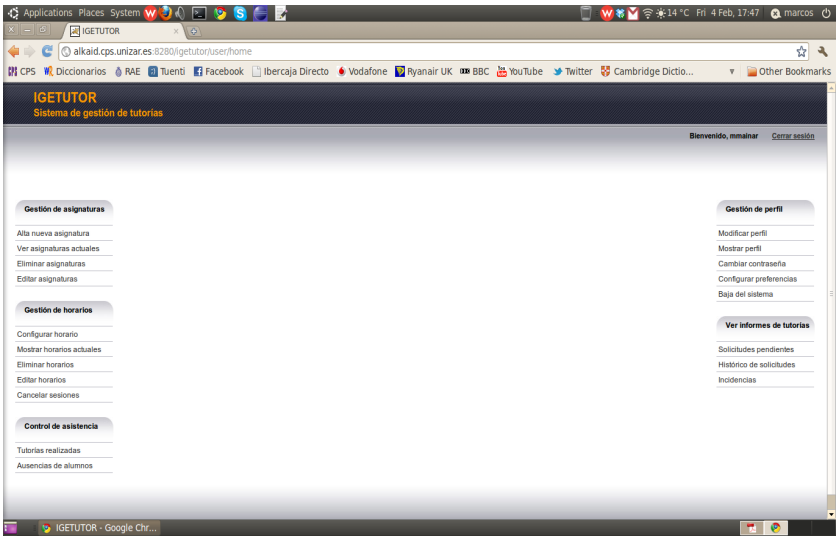


Figura 3.14. Pantalla con el menú principal del profesor

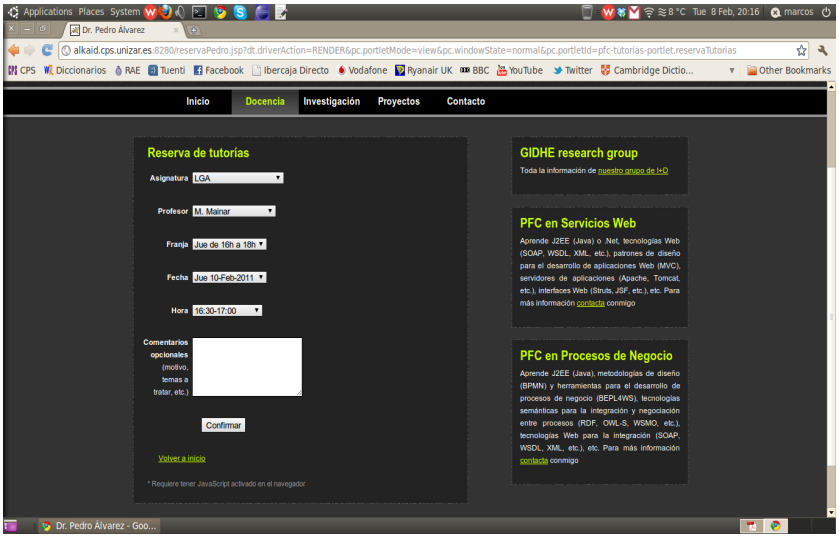


Figura 3.15. Pantalla de reserva de tutorías

3.6.2 Implementación de la capa de presentación

La capa de presentación de la aplicación Web integra las páginas Web accesibles por los profesores y el administrador y su lógica correspondiente. Para ello, cada página tiene asociada una plantilla y una clase Java. La clase Java actúa como controladora, es decir, se encarga de recibir los eventos relativos a la página (envío de un formulario, *click* en un enlace, etc). En la sección C.4.1 se explica cómo encaja el diseño del sistema con el patrón de diseño *Model View Controller* (MVC) [2]. La plantilla se encarga de generar el código de *markup* (HTML) de la página, que define su estructura y contenido. Esta capa también utiliza un plantilla global donde se especifica la estructura general de todas las pantallas del profesor. Es decir, define las opciones de menú que se muestran en todas las pantallas una vez que el profesor ha iniciado sesión correctamente. Del mismo modo, existe otra plantilla global donde se especifica la estructura general de todas las pantallas del administrador, es decir, sus opciones de menú una vez ha iniciado sesión como administrador. Por último, se emplea una hoja de estilo en cascada (*Cascading Style Sheet*, CSS) para especificar la presentación de las pantallas. En esta hoja se definen aspectos visuales como los colores, tipos de letra, márgenes, etc. Se utilizó una hoja de estilo en cascada genérica disponible públicamente en la Web [46] y se modificó para adaptarla a las necesidades concretas de la aplicación.

Análogamente, en el caso de la aplicación Portlet, la capa de presentación integra las páginas Web accesibles por los alumnos y su lógica. En este caso, se utiliza una página JSP por cada página Web. Las páginas JSP permiten crear contenido Web dinámico combinando código HTML, CSS, Java, JavaScript, etc. La lógica de la capa de presentación de esta aplicación se define en una clase Java que utiliza la API de Portlets. En esta clase se especifican las peticiones o eventos que puede recibir el Portlet y se definen las acciones que se ejecutan en respuesta a estos eventos, utilizando para ello los métodos de los componentes de la capa de lógica de negocio. Para las pantallas de la aplicación Portlet, no se utiliza ninguna hoja de estilo en cascada. En su lugar, se pretende que las pantallas del Portlet utilicen los estilos definidos en la hoja de estilos del portal Web en el que se integre. También, al realizar la integración, el usuario puede definir nuevos estilos para los diferentes elementos del Portlet (título y contenido) si lo considera conveniente. Se explica cómo realizar esto en el manual de instalación y mantenimiento del sistema incluido en el anexo E.

Capítulo 4

Gestión del proyecto

Este capítulo tiene como objetivo describir las actividades de gestión del proyecto. En la sección 4.1 se describe la metodología utilizada en el proyecto. En la sección 4.2 se detalla la planificación del proyecto, describiendo para ello las diferentes etapas con sus objetivos y resultados y mostrando las estimaciones de duración temporal realizadas. En la sección 4.3 se presentan los datos sobre el esfuerzo real dedicado al proyecto y se realiza un análisis de las desviaciones del tiempo real con respecto a las estimaciones, reflexionando acerca de sus causas.

4.1 Metodología

En este proyecto de software se ha seguido un ciclo de vida basado en un modelo incremental [67]. Este modelo se basa en la filosofía de construir incrementando las funcionalidades del sistema.

Al final de la fase de formación y análisis de tecnologías, se empezó a realizar el análisis del sistema, especificando los requisitos funcionales y no funcionales del mismo. Después, se llevó a cabo un primer diseño del sistema. Se definió la arquitectura, el modelo de datos, los componentes y sus interfaces y se modeló la interfaz de usuario. Durante la fase de implementación, surgieron nuevos requisitos o modificaciones de los iniciales. Todo ello se incorporaba a los documentos de análisis y posteriormente se revisaba el diseño del sistema para satisfacer dichos requisitos.

Además, para la implementación, se aplicaron ideas y técnicas provenientes de metodologías de desarrollo ágil de software [62] [13] [17] en la medida de lo posible dentro del marco típico de un proyecto fin de carrera (proyectos de una única persona donde el director del proyecto se puede considerar el cliente). Estas metodologías van en una línea similar al modelo incremental. Su idea fundamental consiste en aplicar iteraciones a lo largo de todo el ciclo de vida del proyecto. Entre las metodologías ágiles más conocidas se encuentran la Programación Extrema

(*eXtreme Programming*, XP) [63] [15], Scrum [64] y el Agile Unified Process [65] (versión simplificada y ágil del Rational Unified Process [66]).

Algunas de las estrategias típicas de implementación de este tipo de metodologías y que se han aplicado en el proyecto son: simplicidad en el código, frecuente refactorización del mismo, pruebas unitarias y de integración continuas y evaluación periódica del estado real del sistema por el cliente.

El objetivo principal al introducir estas prácticas en el proyecto era que el director del mismo pudiera ir viendo la aplicación real, interactuar con ella, detectar errores, proponer cambios, modificaciones, mejoras, etc., desde el principio de la fase de implementación. De esta forma, se buscaba una mayor seguridad de que el sistema entregado al final del proyecto coincidiese con el esperado. Otros objetivos eran: simplificar la implementación y el mantenimiento del código, obtener formación a nivel teórico de estas estrategias y ponerlas en práctica en un proyecto de software real.

Para ello, antes del comienzo de la fase de implementación, se priorizaron en detalle las funcionalidades del sistema. Se estimó el tiempo requerido para implementar cada funcionalidad, se llevó a cabo un análisis de los riesgos de implementación y se definieron estrategias de mitigación de dichos riesgos. Los diferentes casos de uso se fueron desarrollando por orden de prioridad. En las reuniones con el director del proyecto se revisaba la funcionalidad implementada hasta el momento. Conforme surgían nuevos requisitos o era necesaria la modificación o refinamiento de requisitos previos, se revisaba el diseño y se implementaba o modificaba la funcionalidad correspondiente. Con este desarrollo incremental, se conseguía una mayor flexibilidad ante cambios en los requerimientos durante el desarrollo. Asimismo, el sistema era utilizable en todo momento.

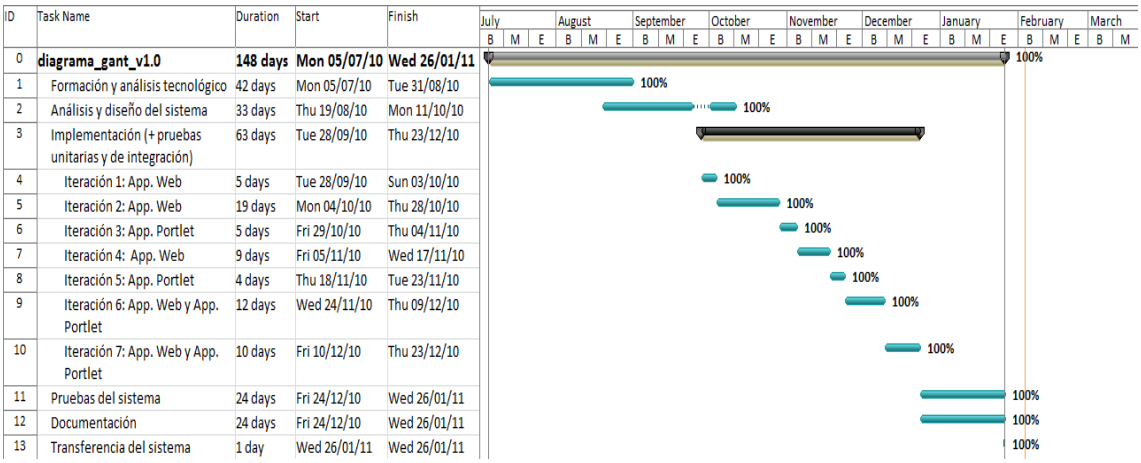
Por último, para elaborar la presente memoria también se llevaron a cabo varias iteraciones. Se realizó una primera versión que luego se fue corrigiendo y mejorando hasta alcanzar el objetivo deseado.

4.2 Planificación

La planificación de cada una de las fases del proyecto se puede observar en el diagrama Gantt de la figura 4.1. Como se puede ver, se estimó una fecha de comienzo y de finalización para cada fase. Esto implica una duración determinada, considerando como días hábiles de lunes a viernes (ambos incluidos) y una dedicación media por día hábil de 6 horas. A continuación, se describen brevemente los objetivos y resultados de cada una de las fases del proyecto.

4.2.1 Fase de formación y análisis de tecnologías

Esta fase tenía dos objetivos principales. El primero era adquirir los conocimientos técnicos necesarios para desarrollar Portlets y aplicaciones Web. El segundo objetivo



ID	Task Name	Duration	Start	Finish
0	**diagrama_gant_v1.0**	**148 days**	**Mon 05/07/10**	**Wed 26/01/11**
1	Formación y análisis tecnológico	42 days	Mon 05/07/10	Tue 31/08/10
2	Análisis y diseño del sistema	33 days	Thu 19/08/10	Mon 11/10/10
3	Implementación (+ pruebas unitarias y de integración)	63 days	Tue 28/09/10	Thu 23/12/10
4	Iteración 1: App. Web	5 days	Tue 28/09/10	Sun 03/10/10
5	Iteración 2: App. Web	19 days	Mon 04/10/10	Thu 28/10/10
6	Iteración 3: App. Portlet	5 days	Fri 29/10/10	Thu 04/11/10
7	Iteración 4: App. Web	9 days	Fri 05/11/10	Wed 17/11/10
8	Iteración 5: App. Portlet	4 days	Thu 18/11/10	Tue 23/11/10
9	Iteración 6: App. Web y App. Portlet	12 days	Wed 24/11/10	Thu 09/12/10
10	Iteración 7: App. Web y App. Portlet	10 days	Fri 10/12/10	Thu 23/12/10
11	Pruebas del sistema	24 days	Fri 24/12/10	Wed 26/01/11
12	Documentación	24 days	Fri 24/12/10	Wed 26/01/11
13	Transferencia del sistema	1 day	Wed 26/01/11	Wed 26/01/11

Figura 4.1. Diagrama Gantt con la planificación estimada del proyecto

era analizar y probar las tecnologías y herramientas más utilizadas para desarrollar estas aplicaciones con el fin de elegir aquellas más adecuadas para el proyecto. Esta fase culminó con la presentación de un seminario al director del proyecto sobre la tecnología de Portlets (dicho seminario está incluido en el anexo F). Sin embargo, la formación a lo largo de todo el proyecto ha sido continua ya que conforme se implementaba era necesario adquirir formación sobre las diferentes tecnologías y herramientas utilizadas.

4.2.2 Fase de análisis del problema

Los objetivos de esta fase eran: analizar las necesidades de los usuarios del sistema y capturar los requisitos del mismo. Para ello se utilizaron tres tipos de técnicas: reuniones con el director, tablas de requisitos y casos de uso. Esta fase no fue especialmente compleja dado que muchos de los requisitos funcionales del sistema eran conocidos previamente por el director del proyecto.

4.2.3 Fase de diseño de la solución

El objetivo de esta fase era definir la arquitectura general del sistema y diseñar cada uno de los elementos que conforman dicha arquitectura. Se realizó un diseño de abajo a arriba, empezando con el modelado de los datos y finalizando con el modelado de la interfaz de usuario. Los resultados de esta fase fueron: el diseño multicapa realizado, el modelado de los de datos del sistema, el diseño de los componentes de alto nivel y las clases del sistema y el modelado de la interfaz de usuario.

Una vez elaborado un primer diseño del sistema, se planificó la fase de implementación. Se priorizaron las funcionalidades del sistema y se estimó su

duración en jornadas de trabajo. En el diagrama Gantt de la figura 4.1 se muestra la fase de implementación desglosada en sus iteraciones y la aplicación afectada en cada iteración.

4.2.4 Fase de implementación

El objetivo de esta fase era codificar el sistema siguiendo el diseño elaborado en la fase anterior. Esta fase se dividió en 7 iteraciones. En la iteración 1 se implementó el esqueleto de la aplicación Web con una interfaz básica que ofrecía una visión completa de las diferentes opciones de menú (siguiendo la filosofía de hacer que el sistema completo funcionase antes de hacerlo atractivo a nivel de interfaz de usuario). En la iteración 2 se implementó la funcionalidad principal o básica de la aplicación Web (registro e inicio de sesión, alta de asignaturas, alumnos, horarios, etc). En la tercera iteración, se implementó el esqueleto y funcionalidad básica de la aplicación Portlet (inicio de sesión y solicitar tutorías). En la cuarta y quinta iteraciones, se implementó la funcionalidad secundaria de la aplicación Web (preferencias de configuración, criterios de visualización de informes, generación de informes en formato PDF, etc.) y de la aplicación Portlet (cancelar y modificar solicitudes de tutorías, solicitud de nueva contraseña, etc.), respectivamente. En la iteración 6 se codificaron aspectos que afectaban al sistema completo: las copias de seguridad, la generación de ficheros de *log*, las operaciones de mantenimiento del sistema cada curso académico, etc. Finalmente, en la séptima iteración, se mejoró la interfaz gráfica del sistema completo.

Además, durante la implementación se fueron realizando pruebas unitarias de los diferentes componentes así como pruebas de integración. El resultado de esta fase es el sistema funcionando de forma local en la máquina del alumno.

4.2.5 Fase de pruebas

El objetivo de esta fase era detectar y posteriormente corregir errores en el sistema. Las pruebas abarcaron toda la funcionalidad del sistema. También se simuló un cambio de curso académico para comprobar que las operaciones de mantenimiento del sistema se realizaban correctamente. Del mismo modo, se simuló la pérdida de todos los datos de la aplicación para comprobar su correcta restauración mediante las copias de seguridad realizadas por el sistema. Como resultado de esta fase, se corrigieron errores de implementación que habían pasado desapercibidos en la fase anterior.

4.2.6 Fase de documentación

Durante esta fase se escribió esta memoria con el resumen del proyecto así como los anexos, utilizando como base toda la documentación que había sido generada durante las diferentes fases del proyecto.

4.2.7 Fase de transferencia del sistema

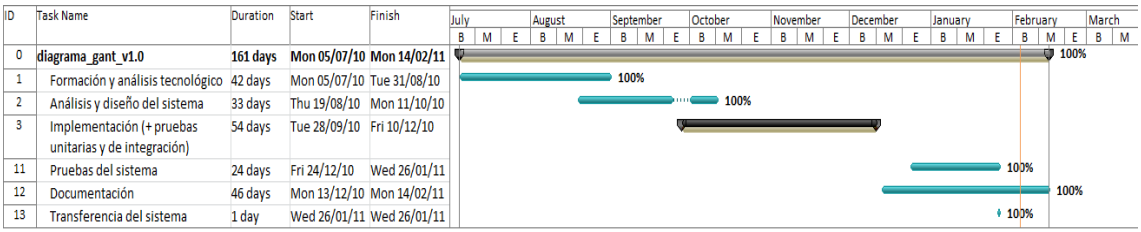
En esta fase se instaló la aplicación Web en un servidor de la Universidad para ponerla a disposición de los profesores. Además, se integró en la página Web del director del proyecto la aplicación Portlet de petición de cita de tutoría para ponerla a disposición de los alumnos.

4.3 Esfuerzo real dedicado

Al inicio del proyecto (julio de 2010), se estimó una duración del mismo de entre 5 y 7 meses, equivalentes a entre 650 y 800 horas de trabajo efectivo. Esta estimación estaba basada fundamentalmente en la experiencia del director del proyecto con otros proyectos similares de desarrollo de un sistema con un fuerte componente de aprendizaje tecnológico. Por tanto, se estableció como objetivo el depósito del proyecto en febrero de 2011 para concurrir a la convocatoria de marzo de 2011.

La duración real de cada una de las fases del proyecto se puede observar en el diagrama Gantt de la figura 4.2. Si se compara con el de la figura 4.1, se observa que las fases que presentan diferencias en cuanto a su duración son la de implementación y la de documentación. La fase de implementación duró 9 jornadas de trabajo menos de lo planificado. Esta disminución se debe a que se sobrestimaron ligeramente los esfuerzos de implementación requeridos dada la inexperiencia del alumno con la mayoría de tecnologías utilizadas en el momento de realizar las estimaciones. Como consecuencia de la disminución de la duración de esta fase, la fase de documentación adelantó su fecha de inicio. Sin embargo, dicha fase duró 22 jornadas de trabajo más de lo planificado. La razón de este desvío reside en la menor dedicación al proyecto durante las 2 semanas correspondientes a las vacaciones de Navidad.

En la figura 4.3 se muestra el tiempo en horas y el porcentaje dedicado a cada fase del proyecto. En total, el número de horas empleadas en el proyecto ha sido de 824. Esto implica un desvío de 24 horas con respecto a la cota superior de 800 horas estimada al inicio del proyecto. Este desvío se debe, fundamentalmente, al trabajo con numerosas tecnologías y herramientas novedosas, al interés del alumno por aprenderlas aun cuando no se utilizaran todas sus posibilidades de forma directa en el proyecto y a las revisiones de diseño necesarias en ciertas etapas de la fase de implementación.



ID	Task Name	Duration	Start	Finish
0	**diagrama_gant_v1.0**	**161 days**	**Mon 05/07/10**	**Mon 14/02/11**
1	Formación y análisis tecnológico	42 days	Mon 05/07/10	Tue 31/08/10
2	Análisis y diseño del sistema	33 days	Thu 19/08/10	Mon 11/10/10
3	Implementación (+ pruebas unitarias y de integración)	54 days	Tue 28/09/10	Fri 10/12/10
11	Pruebas del sistema	24 days	Fri 24/12/10	Wed 26/01/11
12	Documentación	46 days	Mon 13/12/10	Mon 14/02/11
13	Transferencia del sistema	1 day	Wed 26/01/11	Wed 26/01/11

Figura 4.2. Diagrama Gantt con la duración real del proyecto

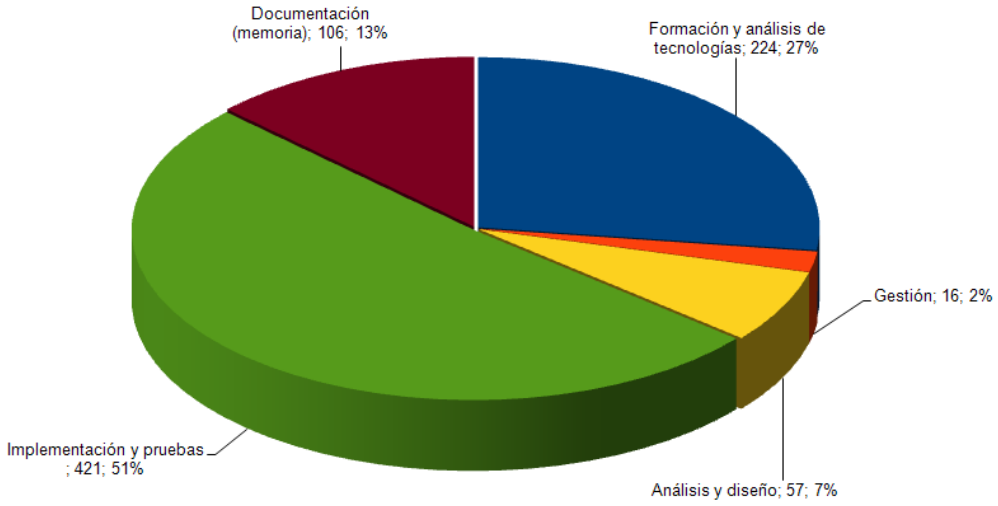


Figura 4.3. Horas y porcentaje de esfuerzo dedicados a cada fase del proyecto

Capítulo 5

Conclusiones

En este último capítulo se reflexiona y concluye acerca del proyecto. En la sección 5.1 se resume el trabajo realizado en el proyecto, cumplimiento de los objetivos, principales problemas encontrados, etc. En la sección 5.2 se presentan las reflexiones personales del alumno sobre el proyecto. En la sección 5.3 se exponen una serie de trabajos futuros relacionados con este proyecto que podrían resultar interesantes, ya sea como proyectos fin de carrera o en otro marco de trabajo o investigación.

5.1 Resultados

El propósito de este proyecto fin de carrera era diseñar e implementar un sistema accesible vía Web que facilitara a profesores y alumnos la gestión de la asistencia a las tutorías.

Para ello, se ha diseñado e implementado una aplicación Web de configuración en la que los profesores pueden configurar las asignaturas para las cuales quieren utilizar el sistema y sus horarios de tutorías. También se ha desarrollado una aplicación Portlet para la reserva de cita de tutoría por los alumnos dados de alta por los profesores mediante la aplicación anterior. La aplicación Web de configuración se ha instalado en el servidor Alkaid y es accesible desde la URL `http://alkaid.cps.unizar.es:8280/igetutor/`. La aplicación Portlet se ha instalado en la página Web del director del proyecto, accesible a través de la URL `http://webdiis.unizar.es/~alvaper/`.

Estas dos aplicaciones han permitido cumplir con los objetivos del proyecto detallados en la sección 2.1. Se ha conseguido desarrollar toda la funcionalidad especificada en los requisitos funcionales del sistema. También se han conseguido todos los aspectos de naturaleza no funcional descritos en los requisitos no funcionales. En cuanto a los objetivos de gestión, se ha cumplido la fecha de entrega planteada al principio del proyecto.

Los principales problemas encontrados a lo largo del proyecto han sido fundamentalmente consecuencia del aprendizaje y adaptación del alumno a las tecnologías y herramientas utilizadas. Es por ello que el proyecto ha tenido un fuerte componente formativo, tal y como se puede observar en los esfuerzos dedicados incluidos en el capítulo anterior. Conviene también resaltar que, en ocasiones, la formación adquirida a través de la lectura de libros y documentos no ha tenido aplicación directa en el proyecto. Sin embargo, el alumno consideraba este trabajo como una buena oportunidad para aprender otras cuestiones aunque no estuvieran directamente implicadas en el proyecto. Aparte de estos, no ha existido ningún otro problema relevante.

5.2 Opinión personal

La realización del presente proyecto ha supuesto una experiencia muy enriquecedora para su autor desde varias perspectivas.

Desde una perspectiva técnica, uno de los objetivos principales del proyecto para el alumno era el aprendizaje de nuevos tipos de aplicaciones, tecnologías y herramientas usadas ampliamente en la actualidad en la industria del software.

En cuanto a los tipos de aplicaciones, esta es la primera aplicación Web desarrollada por el alumno, ya que el resto de aplicaciones realizadas durante la carrera habían sido aplicaciones de escritorio con interfaces gráficas o de línea de comandos. Para su diseño e implementación se han tenido que estudiar aplicaciones Web existentes y patrones de diseño. Esto ha incrementado el conocimiento del alumno de sistemas informáticos.

En cuanto a las tecnologías, el uso de tecnologías Web como HTML, CSS, JavaScript y AJAX (combinación de las anteriores con JSON), ha permitido al alumno adquirir experiencia de gran valor a la hora de su futura incorporación al mercado laboral. La tecnología de Portlets utilizada es una tecnología emergente y, como tal, resulta difícil predecir en este momento sus perspectivas de futuro. El trabajo con esta tecnología novedosa ha permitido al alumno experimentar las dificultades típicas que existen al empezar a trabajar con una tecnología de reciente creación. Algunos ejemplos de estas dificultades son: la escasez de bibliografía y en general de documentación, el estado inicial de muchas herramientas de desarrollo, la falta de expertos en la tecnología con los que consultar dudas, etc. En cualquier caso, el uso de esta tecnología ha supuesto además la toma de contacto con otra serie de tecnologías relacionadas como por ejemplo los Servlets y las JavaServer Pages.

Finalmente, se ha aprendido a utilizar herramientas ampliamente conocidas como los frameworks Hibernate y Spring. Estas herramientas son muy demandadas en la actualidad por las empresas del sector [12] por lo que su aprendizaje tiene de nuevo un gran valor para el futuro profesional del autor. También el uso de un

framework Web como Tapestry, no tan usado como los frameworks anteriores u otros frameworks Web como Struts, pero de gran utilidad práctica. De este modo, se ha complementado la formación adquirida durante la carrera.

Desde la perspectiva de gestión del proyecto, se han aplicado los conocimientos teóricos y prácticos de gestión de proyectos adquiridos a lo largo de estos años. Actividades del proyecto como la planificación, la realización de estimaciones, el análisis de los posibles riesgos y la definición de estrategias de mitigación de los mismos, el control estricto de la gestión de esfuerzos, la gestión de las configuraciones de software, etc., se han puesto de relieve en el proyecto.

Por último, en cuanto a los objetivos personales de índole no exclusivamente técnica, uno de ellos era la adquisición de mayor experiencia en la comunicación tanto de cuestiones técnicas de un sistema como de los aspectos de gestión. Esto se ha conseguido a través de las reuniones y correos electrónicos con el director del proyecto y se seguirá practicando con la defensa del proyecto. Por otro lado, el autor desea que esta aplicación sea de utilidad tanto a profesores como alumnos y que pueda ser ampliada en el futuro ante nuevas necesidades.

5.3 Trabajo futuro

En esta sección se describen posibles trabajos futuros que podrían complementar o ampliar el proyecto realizado.

5.3.1 Comparativas de tecnologías

Desde el principio, este proyecto se centró en la tecnología estándar de Java Portlets. Sin embargo, podría resultar interesante realizar una comparativa entre esta tecnología y su equivalente más cercano en la plataforma .NET, las Web Parts [31]. Se podría realizar un proyecto de software de pequeño tamaño utilizando ambas tecnologías para adquirir experiencia en su manejo y realizar un análisis de ambas tecnologías, comparando sus ventajas e inconvenientes, los tiempos de aprendizaje y desarrollo empleado en el proyecto con ambas tecnologías, etc.

En la misma línea, también se podrían estudiar y comparar a nivel teórico y práctico los Portlets, los Google Gadgets [72] y los Web Widgets [73]. El *World Wide Web Consortium* (W3C) [74] se encuentra actualmente trabajando en la descripción de un conjunto de estándares para los Web Wigets [75]. Además, algunos trabajos en esta línea muestran que las relaciones entre Portlets y Widgets son aún difusas [76] por lo que este campo se encuentra totalmente abierto a nuevas publicaciones.

5.3.2 Ampliaciones del sistema actual

Algunas de las ampliaciones que se podrían realizar al sistema actual son:

- Mejorar el formato de presentación de los informes en PDF utilizando herramientas avanzadas como por ejemplo JasperReports [61].
- Mejorar la interfaz de la aplicación Portlet utilizando componentes AJAX avanzados (calendarios, pestañas, etc.) proporcionados por librerías JavaScript como Dojo [32] y jQuery [33].
- Mejorar el formato de los correos electrónicos enviados a profesores y alumnos.
- Dar la posibilidad al administrador de configurar la frecuencia con la que se realizan las copias de seguridad.
- Ofrecer la aplicación Portlet en otros idiomas para los estudiantes Erasmus.
- Permitir al administrador enviar notificaciones a todos los profesores registrados en el sistema.
- Sincronizar las solicitudes de tutoría pendientes con aplicaciones como iCal [70] o Google Calendar [71] para que los profesores puedan recibir notificaciones por SMS.

