Universidad Carlos III de Madrid
Escuela Politécnica Superior

Ingeniería Técnica de Telecomunicaciones,
Especialidad en Telemática

Proyecto Fin de Carrera

Diseño e implementación de una aplicación Liferay 6.2 CE para gestión de Notas.

Autor: Sara Ostos Lobo

Tutor: Pablo Basanta Val

CURSO ACADÉMICO 2014 / 2015

Proyecto Fin de Carrera

Diseño e implementación de una aplicación Liferay 6.2 CE para gestión de Notas.

Autor

SARA OSTOS LOBO

Director

PABLO BASANTA VAL

El TRIBUNAL:

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de _____, en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

SECRETARIO **VOCAL** **PRESIDENTE**

"Un sueño no se hace realidad a través de magia:

Conlleva sudor, determinación y trabajo duro"

Colin Powell

Agradecimientos

La finalización de la carrera y entrega del proyecto supone para mí una gran satisfacción que no habría podido lograr sin la ayuda de toda la gente que me ha mostrado su apoyo durante mis años de andadura.

A mis padres, Maria de la Paz y Santiago y hermano por toda la paciencia que han tenido conmigo, por los esfuerzos que han realizado durante todo este tiempo, y sobre todo, por su apoyo incondicional y facilidades que me han brindado para que todo esto fuese posible algún día.

A mi tutor Pablo, por su orientación, disponibilidad y sus consejos durante todo el proyecto y por haberme brindado la oportunidad de realizar este proyecto que tan provechoso e interesante me ha resultado.

Por último y muy especialmente, a A.V.M, la persona que ha cambiado mi vida y que me ha enseñado “el Código”. El cariño, estabilidad y sin duda, la felicidad que me ha aportado han sido determinantes para afrontar este reto.

En definitiva, Gracias a toda la gente que ha estado a mi lado y que ha hecho que mi sueño se convierta en realidad.

Resumen

El aumento progresivo del uso de las redes de comunicaciones ha permitido a una gran cantidad de usuarios tener a su alcance información y servicios. En los últimos tiempos ha nacido el concepto de sistema de gestión de contenidos (Content Management Systems o CMS) que conoceremos con mayor detalle durante el desarrollo del documento. Se trata de una herramienta que permite la creación y administración de contenidos principalmente en páginas web.

En la actualidad, se ha detectado que el modo tradicional de publicación de calificaciones en los "tablones de anuncios" de los centros docentes podría ocasionar problemas de privacidad, por ello ha surgido la necesidad de desarrollar una plataforma que permita gestionar esta información de un modo más confidencial y ágil para los usuarios.

En este contexto, el presente proyecto consiste en el desarrollo de una aplicación que permite a los docentes de un colegio, instituto o universidad gestionar las asignaturas, los alumnos y sus calificaciones con el fin de tener esta información recogida en una plataforma web.

La aplicación será personalizable por profesor y permite que varios docentes puedan conectarse simultáneamente sin compartir la información, pero cabe la posibilidad de compartir esta información asignando el rol necesario a otros profesores.

La tecnología empleada para el desarrollo de la aplicación Web es el popular CMS Liferay. Es interesante su uso porque permite manejar de manera independiente el contenido y el diseño, lo cual facilita la renovación de su aspecto sin que exija la revisión de su arquitectura de datos o estructura de contenidos. En la parte del servidor, se ha usado Apache Tomcat 7.0 como contenedor de servlets y JSP principal y se ha hecho una incursión sobre Websphere 7.0. Para el almacenamiento de datos persistente se ha hecho uso de una base de datos MySQL. El cliente será, sencillamente, un navegador Web convencional.

Como resultado de esta evolución se ha realizado un proyecto en el que se pueden distinguir dos fases:

I. La primera, orientada a la investigación y estudio del arte, realizando un exhaustivo análisis de las tecnologías existentes en la actualidad en cuanto a herramientas software y gestores de Contenidos se refiere.

   Tras realizar comparativas entre los diversos CMS`S se optó por la opción de crear una aplicación de *portlets* propia y desarrollar los diversos *plugins* que proporciona la herramienta para aprovechar el potencial que nos ofrece.