Como se puede ver, algunas de estas ampliaciones tienen que ver con la mejora de cuestiones estéticas del sistema y otras con la adición funcionalidad para futuras versiones del mismo.

Anexo A

Análisis de la tecnología de Portlets

En este anexo se presenta un análisis completo de la tecnología de Portlets elaborado a partir del seminario de Portlets realizado durante la fase de formación y análisis de tecnologías del proyecto. El seminario original se incluye en el anexo F. En la sección A.1 se introduce la tecnología de Portlets. En la sección A.2 se ofrece una visión general de los estándares de Portlets. La sección A.3 se enfoca hacia la implementación de Portlets, analizando diferentes herramientas de desarrollo. Por último, en la sección A.4 se presentan una serie de reflexiones sobre esta tecnología.

A.1 Introducción

Un Portlet es una aplicación Web que proporciona un servicio o información y que se incluye en un portal Web. Facilitan por tanto la integración de aplicaciones en páginas de portales Web. Son utilizados por los portales Web como componentes integrables a nivel de interfaz de usuario, proporcionando una capa de presentación a sistemas de información.

Son gestionados por un contenedor de Portlets. También se pueden considerar como servicios Web orientados a presentación. Esta visión se explicará en la sección A.2.2.

Son una extensión de los Servlets (otra capa de abstracción por encima) aunque tienen importantes diferencias con respecto a éstos [43]. Por ejemplo, en un Portlet el desarrollador no se debe preocupar de qué método HTTP ha utilizado el cliente (GET, POST, etc.), ni de crear su propia infraestructura para capturar los eventos del cliente (por ejemplo, cuando se pulsa un botón). Los Portlets complementan y coexisten con las aplicaciones Web tradicionales, no están pensados para reemplazarlas.

Algunas de las características funcionales más relevantes de un Portlet son:

- Producen fragmentos de código *markup* (por ejemplo, HTML, XHTML, WML), a diferencia de los Servlets que generan documentos completos.
- No son directamente direccionables a través de una URL (a diferencia de los Servlets). Lo que sí es direccionable, es la página del portal Web donde esté instalado el Portlet.
- Generan contenido Web estático o dinámico.
- Se renderizan como una tabla HTML.
- Pueden ser locales o remotos al portal Web.
- Interaccionan con el cliente Web (navegador) mediante el paradigma petición/respuesta.
- Tienen 2 fases de interacción diferentes: fase de acción y fase de renderización. En cada fase se tratan las peticiones correspondientes.
- No pueden generar contenido arbitrario al ser parte de una página Web de un portal. Por ejemplo, si el portal solicita contenido de tipo *html/text*, todos los Portlets del portal deben generar contenido con ese formato.
- Para ejecutarlos se necesita un servidor de portales, o un contenedor de Portlets que contenga un *driver* que simule algunas de las tareas de un servidor de portales.

Los Portlets están formados por dos partes:

1. Decoración: Barra de título, controles (maximizar, minimizar, cerrar, etc.) y bordes de las ventanas.
2. Contenido del Portlet: La parte generada por la aplicación Portlet.

En la figura A.1 se puede ver un ejemplo de un Portlet muy simple que muestra un listado de empleados. Se puede observar la decoración del Portlet con su barra de título, botones de cerrar, minimizar, personalizar, ayuda y acerca de. También se muestra el contenido del Portlet que en este caso es el listado de empleados.

A.1.1 Portal Web

Un portal Web [34] es una aplicación Web que presenta información de diversas fuentes de forma unificada y contiene la capa de presentación de diferentes sistemas de información. Constituyen un punto de acceso único a información agregada.

El objetivo principal de un portal es servir información de interés a sus usuarios, de forma organizada, proporcionando una visión integrada de un conjunto de aplicaciones. Por lo tanto, son plataformas de integración de aplicaciones.

Los portales se pueden dividir en dos clases:

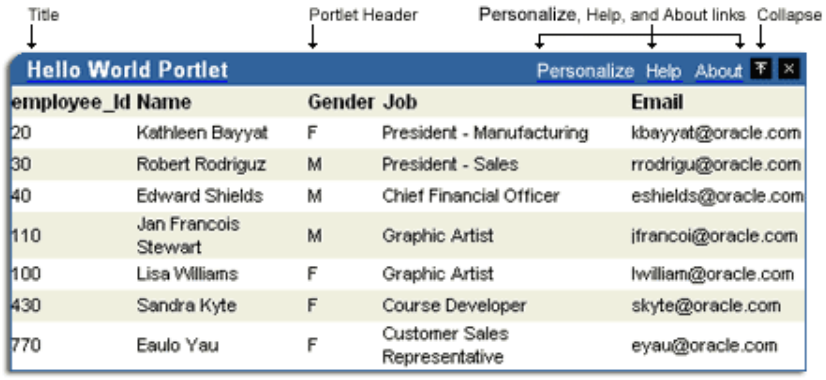


Figura A.1. Ejemplo de un Portlet

1. Portales Intranet o corporativos: Son utilizados por muchas empresas como recurso central para proveer de información de la compañía a sus empleados.

2. Portales Internet: Son utilizados por cualquier usuario particular para agregar información y servicios de interés. Ejemplos de este tipo de portales son los portales de Google (iGoogle), Yahoo (myYahoo), MSN, etc.

Un portal Web de Internet ofrece una serie de servicios típicos: buscador Web, correo electrónico, noticias, partes meteorológicos, calendario, agenda personal, *blogs*, foros, cotizaciones bursátiles, etc.

Los portales integran aplicaciones a nivel de interfaz de usuario. Cada una de las aplicaciones que integra un portal es un Portlet. Una página de un portal se muestra como una colección de Portlets. El portal se encarga de construir cada página Web agregando las respuestas de los Portlets que contiene (los fragmentos de código *markup* que generan los Portlets).

Las tres funcionalidades típicas que caracterizan a un portal son:

1. Inicio de sesión *Single Sign-On*: Mediante este tipo de inicio de sesión, el usuario sólo tiene que introducir una sola vez sus datos de inicio. El portal se encarga de iniciar sesión en todas las aplicaciones que contiene (correo electrónico del usuario, blogs, agenda personal, etc). De esta forma, el usuario accede a varios sistemas independientes con una única instancia de autenticación.

2. Agregación de contenido: El usuario puede agregar contenido procedente de diferentes fuentes (*blogs*, fuentes de noticias, servicios de correo electrónico como Gmail, Hotmail, etc).

3. Personalización de contenido: El usuario puede personalizar el contenido agregado (temática de las noticias, ciudad de los partes meteorológicos, etc.) y elegir los Portlets que requiere en las páginas de las que dispone.

En la figura A.2 podemos ver una captura de pantalla del portal MyYahoo.

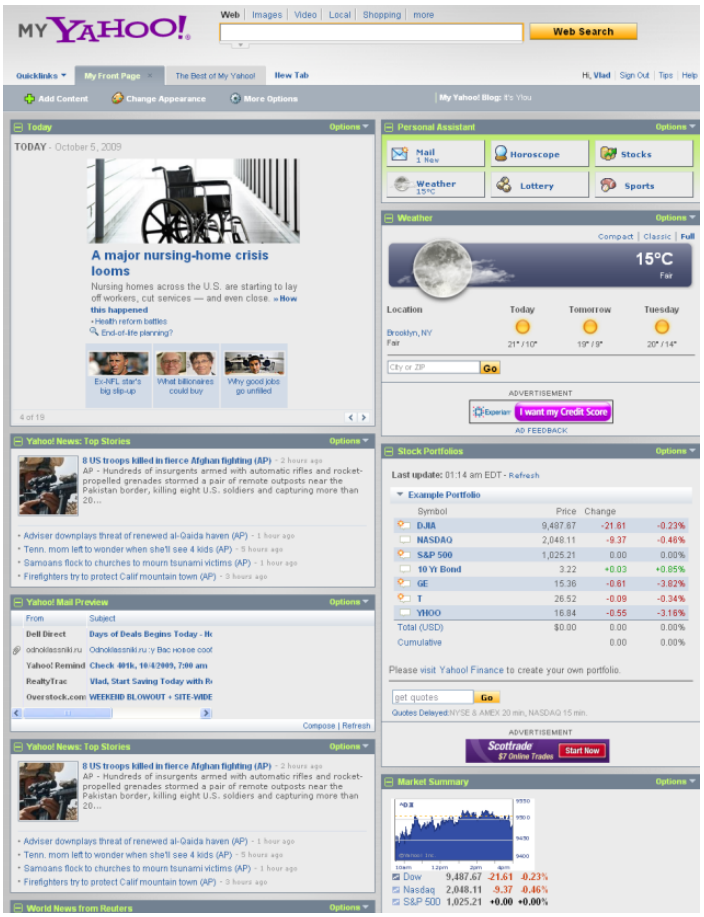


Figura A.2. Ejemplo de un portal Web

A.1.2 Servidor de portales Web

Un servidor, framework o plataforma de portales Web es una aplicación que proporciona un portal preconstruido donde se pueden instalar Portlets. Para ello, tienen un contenedor de Portlets instalado.

En la actualidad existen numerosos servidores de portales, algunos de los más conocidos son:

- Comerciales: IBM WebSphere Portal, Oracle Portal, Microsoft Sharepoint Portal Server, Liferay Portal Enterprise Edition, etc.
- Open Source: JetSpeed2, Liferay Portal Community Edition, uPortal, GateIn Portal (fusión de JBoss Portal y eXo Portal), etc.

Los servidores de portales se instalan en un servidor de aplicaciones J2EE (GlassFish, JBoss, WebLogic, etc.) o en un servidor Web con soporte para Servlets y JavaServer Pages (Tomcat, Jetty, etc). La mayoría de servidores de portales no son más que aplicaciones Web desplegadas en un servidor de aplicaciones que utilizan un Servlet para gestionar las peticiones que se dirigen al servidor de portales.

Los servidores de portales deciden el aspecto global de las páginas del portal Web (logotipo, color de las barras de título de los Portlets, iconos de los controles de los Portlets, etc). Sin embargo, dicho aspecto puede ser configurado por el usuario cambiando hojas de estilo CSS y páginas JSP.

A.1.3 Contenedor de Portlets

Un contenedor de Portlets representa la interfaz entre los Portlets y el portal Web. La idea del contenedor de Portlets es similar a la del contenedor de Servlets, de hecho, es una extensión de éste. El contenedor de Portlets implementa la API de Portlets, que se puede considerar como el "contrato" entre los Portlets y los portales [42].

Los Portlets se instalan en un contenedor de Portlets. Típicamente, el contenedor de Portlets es un componente que forma parte del servidor de portales. El contenedor de Portlets se encarga fundamentalmente de las siguientes tareas:

- Gestionar y ejecutar los Portlets.
- Proveer al Portlet de los recursos necesarios y de su entorno de ejecución.
- Controlar su ciclo de vida:
  - Inicializa y destruye los Portlets.
  - Recibe las peticiones del usuario y las traslada al Portlet.

Dos ejemplos de contenedores de Portlets son: OpenPortal Portlet Container y Apache Pluto.

El contenedor OpenPortal Portlet Container ha sido utilizado en este proyecto ya que cuenta con un *driver* que simula el funcionamiento de un servidor de portales, proporcionando un entorno ligero para desplegar Portlets. En la figura A.3 se puede ver este contenedor con una serie de Portlets de ejemplo instalados.

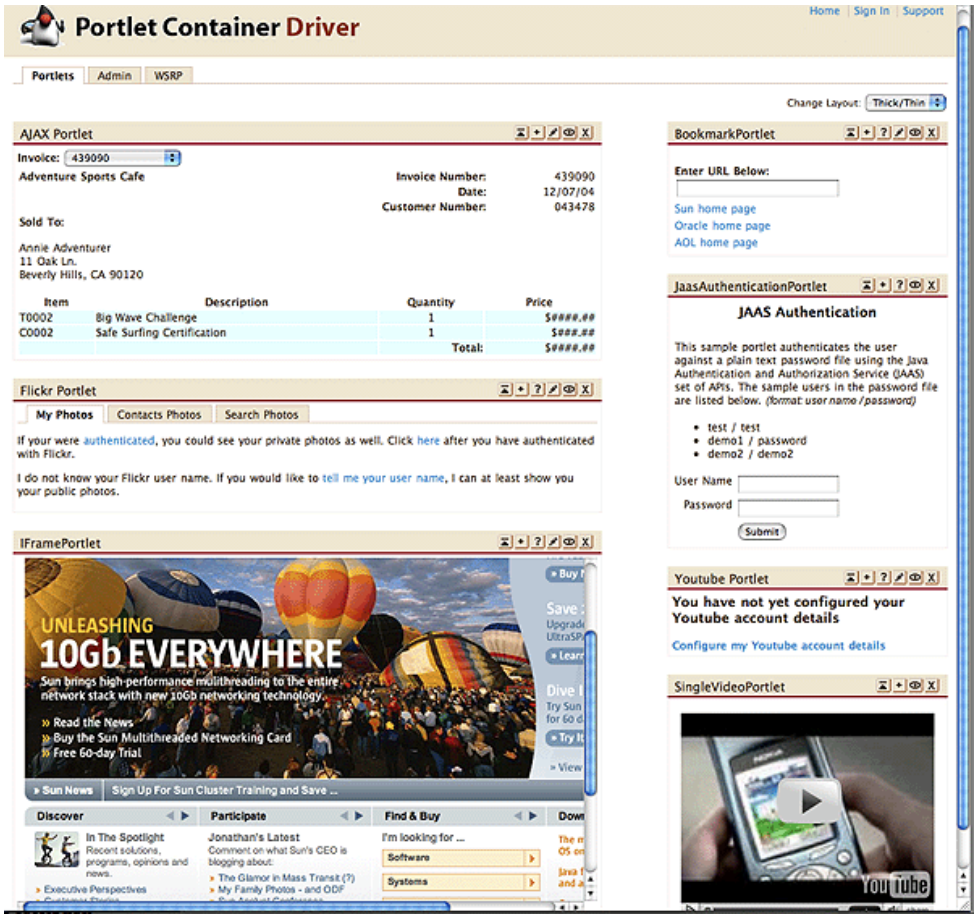


Figura A.3. OpenPortal Portlet Container con ejemplos de Portlets

A.2 Estándares de Portlets

En la actualidad hay dos estándares de Portlets. Estos dos estándares surgieron con el fin de solucionar dos problemas tradicionales que existían en los primeros servidores de portales.

Estos dos problemas se explican a continuación:

1. Los Portlets desarrollados en un determinado portal Web no podían ser instalados en otro portal. Es decir, no había compatibilidad de Portlets entre los diferentes portales disponibles en el mercado. Esto se solucionó con la creación del estándar denominado *Java Portlet Specification* (JPS).

2. Un portal Web no podía consumir remotamente los Portlets instalados en otro portal. Todos los Portlets debían ser locales al portal. Para solucionar este

segundo problema se creó el estándar denominado *Web Services for Remote Portlets* (WSRP).

Hoy en día, la mayor parte de los servidores de portales disponibles en el mercado soportan ambos estándares. Además, hay que destacar que ambos estándares están fuertemente ligados y se han desarrollado de forma paralela en el tiempo:

- WSRP 1.0 - Agosto de 2003.
- JPS 1.0 (más comúnmente conocida JSR-168) - Octubre de 2003.
- WSRP 2.0 - Abril de 2008.
- JSP 2.0 (más comúnmente conocida como JSR-286) - Junio de 2008.

Ambos estándares son totalmente compatibles.

A.2.1 Java Portlet Specification (JPS)

El estándar JPS define la API de Java que permite desarrollar Portlets locales. Esta API ha sido utilizada en el proyecto para desarrollar el Portlet de reserva de cita de tutoría utilizado por los alumnos. Estandariza la API que ofrece el contenedor de Portlets a los Portlets. Permite delegar la generación de código de *markup* en un Servlet de la aplicación Portlet o en páginas JSP (estrategia utilizada en el proyecto).

Como se ha comentado anteriormente, este estándar surgió para solucionar el primer problema que existía en los primeros servidores de portales. Como consecuencia, permite interoperabilidad de Portlets entre diferentes portales.

El estándar abarca las siguientes áreas de los Portlets:

- Agregación
- Personalización
- Presentación
- Seguridad

Hasta la fecha, este estándar cuenta con dos versiones, la 1.0 y la 2.0. A continuación se enumeran brevemente algunas de las características que definía la versión 1.0 del estándar:

- Permite guardar las preferencias de los usuarios del Portlet en una base de datos interna gestionada por el contenedor de Portlets (transparente al desarrollador).
- Modos del Portlet (formas de funcionamiento):
  - Estándares: *View* (tarea principal del Portlet), *Edit* (configuración de preferencias del Portlet) y *Help* (ayuda del Portlet).
  - Modos a medida (creados por el desarrollador).
- Estados de la ventana del Portlet (cantidad de espacio que el portal asigna al fragmento generado por el Portlet):
  - Estándares: Maximizada, minimizada y normal.
  - Estados a medida (definidos por el desarrollador).
- Procesamiento y manejo de las peticiones del Portlet refinado:
  - *RenderRequest*: Procesamiento de peticiones de renderización (producen el contenido que se muestra al usuario, es decir, los fragmentos de *markup*).
  - *ActionRequest*: Procesamiento de peticiones de acción (cambian el estado del sistema y no producen *markup*).

En el momento en que los servidores de portales comenzaron a integrar la versión 1.0 de la especificación, empezaron a detectarse carencias importantes de funcionalidad que el estándar no contemplaba. Pronto comenzó el desarrollo de la versión 2.0 del estándar con el fin de subsanar estas carencias y proveer de nuevas funcionalidades a los Portlets.

Entre las nuevas funcionalidades que introdujo JPS 2.0 destacan:

- Dos mecanismos de coordinación entre Portlets:
  1. Eventos: Mecanismo potente que permite la comunicación entre los Portlets de un portal a través del envío y recepción de eventos.
  2. Parámetros públicos de renderización: Mecanismo menos potente que el anterior pero más sencillo de utilizar. Permite a los Portlets especificar qué parámetros de renderización (*render parameters*) desean compartir con otros Portlets.
- Soporte para servir recursos (por ejemplo descarga de ficheros binarios) en el contexto del Portlet, lo que permite soportar interacciones AJAX.
- Filtros de Portlets: Permiten transformar "al vuelo" el contenido de las peticiones y respuestas de los Portlets, crear cadenas de filtros, etc.

- Anotaciones en el código para los métodos que procesan peticiones de acción, renderización, recursos y eventos. Estas anotaciones minimizan el código requerido haciendo uso del principio de convención sobre configuración [14].

Esta versión 2.0 del estándar JPS está totalmente sincronizada con la versión 2.0 del estándar WSRP. También mantiene total compatibilidad hacia atrás con la versión 1.0 del propio estándar, de hecho, es una extensión de ésta. Más detalle acerca de ambas versiones del estándar se puede consultar en los propios estándares [36] [37] o en artículos explicativos [39].

A.2.2 Web Services for Remote Portlets (WSRP)

El estándar WSRP es un protocolo para la comunicación con Portlets remotos. Como se ha indicado anteriormente, los Portlets se pueden considerar como servicios Web orientados a presentación ya que integran tanto los datos como la lógica de presentación. Este enfoque es opuesto al tradicional de servicios Web orientados a datos. Los servicios Web tradicionales permiten reusar servicios de *back-end*, es decir, ofrecen reusabilidad a nivel funcional o de lógica de negocio. WSRP permite que los Portlets de un portal sean expuestos como servicios Web orientados a presentación y puedan ser consumidos remotamente. Permite por tanto reutilizar la interfaz de usuario completa.

WSRP está construido sobre los estándares existentes de servicios Web como SOAP [81], WSDL [80] y UDDI [82]. Por ejemplo, define cómo usar SOAP para solicitar y recibir fragmentos HTML. De nuevo, este enfoque es diferente al tradicional de invocar un método a través de SOAP y obtener como respuesta típica datos en formato XML o JSON.

El objetivo de WSRP es conseguir que Internet sea un mercado de servicios Web orientados a presentación o visuales (i.e. Portlets) listos para ser integrados en Portales Web.

Los actores que intervienen en este estándar son:

- Portal Web
- Portlet
- Productor WSRP
- Consumidor WSRP

El estándar define el API para exportar los Portlets de un productor a consumidores remotos. Este estándar permite comunicación entre Portlets remotos y por tanto la integración en un portal de Portlets procedentes de diversas fuentes. Además proporciona interoperabilidad, es decir, productor y consumidor de Portlets pueden usar tecnologías diferentes (por ejemplo, JEE y .NET).

A nivel técnico, WSRP define:

- Un conjunto de interfaces WSDL que debe implementar un productor para permitir a los consumidores invocar WSRP:
  - *Service Description*: Esta interfaz WSDL es de implementación obligatoria. Sirve para obtener una descripción del servicio ofrecido por el Portlet remoto.
  - *Markup*: Esta interfaz WSDL es de implementación obligatoria. Sirve para permitir la interacción con el Portlet remoto.
  - *Registration*: Interfaz opcional que permite al consumidor del Portlet remoto registrarse con el productor.
  - *Portlet Management*: Interfaz opcional que sirve para gestionar el ciclo de vida del Portlet remoto (por ejemplo para ofrecerlo sólo durante un periodo de tiempo).
- Métodos para publicar, encontrar y asociar WSRP y metadatos.
- Reglas para los fragmentos de *markup* emitidos por los WSRP.

Para una información más detallada, se puede consultar la versión 2.0 del estándar WSRP [35].

A.3 Desarrollo de Portlets

Una aplicación Portlet es una extensión de una aplicación Web JEE que se empaqueta en un fichero WAR (*Web application Archive*). La aplicación Portlet se instala en un servidor de portales o en un contenedor de Portlets. Cada aplicación Portlet puede constar de uno o más Portlets.

Un Portlet no es más que una clase Java que implementa la interfaz *javax.portlet.Portlet*, o hereda de la clase *GenericPortlet* que implementa la interfaz anterior (estrategia usada en el proyecto). Las aplicaciones Portlet contienen un descriptor de despliegue que consiste en un fichero XML donde se especifican los Portlets disponibles para el contenedor y qué clase se debería utilizar para instanciarlos. Al ser un tipo especial de aplicación Web, también contienen el fichero XML que sirve como descriptor de despliegue para toda aplicación Web (web.xml) donde se define su estructura.

Para empezar a desarrollar Portlets, es necesario tener conocimientos previos de X/HTML, XML, Servlets y páginas JSP. En la figura A.4 se puede observar la estructura típica de una aplicación Portlet. Como se puede ver, la estructura es similar a la de una aplicación Web tradicional.

En la actualidad, no existen prácticamente herramientas gratuitas en estado de madurez que ayuden al desarrollo de Portlets. Las herramientas que se analizaron y probaron en el proyecto se describen a continuación.

Los entornos de desarrollo NetBeans y Eclipse cuentan con un *plugin* denominado Portal Pack que facilita ligeramente el desarrollo de Portlets.

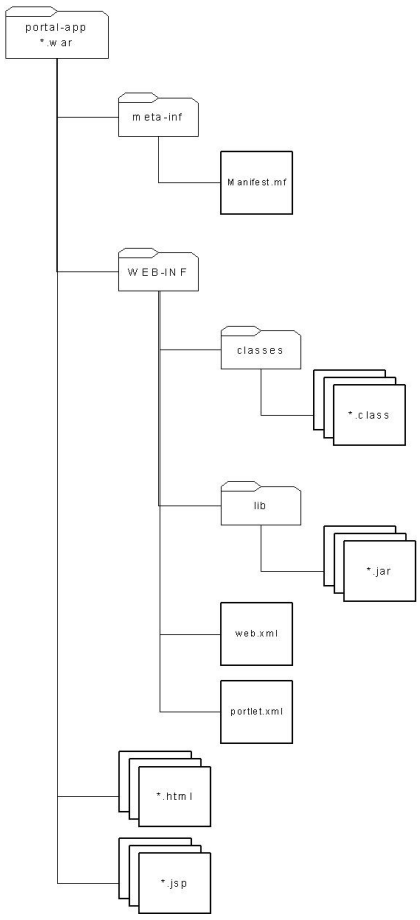


Figura A.4. Estructura de una aplicación Portlet

A.3.1 NetBeans Portal Pack

El *plugin* Portal Pack de NetBeans [44] permite:

- Creación automática de la estructura inicial de la clase Java, el descriptor XML y las páginas JSP del Portlet, añadiendo el tipo de proyecto Portlet a los proyectos de NetBeans.
- Integración de NetBeans con el servidor de portales Liferay, funcionando sobre el servidor de aplicaciones J2EE GlassFish.
- Integración de NetBeans con el servidor de portales Sun Java System Portal Server.
- Integración de NetBeans con el contenedor de Portlets OpenPortal Portlet Container.

- Integración de NetBeans con Spring Portlet MVC.
- Desarrollo de aplicaciones Portlet con JavaServer Faces (JSF) [19], utilizando un editor visual (WYSIWYG).

La versión actual de esta herramienta es la 3.0.4.

A.3.2 Eclipse Portal Pack

El *plugin* Portal Pack de Eclipse [45] está menos avanzado que el de NetBeans.

Ofrece las siguientes funcionalidades:

1. Creación automática de la estructura inicial de la clase Java, el descriptor XML y las páginas JSP del Portlet, añadiendo el tipo de proyecto Portlet a los proyectos de Eclipse.
2. Integración de Eclipse con el servidor de portales WebSpace, funcionando sobre el servidor de aplicaciones J2EE GlassFish.
3. Integración de Eclipse con el contenedor de Portlets OpenPortal Portlet Container.

La versión actual es la 2.0.1.

A.3.3 Eclipse Portlet Tools

Eclipse Portlet Tools [41] es otro *plugin* para Eclipse. Permite crear automáticamente el descriptor XML, completado de etiquetas en este fichero, etc., pero aún no permite integración con servidores de portales o contenedores de Portlets.

La versión actual es la 0.2.0.

A.3.4 Spring Portlet MVC

El framework Spring cuenta con un módulo Web que además de soportar el desarrollo de aplicaciones Web convencionales (basadas en Servlets), también soporta el desarrollo de Portlets según la versión inicial del estándar JPS (1.0).

Spring Web MVC es un framework Web orientado a acciones (*request-driven Web MVC framework*), en oposición a los framework Web orientados a componentes (*event-driven Web UI frameworks*) como Tapestry. Por tanto, requiere que el desarrollador trabaje con cuestiones de más bajo nivel.

El desarrollo de Portlets usando Spring Portlet MVC [40] requiere experiencia previa, en general con el framework Spring y en particular con el módulo Spring Web MVC.

A.3.5 Tapestry Portlet Support

Tapestry es el framework Web que se ha utilizado para el desarrollo de la capa de presentación de la aplicación Web de configuración.

La versión 4.1 del framework Web Tapestry tenía soporte para el estándar de Portlets de Java 1.0 (JSR-168) [38]. Sin embargo, la versión actual de dicho framework (versión 5) sufrió un profundo cambio y los desarrolladores de Tapestry tuvieron que desechar el soporte antiguo de Portlets. Actualmente, el soporte para Portlets continúa como uno de los objetivos a corto o medio plazo de la versión 5 de este framework Web.

El soporte de Tapestry para Portlets podría facilitar considerablemente el desarrollo de la interfaz de usuario de los Portlets y permitir trabajar con plantillas HTML en lugar de tener que utilizar directamente páginas JSP. Éstas resultan tediosas de implementar ya que mezclan código Java y HTML, lo que dificulta su mantenibilidad.

A.4 Conclusiones

Los Portlets son una tecnología muy novedosa con una alta curva de aprendizaje, especialmente si no se tiene experiencia previa con otras tecnologías Web relacionadas como los Servlets, las JSP, la librería de *tags* de las JSP (JSTL), los servidores de portales Web, etc.

Es una tecnología completamente ligada a los portales Web. Los Portlets permiten integración y reutilización a nivel de interfaz de usuario en portales.

En la actualidad, no hay herramientas disponibles de forma gratuita para los desarrolladores, que simplifiquen el desarrollo de aplicaciones Portlet en la misma medida que ocurre con las aplicaciones Web convencionales. La mayoría de las herramientas todavía soportan únicamente la versión 1.0 del estándar JPS y no la nueva versión 2.0 que corrige carencias, fallos e incrementa notablemente la funcionalidad de la primera versión.

En el caso del proyecto, se ha utilizado únicamente el *plugin* Portal Pack para el entorno de desarrollo Eclipse, ya que éste era el entorno de desarrollo elegido para el proyecto.

En cuanto a las perspectivas de futuro de la tecnología de Portlets, resulta muy complicado de predecir en el actual estado emergente de la tecnología. La idea de los Portlets como servicios Web orientados a presentación que permiten reutilizar tanto la lógica de negocio como la interfaz de usuario, resulta muy atractiva. En el futuro se verá si este tipo de servicios Web se imponen como ocurrió en el pasado con los servicios Web tradicionales.

Anexo B

Análisis de tecnologías

En este anexo se presenta un análisis completo de las principales tecnologías utilizadas en el proyecto. La tecnología de Portlets tiene un anexo propio, el A. En la sección B.1 se describen los requisitos tecnológicos del sistema. En la sección B.2 se ofrece un listado completo de todas las tecnologías y herramientas utilizadas en el proyecto junto con una breve explicación de su función. La sección B.3 describe las principales tecnologías elegidas para satisfacer los requisitos tecnológicos del sistema.

B.1 Requisitos tecnológicos

Un primer requisito tecnológico es la necesidad de utilizar una plataforma de desarrollo estable y contrastada que soporte el paradigma de programación orientado a objetos predominante en la mayoría de aplicaciones actuales. También debe permitir la creación de aplicaciones Web modulares. Además, el sistema va a estar formado por varias capas o componentes que se han de coordinar entre si. Para facilitar su implementación, se consideró que sería también interesante utilizar algún framework o librería que facilite el desarrollo de aplicaciones en la plataforma elegida.

El segundo requisito tecnológico que surge a la vista del análisis del problema es la necesidad de almacenar información relativa al dominio del sistema. Se requiere guardar datos de los profesores, alumnos, asignaturas, grupos de docencia, horarios de tutorías, etc. Para ello, hace falta un repositorio permanente de información así como tecnologías que permitan acceder a ese repositorio desde un lenguaje de programación de alto nivel. En definitiva, se precisan tecnologías de soporte al almacenamiento de información.

El tercer requisito tecnológico es la necesidad de utilizar una tecnología Web orientada a servicios que facilite la integración en portales Web para que los

profesores puedan poner el sistema a disposición de sus alumnos de forma sencilla. Además, esta tecnología debe estar basada en estándares para incrementar la compatibilidad y estar soportada por la plataforma de desarrollo elegida.

El cuarto requisito tecnológico surge de la necesidad de desarrollar interfaces Web de usuario dinámicas. Se necesitan tecnologías abiertas y basadas en estándares que estén soportadas por cualquier navegador Web para maximizar la accesibilidad del sistema.

Por último, se requieren tecnologías que sirvan como entorno de ejecución del sistema. Dado que el sistema a construir es accesible vía Web, se necesita un servidor Web en el que residirá la aplicación. Este servidor debe soportar la tecnología concreta utilizada para desarrollar la aplicación Web.

B.2 Listado de tecnologías y herramientas utilizadas

Las tecnologías y herramientas utilizadas en el proyecto han sido las siguientes:

- Java EE [16] como plataforma de desarrollo.
- Java como lenguaje de programación del sistema.
- Portlets como tecnología para la aplicación de reserva de cita de tutoría utilizada por los alumnos.
- MySQL [28] como sistema gestor de base de datos relacional.
- Hibernate Framework [26] como herramienta de mapeo objeto-relacional.
- Hibernate Tools [48] como herramienta para la generación automática de clases Java del dominio.
- Spring Framework [27] como herramienta de desarrollo de aplicaciones para la plataforma Java EE.
- Apache Tapestry [24] como framework Web para el desarrollo de la interfaz de la aplicación Web de configuración utilizada por los profesores.
- Librerías Chenillekit [77] y Ioko [78] de Tapestry para ofrecer componentes AJAX [4] en la aplicación Web.
- Apache Tomcat [21] como contenedor Web.
- OpenPortal Portlet Container [20] como contenedor de Portlets.
- Liferay Portal Server [55] como servidor de portales para realizar pruebas con Portlets.

- Java Persistence API (JPA) [18] como API de persistencia para la plataforma Java EE.
- HTML como lenguaje de *markup* para las páginas Web de la aplicación.
- CSS como lenguaje para definir el estilo de las páginas Web de la aplicación.
- JavaScript como lenguaje de programación para mejorar la interactividad de la aplicación Web.
- JavaScript Object Notation (JSON) [57] como formato ligero para intercambio de datos.
- JavaServer Pages (JSP) [5] y JavaServer Pages Standard Tag Library (JSTL) [54] como tecnologías para la interfaz de la aplicación Portlet de reserva de cita de tutoría.
- Java Servlet [22] como tecnología subyacente de la aplicación Web.
- XML como lenguaje declarativo para los ficheros de configuración del sistema.
- DWR (Direct Web Remoting) [53] como librería que permite invocar métodos de clases Java ubicadas en el servidor Web desde código JavaScript ubicado en el cliente (navegador Web).
- Subversion como sistema de control de versiones.
- Maven [59] para gestionar y compilar el proyecto software.
- Eclipse como entorno de desarrollo para la implementación y depuración del sistema.
- JUnit [56] para las pruebas unitarias del sistema.
- JavaMail [58] para el envío de correos electrónicos desde la aplicación.
- Quartz [49] para la planificación de tareas en la aplicación.
- iTextPDF [50] para la generación y manipulación de documentos PDF desde la aplicación.
- SLF4J (Simple Logging Facade for Java) [51] con Log4J [52] para la generación de ficheros de *log* del sistema.
- Microsoft Visio, ArgoUML y ObjectAid UML Explorer para la elaboración de diagramas de análisis y diseño del sistema.
- Microsoft Project para la elaboración del diagrama Gantt con la planificación del proyecto.

- Edge Diagrammer para la representación de diagramas entidad-relación del modelo de datos del sistema.
- OpenOffice para las hojas de cálculo de control de esfuerzos, diario del proyecto, tareas, etc.
- LaTeX con el editor Kile [60] para la elaboración de la memoria.
- Mozilla Firefox, Google Chrome, Microsoft Internet Explorer y Apple Safari como navegadores Web con los que se ha probado la aplicación Web.
- Ubuntu Linux y Microsoft Windows como sistemas operativos.

En la sección B.3 se ofrece una explicación detallada de las tecnologías más relevantes en el proyecto de entre las anteriores. Se justifican las razones de su elección y se exponen otras tecnologías alternativas consideradas.

B.3 Tecnologías principales

En esta sección se describen las tecnologías utilizadas en el proyecto, explicando en qué consisten, su utilización en el contexto del proyecto y justificando su elección.

B.3.1 Java

Java es un lenguaje de programación orientado a objetos y multiplataforma. Es uno de los lenguajes más populares actualmente. Numerosas empresas, universidades y todo tipo de organizaciones usan este lenguaje de programación.

La elección de Java para la implementación del sistema se basó en varias razones: en la experiencia previa del alumno con este lenguaje obtenida durante la carrera y en que la tecnología de Portlets es un estándar de Java, por tanto, requería el uso de este lenguaje para la aplicación de reserva de cita de tutorías utilizada por los alumnos. Por ello, se decidió utilizar este lenguaje para el sistema completo.

B.3.2 MySQL

MySQL [28] es el sistema de gestión de base de datos relacional utilizado en el proyecto. Es muy popular en todo tipo de aplicaciones Web. En el caso de este proyecto se utiliza configurado con el motor transaccional InnoDB [29], que permite transacciones ACID (*Atomicity, Consistency, Isolation and Durability*), bloqueo de registros y respeta las restricciones de integridad referencial.

La elección de este gestor se basó en su contrastado correcto funcionamiento en numerosas aplicaciones Web disponibles en el mercado. También, como parte del aprendizaje del alumno, ya que anteriormente había utilizado el sistema gestor de base de datos Oracle.

B.3.3 Hibernate

Hibernate [26] es el mapeador objeto-relacional (ORM) por excelencia de la comunidad Java. Un mapeador objeto-relacional es una herramienta que facilita la conversión entre las entidades almacenadas en una base de datos relacional y las clases del dominio de la aplicación. Existen además herramientas como Hibernate Tools que permiten generar automáticamente, a partir de una base de datos y del fichero de configuración de Hibernate, las clases del dominio de la aplicación con las anotaciones requeridas.

El uso de Hibernate ofrece, entre otras, dos ventajas fundamentales. En primer lugar, proporciona independencia con respecto al sistema de gestión de base de datos relacional utilizado (MySQL, Oracle, Derby, PostGres, etc). De esta forma, se facilita el cambio de un sistema de base de datos relacional a otro (también relacional), que se puede hacer efectivo cambiando solamente tres líneas en el fichero XML de configuración de Hibernate, sin necesidad de modificar el código fuente.

En segundo lugar, proporciona un lenguaje de consultas para Java, el Hibernate Query Language (HQL), que es utilizado para las operaciones de acceso a datos. El uso de este lenguaje de consultas permite, por un lado, que la aplicación sea independiente de los distintos dialectos del lenguaje SQL existentes en el mercado, y por otro lado, simplifica la construcción de consultas, dado que éstas tienen una notación más cercana al modelo de objetos, de más alto nivel que el modelo de datos, y por tanto, más intuitivo para el desarrollador. Además, el uso de HQL con consultas que tienen parámetros con nombre protege al sistema contra determinados ataques comunes como SQL Injection [7].

La elección de Hibernate se basó en varias razones. La primera, en que el autor tenía interés personal en tomar contacto a nivel práctico con una herramienta que había estudiado previamente de forma teórica en asignaturas de la carrera. La segunda, en que el uso de Hibernate encajaba perfectamente en el tipo de sistema a desarrollar dado que utiliza una base de datos relacional y se desea que el sistema sea independiente del gestor concreto utilizado. Por último, se integra sin ningún problema con los otros frameworks utilizados, Spring y Tapestry.

B.3.4 Spring

Spring [27] es un framework de desarrollo de aplicaciones para la plataforma Java. Favorece la adopción de buenas prácticas de programación en Java. Empezó siendo un contenedor ligero de inversión de control (*Inversion of Control*, IoC) o inyección de dependencias (*Dependency Injection*, DI) [10]. La inversión de control es un principio de diseño que permite desacoplar la configuración y especificación de dependencias de la lógica de la aplicación. De esta forma, permite construir aplicaciones complejas a partir de componentes Java simples con bajo acoplamiento entre sí.

En la actualidad, Spring está compuesto por un amplio número de módulos que ofrecen numerosas posibilidades: programación orientada a aspectos (*Aspect Oriented Programming*, AOP); integración con diferentes proveedores de persistencia (Hibernate, JPA, JDO, TopLink, iBatis); desarrollo de aplicaciones Web siguiendo el patrón de diseño MVC, etc.

La elección de Spring se basó en razones similares a las de Hibernate: conocimiento previo a nivel teórico, adecuación al tipo de sistema a desarrollar y compatibilidad total con los otros frameworks a utilizar.

B.3.5 Tapestry

Tapestry [24] es un framework Web orientado a componentes, de forma que modela cada página Web como un componente que puede reaccionar a diversos eventos. Existen otro tipo de frameworks Web orientados a acción, es decir, a procesar peticiones HTTP individuales. Ejemplos de este tipo de frameworks Web son Spring MVC y Struts. Sin embargo, estos requieren que el desarrollador se ocupe de aspectos de más bajo nivel.

Los framework Web proporcionan soporte para aspectos típicos de las aplicaciones Web como: internacionalización, gestión de formatos, aplicación de políticas globales al procesamiento de peticiones HTTP, etc.

Las aplicaciones Web desarrolladas con Tapestry sólo requieren para su ejecución la API de Servlets. Esto implica que pueden utilizarse dentro de cualquier contenedor Web "ligero" como por ejemplo Tomcat o Jetty.

Algunas de sus características más relevantes son:

- Soporta la integración directa con Hibernate o a través de Spring.
- Validación de datos de entrada.
- Soporte para inyección de dependencias.
- Reutilización de elementos entre múltiples páginas de una aplicación Web.
- Contiene componentes para interacciones AJAX.
- Soporta el uso de anotaciones (enfoque POJO).
- Soporte para internacionalización a través de ficheros de propiedades.
- Sigue convenciones de nombrado.

La elección del Framework Tapestry se basó en su relativa baja curva de aprendizaje en comparación con otros frameworks Web similares como Struts, y en la necesidad de escribir menos código para cada página Web a desarrollar.

B.3.6 Tomcat

Tomcat [21] es un servidor Web multiplataforma con soporte para Servlets [22] y JavaServer Pages (JSP) [5]. Es frecuentemente referido como un contenedor Web "ligero" ya que, al contrario que los servidores de aplicaciones J2EE como JBoss, GlassFish, etc., no requiere tantos recursos para su ejecución.

La elección de Tomcat se basó en su utilización de pocos recursos y su integración con el contenedor de Portlets OpenPortal Portlet Container. Se consideró también la posibilidad de usar Jetty, otro servidor Web con soporte para Servlets. Sin embargo, la disponibilidad de mejor documentación para Tomcat hizo que la decisión se decantara a favor de éste.

B.3.7 OpenPortal Portlet Container

OpenPortal Portlet Container [20] es un contenedor de Portlets de código abierto que implementa la API de Portlets 2.0 (JSR-286) [42].

La elección de esta herramienta se basó en varias razones. Primero, en la posibilidad de instalarla en el servidor Web Tomcat muy fácilmente. En segundo lugar, en la posibilidad de incluir Portlets en una página JSP. Esto permitía desplegar el Portlet en la página Web del director del proyecto cumpliendo así uno de los requisitos del proyecto. Por último, también la posibilidad de elegir otros contenedores Web para instalar el contenedor de Portlets como JBoss, Jetty, Oracle WebLogic y GlassFish, se tuvo en cuenta para la elección. Todas estas posibilidades de instalación del contenedor se explican en su guía de usuario [8].

Anexo C

Diseño del sistema

En este anexo se ofrece información complementaria sobre el diseño del sistema desde una perspectiva cercana a la implementación.

C.1 Estructura de paquetes

La estructura de paquetes ayuda a comprender el diseño realizado y ver la correspondencia entre las capas del sistema y las clases de los diferentes paquetes. En la figura C.1 se puede observar la estructura de paquetes de la aplicación Web del sistema.

A continuación, se explica el contenido de cada subpaquete dentro del paquete de primer nivel *model*. Se ha denominado de esta forma dado que se corresponde con el modelo de la aplicación Web según el patrón Model View Controller (MVC). Dentro del diseño multicapa realizado, el modelo agrupa las capas de dominio, acceso a datos y lógica de negocio. El patrón MVC se explicará con más detalle en la sección C.4.1.

Los subpaquetes del modelo son:

- *dao*: contiene las interfaces Java de la capa de acceso a datos utilizando el patrón de diseño DAO. Cada DAO se puede ver como un gestor responsable de obtener los objetos de una entidad concreta de la base de datos.
  - *hibernate3*: contiene las clases que implementan las interfaces Java anteriores de la capa DAO utilizando el framework Hibernate (en su versión 3).
- *domain*: contiene las clases persistentes o entidades que conforman la capa de dominio.

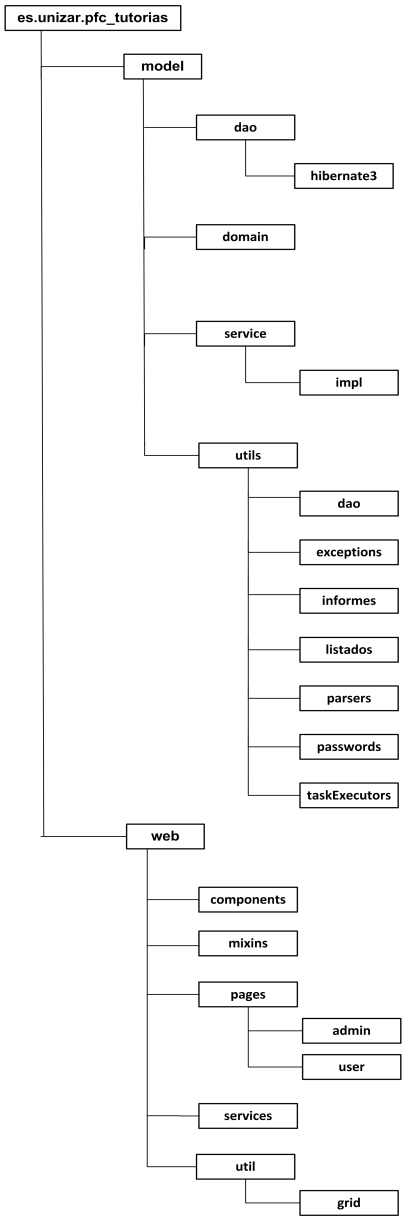


Figura C.1. Estructura de paquetes de la aplicación Web

- *service*: contiene las interfaces de componentes de la capa de lógica de negocio.
  - *impl*: contiene las implementaciones de las interfaces anteriores utilizando el framework Spring.
- *utils*: contiene clases de utilidad genérica empleadas por los componentes de la capa de lógica de negocio.

  - *dao*: contiene la interfaz genérica *GenericDao* y su implementación genérica *GenericDaoHibernate*.
  - *exceptions*: contiene clases de excepciones personalizadas para la aplicación como *InstanceNotFoundException* o *DuplicateInstanceException*.
  - *informes*: contiene las clases utilizadas para generar los informes de solicitudes de tutorías, históricos de solicitudes, incidencias y los informes resumen del curso académico.
  - *listados*: contiene clases que se utilizan para transformar y listar datos recuperados de la base de datos.
  - *parsers*: contiene los dos analizadores sintácticos que procesan los ficheros de festivos y de alumnos matriculados.
  - *passwords*: contiene las clases relacionadas con la generación automática de contraseñas y cifrado de las mismas utilizando la utilidad Unix *crypt*.
  - *taskExecutors*: contiene clases que ejecutan tareas en segundo plano (por ejemplo, envío de correos electrónicos de notificaciones).

A continuación, se detallan los subpaquetes del paquete de primer nivel *web*. Este paquete contiene clases y plantillas que implementan la capa de presentación de la aplicación Web. Las clases y plantillas de estos paquetes realizan las funciones de Controlador y Vista del patrón MVC, respectivamente.

Los subpaquetes de la capa de presentación son los siguientes:

- *components*: contiene las clases Java del framework Web Tapestry que actúan como controladoras para los componentes de las páginas. Por ejemplo, para el *Layout* de las páginas de los profesores y el *Layout* de las páginas del administrador.
- *mixins*: contiene la clase Java *Confirm* que invoca código JavaScript para mostrar mensajes de confirmación al usuario en ciertas pantallas.
- *pages*: contiene las clases Java de Tapestry que actúan como controladoras para las páginas Web comunes a todos los usuarios de la aplicación (Home, Index, etc).
  - *admin*: contiene las clases Java de Tapestry que actúan como controladoras para las páginas accesibles por el administrador.
  - *user*: contiene las clases Java de Tapestry que actúan como controladoras para las páginas accesibles por los profesores.
- *services*: contiene clases Java encargadas de gestionar aspectos de la aplicación como: *cookies*, políticas de autenticación para permitir o denegar el acceso a las diferentes páginas Web de la aplicación según el perfil del usuario (profesor o administrador), el *locale* de la interfaz de usuario (que define el idioma y el país), etc.

- *util*: contiene clases Java de utilidad genérica. Por ejemplo, clases para mantener la sesión del profesor y del administrador en el sistema, permitir descargar ficheros en formatos ZIP, PDF, texto, etc.
  - *grid*: contiene clases Java del componente Grid de Tapestry para diferentes pantallas de la aplicación según el tipo de datos a mostrar. El componente Grid de Tapestry permite mostrar datos en una tabla.

Además, por cada clase Java de los paquetes *components*, *pages*, *pages.admin* y *pages.user*, existen plantillas de Tapestry (ficheros con la extensión .tml, Tapestry Markup Language) que definen utilizando XML y HTML, el formato visual de las diferentes páginas Web de la aplicación.

Por último, la aplicación Portlet sólo tiene 2 paquetes o subsistemas con código Java:

- *portlet*: contiene la clase *PortletReservaTutorias* que hereda de *GenericPortlet*. Actúa como Controlador y utiliza el componente de los alumnos (*AlumnoService*) del Modelo.
- *utils*: contiene la clase *Constants* que define constantes de uso general en el Portlet.

En el directorio de recursos del proyecto correspondiente a la aplicación Portlet se encuentran todas las páginas JSP, que generan la parte visual de la interfaz de usuario del Portlet.

C.2 Componentes del sistema

En esta sección se describen los componentes principales del sistema.

C.2.1 Desglose de los componentes

A continuación, se enumeran los componentes principales del sistema. Estos componentes principales del sistema forman parte de la capa de lógica de negocio.

- *AdminService*: Implementa la funcionalidad del administrador del sistema.
- *AlumnoService*: Implementa la funcionalidad de los alumnos.
- *ProfesorService*: Implementa las operaciones de los profesores.
- *TutoriaService*: Implementa la funcionalidad relacionada con la gestión de tutorías por parte de los profesores.

- *StatsService*: Implementa la funcionalidad relacionada con las estadísticas generales del sistema.

- *InformeService*: Implementa la funcionalidad relacionada con los informes o notificaciones del sistema.

- *MantenimientoService*: Implementa las operaciones de mantenimiento del sistema (operaciones a realizar cuando cambia el curso académico, copias de seguridad del sistema, etc).

- *EnvioMails*: Permite enviar correos electrónicos utilizando la librería JavaMail. Envía tanto correos sin ficheros adjuntos como correos con ficheros adjuntos en formato PDF (para los informes o notificaciones).

- *GeneracionCalendarioSesionesTutorias*: Se encarga de calcular las sesiones de tutorías del curso académico correspondientes a un determinado horario de tutorías.

- *PasswordGenerator*: Genera contraseñas aleatorias para los alumnos del sistema.

- *PasswordEncrypter*: Componente para cifrar las contraseñas almacenadas en la base de datos del sistema utilizando el cifrado de la utilidad Unix *crypt*.

- *ParserFicheroAlumnos*: Componente que implementa un analizador sintáctico sencillo para procesar los ficheros de alumnos matriculados, importados en el sistema por los profesores.

- *ParserFicheroFestivos*: Componente utilizado para procesar sintácticamente los ficheros de festivos que no se corresponden con fechas fijas durante un curso académico.

- Componentes *TaskExecutor*: Conjunto de componentes que se encargan de crear una tarea o proceso en segundo plano para realizar operaciones que pueden ser costosas en tiempo. Ejemplos de operaciones costosas son: el envío de correos electrónicos a usuarios del sistema y la generación del calendario de sesiones de tutorías.

Todos los componentes del sistema están empaquetados en un fichero JAR. Esto permite que puedan ser utilizados tanto por la aplicación Web, como por la aplicación Portlet, incluyendo el fichero anterior como librería en el proyecto correspondiente a cada aplicación.

C.2.2 Interfaces de los componentes

Durante la fase de diseño se decidieron las operaciones de los componentes del sistema. Las operaciones de los componentes funcionales del sistema se pueden observar en el diagrama de clases de la figura C.2. Las operaciones de los componentes de soporte del sistema se pueden observar en el diagrama de clases de la figura C.3.

C.3 Diagrama de clases del dominio

En la figura C.4 se muestra el diagrama de clases del dominio. Para indicar las asociaciones entre clases no se han utilizado flechas ya que esto disminuye considerablemente la legibilidad del diagrama. En su lugar, las relaciones entre clases pueden obtenerse a partir de los atributos de cada clase. Todo atributo cuyo tipo sea de una clase del dominio (o un conjunto, *Set*, de una clase del dominio), se corresponde con una relación. Tampoco se han indicado las operaciones de cada clase ya que éstas son únicamente constructores y métodos de acceso a los atributos (*getters* y *setters*).

C.4 Patrones de diseño

Además del patrón de diseño DAO explicado en la sección 3.3.2, también se han aplicado otra serie de patrones en diferentes partes del sistema. En esta sección se presentan dichos patrones.

C.4.1 Model View Controller (MVC)

La aplicación Web sigue un diseño multicapa tal y como se ha explicado en el capítulo correspondiente. El diseño realizado, también se puede identificar con el patrón Model View Controller [2]. A continuación se explica este patrón y su correspondencia con el diseño realizado.

MVC es un patrón de presentación utilizado ampliamente en las aplicaciones Web. Divide la interacción con la interfaz de usuario en 3 roles distintos: el Modelo, la Vista y el Controlador.

El Modelo es la parte de la aplicación que representa información acerca del dominio del sistema. En el caso de la aplicación Web diseñada, el Modelo se corresponde con las capas de dominio, acceso a datos y lógica de negocio. Contiene toda la información (datos) y el comportamiento de la aplicación.

La Vista se corresponde con la representación del Modelo en la interfaz de usuario. Sólo se encarga de mostrar la información, no de gestionar los cambios

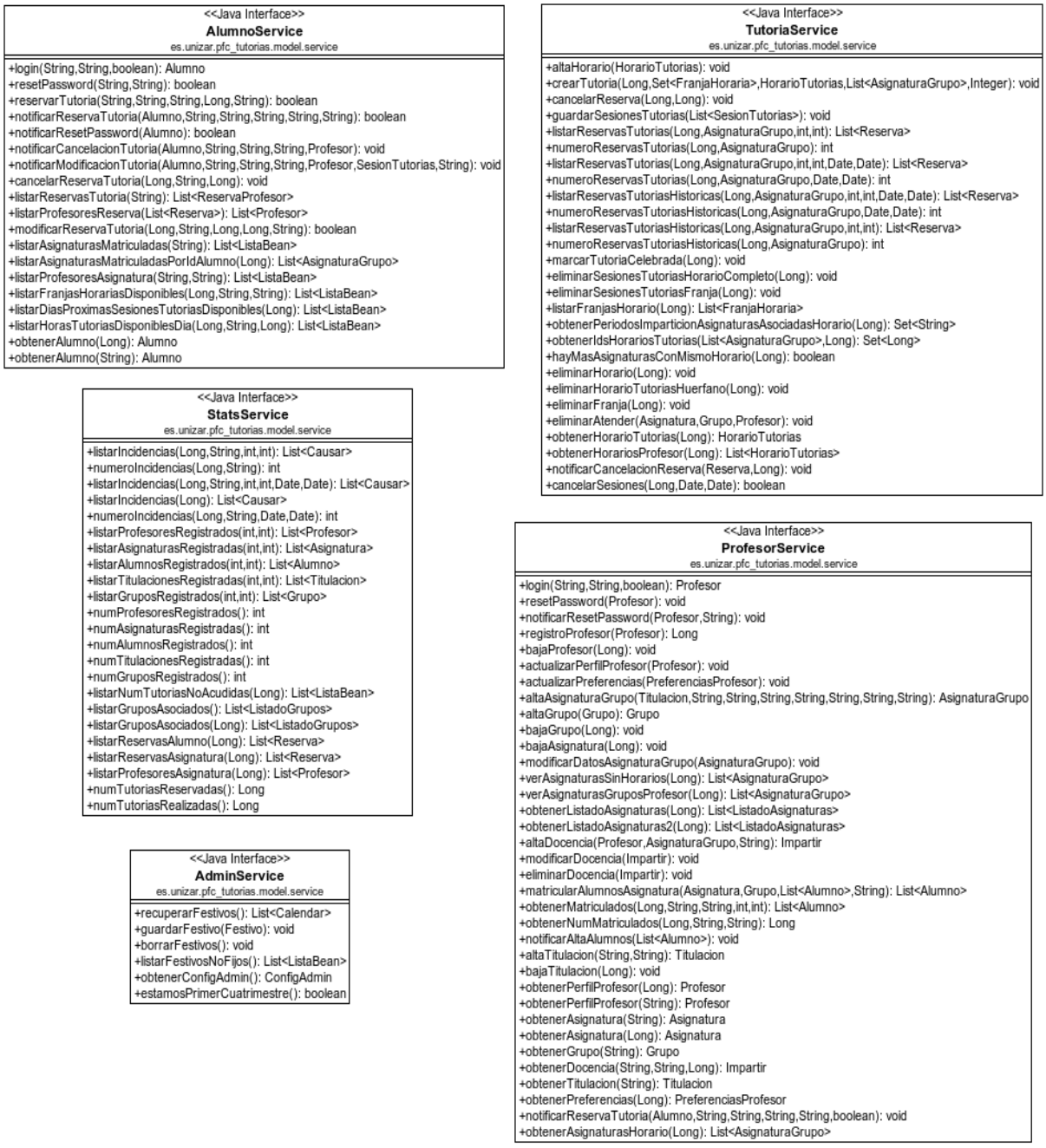


Figura C.2. Interfaces de los componentes funcionales del sistema

de dicha información, ya que esa es precisamente la tarea del tercer rol del patrón,

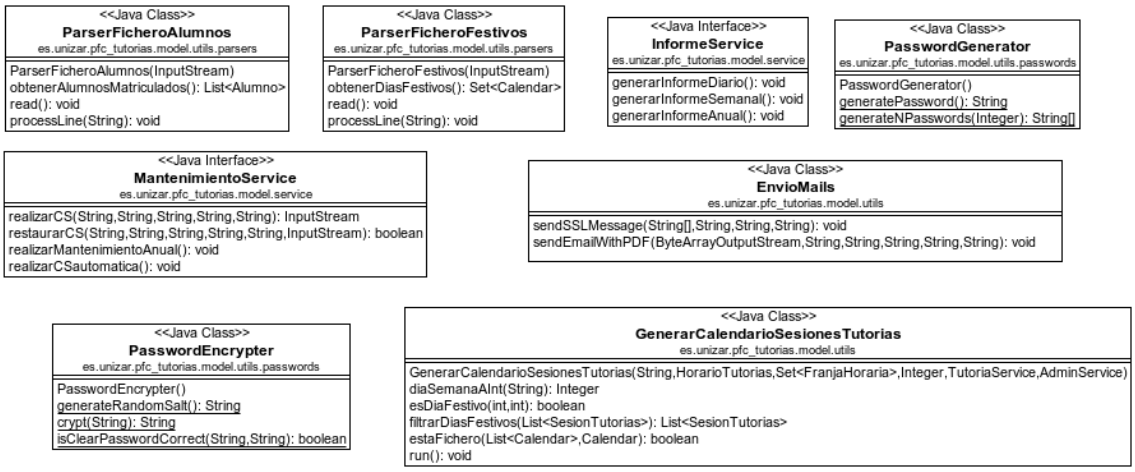


Figura C.3. Interfaces de los principales componentes de soporte del sistema

el Controlador.