II. La segunda, consistirá en una inmersión en las tecnologías seleccionadas y en la descripción del proyecto elaborado.

Para finalizar el proyecto se completa con la documentación de la parte práctica, que consistió en la elaboración de manuales de instalación y administración.

Abstract

The progressive increase in the use of communication networks has led to a lot of users have at their disposal information and services. In recent times has emerged the concept of content management system (Content Management Systems or CMS) to know in greater detail throughout the document. It is a tool for creating and managing content on web pages mainly.

At present, it seems that the traditional mode of publication of results in the "taboo" of schools could lead to privacy concerns, so it has become necessary to develop a platform to manage this information in a manner more sensitive and responsive to users.

In this context, this project involves the development of an application that allows teachers from a school, college or university managing subjects, students and qualifications in order to have this information in a web platform.

The application is customizable by professor and allow several teachers can connect simultaneously without sharing information, but it is possible to share this information by assigning the required role to other teachers.

The technology used for the development of the Web application is the popular *CMS Liferay*. It is interesting because it allows use independently manage the content and design, which facilitates the renewal of its appearance without requiring the review of the architecture of data or content structure. On the server side, we used Apache *Tomcat 7.0* as servlet container main JSP and made a raid on *Websphere 7.0*, and for storing persistent data we used *MySQL* database. The customer is simply a conventional Web browser.

As a result of this evolution has done a project in which we can distinguish two phases:

I. The first, research-oriented and art studio, performing a thorough analysis of existing technologies today in terms of software and tools Content managers are concerned. After making comparisons between different CMS`S we chose the option to create an application of own *portlets* and develop the various plugins that provides the tool to exploit the potential it offers.

II. The second will consist of an immersion in the selected technologies and project description elaborate.

To finish the project is completed with the documentation of the practical part, which consisted in preparing manuals for installation and administration.

Tabla de contenidos

Índice de figuras

Índice de Código

Índice de tablas

Capítulo 1

Introducción

A lo largo de esta memoria se repasan las razones y motivaciones que han llevado a la elección del proyecto, se evaluarán distintas herramientas existentes en el mercado y finalmente, se profundizará en la seleccionada para su desarrollo. Se pretende realizar una aplicación web que permita a un profesor gestionar las distintas calificaciones de un año académico.

En este capítulo se describe el punto inicial y los precedentes de este proyecto.

1.1 Motivación

Son diversas las motivaciones que influyeron en la selección de este PFC:

- La primera de ellas fue la posibilidad de crear un portal Web con herramientas Open Source donde no hubiese coste económico, es decir, una plataforma libre que ofreciese licencias abiertas. Además, para el caso de extensiones futuras, sería un punto importante para la interacción.

- Otro de los aspectos y quizás el más relevante, fue el poder sacar partido a los conocimientos laborales como experto en administración Liferay. Se trata de una herramienta con mucho auge en los últimos tiempos y era muy atractivo el hecho de poder investigar y aumentar los conocimientos en Liferay, con la finalidad de aplicar estas nuevas capacidades adquiridas en la empresa en la que trabajo y aumentar mí potencial.

- El lenguaje de programación, Java, me resultaba familiar por lo tanto era interesante su uso para adaptarlo a este nuevo Gestor de Contenidos Web, desarrollando los componentes, un componente de cada uno de los existentes para valorar la complejidad y el grado de personalización.

- Otro punto clave fue la finalidad del uso de la tecnología. Es decir, crear un espacio de trabajo en un entorno distribuido, donde se facilite el trabajo colaborativo, se disponga de una información actualizada y donde exista una comunicación fluida entre sus participantes.

- Académicamente, era interesante aprovechar la oportunidad que se me ha brindado de profundizar en un tema tan interesante como es el desarrollo Web, y que puede ser de gran utilidad en un futuro.

Y, por último me interesaba experimentar con Liferay que es uno de los CMS mejor considerado del momento, así como evaluar algunos de sus límites y su idoneidad para el contexto académico y científico, en particular para una página universitaria simple que podría extenderse a la red universitaria de la Universidad Carlos III.