El Controlador se encarga de recibir las acciones del usuario, manipular el Modelo y hacer que la Vista se actualice de forma apropiada. Por ello, la interfaz de usuario es una combinación de la Vista y el Controlador.

En este proyecto, en el caso de la aplicación Web, la Vista es generada a partir de plantillas del framework Web Tapestry. Las clases Java de Tapestry de la capa de presentación se encargan de recibir los eventos del usuario, actuando como Controlador. Tapestry se encarga de procesar adecuadamente estos eventos utilizando la API de Servlets.

En el caso de la aplicación Portlet, la Vista es generada a partir de páginas JSP que al ser compiladas se transforman en Servlets. La clase Java del Portlet (que hereda de la clase GenericPortlet) actúa como controladora, gestionando las peticiones de renderización y acción que llegan al Portlet a través del usuario.

Este patrón tiene 2 ventajas fundamentales:

1. Separación de la presentación y el Modelo: esta separación es una de las estrategias más importantes de diseño de software. En esencia, consiste en la separación de la interfaz de usuario y la lógica de las aplicaciones.

   La importancia de esta separación se explica por varias razones:

   - La presentación y el Modelo son cuestiones diferentes de un sistema. A la hora de desarrollar la Vista, el desarrollador se preocupa de mecanismos de la interfaz de usuario como por ejemplo, qué componentes visuales puede utilizar, y cómo se colocarán éstos, de forma que la interfaz

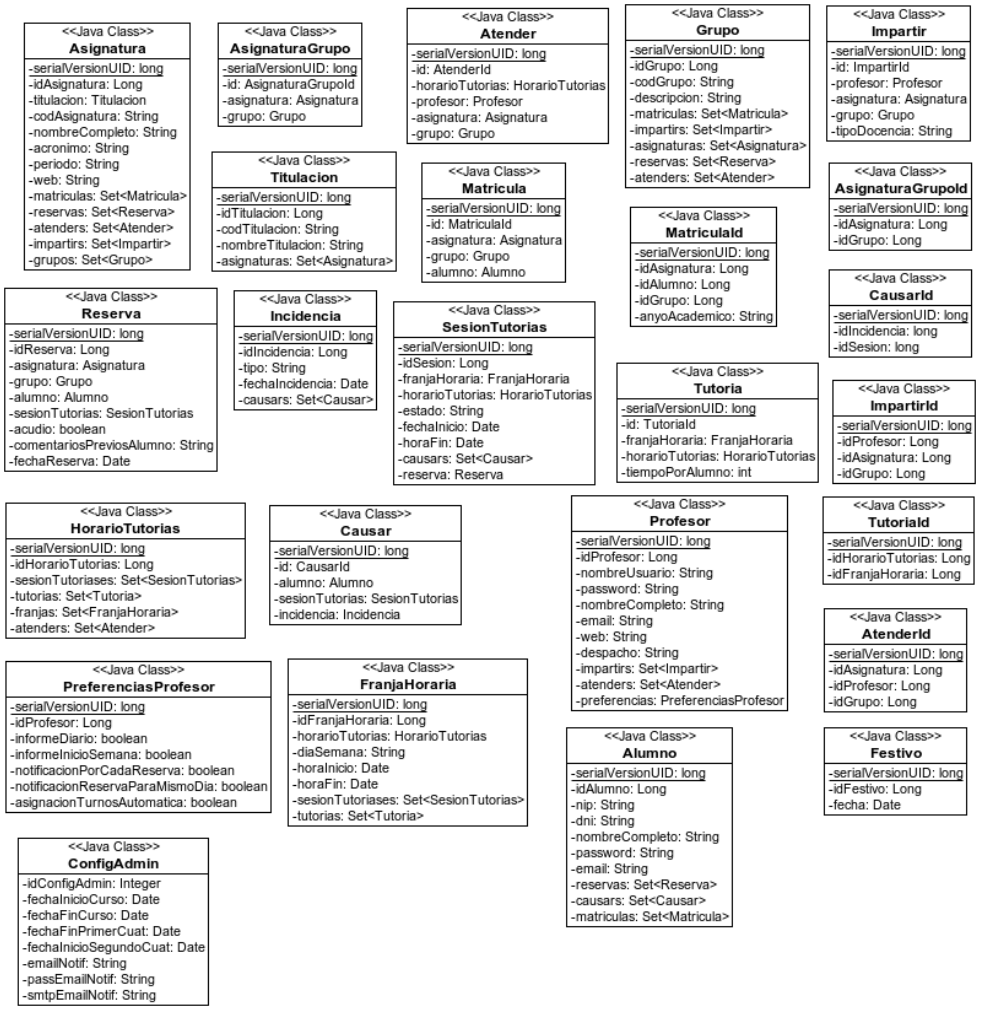


Figura C.4. Diagrama de clases detallado de la capa de dominio

sea amigable y respete principios básicos de usabilidad (consistencia, sencillez, mensajes de error informativos, etc). Sin embargo, a la hora de trabajar en el Modelo, el programador tiene que pensar en cuestiones como la lógica de negocio, interacciones con la base de datos del sistema, transacciones, etc. Además, típicamente, el diseño de la interfaz de usuario y la lógica de un sistema son tareas realizadas por personas con un perfil y formación diferentes.

- Permite desarrollar múltiples presentaciones (diferentes interfaces de usuario) utilizando el mismo código del Modelo. Dependiendo del contexto del sistema, la interfaz de usuario puede tomar formas muy diferentes. Es posible tener interfaces de usuario de línea de comandos,

gráficas de escritorio, interfaces Web, interfaces remotas, etc.

- Los objetos no visuales suelen ser más fáciles de probar de forma automatizada que los objetos visuales. Esta separación, facilita realizar pruebas sobre la lógica de negocio de las aplicaciones, sin tener en cuenta, en un primer momento, la interfaz de usuario.

Es importante mencionar que la presentación depende del Modelo, pero éste, en cambio, no depende de la presentación. De esta forma, resulta más fácil añadir nuevas interfaces de usuario más adelante, o cambiar las existentes sin alterar para ello el Modelo.

2. Separación del Controlador y la Vista: esta separación es bastante menos importante que la anterior. De hecho, en muchas implementaciones de este patrón no se lleva a cabo. Un ejemplo de utilidad podría ser el tener comportamiento editable y no editable en una interfaz. Esto se podría lograr con una única vista y 2 controladores para los 2 casos.

C.4.2 Patrones objeto-relacional (The ORM Patterns)

Los patrones objeto-relacional [2] se corresponden con un conjunto de patrones de diseño estructurales y de comportamiento. Están implementados en los mapeadores objeto-relacional que realizan la correspondencia entre objetos de un lenguaje de programación orientado a objetos y entidades de una base de datos relacional.

En este proyecto se ha utilizado el mapeador Hibernate. En la figura C.5 se muestra la arquitectura de una aplicación que utiliza Hibernate. En ella, se puede observar cómo los objetos persistentes (*Persistence Objects*), son accesibles a todo el resto de la aplicación que utiliza Hibernate. Estos objetos persistentes se corresponden con la capa de dominio del sistema, transversal al resto de capas. Hibernate también utiliza un fichero de configuración (*Hibernate.properties*), donde se especifican las propiedades de la base de datos relacional usada, y ficheros XML (*XML Mapping*) que realizan el mapeo entre las entidades de la base de datos y las clases persistentes. En el caso del proyecto, se han utilizado anotaciones en las clases en lugar de ficheros XML para facilitar el mantenimiento, al tener el código y las anotaciones en el mismo sitio.

A continuación se explican los patrones objeto-relacional utilizados en el proyecto.

C.4.2.1 Identity Field

El patrón Identity Field [2] es un patrón objeto-relacional de tipo estructural. Es muy sencillo de utilizar y entender. Se basa únicamente en guardar la clave primaria de las entidades de la base de datos como un campo de identidad (*Identity Field*)

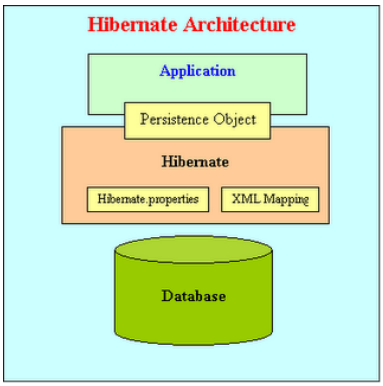


Figura C.5. Arquitectura de Hibernate

en el objeto persistente que representa a dicha entidad. De esta forma, se conserva la identidad entre el objeto en memoria y la tupla de la base de datos.

En el caso del proyecto, se ha utilizado en los objetos persistentes un campo de tipo *Long* para almacenar la clave primaria de las entidades. Para los objetos persistentes que representan relaciones de la base de datos, se ha utilizado una clase que contiene las claves primarias de todas las entidades que intervienen en la relación. Esto se puede observar en la figura C.4.

C.4.2.2 Foreign Key Mapping

El patrón Foreign Key Mapping [2] es también un patrón objeto-relacional de tipo estructural. Se utiliza para hacer corresponder las asociaciones entre objetos del dominio en claves ajenas (*foreign keys*) que hacen referencia a tablas de la base de datos.

Por ejemplo, a nivel de objetos, la clase *Asignatura* tiene una referencia a la titulación a la que pertenece. Cuando se cree una nueva asignatura y se almacene en la base de datos, esa referencia a la titulación se transformará en la clave primaria de la titulación que referencia, constituyendo una clave ajena.

C.4.2.3 Association Table Mapping

El patrón Association Table Mapping [2] es otro patrón objeto-relacional de tipo estructural. Se basa en utilizar una nueva entidad con claves ajenas para almacenar asociaciones Muchos a Muchos, dado que éstas no se pueden tratar utilizando el patrón anterior. Esta entidad, frecuentemente denominada *Link Table*, contendrá únicamente las claves primarias de las 2 entidades relacionadas.

Este tipo de tablas pueden tener o no corresponencia con objetos en memoria. Si la asociación tiene información acerca de la misma (atributos de la asociación),

se utilizará un objeto del dominio para representarla. Si no, se puede adoptar la decisión de no tener objeto correspondiente en memoria.

Por ejemplo, en el caso del proyecto, se utiliza la entidad Matricula para representar la relación *matricular* entre *Alumno* y la entidad agregada *Asignatura-Grupo* (ver figura 3.5 con el diagrama entidad-relación). Además, esta asociación contiene información adicional, en este caso, el curso académico en el que se produce la matrícula del alumno en la asignatura y grupo correspondiente. Por ello, en la capa de dominio del sistema, existe también la clase Matricula para representar datos de este tipo, tal y como se puede observar en el diagrama de clases de dominio de la figura C.4. Esta clase tiene un objeto (de la clase MatriculaId) con la clave primaria compuesta, formada por la clave primaria del *Alumno*, la de la *Asignatura*, la del *Grupo* y el curso académico.

C.4.2.4 Lazy Load

El patrón Lazy Load [2] es un patrón de comportamiento objeto-relacional utilizado ampliamente en el ámbito de los mapeadores como Hibernate. La idea básica de este patrón consiste en tener un objeto que no contiene todos los datos que se necesitan en un momento dado, sin embargo, sabe cómo conseguirlos.

Se utiliza para cargar datos de una base de datos a memoria. Cuando se carga en memoria un objeto a partir de los datos almacenados en una entidad de la base de datos, también se suelen cargar los objetos relacionados (a través de las entidades que tienen asociaciones con la que se carga). De esta forma, el desarrollador que usa el objeto no tiene que cargar explícitamente los objetos relacionados en el caso de que los necesite. Sin embargo, esto puede afectar considerablemente al rendimiento del sistema, ya que se puede acabar cargando en memoria listas enormes de objetos que luego no se lleguen a utilizar. Para solucionar esto se utiliza el patrón Lazy Load.

Este patrón no carga todos los objetos relacionados con el que se quiere cargar. En su lugar, deja una marca en la estructura del objeto, de forma que si se necesitan los objetos relacionados, estos puedan ser cargados en el momento exacto en el que se necesiten. La ventaja radica en que si no se requieren finalmente los objetos relacionados con el que se ha cargado, no existe penalización en el rendimiento tanto desde el punto de vista de la cantidad de memoria utilizada, como del tiempo de ejecución.

Hibernate utiliza por defecto esta estrategia (que denomina $LAZY$) para ciertos tipos de relaciones entre entidades. En concreto, se utiliza para todas las relaciones Uno a Muchos y Muchos a Muchos. En este caso Hibernate devuelve una colección (*List*, *Set*, etc.) para modelar la relación. Esta colección se inicializa (es decir, se recuperan las instancias relacionadas de la base de datos) cuando se invoca alguna operación de la colección (*size*, *contains*, iterar, etc).

Sin embargo, en el caso de las relaciones Muchos a Uno y Uno a Uno, se utiliza la estrategia denominada *EAGER* (por cuestiones de compatibilidad con el estándar de persistencia de Java, JPA). Con esta estrategia, cuando Hibernate recupera un objeto persistente, también recupera todos los objetos persistentes que presentan este tipo de asociación. Si en estas relaciones se especifica en su lugar la estrategia *LAZY* de aplazar la recuperación, Hibernate no devuelve el objeto relacionado completo, sino un *proxy* del mismo que contiene la clave primaria de la entidad. Estos *proxies* son generados dinámicamente en tiempo de ejecución. En el momento en que se invoque cualquier método sobre el *proxy* (excepto recuperar la clave primaria), Hibernate lo inicializará, es decir, recuperará el estado de la base de datos. Este es un caso de aplicación del patrón de diseño Proxy [25].

La documentación de Hibernate aconseja utilizar siempre la estrategia *LAZY* para todas las relaciones, para evitar que cada vez que se carga un objeto persistente en memoria se tengan que recuperar los relacionados, lo que podría desencadenar demasiadas consultas o consultas con muchos *JOIN* (costosas en tiempo de ejecución). Es decir, se aplaza la recuperación de la entidad hasta que realmente se necesite.

En el caso del proyecto, se ha utilizado la estrategia *LAZY* para todas las relaciones de la base de datos excepto para las de la entidad Reserva (de tutorías). En este caso, siempre que se recupera una reserva se accede a los datos relacionados: alumno que la realiza, asignatura, grupo y sesión de tutorías a las que corresponde. Por ello, tiene sentido que en este caso se utilice esta estrategia. Con ella, cuando se recupere un objeto persistente de la clase Reserva, también se recuperarán al mismo tiempo todos los objetos persistentes que presenten este tipo de asociación.

C.4.3 Open Session In View

El patrón Open Session In View [9] se suele utilizar en aplicaciones Web para solucionar un problema típico relacionado con el patrón Lazy Load. El problema se explica a continuación por medio de un ejemplo.

Supongamos que se quieren listar las asignaturas que imparte un profesor. La interfaz gráfica mostrará los datos más relevantes de cada asignatura (código, nombre, titulación, etc.) junto con la lista de alumnos matriculados. Al recuperar las asignaturas de la base de datos utilizando Hibernate, la relación entre asignatura y alumnos matriculados es Muchos a Muchos, luego se utilizará la estrategia *LAZY* por defecto. Cuando la interfaz de usuario necesite acceder a los alumnos de una determinada asignatura para mostrarlos, se encontrará con una colección sin inicializar. Según la estrategia *LAZY*, Hibernate recuperaría en este momento los datos de la base de datos. Sin embargo, la transacción anterior del caso de uso "listar las asignaturas que imparte un profesor" habrá terminado cuando se recupera la última asignatura. Por tanto, la sesión de Hibernate (la sesión en este caso es la combinación de la conexión a la base de datos y la transacción) estará cerrada. Como

consecuencia, Hibernate lanzará una excepción del tipo *LazyInitializationException* al intentar listar los alumnos.

Para solucionar esto se utiliza el patrón Open Session In View. La idea de este patrón consiste en que cada vez que llega una petición HTTP, se abre una sesión Hibernate que no se cierra hasta que termine de procesarse la petición. La lógica de negocio de la aplicación asume que la interfaz gráfica podrá navegar por las relaciones de entidades aunque éstas estén sin inicializar.

La implementación de este patrón realizada en el proyecto consiste en el uso de un filtro proporcionado por el framework Spring denominado *OpenSessionInView-Filter*. Este filtro, para cada petición HTTP que intercepta:

- Abre una sesión de Hibernate.
- Deja fluir la petición.
- Cierra la sesión de Hibernate.

El filtro se especifica en el descriptor XML de la aplicación Web, indicando que todas las peticiones HTTP lo utilicen (en el caso de la aplicación Portlet no es necesario ya que sólo lista las reservas y éstas utilizan la estrategia *EAGER*). Como inconveniente, toda petición abrirá una sesión de Hibernate que consume una conexión a la base de datos. Sin embargo, aquellas peticiones que no necesiten acceder a la base de datos consumirán una conexión innecesariamente. Para solucionarlo, se emplea un *proxy* de la base de datos real (*LazyConnectionDataSourceProxy*), proporcionado también por Spring. Este *proxy* retrasa la petición de conexión a la base de datos hasta el momento en que se intenta lanzar una consulta.

C.4.4 Page by Page Iterator

El patrón Page by Page Iterator [9] sirve para acceder a una lista grande de objetos de forma eficiente. Se utiliza para evitar recuperar de una sola vez una gran cantidad de datos de la base de datos.

Por ejemplo, se usa en los listados de la aplicación que devuelven un número potencialmente elevado de objetos dado que la lista completa no cabe en memoria ni en pantalla. Por ello, se trocea la lista y se va recuperando por trozos de la base de datos. Cada página muestra únicamente un trozo.

Para implementar este patrón se utiliza el componente Grid del framework Web Tapestry. Este componente muestra, en una tabla por página, el número de datos que el programador especifique. Además, de forma automática crea enlaces a cada página de datos para que el usuario pueda ir hacia delante y hacia atrás visualizando el resto de la información. Se recuperan tantos datos como se especifica que van a aparecer en cada página, de forma que en un momento dado no se tengan todos los datos cargados en memoria, sino sólo aquellos que se están visualizando en la pantalla.

Anexo D

Manual de Usuario

Este anexo incluye el manual de usuario del sistema de gestión de tutorías IGETUTOR. El manual se divide en cuatro secciones. La sección D.1 presenta los requisitos necesarios para utilizar el sistema. La sección D.2 se corresponde con el manual del profesor. La sección D.3 corresponde al manual del alumno. Por último, la sección D.4 es el manual del administrador.

D.1 Requisitos generales para utilizar el sistema

Para acceder al sistema, únicamente es necesario tener instalado un navegador Web en su ordenador.

IGETUTOR ha sido probado con los siguientes navegadores Web:

- Google Chrome 8.0 y 9.0
- Mozilla Firefox 3.6.13
- Microsoft Internet Explorer 8.0
- Apple Safari 5.0.3

Puede utilizar cualquiera de los 4 navegadores anteriores.

Además, es necesario tener JavaScript activado en su navegador Web. Consulte el manual correspondiente a su navegador para obtener información sobre cómo activarlo. Por defecto, la mayoría de los navegadores Web tienen JavaScript activado, de modo que si no lo ha desactivado debería poder utilizar el sistema.

D.2 Manual del Profesor

D.2.1 Primeros pasos

Para comenzar a utilizar el sistema, abra una ventana de su navegador Web preferido. Una vez abierta, introduzca la siguiente dirección: `http://alkaid.cps.unizar.es:8280/igetutor/`. Se recomienda guardar esta dirección en los favoritos del navegador. Al cargarse la dirección abierta, aparecerá la pantalla que se observa en la figura D.1.

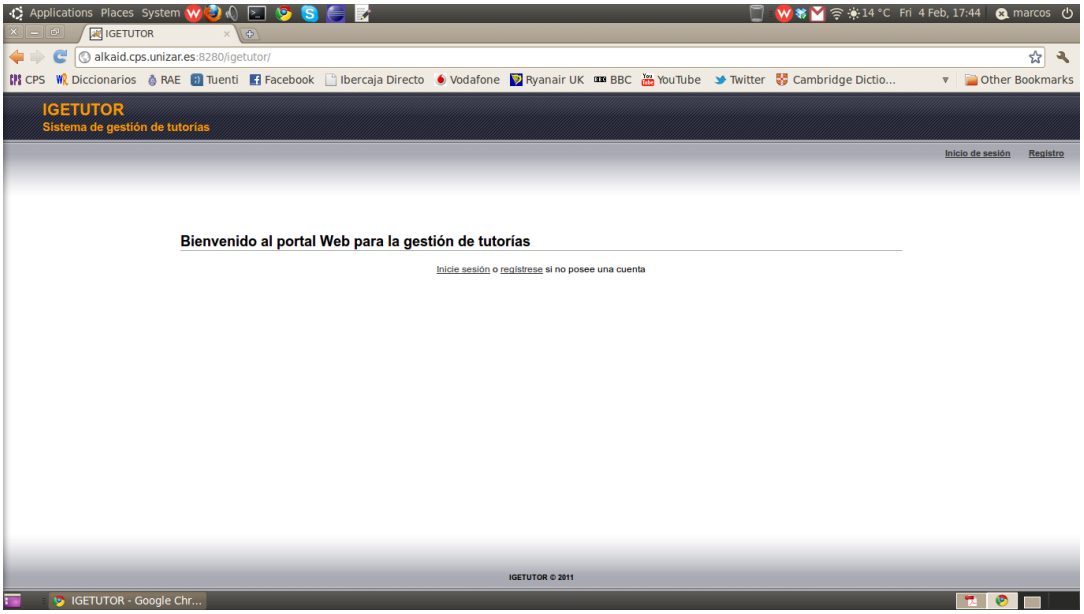


Figura D.1. Pantalla inicial de la aplicación Web

Una vez se encuentre en la pantalla inicial, el primer paso será registrarse. Para ello, pulsaremos en el enlace *Registro* en la parte superior derecha de la pantalla. Al pulsar en este enlace se solicitará un nombre de usuario y contraseña para poder registrarse (solicitar la contraseña al director del proyecto). Tras introducir los datos correctamente, aparecerá la pantalla que se observa en la figura D.2.

En esta pantalla debe introducir todos los datos requeridos por el formulario. En todos los formularios se ha utilizado la misma regla: señalar los campos opcionales en lugar de los campos requeridos, dado que éstos son mayoría en todos los formularios del sistema. Para señalar los campos opcionales se indica textualmente después del nombre del campo con la palabra *opcional* entre paréntesis. Se puede observar en el caso del campo correspondiente a la página Web del formulario de registro.

La contraseña escogida debe tener entre 4 y 8 caracteres. Como consejo de seguridad, trate de escoger una contraseña lo suficientemente segura, donde se combinen letras minúsculas, mayúsculas y números.

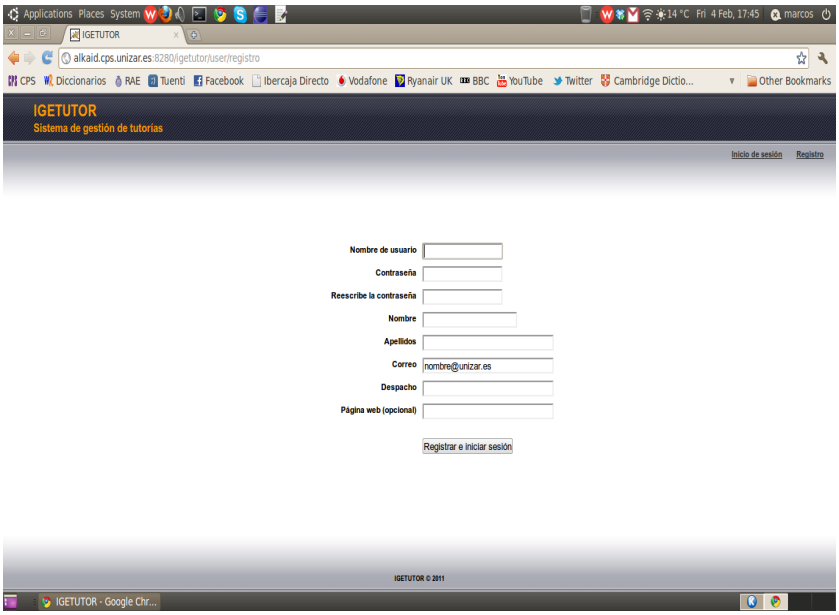


Figura D.2. Pantalla de registro

Es importante que compruebe que la dirección de correo electrónico introducida es correcta y que tiene acceso a ella, ya que será utilizada por el sistema para diferentes tareas: envío de las notificaciones e informes de tutorías, envío de una nueva contraseña en caso de olvido, etc.

Una vez introducidos todos los datos de forma correcta en el formulario de registro, pulsaremos en el botón *Registrar e iniciar sesión*. De esta forma, estaremos registrados en el sistema. Además, el sistema nos conducirá a la pantalla principal que se puede observar en la figura D.3. Esta pantalla contiene todas las opciones disponibles para el profesor en menús organizados por categorías.

En lo sucesivo, una vez registrados en el sistema, para iniciar sesión en el mismo, pulsaremos en la opción *Inicio de sesión*, situada en la zona superior derecha de la pantalla inicial. Aparecerá la pantalla de la figura D.4. Una vez ahí, introduciremos nuestros datos de inicio de sesión, es decir, nuestro nombre de usuario y contraseña. En esta pantalla existe la posibilidad de solicitar que nuestro navegador recuerde nuestros datos de inicio de sesión. Para ello, debemos tener activadas las *cookies* en el navegador. De esta forma, al volver a cargar la dirección Web de IGETUTOR, el sistema usará las *cookies* almacenadas en nuestro navegador para iniciarnos sesión automáticamente. Aparecerá la pantalla principal con las opciones de menú.

En el caso de que hayamos olvidado nuestra contraseña, en la pantalla de inicio de sesión debemos pulsar en el enlace *Olvidé mi contraseña*. Se mostrará la pantalla de la figura D.5. En esta pantalla, se solicita nuestro nombre de usuario. En caso de que haya olvidado también su nombre de usuario, contacte con el administrador

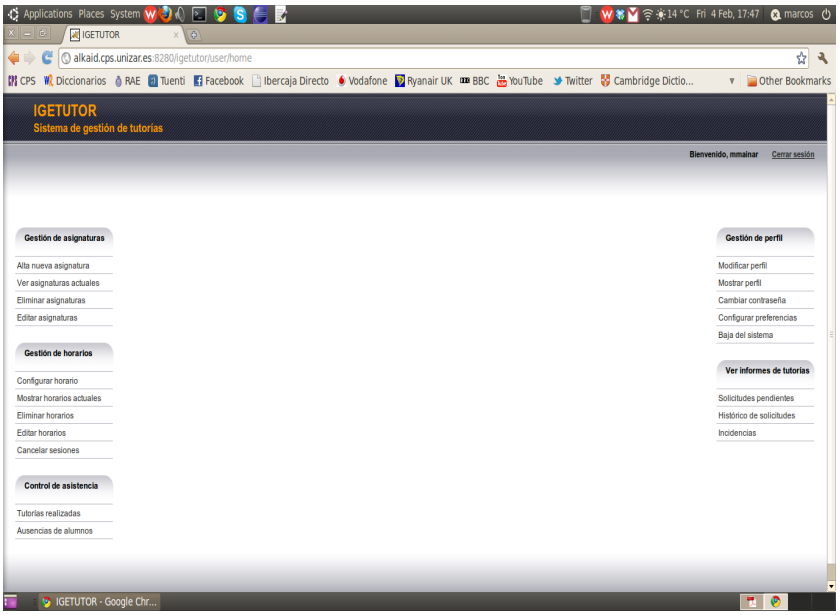


Figura D.3. Pantalla principal del profesor

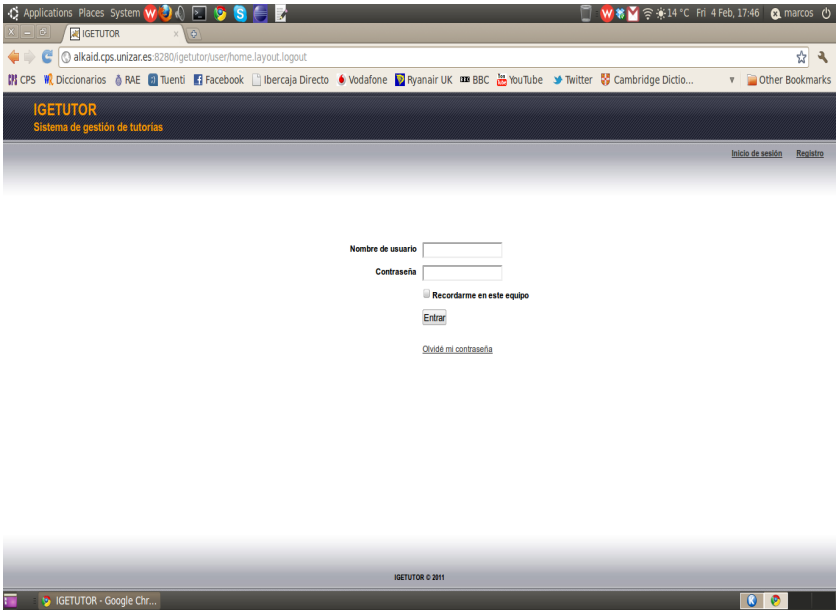


Figura D.4. Pantalla de inicio de sesión

del sistema quien se lo indicará. A partir del nombre de usuario, el sistema enviará una nueva contraseña a la cuenta de correo electrónico asociada.

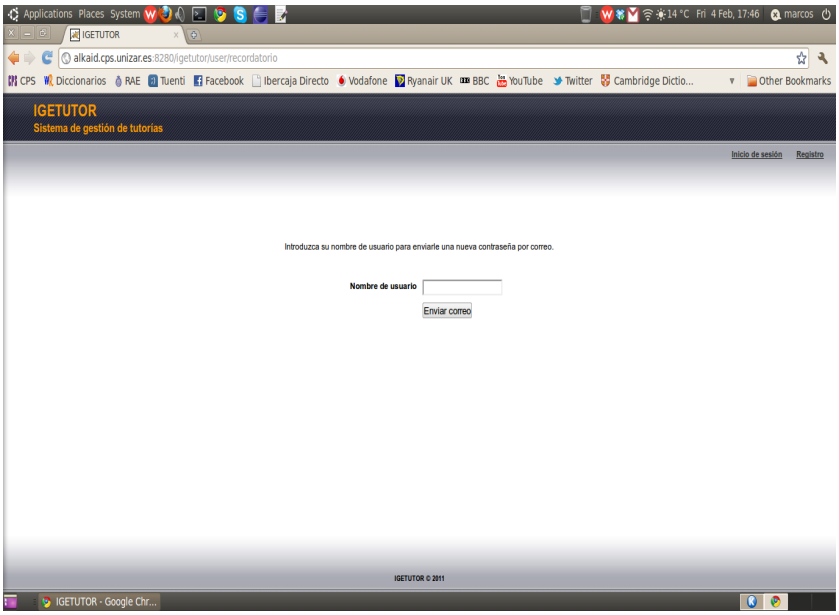


Figura D.5. Pantalla de olvido de contraseña

Los primeros pasos a realizar una vez que haya iniciado sesión en el sistema por primera vez serán:

1. Configurar sus asignaturas.
2. Configurar sus horarios de tutorías.
3. Configurar sus preferencias.

D.2.2 Configuración de asignaturas

Antes de configurar las asignaturas, debemos asegurarnos de que podemos descargarnos del sistema de la Universidad el fichero de alumnos matriculados en las asignaturas que queramos dar de alta. No es posible configurar las asignaturas sin tener el listado de alumnos matriculados.

Una vez haya iniciado sesión en el sistema por primera vez, el primer paso será configurar las asignaturas para las cuales quiere gestionar sus tutorías de forma automática. Para ello, en el menú de Gestión de asignaturas pulsaremos en la opción *Alta nueva asignatura*. Esto nos mostrará la pantalla que aparece en la figura D.6. En ella aparece un formulario donde introducir los datos de la asignatura a dar de alta.

El primer paso es introducir el código de la asignatura. Si la asignatura ya estaba registrada previamente en el sistema (por otro profesor), al introducir el código en su

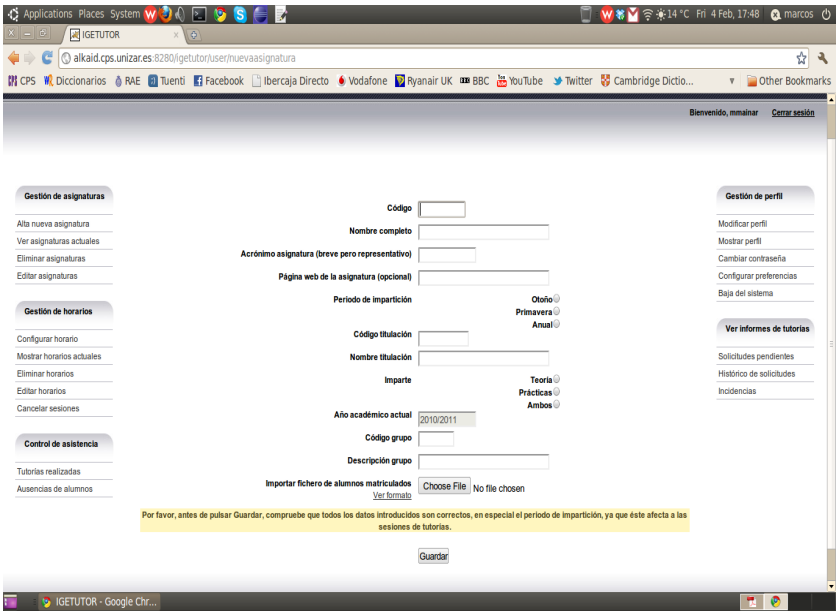


Figura D.6. Pantalla de alta de asignatura

correspondiente campo y cambiar de campo en el formulario, el sistema recuperará el resto de datos de la asignatura y rellenará de forma automática la mayoría de los campos. Sin embargo, todavía deberemos rellenar el tipo de impartición que realizamos (teoría, prácticas o ambos) y los datos del grupo de la asignatura que impartimos. De nuevo, si el grupo ya estuviera registrado, aparecerá la descripción que éste tuviera asociada. En el caso de que imparta más de un grupo de una misma asignatura, se tratará como 2 asignaturas diferentes, es decir, habrá que dar de alta ambos de forma independiente. Será necesario volver a repetir estos pasos con la asignatura e introducir el otro grupo. Al haber registrado previamente la asignatura, ahora sólo tendremos que introducir los datos del grupo y el tipo de impartición que realizamos.

Por último, introduciremos la ruta del fichero de alumnos matriculados en la asignatura. El formato de este fichero se puede consultar pulsando en el enlace *Ver formato* (luego podemos volver atrás, al formulario que estábamos rellenando, pulsando el correspondiente botón del navegador). Este fichero se creará de forma muy sencilla a partir del fichero de alumnos matriculados que nos hayamos descargado del sistema de la Universidad, tal y como se indica en las instrucciones al pulsar el enlace *Ver formato*.

D.2.3 Configuración de horarios de tutorías

Una vez haya configurado las asignaturas para las que quiere utilizar el sistema, el siguiente paso es configurar sus horarios de tutorías. El sistema permite flexibilidad

de forma que es posible tener:

- Un horario de tutorías distinto para cada asignatura configurada.
- El mismo horario para todas las asignaturas configuradas.
- Horarios diferentes para grupos de asignaturas (por ejemplo, un horario para las asignaturas del primer cuatrimestre y otro para las del segundo).

Para configurar los horarios de tutorías, iremos al menú Gestión de horarios y pulsaremos en la opción *Configurar horario*. Se mostrará la pantalla que aparece en la figura D.7.

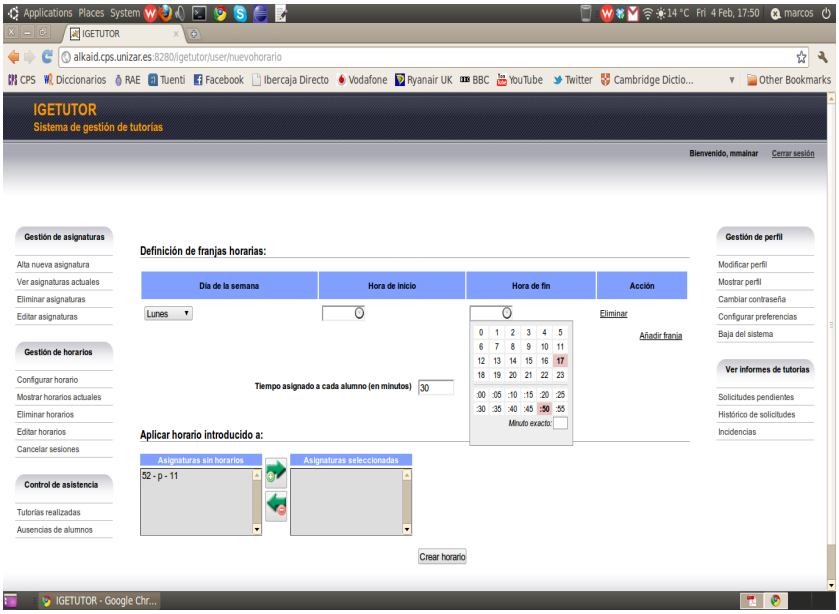


Figura D.7. Pantalla de configuración de nuevo horario de tutorías

En esta pantalla, primero hay que introducir las franjas del horario de tutorías. Inicialmente aparece una única franja y la opción de añadir nuevas franjas.

Para cada franja que se quiera añadir, se seleccionará el día de la semana del menú desplegable. A continuación, se seleccionará la hora de inicio y la hora de finalización de la franja. Para ello, pulsaremos en el icono del reloj donde se elige en primer lugar la hora. A continuación se mostrará la hora pulsada y el minuto actual. Para cambiar los minutos, pulsaremos de nuevo en el reloj y seleccionaremos los minutos. Alternativamente, también se puede introducir directamente la hora escribiéndola en formato *hh:mm* (siempre la hora y los minutos con 2 cifras y separadas por el carácter dos puntos).

Para crear una nueva franja, pulsaremos en el enlace *Añadir franja*. Aparecerá una nueva franja donde introducir sus datos. Si se desea eliminar una franja concreta, pulsaremos en el enlace *Eliminar* de la franja concreta.

Una vez configuradas todas las franjas del horario de tutorías, introduciremos el tiempo (en minutos) que deseamos asignar a cada tutoría. Por defecto, aparece 30 minutos pero se puede cambiar a cualquier otro valor (siempre que no sea mayor que la duración de la franja más corta del horario).

Por último, seleccionaremos las asignaturas a las que queremos aplicar el horario de tutorías introducido. Para ello, se utiliza un componente que contiene 2 cuadros de texto. En el de la izquierda aparecen las asignaturas que hayamos configurado y que todavía no tienen horario de tutorías asignado. Podemos elegir entre estas asignaturas a cuáles queremos aplicar el horario haciendo doble click en el nombre de la asignatura o bien pulsando el botón que muestra la flecha apuntando hacia la derecha. También podremos cambiar asignaturas que habíamos elegido previamente utilizando la flecha que apunta hacia la izquierda o haciendo doble click en la asignatura en el cuadro de la derecha. Las asignaturas que queden finalmente en el cuadro de la derecha serán las que tengan el horario introducido.

Finalmente, pulsaremos el botón *Crear horario* para configurar el horario definitivamente.

D.2.4 Configuración de las preferencias

Una vez haya configurado sus horarios de tutorías, es recomendable configurar sus preferencias. Para ello, pulse en la opción *Configurar preferencias* del menú Gestión de perfil. Aparecerá la pantalla que se muestra en la figura D.8.

Primero, configuraremos los tipos de informes y notificaciones de solicitudes de tutorías que deseamos recibir por correo electrónico (se envían en un fichero adjunto en formato PDF con el fin de facilitar su impresión). Al registrarnos en el sistema, se añaden dos tipos de notificaciones automáticamente: el sistema le enviará un informe diario si hay tutorías reservadas para ese mismo día y también le enviará una notificación si se reservan tutorías durante el día actual que vayan a tener lugar ese mismo día. Sin embargo, estas opciones podrán ser personalizadas según sus preferencias. Además de las opciones anteriores, el sistema también permite recibir una notificación individual por cada nueva tutoría solicitada y un informe el primer día de la semana con todas las tutorías pendientes de la semana.

Después, configuraremos si se quiere que los alumnos puedan elegir hora de tutoría entre las disponibles, o si preferimos que el sistema asigne automáticamente una hora a cada alumno con el fin de que las tutorías asignadas tengan lugar de forma consecutiva (y así evitar huecos libres).

Una vez seleccionadas todas las preferencias, pulsaremos en el botón *Guardar preferencias* para almacenarlas.

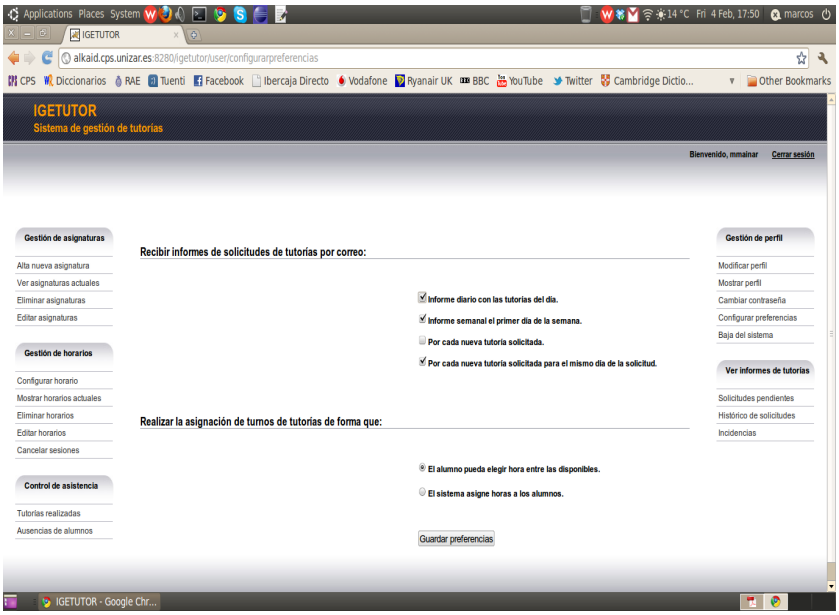


Figura D.8. Pantalla de configuración de las preferencias

D.2.5 Gestión de la asistencia a las tutorías

El sistema permite registrar si las tutorías solicitadas han tenido lugar o no. Para ello, hay que pulsar en la opción *Tutorías realizadas* del menú Control de asistencia. Aparecerá una pantalla como la que se muestra en la figura D.9. En ella, aparece una tabla con las tutorías realizadas hasta la fecha y una columna donde indicar si el alumno acudió a la tutoría o no. Una vez indicadas las tutorías realizadas, pulsaremos en el botón *Guardar* para que el sistema almacene la información introducida.

D.2.6 Consulta de ausencias de alumnos

Para consultar los alumnos que no acuden a las tutorías solicitadas, pulsaremos en la opción *Ausencias de alumnos* del menú Control de asistencia. Aparecerá una pantalla como la que se muestra en la figura D.10. En ella aparece un listado con aquellos alumnos que no han acudido a alguna tutoría y el número de tutorías a las que no han acudido.

D.2.7 Ver informes de tutorías

El sistema permite consultar informes de las solicitudes pendientes, el histórico de solicitudes y las incidencias de los alumnos. Además, también ofrece la opción

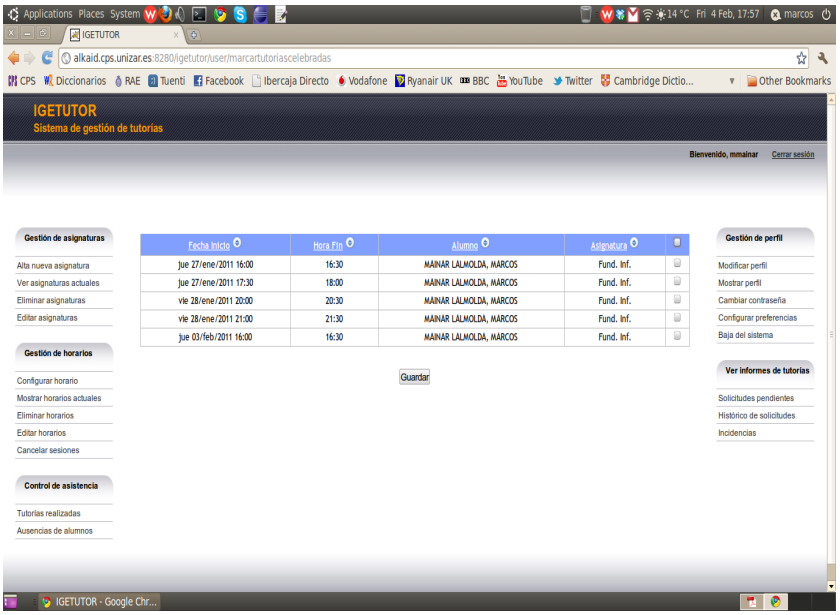


Figura D.9. Pantalla de gestión de la asistencia a las tutorías

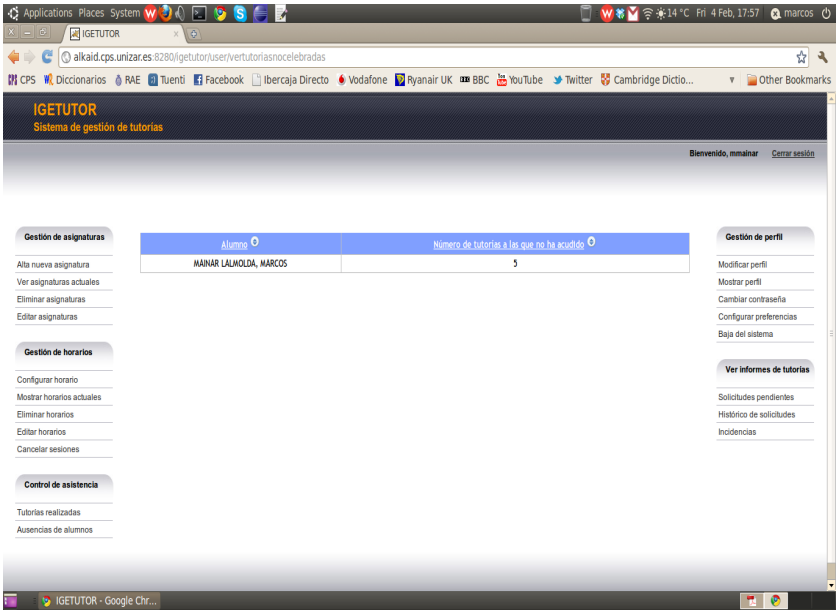


Figura D.10. Pantalla de visualización de las ausencias de alumnos

de descargar un fichero en formato PDF con el informe consultado con el fin de facilitar su impresión. Para ello, aparece el enlace *Descargar informe en PDF*. En la

figura D.11 se muestra un ejemplo con el listado de solicitudes de tutorías pendientes.

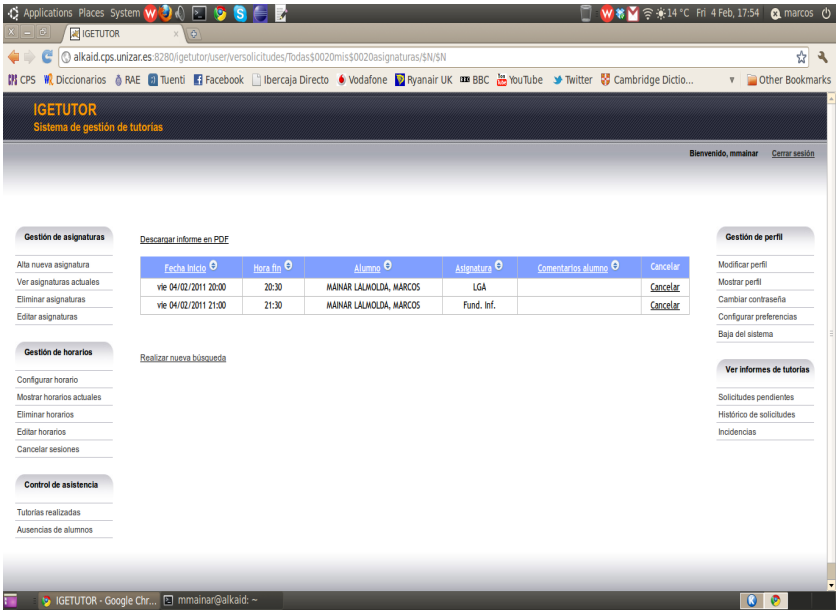


Figura D.11. Pantalla de visualización de las solicitudes de tutoría pendientes

Las solicitudes pendientes y el histórico de solicitudes se pueden consultar utilizando 2 criterios: la asignatura y un rango de fechas. Para visualizar todas las solicitudes, simplemente dejaremos la opción por defecto en el campo Asignatura (Todas mis asignaturas) y no introduciremos ninguna fecha. En el listado de solicitudes pendientes existe la posibilidad de cancelar determinadas tutorías. Para ello, basta con pulsar el enlace *Cancelar* de la tutoría concreta. El sistema enviará una notificación por correo electrónico al alumno informándole de la cancelación de la tutoría.

Las incidencias (modificaciones o cancelaciones de tutorías llevadas a cabo por los alumnos) pueden consultarse también utilizando 2 criterios: el tipo de incidencia (modificación o cancelación) y el rango de fechas en el que tuvo lugar la incidencia.

D.2.8 Cancelar sesiones

El sistema también ofrece la opción de cancelar sus sesiones de tutorías entre 2 fechas determinadas a través de la pantalla que se muestra en la figura D.12. Para ello, seleccionaremos la opción *Cancelar sesiones* del menú Gestión de horarios. Aparecen 2 campos de tipo fecha donde podemos seleccionar en un calendario desplegable las fechas entre las cuales queremos cancelar nuestras sesiones. Una vez introducidas las 2 fechas, hay que pulsar en el botón *Cancelar*. Si hubiera tutorías reservadas para alguna de las sesiones canceladas, se enviará un correo electrónico al alumno informándole de la cancelación de la tutoría.

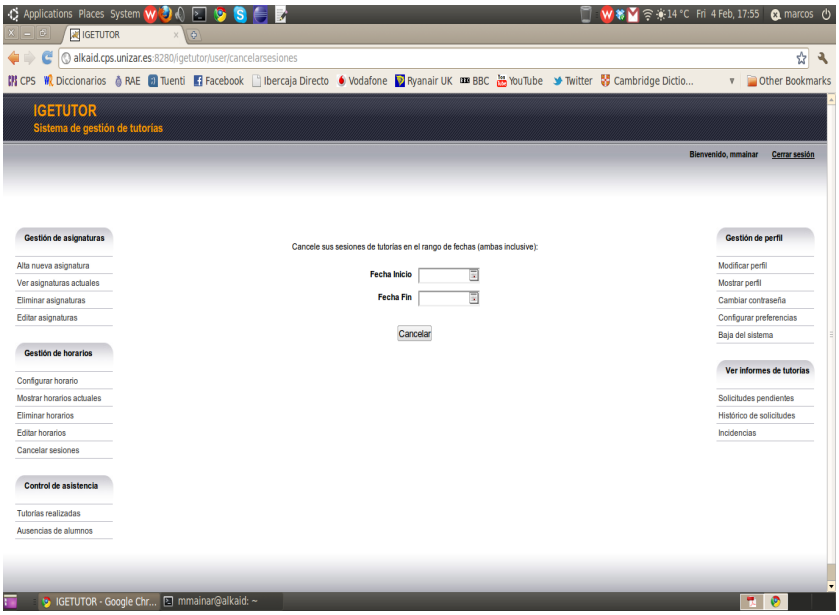


Figura D.12. Pantalla de cancelación de sesiones de tutorías

D.2.9 Ver asignaturas configuradas

Para ver las asignaturas configuradas pulsaremos en la opción *Ver asignaturas actuales* del menú Gestión de asignaturas. Se mostrará la pantalla que aparece en la figura D.13. En ella, aparece una tabla con las asignaturas. Desde esta tabla se puede consultar el listado de alumnos matriculados en cada asignatura pulsando en el número de alumnos que aparece en la columna *Alumnos* de la asignatura.

D.2.10 Eliminar asignaturas configuradas

Para eliminar asignaturas configuradas (en el caso de que haya dejado de impartirlas), pulsaremos en la opción *Eliminar asignaturas* del menú Gestión de asignaturas. Se mostrará la pantalla que aparece en la figura D.14. En ella, se muestran en una tabla todas las asignaturas configuradas. Para eliminar una asignatura, basta con pulsar el enlace *Eliminar* de la asignatura correspondiente. El sistema pedirá confirmación antes de proceder a eliminar la asignatura de forma definitiva.

D.2.11 Editar asignaturas configuradas

Para editar asignaturas configuradas, pulsaremos en la opción *Editar asignaturas* del menú Gestión de asignaturas.

Los datos que se pueden editar de una asignatura son:

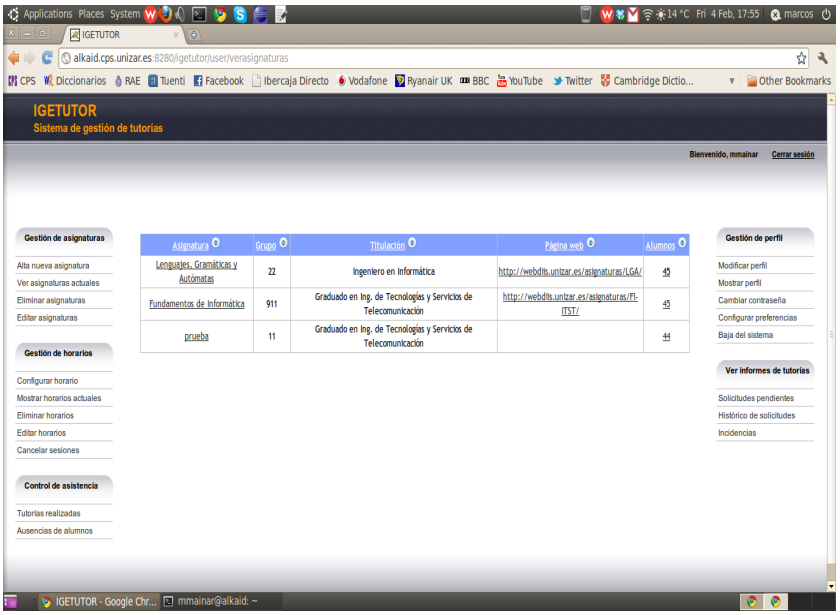


Figura D.13. Pantalla de visualización de las asignaturas configuradas

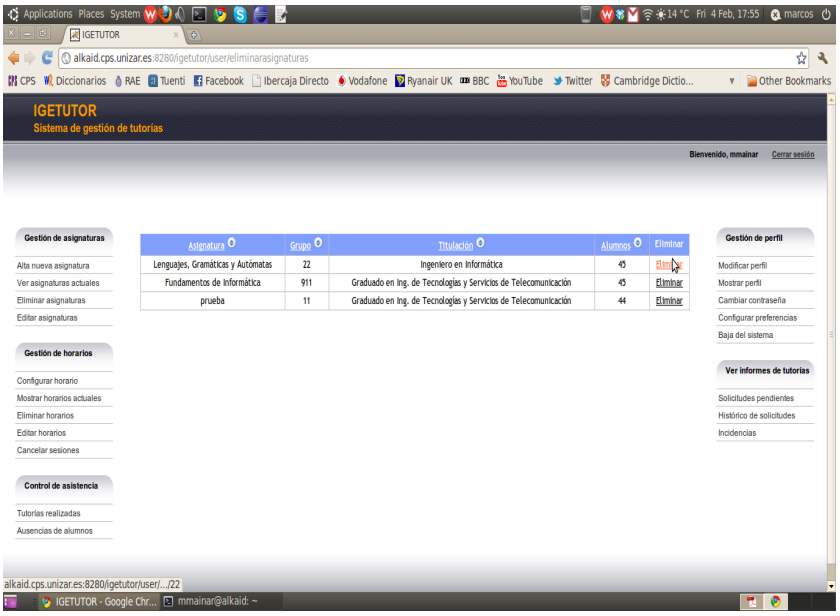


Figura D.14. Pantalla para eliminar asignaturas configuradas

- El acrónimo.
- La página Web.

- El tipo de impartición que realizamos (teoría, prácticas o ambos).
- El grupo y la descripción del mismo.
- Los alumnos matriculados.

Esta opción se debe usar para importar en cada curso académico a los nuevos alumnos que estén cursando la asignatura.

Al editar asignaturas se mostrará la pantalla que aparece en la figura D.15. En ella se muestran en una tabla todas las asignaturas que tuviéramos configuradas. Para editar una asignatura, basta con pulsar el enlace *Editar* de la asignatura correspondiente. El sistema mostrará el mismo formulario de configuración de una nueva asignatura con los datos de la asignatura ya introducidos en los campos correspondientes.

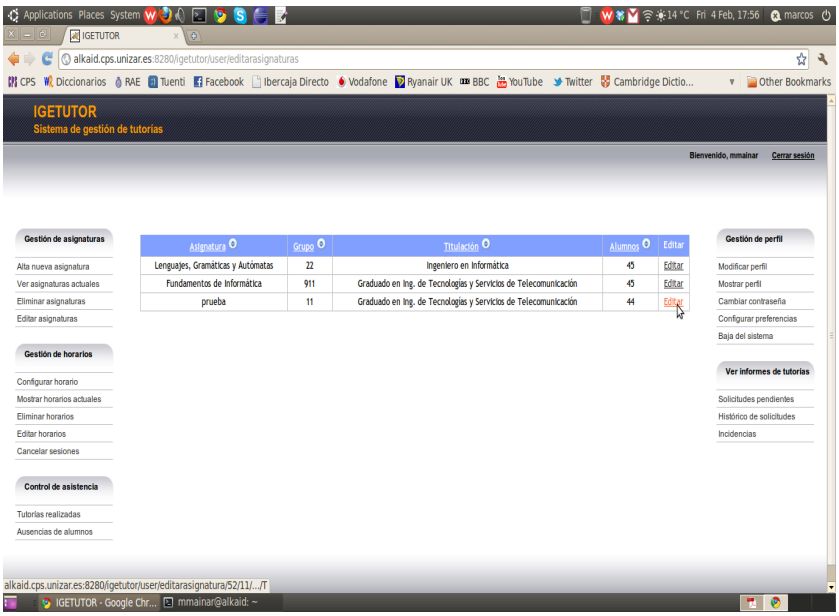


Figura D.15. Pantalla de edición de asignaturas configuradas

D.2.12 Ver horarios de tutorías

Para ver los horarios de tutorías configurados actualmente, pulsaremos en la opción *Mostrar horarios actuales* del menú Gestión de horarios. Se mostrarán los horarios y las asignaturas a las que se aplican, tal y como se puede ver en la figura D.16.