1.2 Objetivos

El objetivo principal de este trabajo es la realización de una interfaz Web para la gestión de notificaciones mencionado, mediante el uso de la tecnología *Liferay* en su versión 6.2 ante la necesidad que presentaban los docentes de gestionar tanta información y los alumnos de obtener sus calificaciones.

Los distintos objetivos marcados con el fin de lograr el desarrollo de la aplicación son:

- CMS Novedoso

Se pretende emplear para el desarrollo de nuestra aplicación web el CMS Liferay por su gran auge, potencial y liderazgo en el mercado.

- Plataforma Web

Desarrollar una herramienta que permita a profesores gestionar de forma sencilla los alumnos a los que imparta docencia, las asignaturas y sus calificaciones.

- Interfaz Sencilla

Es cierto que con respecto a lo que concierne al cliente, la interfaz es el producto, pero se debe lograr un diseño de esta claro, sencillo, vistoso, amigable y principalmente intuitivo, que ofrezca una agradable experiencia de usuario. Sencillamente, se busca potenciar **el poder de lo simple** puesto que se presupone que el usuario no posee conocimientos previos sobre la tecnología desarrollada.

- Personalizar Espacio de Trabajo

Será un requisito importante que cada usuario pueda personalizar su espacio de trabajo (Site) de modo independiente incluso será aplicado a múltiples dispositivos (Tablet, Dispositivos Móviles)

- Diseño e implementación de la aplicación

Una vez cumplidos los objetivos anteriores estaremos en disposición de diseñar e implementar la aplicación. Se crearán las clases Java y otros ficheros necesarios de la arquitectura y se desplegarán dentro de la estructura de Apache Tomcat.

1.3 Estructura de la Memoria

Para facilitar la legibilidad de la memoria, se ha distribuido en diversos bloques, esquematizados del siguiente modo:

- Bloque I: Introducción

Bloque inicial donde se presenta la motivación del proyecto y se establecen los objetivos a alcanzar, dando una idea global del trabajo que se va a realizar. Se describen los recursos software y hardware necesarios, así como un esquema de la estructura de esta memoria.

- Bloque II: Estado del arte

En los capítulos que conforman este bloque se describen los distintos componentes empleados para la realización del portal web. Conoceremos diversos servidores web, sistemas de gestión de bases de datos relacionales, sistemas de gestión de contenidos web así como tipos de autenticación que ofrece Liferay y un *framework* propio del producto (*AlloyUI*) que facilita la creación de componentes dentro del portal.

- Bloque III: Requisitos y Casos de Uso

Detalle de los requisitos funcionales de la aplicación implementada. Se indica el acceso a la aplicación, los tipos de usuarios existentes así como los permisos que le otorgaremos a cada uno de ellos, los requisitos de creación, edición y búsqueda de datos así como la visualización de los contenidos dentro del portal.

- Bloque IV: Portal Web - Liferay

Este bloque es sin duda uno de los más significativos y de los que mayor contenido nos ocupará. Se realizará un estudio detallado del producto, Liferay, conociendo sus principales características, su arquitectura lógica, el tipo de licencias que ofrece (CE y EE) y se realizará una comparativa entre las dos versiones más actuales de producto (6.1 y 6.2), destacando el motivo que nos ha llevado a decantarnos por la segunda.

Se describirán los módulos existentes y se profundizará en el más característico, conocido como '*Portlet/Widget*'. Conoceremos su ciclo de vida, la gestión de portlets y la arquitectura de estos componentes.

- Bloque V: Entorno de Desarrollo

Se mostrará el entorno que se ha empleado y las herramientas de desarrollo.

- Bloque VI: Análisis y diseño de la aplicación

Se detalla el diseño de las distintas partes de la aplicación: tanto a nivel de datos como la interfaz Web. Se emplearán diagramas tipo UML que explican el funcionamiento de la aplicación, y se mostrará un diseño global.

- Bloque VII: Implementación del portal para la gestión de Notas

Describiremos los diversos módulos de software desarrollados. Se conocerán diversos *portlets*, con diferentes modos de acceso a los datos, un *layout* y un *theme*.