D.2.13 Modificar horarios de tutorías

Para modificar los horarios de tutorías configurados, pulsaremos en la opción *Editar horarios* del menú Gestión de horarios. Se mostrarán los horarios tal y como se puede ver en la figura D.16. Después de cada horario, se muestran dos enlaces: Eliminar horario completo y Modificar horario. Pulsaremos en este último y nos aparecerá la pantalla que se muestra en la figura D.17.

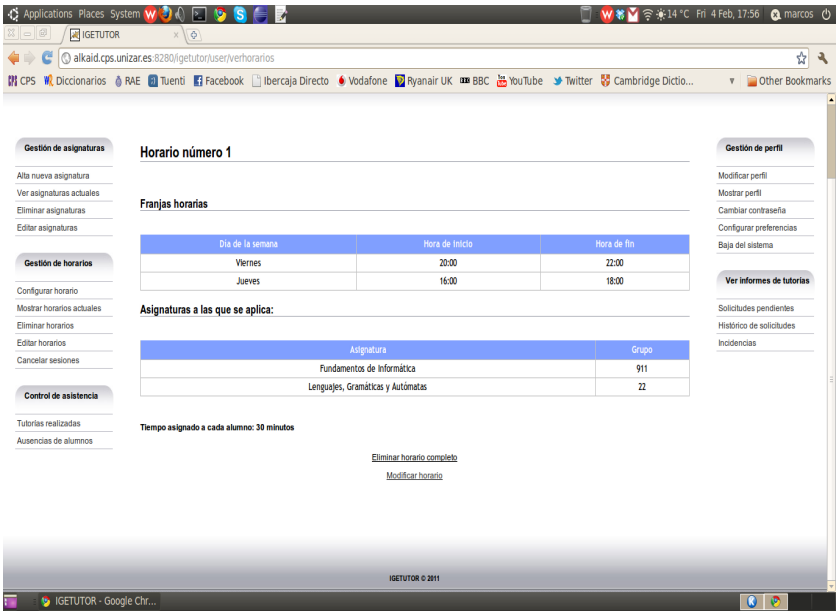


Figura D.16. Pantalla de visualización de los horarios de tutorías

En la nueva pantalla aparecerán los datos del horario que queremos modificar, con las franjas del horario y las asignaturas a las que se aplica el horario. Podremos crear nuevas franjas, eliminar o modificar franjas existentes. También podremos cambiar las asignaturas a las que se aplica el horario, añadiendo nuevas asignaturas que no tuvieran horario configurado todavía, o eliminando algunas de las asignaturas para las que se aplicaba el horario actual.

Recuerde que al modificar un horario de tutorías, el sistema necesita volver a calcular las sesiones de tutorías asociadas a dicho horario. Esto implica que aquellas tutorías reservadas para las sesiones del horario original desaparecerán.

D.2.14 Eliminar horarios de tutorías

Para eliminar horarios de tutorías que tengamos configurados, pulsaremos en la opción *Eliminar horarios* del menú Gestión de horarios. Se mostrarán los horarios que teníamos configurados. Después de cada horario, se muestran dos enlaces:

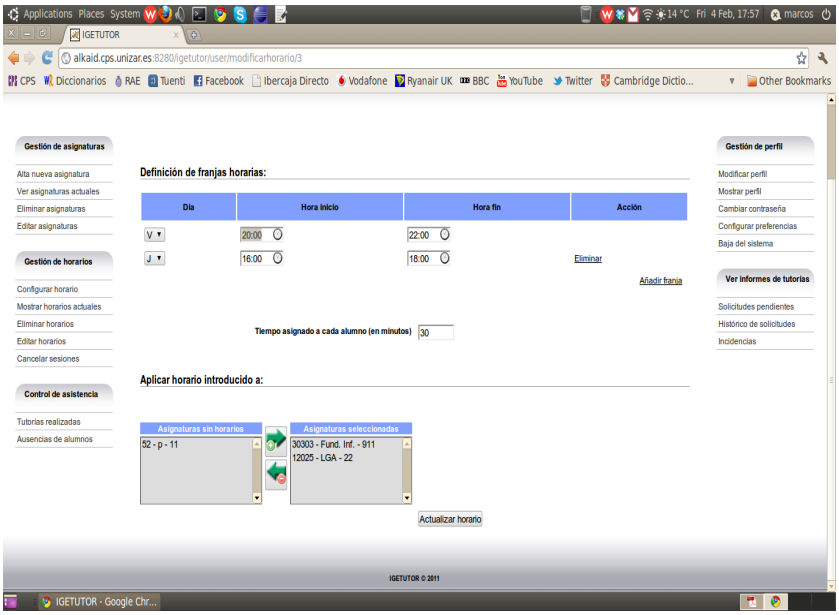


Figura D.17. Pantalla de modificación de un horario

Eliminar horario completo y Modificar horario. Pulsaremos en la primera opción si deseamos eliminar el horario por completo. El sistema solicitará confirmación del usuario antes de llevar a cabo la operación.

Recuerde que al eliminar el horario de tutorías también se eliminarán las reservas de tutorías asociadas a ese horario.

D.2.15 Baja del sistema

Para darse de baja en el sistema, pulsaremos la opción *Baja del sistema* en el menú Gestión de perfil. Cabe recordar que al darse de baja del sistema, se eliminarán todos los datos asociados a nuestro perfil (asignaturas impartidas, horarios configurados, reservas de tutorías, etc).

Una vez pulsada la opción, aparecerá la pantalla que se muestra en la figura D.18. En ella, deberemos introducir nuestro nombre de usuario y contraseña y pulsar en el botón *Baja* para hacer efectiva la baja. El sistema nos devolverá a la pantalla inicial del sistema.

D.2.16 Cambio de contraseña

Para cambiar la contraseña, pulsaremos la opción *Cambiar contraseña* en el menú Gestión de perfil. Aparecerá la pantalla que se muestra en la figura D.19. En

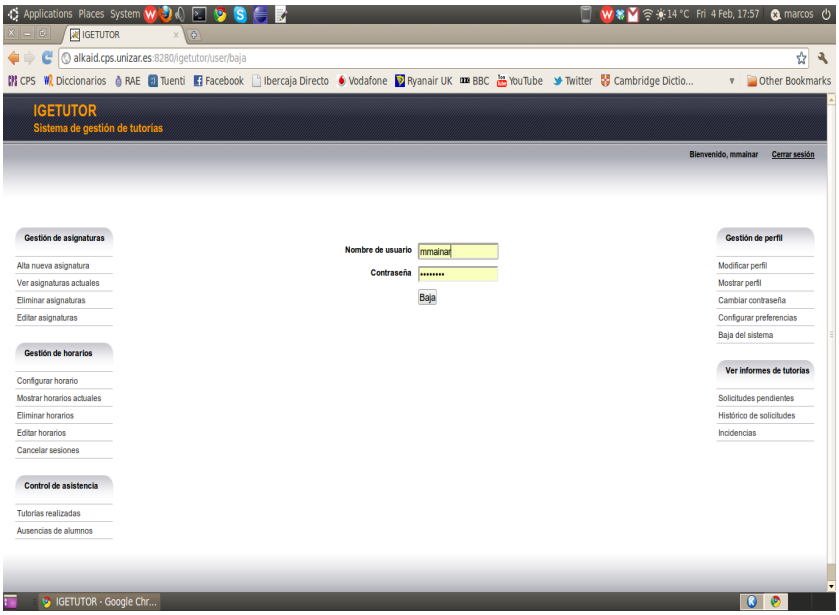


Figura D.18. Pantalla de baja del sistema

ella, deberá introducir su contraseña antigua y su nueva contraseña. Como consejo de seguridad, trate de elegir una contraseña lo suficientemente segura, donde se combinen letras minúsculas, mayúsculas y números.

D.2.17 Ver perfil

Para ver los datos de su perfil, pulsaremos la opción *Mostrar perfil* en el menú Gestión de perfil. Aparecerá la pantalla que se muestra en la figura D.20

D.2.18 Modificar perfil

Para modificar su perfil, pulsaremos la opción *Modificar perfil* en el menú Gestión de perfil. Modifique los datos que desee y finalmente pulse *Actualizar* para guardar su perfil con los datos actualizados.

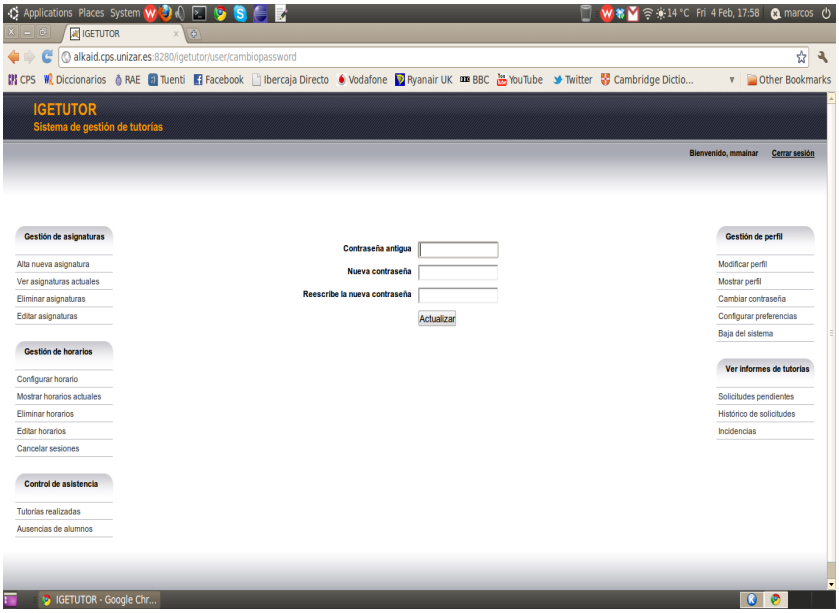


Figura D.19. Pantalla de cambio de contraseña

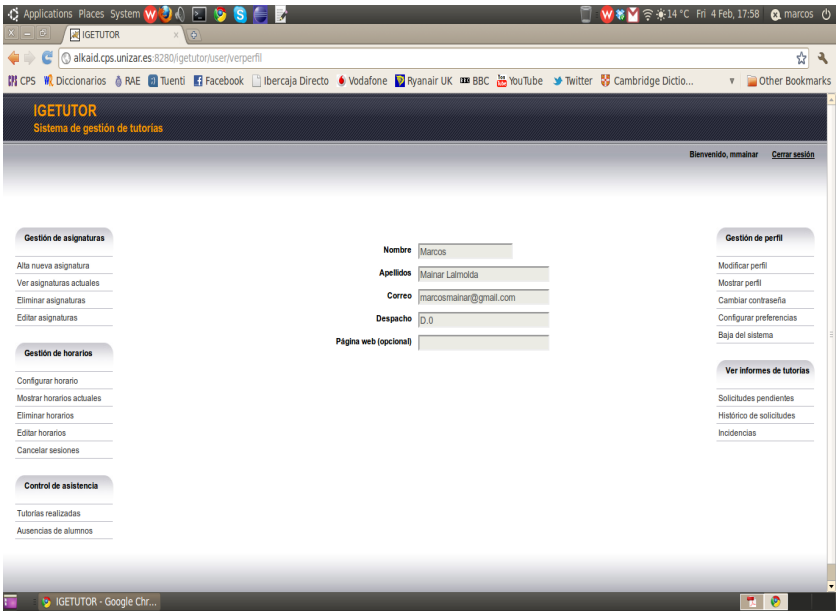


Figura D.20. Pantalla de visualización del perfil

D.3 Manual del Alumno

D.3.1 Primeros pasos

Para comenzar a utilizar el sistema, abriremos nuestro navegador Web favorito e introduciremos la siguiente dirección: `http://alkaid.cps.unizar.es:8280/reservaPedro.jsp`. Aparecerá la pantalla que se muestra en la figura D.21.

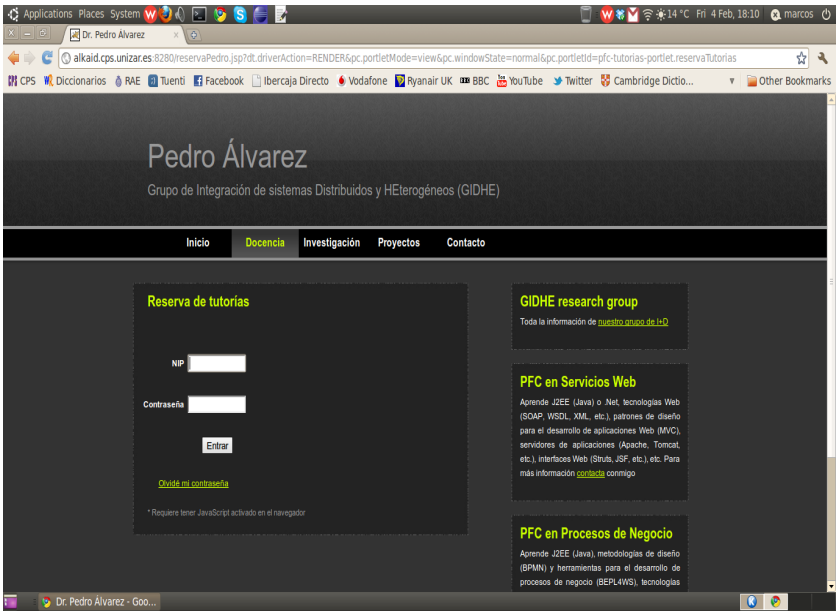


Figura D.21. Pantalla inicial de la aplicación Portlet

La cuenta de correo electrónico que utiliza el sistema para todas las notificaciones que se envían al alumno (alta en el sistema, confirmaciones de tutoría reservada/modificada/cancelada y recordatorio de contraseña), es la oficial de la Universidad de Zaragoza (NIP@celes.unizar.es).

D.3.2 Inicio de sesión

Para iniciar la sesión, introduciremos nuestro NIP y contraseña. La contraseña habrá sido enviada a la cuenta de correo electrónico del alumno. En caso contrario, puede solicitar una nueva contraseña pulsando en el enlace *Olvidé mi contraseña*.

Una vez en el sistema, se mostrará la pantalla con el menú principal de la aplicación. Esta pantalla se puede ver en la figura D.22. Desde esta pantalla podremos acceder a toda la funcionalidad disponible.

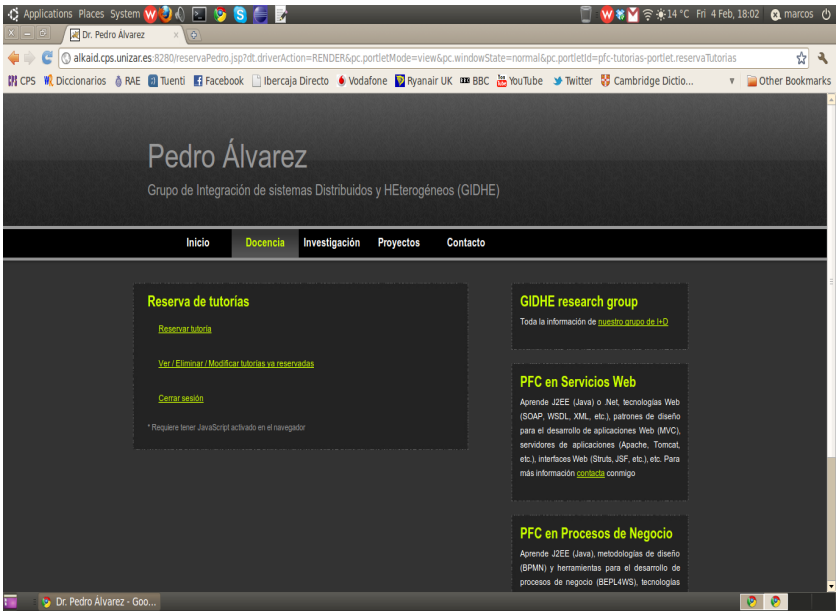


Figura D.22. Menú principal de la aplicación

D.3.3 Olvido de contraseña

Para recibir una nueva contraseña por correo electrónico, pulsaremos en la opción *Olvidé mi contraseña* del menú principal de la aplicación. Aparecerá la pantalla que se muestra en la figura D.23. En ella, introduciremos nuestro NIP y DNI (incluida la letra). El sistema enviará una nueva contraseña a nuestro correo electrónico si los datos introducidos son correctos.

D.3.4 Reserva de cita de tutoría

Para reservar una cita de tutoría, pulsaremos en la opción *Reservar tutoría* del menú principal de la aplicación. Aparecerá la pantalla que se muestra en la figura D.24. En ella, tenemos que introducir, en orden, los datos de los menús desplegables. Opcionalmente podremos introducir, también, comentarios de interés para el profesor acerca de la tutoría. Una vez introducidos todos los datos necesarios, pulsaremos en el botón Confirmar. Se mostrará una nueva pantalla indicando si la tutoría se ha reservado correctamente o no. En el caso de que la reserva haya sido correcta, el sistema enviará un correo electrónico al alumno confirmándole los datos de la misma.

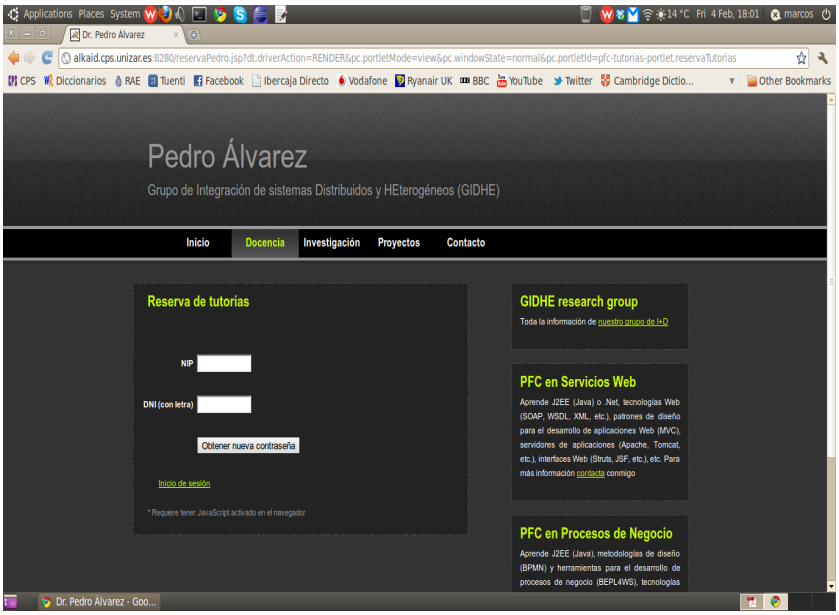


Figura D.23. Pantalla de olvido de contraseña del alumno

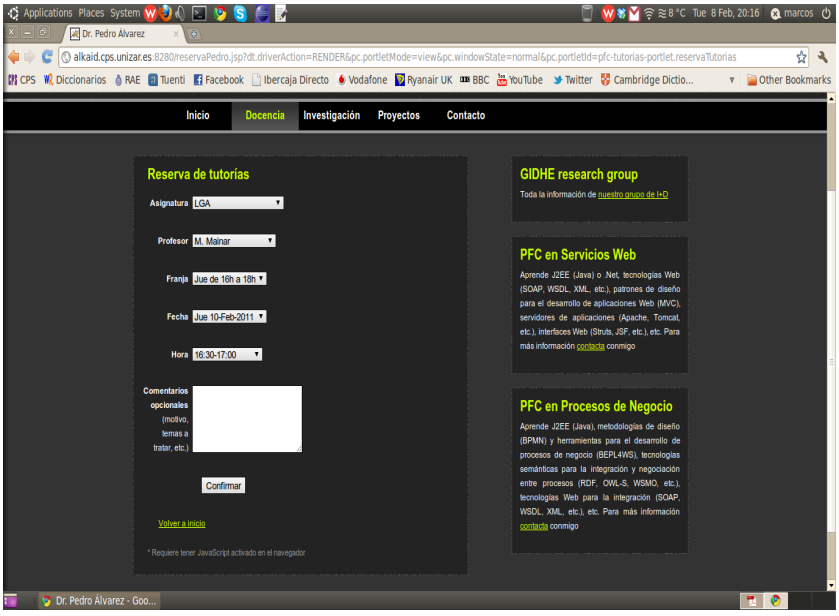


Figura D.24. Pantalla de reserva de cita de tutoría

D.3.5 Ver tutorías asignadas

Para ver las tutorías asignadas pendientes de realizarse, pulsaremos en el enlace *Ver/Eliminar/Modificar tutorías ya reservadas*. Aparecerá la pantalla que se muestra en la figura D.25. En ella, podemos ver en una tabla todas las tutorías asignadas. Desde esta pantalla también hay enlaces para acceder a modificar o cancelar las tutorías asignadas.

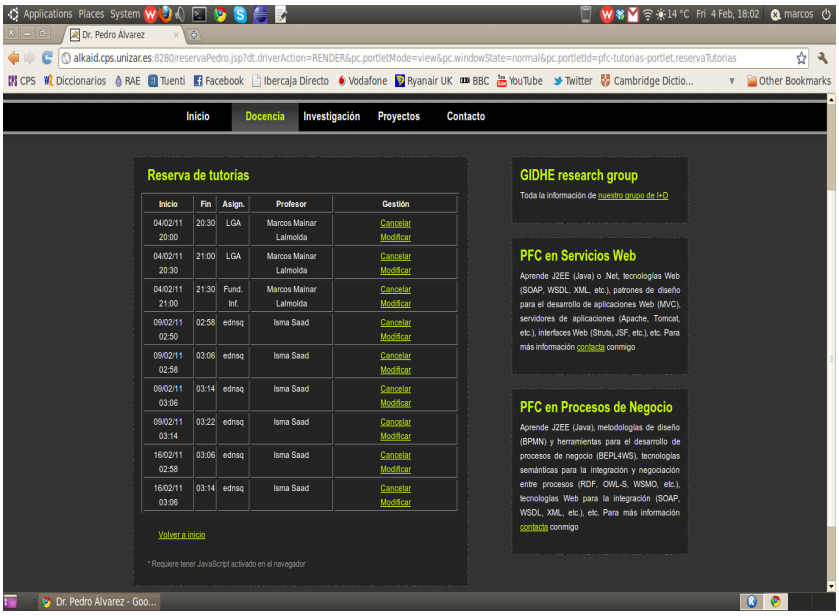


Figura D.25. Pantalla de ver, eliminar o modificar tutorías asignadas

D.3.6 Modificar la fecha y hora de tutorías asignadas

Para poder modificar una tutoría asignada, debe existir al menos un día de antelación con respecto a la fecha y hora de comienzo de la tutoría. Los datos que se podrán modificar son su fecha y hora.

Para modificar el horario de una tutoría, pulsaremos en el enlace *Modificar* de la fila de la tabla correspondiente a la tutoría. El sistema nos preguntará si estamos seguros de que deseamos modificar esta tutoría. Al pulsar el botón *OK* se mostrará la pantalla de la figura D.26 donde podremos modificar los datos de la tutoría. Una vez realizados los cambios, pulsaremos en el botón *Modificar*. Si la modificación se realiza correctamente, el sistema enviará un correo electrónico al alumno con los datos de la tutoría modificados. El profesor también recibirá un correo si tenía configurada la opción de recibir una notificación por cada nueva solicitud de tutoría.

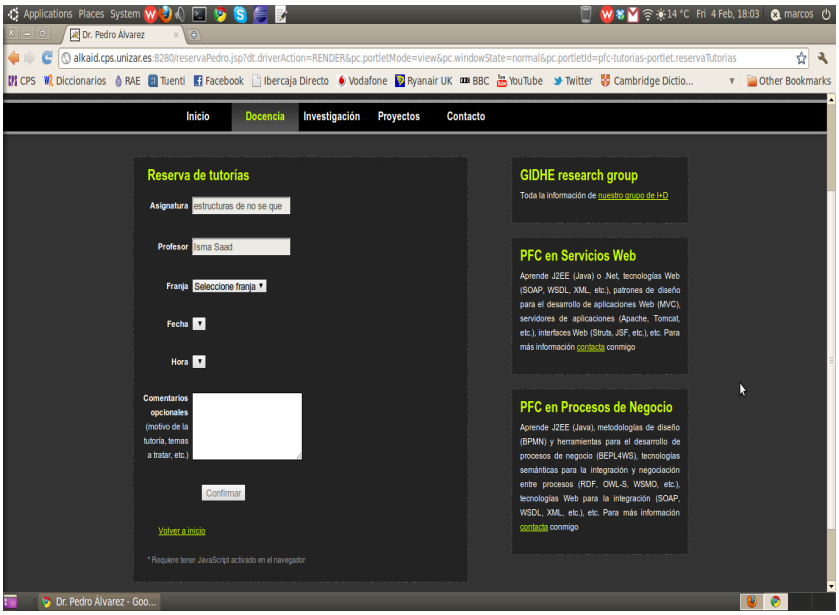


Figura D.26. Pantalla de modificación de una tutoría reservada

D.3.7 Cancelar tutorías asignadas

Para cancelar una tutoría asignada debe existir al menos un día de antelación con respecto a la fecha y hora de comienzo de la tutoría. Si queremos cancelar una tutoría, pulsaremos en el enlace *Cancelar* en la fila de la tabla correspondiente a la tutoría. El sistema nos preguntará si estamos seguros de que deseamos cancelar esta tutoría. Para hacer efectiva la cancelación, basta pulsar en el botón *OK*. Si la cancelación se realiza correctamente, el sistema enviará un correo electrónico al alumno con los datos de la tutoría cancelada. El profesor también recibirá un correo si tenía configurada la opción de recibir una notificación por cada nueva solicitud de tutoría.

D.3.8 Cerrar sesión

Para salir del sistema, en la pantalla principal de la aplicación que se muestra en la figura D.22, pulsaremos en la opción *Cerrar sesión*.

D.4 Manual del Administrador

D.4.1 Primeros pasos

Para comenzar a utilizar el sistema, abra una ventana de su navegador Web preferido. Una vez abierta, introduzca la siguiente dirección: `http://alkaid.cps.unizar.es:8280/igetutor/`. Se recomienda guardar esta dirección en los favoritos del navegador. Al cargarse la dirección anterior, aparecerá la pantalla que se observa en la figura D.1.

Una vez se encuentre en la pantalla inicial, el primer paso será iniciar sesión como administrador en el sistema. Para ello, pulsaremos en el enlace *Inicio de sesión* situado en la parte superior derecha de la pantalla. Aparecerá la pantalla que se observa en la figura D.4. En ella, introduciremos los datos de inicio de sesión del administrador.

Una vez iniciada la sesión, aparece la pantalla principal que se puede observar en la figura D.27. Esta pantalla contiene todas las opciones disponibles para el administrador en menús organizados por categorías.

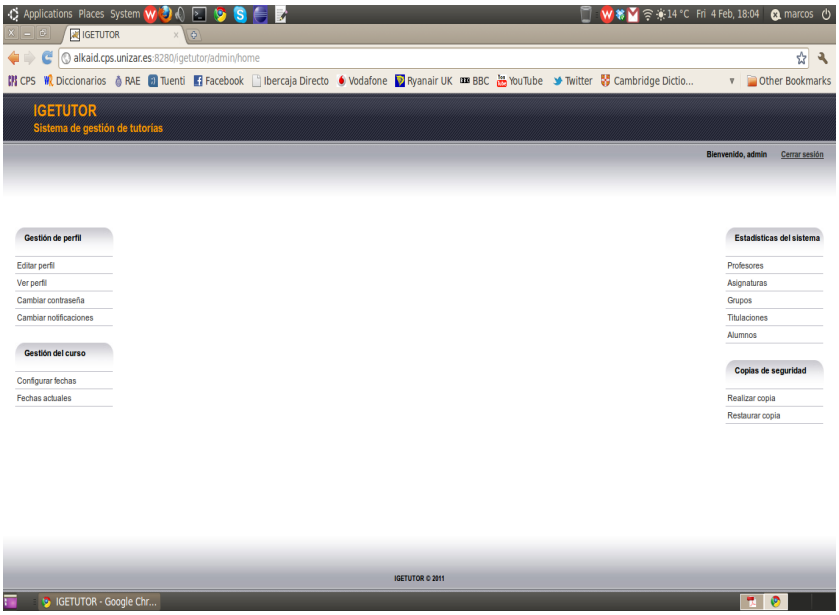


Figura D.27. Pantalla principal del administrador

Los primeros pasos que debería realizar el administrador una vez ha iniciado sesión del sistema por primera vez son:

- Cambiar su contraseña.

- Cambiar los datos de la cuenta de correo electrónico utilizada para las notificaciones del sistema.
- Configurar las fechas del curso académico actual.

D.4.2 Cambiar la contraseña

Para cambiar su contraseña de administrador, pulse en la opción *Cambiar contraseña* del menú Gestión de perfil. Aparecerá la pantalla que se muestra en la figura D.19. En ella, deberá introducir su contraseña antigua y su nueva contraseña. La contraseña escogida debe tener entre 4 y 8 caracteres. Como consejo de seguridad, trate de elegir una contraseña lo suficientemente segura, donde se combinen letras minúsculas, mayúsculas y números.

D.4.3 Cambiar la cuenta de notificaciones

Para cambiar los datos de la cuenta de correo electrónico que se utiliza para enviar las notificaciones a los profesores y alumnos del sistema, pulse en la opción *Cambiar notificaciones* del menú Gestión de perfil. Aparecerá la pantalla que se muestra en la figura D.28. En ella, deberá introducir la dirección de correo electrónico, el servidor SMTP y la contraseña de dicha cuenta.

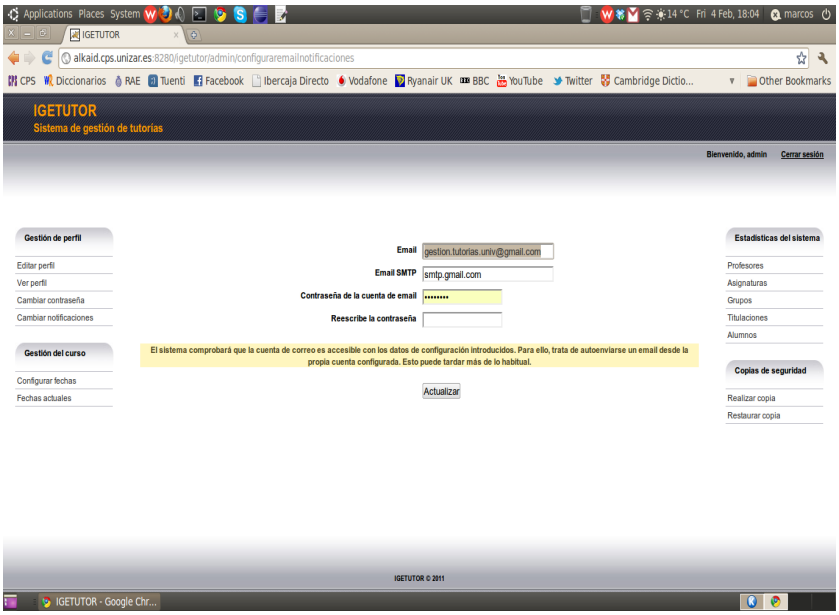


Figura D.28. Pantalla de cambio de la cuenta de notificaciones

El sistema comprobará que la cuenta de correo es accesible con los datos introducidos; lo notificará en caso contrario.

D.4.4 Configurar las fechas del curso

Para cambiar las fechas del curso académico actual, pulse en la opción *Configurar fechas* del menú Gestión del curso. Aparecerá la pantalla que se muestra en la figura D.29. En ella, deberá introducir la fecha de inicio del curso académico, fecha de finalización del primer cuatrimestre, fecha de inicio del segundo cuatrimestre y fecha de finalización del curso académico. Para ello, aparece un icono de calendario que al pulsarlo permite elegir una fecha concreta.

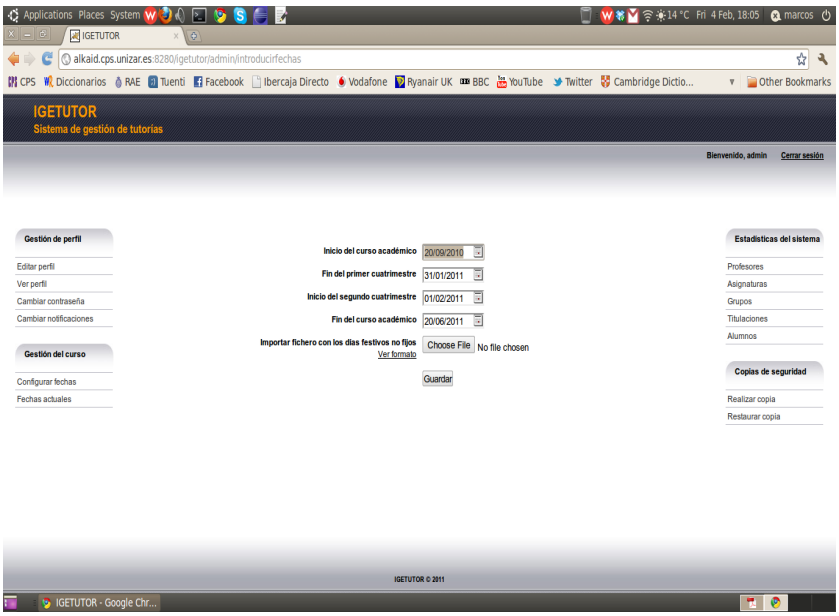


Figura D.29. Pantalla de configuración de las fechas del curso actual

También deberá importar un fichero con las fechas de días festivos del curso académico actual. Si pulsa en el enlace *Ver formato*, se muestra una explicación del formato de dicho fichero y se le permite descargar el fichero de festivos del curso académico 2010/2011, a modo de ejemplo. Puede descargar y editar directamente ese fichero para adaptar las fechas al curso académico actual. Cuando tenga el fichero preparado, pulse en el botón *Choose file* para seleccionar su ubicación. Por último, pulse el botón *Guardar* para almacenar las fechas introducidas.

D.4.5 Ver fechas actuales del curso

Para visualizar las fechas del curso académico actual configuradas en el sistema, pulse en la opción *Fechas actuales* del menú Gestión del curso. Aparecerá la pantalla que se muestra en la figura D.30. En esta pantalla, se muestran todas las fechas introducidas.

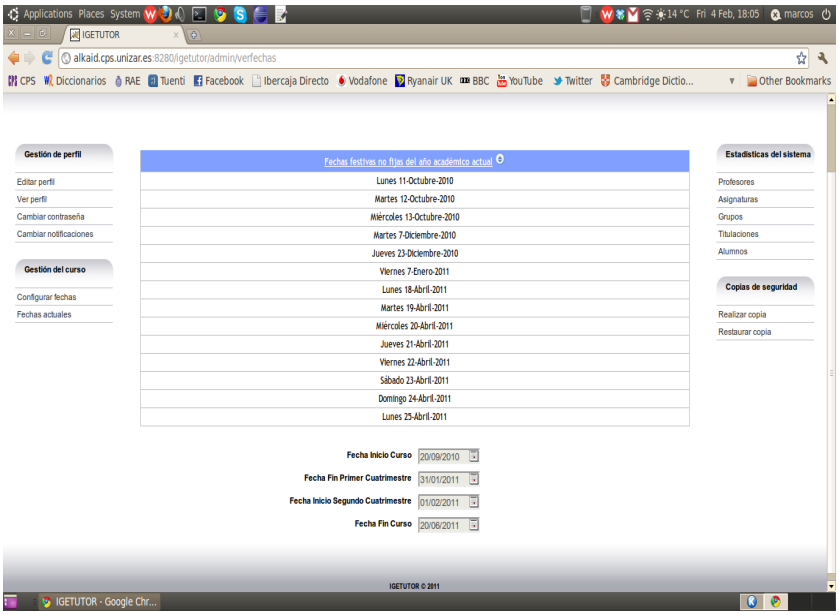


Figura D.30. Pantalla de visualización de las fechas configuradas del curso actual

D.4.6 Realizar copias de seguridad

El sistema realiza de forma automática copias de seguridad de toda su información una vez al mes. Adicionalmente, el administrador puede realizar copias de seguridad de todos los datos del sistema en cualquier momento deseado. Para ello, pulse en la opción *Realizar copia*, del menú Copias de seguridad. Aparecerá la pantalla que se muestra en la figura D.31. Para descargar la copia de seguridad, simplemente pulse en el botón *Descargar copia*. Se realizará la copia y se descargará comprimida en formato ZIP. El nombre del fichero será: *CS.fechaActual.zip* (la fecha actual tendrá formato *dd-mm-aaaa*).

D.4.7 Restaurar copias de seguridad

El administrador puede restaurar todos los datos del sistema a partir de copias de seguridad. Para ello, se proporciona un sencillo script SQL de restauración llamado *restaurar_BD.sql*. Este script se puede encontrar en el directorio src/sql del proyecto pfc-tutorias-model. El script debe tener permiso de ejecución para poder ejecutarse en su sistema. Para utilizarlo, primero descomprima el fichero zip con la copia de seguridad que desea utilizar para la restauración en el mismo directorio donde tenga el script. Esto generará el fichero SQL que contiene el estado de la base de datos para restaurarla. En segundo lugar, abra una consola de comandos e introduzca la siguiente línea, desde el directorio donde estén situados el script y la copia de seguridad:

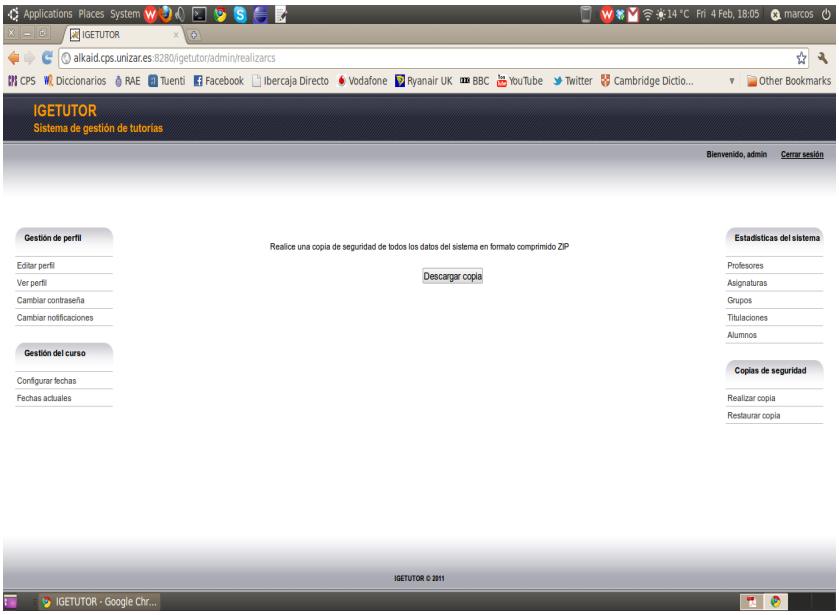


Figura D.31. Pantalla de realización de copias de seguridad del sistema

```
restaurarBD.sql < CS.fecha.sql
```

Por ejemplo, si se quisiera restaurar la base de datos utilizando la copia de seguridad del 24 de diciembre de 2010, se ejecutaría lo siguiente:

```
restaurarBD.sql < CS.24-12-2010.sql
```

Si la ejecución del comando anterior no devuelve error alguno, la base de datos del sistema habrá sido restaurada con la copia de seguridad elegida.

D.4.8 Ver estadísticas del sistema

En el menú Estadísticas del sistema, el administrador puede consultar diferentes datos del sistema navegando a partir de los Profesores, Asignaturas, Grupos, Titulaciones y Alumnos. En la figura D.32 se puede ver una pantalla de ejemplo con estadísticas del sistema.

D.4.9 Ver perfil

Para ver los datos de su perfil, pulsaremos la opción *Ver perfil* en el menú de Gestión de Perfil.

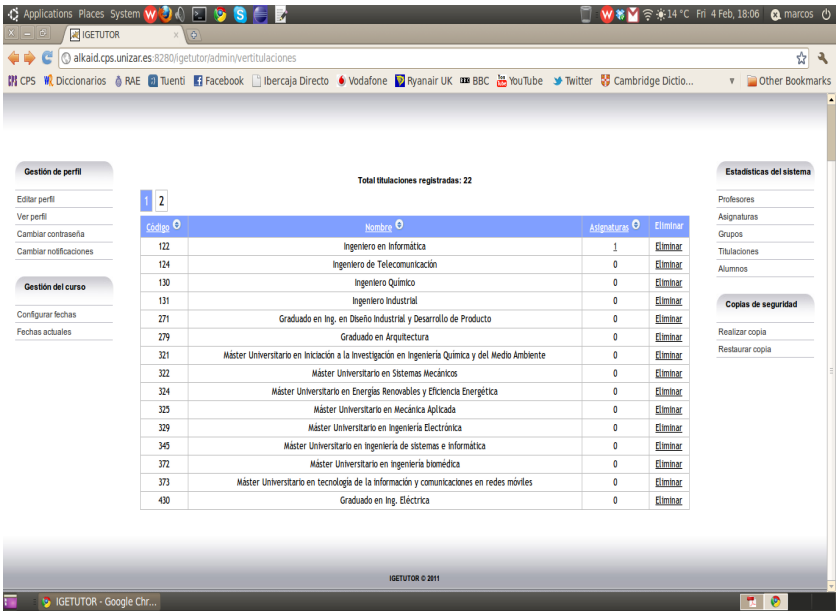


Figura D.32. Pantalla de ejemplo con estadísticas del sistema

D.4.10 Modificar perfil

Para modificar su perfil, pulsaremos la opción *Editar perfil* en el menú de Gestión de Perfil. Modifique los datos que desee y pulse *Actualizar* para guardar su perfil con los datos actualizados.

Anexo E

Manual de Instalación y Mantenimiento

En este anexo se explica el proceso de instalación y mantenimiento del sistema. Se divide en 2 secciones. En la sección E.1 se describe la instalación y mantenimiento de la aplicación Web. En la sección E.2 se explica la instalación y mantenimiento de la aplicación Portlet.

E.1 Instalación y mantenimiento de la aplicación Web

En esta sección se describen los requisitos y pasos necesarios para instalar la aplicación Web.

E.1.1 Requisitos para la instalación

Para instalar la aplicación Web, se requiere:

1. Una base de datos relacional MySQL versión 5 (o superior) con el motor transaccional InnoDB.
2. Un servidor Web con soporte para Servlets (Tomcat, Jetty) o un servidor de aplicaciones J2EE (JBoss, GlassFish, etc).

En este caso se ha elegido el servidor Web Apache Tomcat, dado que requiere pocos recursos y dispone de buena documentación.

E.1.1.1 Instalar MySQL

El primer paso será descargar MySQL Community Server de `http://dev.mysql.com/downloads/mysql/` para nuestro sistema operativo.

Después, procederemos a instalar el servidor de base de datos descargado siguiendo las instrucciones de la página oficial.

E.1.1.2 Crear la base de datos de la aplicación

Para crear la base de datos de la aplicación, se empleará la utilidad de administración *mysqladmin*.

En primer lugar, crearemos la base de datos con nombre *igetutor*. Para ello, ejecutaremos el siguiente comando:

```
mysqladmin -u root -p create igetutor
```

El comando solicitará la contraseña del administrador de la base de datos MySQL. Tras introducirla correctamente, habrá creado la base de datos.

Después, se creará un usuario para acceder a la base de datos. En este manual se considerará que los datos del usuario son:

- Nombre de usuario: user
- Contraseña: hiK9J2lvM

Para ello, hay que entrar como root ejecutando:

```
mysql -u root -p
```

Se pedirá de nuevo la contraseña del administrador de la base de datos. A continuación, aparecerá el *prompt mysql>* para introducir órdenes SQL. Introduciremos el siguiente comando:

```
GRANT ALL PRIVILEGES ON igetutor.* TO 'user'@'%'
IDENTIFIED BY 'hiK9J2lvM';
```

Tras pulsar la tecla *Intro* deberá aparecer la siguiente respuesta: *Query OK, 0 rows affected*. Para salir del *prompt* y volver a la consola de comandos, ejecutaremos:

```
exit;
```

Podemos comprobar que el acceso a la base de datos creada con el usuario es posible ejecutando:

```
mysql -u user --password=hiK9J2lvM igetutor
```

El siguiente paso será crear las tablas e introducir algunos datos iniciales en la base de datos. Para ello, hace falta tener instalada la herramienta Maven. Para descargar e instalar la herramienta Maven en nuestro sistema, seguiremos las instrucciones de la página Web oficial `http://maven.apache.org/run-maven/index.html`.

En el interior del directorio *pfc-tutorias-model*, abriremos el fichero de configuración de Maven, *pom.xml*. Buscaremos la propiedad *dataSource.url* e introduciremos la URL de la base de datos configurada. También buscaremos las propiedades *dataSource.user* y *dataSource.password* e introduciremos los datos del nombre de usuario y contraseña elegidos para acceder a la base de datos. A continuación, desde ese directorio, abriremos una terminal y ejecutaremos el siguiente comando:

```
mvn sql:execute
```

Este comando crea las tablas e introduce algunos datos iniciales en la base de datos. Si el comando se ejecuta correctamente, mostrará lo siguiente:

```
[INFO] 87 of 87 SQL statements executed successfully
[INFO] ------------------------------------------
[INFO] BUILD SUCCESSFUL
```

E.1.1.3 Instalar Apache Tomcat

Para instalar Tomcat, siga los pasos de la página oficial, `http://tomcat.apache.org/`, según su sistema operativo. Básicamente, estos pasos consisten en descargar Tomcat y descomprimirlo en el directorio deseado.

A continuación, se añadirá a Tomcat el *driver* JDBC para permitir el acceso a la base de datos MySQL. Para ello, hay que descargar el connector MySQL Connector/J de `http://dev.mysql.com/downloads/connector/j/`. Se descomprime el fichero descargado y se copia el fichero *mysql-connector-java-version-bin.jar*, al directorio *lib* de Tomcat.

Posteriormente, hay que configurar la base de datos en Tomcat utilizando JNDI [47]. Para ello, tenemos que editar 2 ficheros de configuración de Tomcat: *server.xml* y *context.xml*. En primer lugar, abriremos el fichero *conf/server.xml* y escribiremos lo siguiente en el interior de la etiqueta *<GlobalNamingResources>*:

```
<!-- MySQL -->
<Resource name="jdbc/igetutor"
          auth="Container"
          type="javax.sql.DataSource"
          driverClassName="com.mysql.jdbc.Driver"
          url="jdbc:mysql://localhost:3306/igetutor"
          username="user"
          password="hiK9J2lvM"
          maxActive="-1"
          maxIdle="-1"
          maxWait="10000"
          removeAbandoned="true"
          removeAbandonedTimeout="60"
          logAbandoned="true"
          validationQuery="SELECT COUNT(*) FROM PingTable"/>
```

En la propiedad *url* del código anterior, pondremos la URL de la base de datos configurada. Del mismo modo, en las propiedades *username* y *password* introduciremos el nombre de usuario y contraseña del usuario utilizado para acceder a la base de datos.

En segundo lugar, añadiremos lo siguiente al fichero *conf/context.xml*, dentro de la etiqueta *<Context>*:

```
<ResourceLink name="jdbc/igetutor" global="jdbc/igetutor"
              type="javax.sql.DataSource"/>
```

E.1.2 Instalación de la aplicación Web en Tomcat

Para proceder a instalar la aplicación Web en el servidor Web Tomcat, en primer lugar hay que ejecutar desde el directorio *pfc-tutorias-model* el siguiente comando Maven:

```
mvn clean install
```

Después, se ejecuta desde el directorio *pfc-tutorias-web* el mismo comando anterior.

Estos 2 comandos construyen el proyecto, generando el fichero WAR de la aplicación Web en el directorio *target*. Este fichero es el que se instala en Apache Tomcat.

A continuación, iniciaremos Tomcat. Para ello, desde el directorio *bin*, hay que ejecutar el *script startup*, dependiendo de nuestro sistema operativo. En Windows,

habrá que ejecutar *startup.bat*. En Linux, *startup.sh* (antes tendremos que dar permisos de ejecución a todos los scripts, con el comando *chmod +x *.sh*).

Probar a abrir en el navegador la URL `http://localhost:8080` (suponiendo que hemos instalado Tomcat en nuestra máquina localmente, en otro caso, abrir la URL del servidor correspondiente). Si aparece la pantalla de inicio de Tomcat, éste ha sido configurado correctamente.

A continuación, copiar el fichero *pfc-tutorias-web/target/igetutor.war* al directorio *webapps* de Tomcat. Pasados unos segundos, aparecerá un directorio *igetutor*.

Probar a abrir en el navegador la dirección `http://localhost:8080/igetutor` después de un minuto aproximadamente (durante este tiempo se está iniciando la aplicación Web en Tomcat). Si aparece la pantalla de inicio del sistema IGETUTOR, la instalación ha sido satisfactoria.

Por último, hay que mencionar que los ficheros de *log* de la aplicación Web se llamarán *igetutor_web.log*. Estarán almacenados en el directorio *logs* de Tomcat. Al final de cada día se creará un nuevo fichero de *log* al que se le añadirá la fecha del día de forma que el nombre del fichero será *igetutor_web.log.fecha*.

E.2 Instalación y mantenimiento de la aplicación Portlet

En esta sección se describen los requisitos y pasos necesarios para instalar la aplicación Portlet.

E.2.1 Requisitos para la instalación

Para instalar la aplicación Portlet, se puede optar por utilizar cualquiera de los 2 entornos de instalación siguientes:

1. El contenedor de Portlets OpenPortal Portlet Container sobre un contenedor Web (Tomcat, GlassFish, Jetty, WebLogic, etc.) ó
2. Un servidor de portales Web (Liferay, JetSpeed2, GateIn, WebSphere, SharePoint, etc).

En este manual se explica en primer lugar el proceso de instalación en el primer entorno, utilizando Apache Tomcat como servidor Web. La elección de Tomcat se basa en que es un servidor Web con soporte para Servlets y JSP ligero, es decir, que no requiere excesivos recursos para funcionar y además dispone de buena documentación.

Posteriormente, se explica el proceso en el segundo entorno, con Liferay como servidor de portales Web. Se ha elegido Liferay dado que tiene una versión gratuita,

dispone de buena documentación, puede ser configurado en una gran variedad de servidores (Tomcat, Jetty, JBoss, Geronimo, GlassFish, Resin, JOnAS, etc.), resulta sencillo de utilizar y tiene una amplia comunidad de desarrolladores y usuarios detrás.

Por último, es necesario recordar que la base de datos MySQL debe estar creada e instalada para que la aplicación Portlet funcione correctamente. Para ello, siga los pasos ya explicados en la sección E.1.1.1.

E.2.2 Instalación de la aplicación Portlet en OpenPortal Portlet Container sobre Apache Tomcat

Primeramente, para instalar Tomcat, siga los pasos explicados en la sección E.1.1.3. Para instalar OpenPortal Portlet Container sobre Apache Tomcat, siga los pasos que se especifican en el manual de usuario del contenedor de Portlets [8].

Una vez estén ambos instalados, para instalar la aplicación Portlet en este entorno, basta con desplegar el fichero WAR del Portlet sobre el contenedor. Por tanto, primero se generará el fichero WAR, utilizando la herramienta Maven. Para ello, desde el interior del directorio *pfc-tutorias-model*, ejecute el comando siguiente:

```
mvn clean install
```

A continuación, ejecutaremos el mismo comando desde el interior del directorio *pfc-tutorias-portlet*. Esto último habrá generado el fichero WAR en el directorio *target* de *pfc-tutorias-portlet*.

Para instalar el fichero WAR, se puede emplear la utilidad de administración que posee el contenedor, realizando los pasos siguientes:

1. Abra la dirección Web `http://alkaid.cps.unizar.es:8280/portletdriver/admin`. Desde ahí, se pueden administrar los Portlets instalados en el contenedor previa introducción de la contraseña de administración.
2. En la zona de desplegar Portlets (*Deploy a Portlet*), pulse el botón *Choose file* para seleccionar la ruta del fichero WAR del Portlet. Una vez seleccionada, pulse *Deploy*. Esto se puede observar en la figura E.1.
3. El contenedor desplegará el Portlet pasados unos segundos.
4. Pulsando en la pestaña Portlets, veremos el Portlet recién instalado. Puede tardar unos segundos en aparecer y que al principio se muestre minimizado. Para ver el Portlet completo, pulse en el icono con forma de ojo, correspondiente al modo *View* del Portlet. Esto se puede observar en la figura E.2.

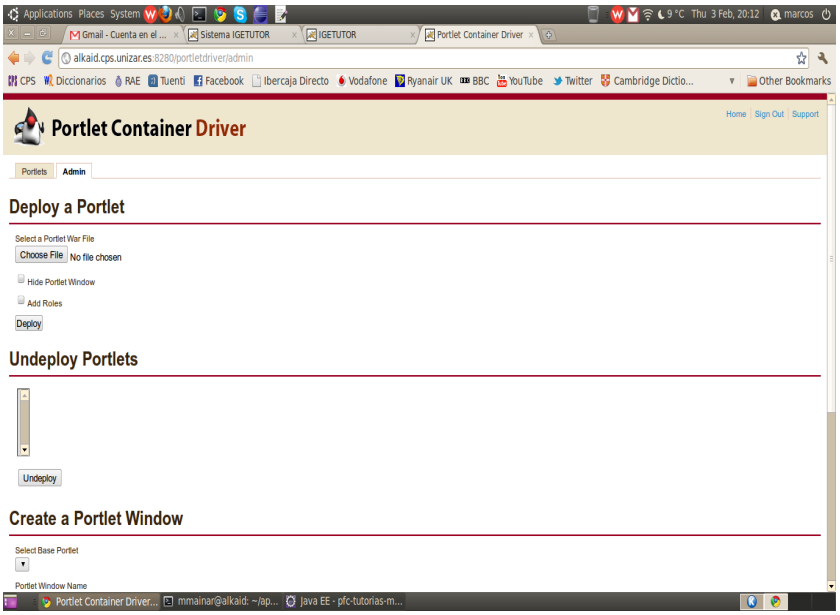


Figura E.1. Pantalla de administración de Portlets de OpenPortal Portlet Container

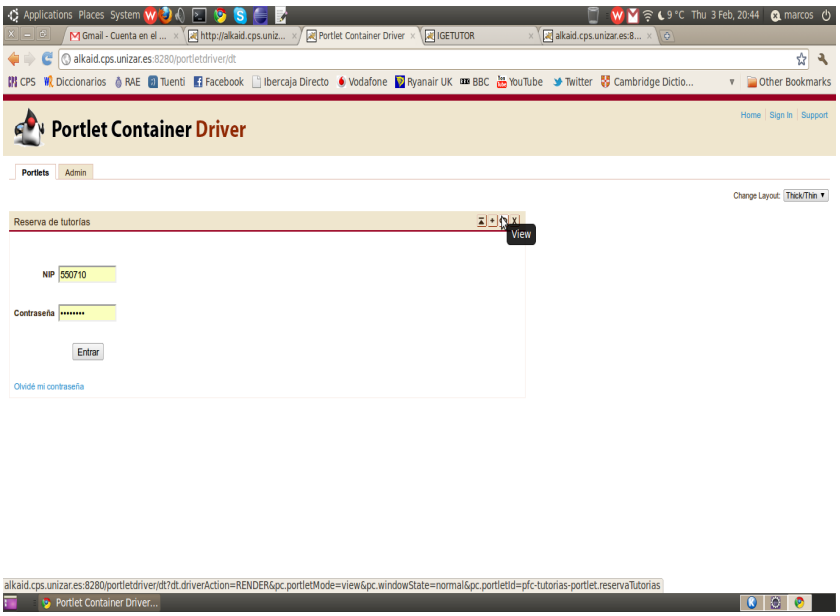


Figura E.2. Pantalla de visualización de Portlets instalados en OpenPortal Portlet Container

Por último, hay que mencionar que los ficheros de *log* de la aplicación Portlet se llamarán *igetutor_portlet.log*. Estarán almacenados en el directorio *logs* de Tomcat.

Al final de cada día se creará un nuevo fichero de *log* al que se le añadirá la fecha del día de forma que el nombre del fichero será *igetutor_portlet.log.fecha*.

E.2.2.1 Integrar el Portlet en una página Web utilizando el driver de OpenPortal Portlet Container

OpenPortal Portlet Container posee un *driver* que hace posible incluir Portlets en páginas JavaServer Pages (JSP). Si tenemos una página Web .html, basta con renombrar el fichero a .jsp para convertirlo en una página JSP.

Las instrucciones que se exponen en este apartado están basadas en la sección correspondiente del manual de usuario del contenedor de Portlets [8].

Los pasos para añadir el Portlet a la página JSP creada son:

1. Añadir en la primera línea de la página JSP creada la referencia al *driver* del contenedor de Portlets. Esto se hace incluyendo la siguiente directiva:

```
<%@taglib uri="http://portlet-container.dev.java.net"
          prefix="pcdriver" %>
```

2. Añadir el código necesario para mostrar el Portlet donde se desea que aparezca. Este código es el siguiente:

```
<pcdriver:portlet portletName="reservaTutorias"
   applicationName="pfc-tutorias-portlet">
   <pcdriver:title/>
   <pcdriver:render/>
</pcdriver:portlet>
```

3. Copiar la página JSP creada al directorio *ROOT* de Apache Tomcat del servidor Alkaid donde está instalada la aplicación Portlet. La ruta del directorio es */home/mmainar/apache-tomcat-6.0.29/webapps/ROOT*. Si no se posee cuenta en dicho servidor, se puede solicitar una al administrador o bien pedir a otro profesor que posea cuenta o al propio administrador que copie la página a la ruta indicada.

4. La página con el Portlet será accesible desde la URL *http://alkaid.cps.unizar.es:8280/nombrePaginaCreada.jsp*. Lo más sencillo será crear un hipervínculo desde nuestra página Web a dicha URL.

El código incluido para mostrar el Portlet irá normalmente entre etiquetas HTML de estilo (*div*, *span*, etc.) utilizando el CSS de nuestra página Web. El Portlet utiliza en todas sus páginas una estructura en forma de tabla. Es posible que haya que definir estilos para la etiqueta *table* en el CSS si no están definidos previamente,

para diferenciar de alguna forma el título del resto de contenido del Portlet (por ejemplo con colores, letra más grande, ...), etc. Por ejemplo, el código anterior con etiquetas de estilo en una página Web concreta podría quedar así:

```
<pcdriver:portlet portletName="reservaTutorias"
          applicationName="pfc-tutorias-portlet" >

  <div class="templatemo_post">

    <div class="templatemo_post_top">
      <h1><b style="color:#ccff00"><pcdriver:title/></b></h1>
    </div>

    <div class="templatemo_post_mid">
         <pcdriver:render/>

         <div class="clear"></div>

    </div>

    <div class="templatemo_post_bottom">
         <span class="post">* Requiere tener JavaScript
            activado en el navegador</span>
    </div>

  </div><!-- end of templatemo_post-->

</pcdriver:portlet>
```

Los estilos para la etiqueta *table* redefinidos en el CSS de esta página Web son:

```
.templatemo_post_mid table{
  /* Para que se vea bien el texto del contenido del Portlet
     con la letra en el color del resto de las páginas */
  color: #FFFFFF;
  /* Para que haya espacio entre el recuadro
     (templatemo_section_box) y el comienzo del texto. */
  margin : 0px 15px 15px 15px;
}
```

E.2.2.2 Integrar el Portlet en una página Web utilizando un IFrame

Esta segunda estrategia de integración se basa en mostrar directamente la aplicación Portlet utilizando la etiqueta HTML *IFrame* como si se tratara de un Web Widget [73].