- Bloque VIII: Pruebas y Evaluación

En este bloque se describen y presentan los resultados de las múltiples pruebas realizadas para verificar funcionamiento y tiempos de respuesta de la aplicación.

- Bloque VIX: Conclusiones y líneas futuras de trabajo

Para finalizar, se muestran las conclusiones sobre el trabajo realizado, una breve valoración sobre la tecnología Liferay, un análisis sobre los conocimientos aplicados y adquiridos así como las posibles mejoras u orientaciones para futuros trabajos en esta temática.

Capítulo 2

Estado del Arte

En este bloque se detallarán las tecnologías utilizadas para la realización del proyecto además de las posibles alternativas a estas tecnologías. Antes de comenzar el proyecto se debe investigar acerca de las tecnologías disponibles, pero principalmente, para conocer las potencialmente utilizables. Puesto que se desea realizar una plataforma web, se deben tener en cuenta una serie de tecnologías predeterminadas por el cliente. Este es el caso se requería el uso de tecnologías como Liferay [1] y MySQL [2] por parte del cliente. Sin embargo, también se ha considerado necesario mencionar otras tecnologías que podrían haber sido empleadas en lugar de las predeterminadas, ya que ofrecen características similares.

2.1 JAVA

La tecnología Java es simultáneamente un lenguaje de programación y una plataforma. El lenguaje Java es un lenguaje de alto nivel orientado a objetos. Una plataforma Java es un entorno en el cual se ejecutan aplicaciones programadas en código Java. Hay básicamente tres plataformas de este lenguaje [3] [4]:

- *Java SE (Java Platform, Standard Edition)* - Versión núcleo empleada por la mayoría de desarrolladores puesto que contiene todas las librerías y *APIs* básicas. Versión originaria desarrollada por *Sun*.
- *Java ME (Java Platform, Micro Edition)* - Se trata de la versión de desarrollo de aplicaciones bajo dispositivos sobre dispositivos móviles.
- *Java EE (Java Platform, Enterprise Edition)* – Versión más amplia empleada para aplicaciones con gran volumetría cliente/servidor así como para el desarrollo de Servicios Web. Extiende de la versión SE y agrega librerías para acceso a base de datos, servicios web, *portlet y servlets*.

Todas ellas están compuestas por una máquina virtual Java [5] (en inglés *Java Virtual Machine*, **JVM**) y una Interfaz de Programación de Aplicaciones (en inglés *Application Programming Interface*, API).

- *Máquina Virtual*

Es el entorno en el que se ejecutan los programas Java y su principal misión es la de garantizar la portabilidad de este tipo de aplicaciones. Era una característica importante que potenciaba al lenguaje Java frente al resto cuando irrumpió en el mercado de los lenguajes de programación; ya

que permitía **escribir y compilar el programa una única vez** y ejecutar ese código en múltiples plataformas.

Aclarar que al ser Java un programa que se interpreta en una máquina virtual, el archivo resultante de la compilación tendrá una extensión *.class* interpretable por la máquina virtual. Este archivo está escrito en un lenguaje de máquina virtual, conocido como *bytecode*. En la Figura 1 se detalla el proceso de compilación de una aplicación Java.

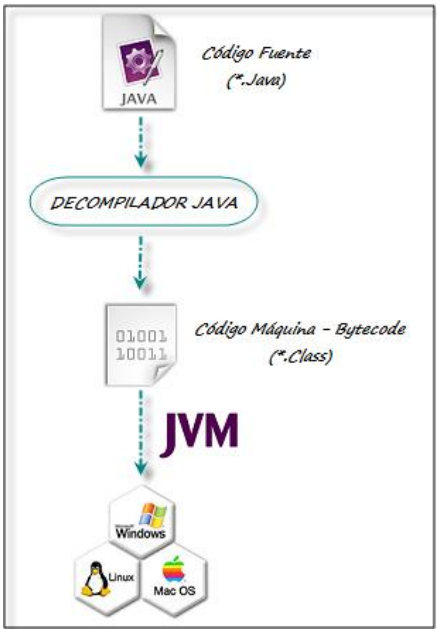


Figura 1 Esquema Compilación Java

La JVM es una de las piezas fundamentales de la plataforma Java. Básicamente se sitúa en un nivel superior al hardware del sistema sobre el que se pretende ejecutar la aplicación, y este actúa como un puente que entiende tanto el *bytecode* como el citado sistema. Así, cuando se escribe una aplicación Java, se hace pensando que será ejecutada en una máquina virtual Java en concreto, siendo ésta la que en última instancia convierte de código *bytecode* a código nativo del dispositivo final.

Cabe destacar la gran ventaja de la máquina virtual, que es la de aportar portabilidad al lenguaje.

- *Interfaz de Programación de Aplicaciones (API)*

Un API [6] no es más que una colección de componentes software que proporciona a los programadores los medios necesarios para desarrollar aplicaciones Java.

Como el lenguaje Java es un Lenguaje Orientado a Objetos, la API de Java provee de un conjunto de clases utilitarias que permiten efectuar toda clase de tareas necesarias dentro de un programa. Dicha API se encuentra organizada en paquetes lógicos, donde cada paquete contiene un conjunto de clases relacionadas semánticamente.

Es muy interesante el empleo de una plataforma Java, puesto que cuenta con las ventajas como: independencia de plataforma, estabilidad, facilidad de desarrollo y seguridad.

Java EE o Java Enterprise Edition es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Java, basándose en componentes de software modulares ejecutados bajo un servidor de aplicaciones.

Java EE incluye varias especificaciones de API, tales como Java Database Connectivity

(JDBC) [7] como interfaz de acceso a base de datos, Servicios Web y define el modo de integrarlos y coordinarlos.

Java EE también configura algunas especificaciones concretas para Java EE para componentes. Éstas incluyen Enterprise JavaBeans, servlets, portlets (siguiendo la especificación de Portlets Java), y varias tecnologías de servicios web. Esto permite al desarrollador crear una Aplicación portable entre plataformas y escalable, a la vez que integrable con tecnologías existentes. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones es el encargado de gestionar la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, es decir, lo que se conoce como el mantenimiento de bajo nivel, dejando al desarrollador el control de la lógica de negocio.

2.1.1 J2EE y Componentes

Java 2 Enterprise Edition (J2EE) es una arquitectura multicapa empleada para implementar aplicaciones de tipo empresarial y aplicaciones basadas en la Web. Se trata de un estándar que dispone de componentes reutilizables, cuenta con un control de transacciones flexibles y proporciona soporte para servicios Web a través de protocolos y estándares basados en XML [8].
En este modelo dichos componentes utilizan servicios proporcionados por el contenedor, que de otro modo tendrían que estar incorporados en el código de la aplicación.

La lógica de la aplicación se divide en componentes según su funcionalidad y estos elementos que constituyen la aplicación J2EE se instalan en diferentes máquinas según la capa a la que pertenezcan como se muestra en la Figura 2. En este marco, nos encontraríamos con las siguientes capas [9]:

- *Componentes de la capa cliente* (*Client Tier*): Ejecutados en la máquina del cliente. Se trata de la interfaz gráfica del sistema y se encarga la interacción con el usuario. J2EE proporciona soporte para diferentes tipos de clientes incluyendo clientes HTML, applets Java y aplicaciones Java.
- *Componentes de la capa Web* (*Web Tier*): Se encuentran en el servidor web y contiene la lógica de presentación que se empleará para proporcionar una respuesta al cliente. Recibe los datos introducidos por el usuario desde la capa cliente y en base a estos genera una respuesta apropiada a la solicitud. Esta capa se apoya en los componentes Java Servlets y JSPs para crear los datos que se enviarán al cliente.
- *Componentes de la capa de Negocio* (*Business Tier*): Se alberga en el servidor de aplicaciones y contiene el núcleo de la lógica del negocio de la APP. Proporciona las interfaces necesarias para emplear el servicio de componentes del negocio. Los componentes de este tipo interactúan con la capa de datos y son típicamente implementados como componentes EJB.
- *Componentes de la capa de Datos*: Software de la capa del sistema de información (*Enterprise Information System Tier, EIS*) que incluye las bases de datos y los sistemas de procesamiento de estos.