Para ello, simplemente deberemos añadir el siguiente código HTML (entre las correspondientes etiquetas de estilo para posicionar el Portlet donde queramos):

```
<iframe src="http://alkaid.cps.unizar.es:8280/pfc-tutorias-portlet"
        frameborder="0" scrolling="no" id="FramePortletReservaTutorias"
        height="100%" width="100%">
        <p>Su navegador no soporta iframes.</p>
</iframe>
```

E.2.2.3 Utilizar el Portlet remotamente usando WSRP

Como tercera y última estragia de integración, el Portlet instalado se ofrece para ser utilizado remotamente gracias al estándar de servicios Web para Portlets remotos (WSRP [35]). Para más información sobre WSRP consultar la sección A.2.2 del anexo sobre la tecnología de Portlets.

Esta posibilidad se puede utilizar para integrar el Portlet si se posee un portal Web o un servidor Web con un contenedor de Portlets instalado. Para ello, habrá que hacer uso del consumidor WSRP del portal Web o contenedor de Portlets concreto. Siga las instrucciones de su portal para realizar esto. La descripción del servicio Web del Portlet en formato WSDL [80] se encuentra en la ruta `http://alkaid.cps.unizar.es:8180/producer/wsrp/wsdl/portletReservaTutoriasRemoto`. Esta dirección será la que habrá que introducir en el consumidor WSRP ofrecido por el portal Web o contenedor de Portlets.

E.2.3 Instalación de la aplicación Portlet en Liferay Portal

En esta sección se asume el uso de Liferay, versión Community Edition (gratuita), sobre Apache Tomcat. Se puede descargar todo junto desde la página web de Liferay `http://www.liferay.com/`. También se asumen ciertas nociones básicas sobre un portal Web y manejo previo básico de Liferay en concreto.

Para instalar el Portlet en Liferay, en primer lugar, hay que configurarlo para que utilice como fuente de almacenamiento una base de datos MySQL por medio de JNDI. Para ello, dentro del directorio de Liferay, iremos al de Tomcat, donde editaremos los 2 ficheros XML de configuración (*server* y *context*) y añadiremos el driver JDBC para MySQL, tal y como se explicó en la sección E.1.1.3. También crearemos en el directorio *webapps/ROOT/WEB-INF/classes* de Tomcat, un fichero de configuración llamado *portal-ext.properties*. En este fichero se especificarán las propiedades de la base de datos a utilizar. El contenido de este fichero será el siguiente:

```
jdbc.default.driverClassName=com.mysql.jdbc.Driver
jdbc.default.url=jdbc:mysql://localhost/igetutor?
useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false
jdbc.default.username=user
jdbc.default.password=hiK9J2lvM
```

En la propiedad *url* del código anterior, pondremos la URL de la base de datos configurada. También se hará lo propio en las propiedades *username* y *password*.

A continuación, se desplegará el Portlet copiando el fichero WAR correspondiente en el directorio *deploy* de Liferay. Para obtener el fichero WAR, se han de seguir los pasos explicados en la sección E.2.2, utilizando la herramienta Maven.

Cuando se inicie el portal, éste instalará automáticamente el Portlet. Para poder añadir el Portlet a una página Web del portal, iniciaremos sesión en el portal e iremos a una página Web pública. Pulsaremos en *Add*, situado en la parte superior izquierda de la pantalla, que despliega un menú donde seleccionaremos la opción *More*. Aparecerán una serie de categorías. Nuestro Portlet se ubicará bajo la categoría *misPortlets*, tal y como se observa en la figura E.3. Pulsando en el nombre de la categoría, aparece el Portlet cuyo nombre es *Reserva de tutorías* y la opción *Add* para añadirlo a la página actual del portal. Tras pulsar Add, el Portlet se añade a la página, tal y como se puede observar en la figura E.4.

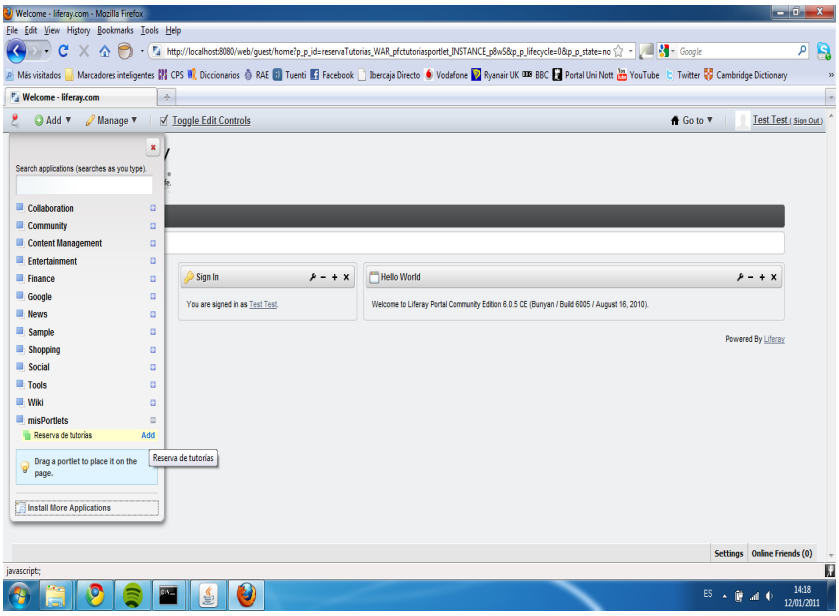


Figura E.3. Pantalla de Liferay para añadir el Portlet a una página Web del portal

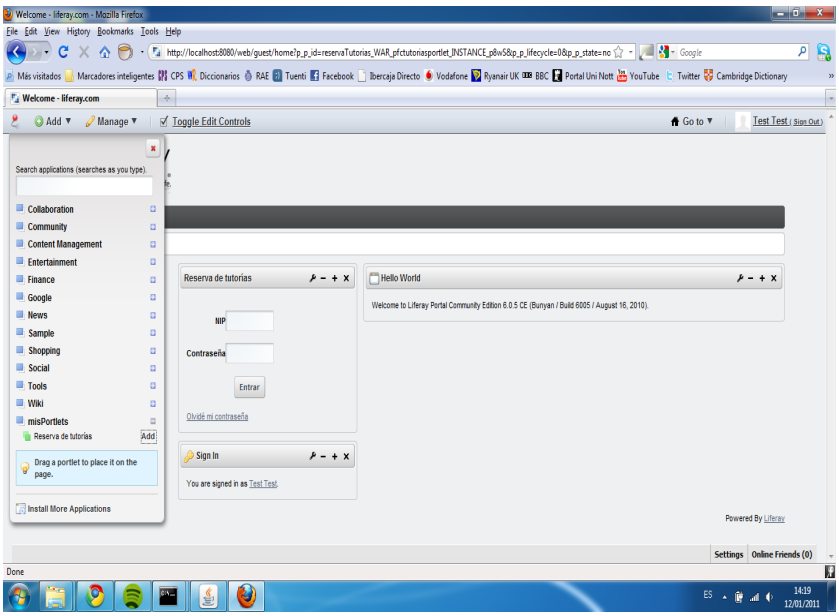


Figura E.4. Página Web de un portal Web Liferay con el Portlet integrado

Anexo F

Seminario de Portlets

En este anexo se incluyen las transparencias del seminario de Portlets elaborado durante la fase de formación y análisis de tecnologías del proyecto.

En la siguiente página comienzan las transparencias. Se han incluido ocho transparencias por cada página.

Seminario de Portlets

Marcos Mainar Lalmolda
Proyecto fin de carrera
4 de Agosto de 2010

Índice de contenidos

Índice de contenidos

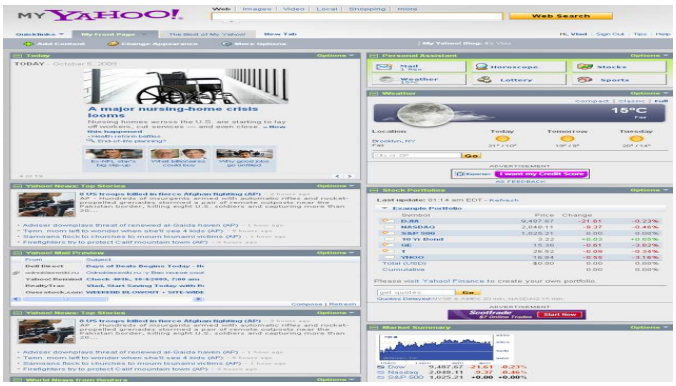
Portales Web (1/2)

- Aplicación Web que presenta información de diversas fuentes de forma unificada
- Portales Intranet (corporativos) vs Internet (iGoogle, Yahoo, MSN)
- Servicios típicos
  - Buscador Web
  - Correo electrónico
  - Noticias
  - Partes meteorológicos
  - Calendario
  - Blogs
  - ...

Portales Web (2/2)

- Integran aplicaciones a nivel de interfaz de usuario
- Funcionalidades genéricas
  - Inicio de sesión *(Single Sign-On)*
  - Agregación de contenido
  - Personalización de contenido
- El portal construye cada página agregando las respuestas de los portlets que contiene

Ejemplo de portal Web



Índice de contenidos

Servidor de portales Web

- Proporcionan un portal pre-construido donde se puede instalar portlets
  - Comerciales: IBM WebSphere Portal, Oracle Portal, Microsoft SharePoint Portal Server, etc.
  - Open Source: Jetspeed2, Liferay, uPortal, GateIn Portal, etc.
- Se instala en un servidor de aplicaciones (GlassFish, JBoss, Weblogic) o en un servidor web con soporte de Servlets y JavaServer Pages (Tomcat, Jetty, etc).
- Decide el aspecto global de las páginas del portal.
- Terminología: *Portal Framework, Portal Platform*, etc.

Índice de contenidos

- Contexto: Portales Web
  - Servidores de portales Web
- Qué son los portlets
  - Contenedor de portlets
- Estándares de portlets
  - Java Portlet Specification (JPS)
  - Web Services for Remote Portlets (WSRP)
- Desarrollo de portlets
- Conclusiones
- Demostraciones de portlets

Qué son los portlets (1/2)

- Componentes software modulares de las interfaces de usuario que proveen una capa de presentación a sistemas de información
- Proporcionan información o un servicio que se incluye en un portal Web
- Producen fragmentos de código *markup* (X/HTML, WML, etc.), no documentos completos
- No son directamente direccionables mediante una URL
- Generan contenido dinámico
- Se renderizan como una tabla HTML
- Locales o remotos al portal Web
- Servicios web orientados a presentación

Qué son los portlets (2/2)

- 2 partes
  1) Decoración: barra de título, controles y bordes de las ventanas de los portlets
  2) Contenido del portlet: la parte generada por la aplicación portlet
- Portlet API: contrato entre los portlets y el portal
- Los portlets interaccionan con el cliente Web mediante el paradigma petición/respuesta
- No pueden generan contenido arbitrario al ser parte de una página de un portal Web
  - Ejemplo: si el portal Web solicita *html/text*, todos los portlets deben generar contenido con formato *html/text*
- Para ejecutarlos se necesita un servidor de portales o al menos un contenedor de portlets

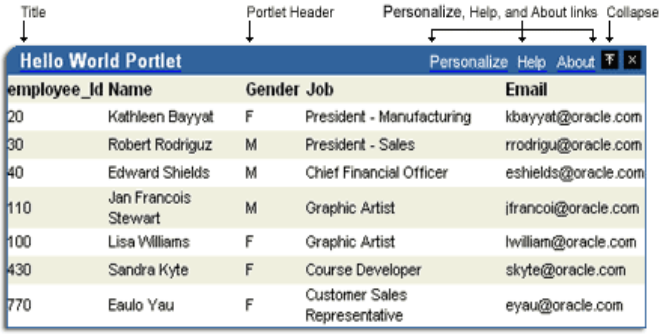
Índice de contenidos

- Contexto: Portales Web
  - Servidores de portales Web
- Qué son los portlets
  - Contenedor de portlets
- Estándares de portlets
  - Java Portlet Specification (JPS)
  - Web Services for Remote Portlets (WSRP)
- Desarrollo de portlets
- Conclusiones
- Demostraciones de portlets

Contenedor de portlets

- Interfaz entre los portlets y el portal Web
- Idea similar al contenedor de servlets
- Donde se instalan los portlets
- Típicamente es un componente del portal Web
- Gestiona y ejecuta los portlets
- Provee al portlet de recursos necesarios y de su entorno de ejecución
- Controla su ciclo de vida
  - Inicializa y destruye los portlets
  - Recibe las peticiones del usuario y las traslada al portlet
- Ejemplos:
  - OpenPortal Portlet Container
  - Apache Pluto

Ejemplo de portlet



Índice de contenidos

- Contexto: Portales Web
  - Servidores de portales Web
- Qué son los portlets
  - Contenedor de portlets
- Estándares de portlets
  - Java Portlet Specification (JPS)
  - Web Services for Remote Portlets (WSRP)
- Desarrollo de portlets
- Conclusiones
- Demostraciones de portlets

Estándares de portlets (1/2)

- Hacen frente a 2 problemas tradicionales que existían en los primeros servidores de portales

  1) Los portlets desarrollados con un determinado portal no podían ser instalados en otro portal
     Estándar: **Java Portlet Specification (JPS)**
  2) Un portal no podía consumir remotamente los portlets instalados en otro portal (todos los portlets debian ser locales al portal)
     Estándar: **Web Services for Remote Portlets (WSRP)**

- La mayor parte de los servidores de portales actuales soportan ambos estándares

Estándares de portlets (2/2)

- Ambos estándares están fuertemente ligados
  - WSRP 1.0 – Septiembre de 2003
  - JPS 1.0 – Octubre de 2003
  - WSRP 2.0 – Abril de 2008
  - JPS 2.0 – Junio de 2008
- Los estándares son totalmente compatibles
- JPS describe el API de Java que permite desarrollar portlets
- WSRP es un protocolo de comunicación entre servidores de portales y contenedores de portlets

Índice de contenidos

- Contexto: Portales Web
  - Servidores de portales Web
- Qué son los portlets
  - Contenedor de portlets
- Estándares de portlets
  - Java Portlet Specification (JPS)
  - Web Services for Remote Portlets (WSRP)
- Desarrollo de portlets
- Conclusiones
- Demostraciones de portlets

Java Portlet Specification 1.0 (1/2)

- API estándar para desarrollar portlets locales en Java
- Estandariza el API que ofrece el contenedor de portlets a los portlets
- Permite delegar la generación de markup en páginas JSP o en un servlet de la aplicación portlet
- Al solucionar el problema 1, permite interoperabilidad de portlets entre diferentes portales Web
- Abarca las áreas de:
  - Agregación
  - Personalización
  - Presentación
  - Seguridad

Java Portlet Specification 1.0 (2/2)

- Permite guardar preferencias de los usuarios en una BBDD interna gestionada por el contenedor de portlets (transparente al desarrollador)
- Modos del portlet
  - Estándares: *View*, *Edit* y *Help*
  - Modos a medida
- Estados de la ventana del portlet
  - Estándares: maximizada, minimizada y normal
  - Estados a medida
- Procesamiento y manejo de las peticiones refinado
  - *Render request* vs *Action request*

Java Portlet Specification 2.0

- Mantiene la compatibilidad hacia atrás con la versión 1.0
- Introduce nuevas funcionalidades:
  - Mecanismos de coordinación entre portlets
    - Eventos
    - Parámetros públicos de renderización
  - Soporte para servir recursos
    - Permite soportar interacciones AJAX
  - Filtros de portlets
  - Anotaciones para los métodos que procesan peticiones de acción, renderización, recursos y eventos
- Sincronizada con WSRP 2.0

Índice de contenidos

- Contexto: Portales Web
  - Servidores de portales Web
- Qué son los portlets
  - Contenedor de portlets
- Estándares de portlets
  - Java Portlet Specification (JPS)
  - Web Services for Remote Portlets (WSRP)
- Desarrollo de portlets
- Conclusiones
- Demostraciones de portlets

Web Services for Remote Portlets (1/2)

- Servicio Web orientado a presentación (datos + lógica de presentación) – opuesto al enfoque tradicional de servicio Web orientado a datos
- Estándar para comunciar con portlets remotos
- Actores
  - Portal Web
  - Productor WSRP
  - Consumidor WSRP
  - Portlet
- Define el API para exportar los portlets de un productor a consumidores remotos
- Permite la integración de portlets procedentes de diversas fuentes

Web Services for Remote Portlets (2/2)

- WSRP define
  - Conjunto de interfaces WSDL que debe implementar un productor para permitir a los consumidores invocar WSRP
    - *Service Description* – obligatorio, descubrimiento
    - *Markup* – obligatorio, interacción
    - *Registration* – opcional
    - *Portlet Management* – opcional, gestionar ciclo de vida
  - Métodos para publicar, encontrar y asociar WSRP y metadatos
  - Reglas para los fragmentos de *markup* emitidos por los WSRP
- Interoperabilidad - productor y consumidor de portlets pueden usar tecnologías diferentes (ejemplo: J2EE y .NET)

Índice de contenidos

- Contexto: Portales Web
  - Servidores de portales Web
- Qué son los portlets
  - Contenedor de portlets
- Estándares de portlets
  - Java Portlet Specification (JPS)
  - Web Services for Remote Portlets (WSRP)
- Desarrollo de portlets
- Conclusiones
- Demostraciones de portlets

Desarrollo de portlets (1/2)

- Una aplicación portlet es una extensión de una aplicación Web J2EE (.war)
- La aplicación Web se instala en el contenedor de portlets o en el servidor de portales
- Cada aplicación portlet contiene 1 o más portlets
- Portlet: clase Java que implementa el interfaz *javax.portlet.Portlet* o hereda de la clase *GenericPortlet*
- Descriptor de despliegue (XML)

Desarrollo de portlets: entorno (2/2)

- NetBeans + Spring + Hibernate + OpenPortal Portlet Container
- Spring Portlet MVC
- Otras herramientas:
  - JUnit
  - HTTPUnit
  - JMeter
  - Doxygen
  - Hudson
  - Maven
  - SVN
  - Trac
- Contenedor de portlets vs servidor de portales

Índice de contenidos

- Contexto: Portales Web
  - Servidores de portales Web
- Qué son los portlets
  - Contenedor de portlets
- Estándares de portlets
  - Java Portlet Specification (JPS)
  - Web Services for Remote Portlets (WSRP)
- Desarrollo de portlets
- Conclusiones
- Demostraciones de portlets

Conclusiones

- Tecnología muy novedosa
- Alta curva de aprendizaje
- Los portlets están totalmente ligados a los portales Web
- Reusabilidad e integración a nivel de interfaz

Bibliografía

- Wikipedia. http://en.wikipedia.org
- Asignatura Tecnología de Portales, Máster de Informática, Universidad de A Coruña. http://www.tic.udc.es/~fbellas/teaching/tp/index.html
- Ashish Sarin, *Portlets in Action*, Manning, 2009-2010. http://www.manning.com/sarin/
- OASIS, *Web Services for Remote Portlets*, OASIS Standard, April 2008. http://docs.oasis-open.org/wsrp/v2/wsrp-2.0-spec.html
- Java Community Process, *JSR 168: Portlet Specification 1.0*, October 2003. http://jcp.org/en/jsr/detail?id=168
- Java Community Process, *JSR 286: Portlet Specification 2.0*, January 2008. http://jcp.org/en/jsr/summary?id=286
- Spring Framework 3.0 Reference Documentation. http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/htmlsingle
- Otras páginas web y blogs de desarrolladores.

Índice de contenidos

- Contexto: Portales Web
  - Servidores de portales Web
- Qué son los portlets
  - Contenedor de portlets
- Estándares de portlets
  - Java Portlet Specification (JPS)
  - Web Services for Remote Portlets (WSRP)
- Desarrollo de portlets
- Conclusiones
- Demostraciones de portlets

Bibliografía

[1] A. Sarin. *Portlets in Action*. Manning Publications Co., Early Access Edition, Publicación estimada para Abril de 2011.

[2] M. Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, Indiana, 2002.

[3] Oracle. Core J2EE Pattern Catalog, Data Access Object. `http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html`

[4] Wikipedia. AJAX. `http://es.wikipedia.org/wiki/AJAX`

[5] Oracle, Sun Developer Network (SDN). JavaServer Pages Technology. `http://java.sun.com/products/jsp/`

[6] P. Mellqvist. IBM Developer works. Don't repeat the DAO. `http://www.ibm.com/developerworks/java/library/j-genericdao.html`

[7] D. Wichers. SQL Injection Prevention Sheet. `http://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet`

[8] Oracle. OpenPortal Portlet Container Project User Guide. `http://java.sun.com/javaee/sdk/portletcontaineru3_install.jsp`

[9] F. Bellas Permuy. Apuntes de la parte de Java de la Asignatura Integración de Sistemas, Máster de Informática, Universidad de A Coruña, Curso 2009-2010. `http://www.tic.udc.es/is-java/is-java-2009-2010/index.html`

[10] M. Fowler. Inversion of Control Containers and the Dependency Injection pattern. `http://www.martinfowler.com/articles/injection.html`

[11] J. Shore. Dependency Injection Demystified. `http://jamesshore.com/Blog/Dependency-Injection-Demystified.html`

[12] D. Elliman. The University of Nottingham. Enterprise Level Computing lecture slides and course notes. `http://www.cs.nott.ac.uk/~dge/G53ELC/index.html`

[13] D. Elliman. The University of Nottingham. Software Engineering Methodologies lecture slides and course notes. http://www.cs.nott.ac.uk/~dge/G52SEM/index.html

[14] Wikipedia. Convención sobre configuración. http://es.wikipedia.org/wiki/Convencion_sobre_Configuracion

[15] Wikipedia. Programación extrema. http://es.wikipedia.org/wiki/Programacion_extrema

[16] Oracle. Java EE Technical Documentation. http://download.oracle.com/javaee/

[17] K. Beck, A. Cockburn, M. Fowler, R.C. Martin et ali. Manifiesto por el desarrollo ágil de software, 2001. http://www.agilemanifesto.org/iso/es/manifesto.html

[18] Oracle. The Java Persistence API. http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html

[19] Oracle. JavaServer Faces Technology. http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

[20] Oracle. OpenPortal Portlet Container. http://java.net/projects/portlet-container/

[21] The Apache Software Foundation. Apache Tomcat official page. http://tomcat.apache.org/

[22] Oracle. Java Servlet Technology. http://www.oracle.com/technetwork/java/index-jsp-135475.html

[23] The Apache Software Foundation. Apache Struts. http://struts.apache.org/

[24] The Apache Software Foundation. Apache Tapestry Home. http://tapestry.apache.org/

[25] E. Gamma, R. Helm, R. Johnson, and J.Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[26] JBoss Community. Hibernate. http://www.hibernate.org/

[27] SpringSource Community. Spring. http://www.springsource.org/

[28] Oracle. MySQL. http://www.mysql.com/

[29] Innobase Oy. InnoDB website. http://www.innodb.com/

[30] M. Fowler. Plain Old Java Objects (POJO). http://www.martinfowler.com/bliki/POJO.html

[31] Microsoft Corporation. ASP.NET Web Parts Overview. http://msdn.microsoft.com/en-us/library/hhy9ewf1.aspx

[32] The Dojo foundation. Dojo toolkit official webpage. http://dojotoolkit.org/

[33] The jQuery project. jQuery official webpage. http://jquery.com/

[34] F. Bellas Permuy. Apuntes de la asignatura Tecnología de Portales, Máster de Informática, Universidad de A Coruña, Curso 2009-2010. http://www.tic.udc.es/~fbellas/teaching/tp/index.html

[35] OASIS. Web Services for Remote Portlets Specification Version 2.0, 2003. http://docs.oasis-open.org/wsrp/v2/wsrp-2.0-spec.html

[36] Java Community Process (JCP). JSR 168 Portlet Specification Version 1.0, October 2003. http://jcp.org/en/jsr/detail?id=168

[37] Java Community Process (JCP). JSR 286 Portlet Specification Version 2.0, January 2008. http://jcp.org/en/jsr/summary?id=286

[38] Apache Tapestry. Tapestry Portlet Support. http://tapestry.apache.org/tapestry4.1/tapestry-portlet/

[39] Oracle. Introducing Java Portlet Specifications: JSR 168 and JSR 286. http://developers.sun.com/portalserver/reference/techart/jsr168/

[40] SpringSource. Spring Framework Reference Documentation. Portlet MVC Framework. http://static.springsource.org/spring/docs/3.0.x/reference/portlet.html

[41] Eclipse. Eclipse Portlet Tools. http://portlet-eclipse.sourceforge.net/

[42] Apache Software Foundation. Portlet 2.0 API documentation. http://portals.apache.org/pluto/portlet-api/apidocs/index.html

[43] K. Ramirez. Portlets and Servlets: What's the difference? https://portlet.dev.java.net/files/documents/1654/11398/tip4.html

[44] NetBeans Portal Pack Plugin. http://contrib.netbeans.org/portalpack/

[45] Eclipse Portal Pack Plugin. https://eclipse-portalpack.dev.java.net/

[46] CssTemplateHeaven. Cool racing CSS template. http://www.csstemplateheaven.com/business-templates/cool-racing/

[47] Oracle. Java Naming and Directory Interface (JNDI). http://www.oracle.com/technetwork/java/index-jsp-137536.html

[48] JBoss Community. Hibernate Tools for Enclipse and Ant. http://www.hibernate.org/subprojects/tools.html

[49] Terracota. Quartz Enterprise Job Scheduler. http://www.quartz-scheduler.org/

[50] iText Software Corp. iTextPDF. http://itextpdf.com/

[51] Quality Open Software (QOS.ch). Simple Logging Facade for Java (SLF4J). http://www.slf4j.org/

[52] The Apache Software Foundation. Apache Log4J. http://logging.apache.org/log4j/

[53] The Dojo Foundation. Direct Web Remoting (DWR). http://directwebremoting.org/dwr/index.html

[54] Oracle. JavaServer Pages Standard Tag Library. http://www.oracle.com/technetwork/java/index-jsp-135995.html

[55] Liferay Inc. Liferay. http://www.liferay.com/

[56] JUnit.org. JUnit. http://www.junit.org/

[57] JSON. JavaScript Object Notation (JSON). http://www.json.org/

[58] Oracle. JavaMail. http://www.oracle.com/technetwork/java/index-jsp-139225.html

[59] The Apache Software Foundation. Apache Maven. http://maven.apache.org/

[60] SourceForge. Kile. http://kile.sourceforge.net/

[61] JasperForge. JasperReports. http://jasperforge.org/projects/jasperreports

[62] Wikipedia. Agile Software Development. http://en.wikipedia.org/wiki/Agile_software_development

[63] Wikipedia. Extreme Programming. http://en.wikipedia.org/wiki/Extreme_Programming

[64] Wikipedia. Scrum. http://en.wikipedia.org/wiki/Scrum_(development)

[65] Wikipedia. Agile Unified Process. http://en.wikipedia.org/wiki/Agile_Unified_Process

[66] Wikipedia. IBM Rational Unified Process Unified Process. http://en.wikipedia.org/wiki/IBM_Rational_Unified_Process

[67] Wikipedia. Desarrollo iterativo y creciente. http://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente

[68] Wikipedia. Cross-cutting concern. http://en.wikipedia.org/wiki/Cross-cutting_concern

[69] Wikipedia. Aspect-oriented programming. http://en.wikipedia.org/wiki/Aspect-oriented_programming

[70] Apple. iCal MacOS calendar. http://www.apple.com/macosx/what-is-macosx/mail-ical-address-book.html

[71] Google. Google Calendar. https://www.google.com/calendar/

[72] Google. Google Gadgets. http://code.google.com/apis/gadgets/

[73] Wikipedia. Web Widget. http://en.wikipedia.org/wiki/Web_widget

[74] World Wide Web Consortium (W3C). Página oficial en español. http://www.w3c.es/

[75] World Wide Web Consortium (W3C). WidgetSpecs. http://www.w3.org/2008/webapps/wiki/WidgetSpecs

[76] E.L-C Law, D. Müller, A.V. Nguyen-Ngoc. *Differentiating and Defining Portlets and Widgets: A survey approach.* 2nd Workshop on Mash-Up Personal Learning Environments (MUPPLE-09), Nice, France, September 29, 2009. http://www.cs.le.ac.uk/people/avnn1/papers/LawMN-Mupple09.pdf

[77] ChenilleKit. ChenilleKit Tapestry. http://www.chenillekit.org/chenillekit-tapestry/index.html

[78] Ioko Tapestry Commons Project. Ioko Tapestry. http://tapestry.ioko.com/

[79] Oracle. JDBC. http://download.oracle.com/javase/tutorial/jdbc/index.html

[80] W3C. Web Services Definition Language (WSDL). http://www.w3.org/TR/wsdl

[81] W3C. SOAP. http://www.w3.org/TR/soap12-part1/

[82] OASIS. UDDI. http://www.uddi.org/pubs/uddi_v3.htm

Todas las referencias Web han sido visitadas por última vez el día 10 de febrero de 2011.