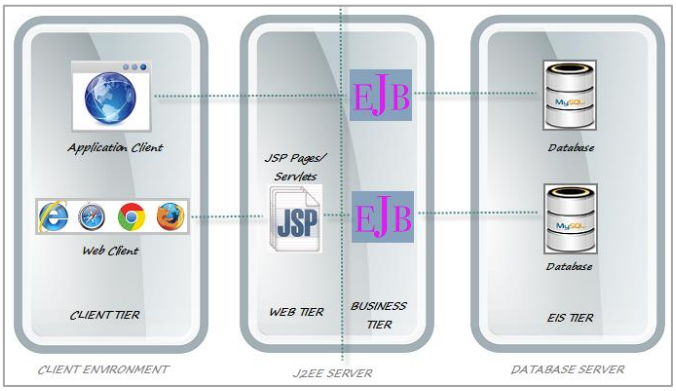


Figura 2 Arquitectura de una aplicación J2EE

Las aplicaciones J2EE suelen ser consideradas en tres capas debido a su distribución: entorno del cliente, servidor J2EE y la base de datos (*backend*).

❖ *Componentes J2EE*

Las aplicaciones J2EE están compuestas de múltiples componentes. Un componente J2EE no es más que una unidad de software funcional auto-contenido que se ensambla dentro de una aplicación J2EE y que se comunica con otros componentes de la aplicación. La especificación J2EE define los siguientes componentes J2EE:

- *Aplicaciones Cliente*: Se trata de programas nativos escritos en lenguaje Java que en general poseen su propia interfaz gráfica.
- *Applets*: Elementos que se ejecutan típicamente en un browser web y proporcionan una interfaz web mejorada para aplicaciones J2EE. Los dos anteriores son componentes que se ejecutan en el lado del cliente.
- *Java Servlet* y *Java Server Pages (JSP)*: Componentes Web que se ejecutan en el lado del servidor para responder a solicitudes HTTP generadas por clientes.
- *Enterprise JavaBeans* (*EJB*): Componentes de negocio que se ejecutan en el servidor de aplicación y soportan transacciones. Encapsulan el acceso al EIS a través del empleo de objetos que proveen la funcionalidad de uso de transacciones y persistencia.

Además de estos componentes principales, J2EE incluye servicios estándar y tecnologías de soporte como:

- *Java Database Connectivity (JDBC): T*ecnología que proporciona acceso a sistemas de bases de datos relacionales de una forma independiente del proveedor.
- *Java Transaction API (JTA)* [10] *o Java Transaction Service (JTS)*: Proporcionan soporte para transacciones a los componentes J2EE.
- *JavaMail*: API que permite la creación de aplicaciones Java para mensajería y envío de correo electrónico con independencia de la plataforma y del protocolo a usar.
- *Java Naming y Directory Interface (JNDI)* [11]*:* Proporcionan el registro y acceso de servicios y objetos. Incluye soporte para *LDAP (Lightweight Directory Access Protocol).*

Para más detalle véase: [12]

2.1.2 Especificaciones Java Portlets

Los portlets o también conocidos como widgets, son "mini" aplicaciones web que se ejecutan en una parte dentro de páginas web. Son los componentes fundamentales ya que es donde reside la funcionalidad de cualquier portal. Liferay es un contenedor de portlets, y como tal, sólo es responsable de la suma del conjunto de portlets que se van a añadir para formar diversas páginas. Esto significa que todas las características y funciones de la aplicación de portal deben estar contenidas en portlets.

Los portlets, como aplicaciones de servlet, se han convertido en un estándar de Java que varios fabricantes de servidores de portales han implementado. El estándar JSR-168 define la especificación inicial, mientras que el estándar JSR-286 define la especificación de portlets 2.0. Un portlet estándar de Java debe poder ser desplegado en cualquier contenedor de portlets que siga la norma. Los Portlets son colocados en las páginas en un orden determinado por el usuario final y son servidos de forma dinámica por el servidor del portal.

Vamos a conocer algunas de las características más destacadas de estas especificaciones [13]:

❖ *JSR 168 (Portlet API 1.0)*

La dependencia con el proveedor iba en contra de la portabilidad de aplicaciones que planteaba J2EE, por ello, en Octubre de 2003 fue aprobado un estándar conocido como Java Specification Request (JSR 168) [14].

De acuerdo con la especificación JSR 168, un Portlet es un componente Web específicamente diseñado para ser agregado en el contexto de una página. Normalmente, diversos Portlets son invocados en una única requisición de una página de un Portal. Cada Portlet produce un fragmento de resultado que es combinado con los resultados de los otros para obtener el resultado de una página de Portal como un conjunto.

La plataforma Java proporciona un estándar que regula el modo en que los portlets interactúan con los contenedores de portlets y garantiza la compatibilidad entre los diferentes productos de portal, siempre y cuando los portlets desarrollados se adhieran a la norma.
La finalidad de la especificación es principalmente la de solventar el problema de la compatibilidad de portlets entre los servidores del portal que ofrecen los distintos proveedores.

Liferay Portal proporciona un contenedor de portlets compatibles 100% que garantiza que cualquier portlet que siga el estándar será capaz de funcionar dentro del portal.

La especificación JSR-168, permite a su vez la interoperabilidad de los portlets entre diferentes portales web. Esta especificación define un conjunto de API`s que permiten la interacción entre el contenedor de portlet y el portlet que direcciona áreas de personalización, seguridad y presentación.
En JSR-168 una aplicación puede agregar varios portlets diferentes y estar empaquetada en un único archivo WAR como una aplicación web estándar de Java. Los portlets de la aplicación se definen en un archivo llamado *portlet.xml* que se coloca en el directorio WEB-INF en el interior del archivo WAR. Este archivo puede ser visto como una extensión al *web.xml* definido en la especificación Java Servlet.

❖ *JSR 286 (Portlet API 2.0)*

El pasado 12 de julio de 2008 se liberó la versión final de la especificación JSR 286 [15]. Se trata de la versión 2.0 de Portlets desarrollada según el *Java Community Process* y creada en alineación con la versión 2.0 de Servicios Web para Portlets Remotos (en inglés Web Services for Remote Portlets *WSRP*). Se trata, como era de esperar, de un evolutivo de la especificación JSR-168.

Con esta revisión lo que se persigue es acercar a la especificación de portlets, otras especificaciones que no estaban disponibles hasta el momento.

Algunas de sus características más importantes son [16]:

- Compatibilidad hacia atrás. Característica realmente importante para la reutilización.
- Comunicación entre portlets empleando Eventos y parámetros públicos: Uno de los problemas que presentaba la versión previa consistía en que la comunicación entre portlets era demasiado básica.
- Soporte básico para AJAX y JSON de forma directa mediante los portlets.
- Soporte a mayor cantidad de frameworks web. A los ya conocidos Spring MVC, Java Server Faces (JSF) o Struts se unen otros más desconocidos como WebWork, empleado para construir interfaces de usuario reutilizables.
- Mejoras en la caché de portlets.

- Definición de CSS a nivel de portlet permitiendo que estos puedan ser portados a otros portales y a la vez se integren mejor en la apariencia del portal.

Para más detalle véase: [17]

2.2 Servidores de Aplicaciones Web

En este punto se indicará el motivo de disponer de un servidor web y que ventajas aporta a nuestra solución.

Como en todos los portales web el uso de un servidor web suele ser altamente recomendable, aunque si se dispone de un servidor de aplicaciones las peticiones no tienen por qué ser necesariamente tratadas por un servidor web.

El servidor Web, también definido en ocasiones como servidor HTTP, es el componente fundamental de cualquier aplicación Web. Gracias a él podemos dar disponibilidad a sitios Web cuando son requeridos por usuarios de la red. Durante el desarrollo del proyecto, un ordenador propio será empleado como hardware del servidor Web. Posteriormente, podrá ser albergada en otro equipo si se desea, en este caso, podría ser situada en un servidor de la universidad Carlos III para su posterior uso por parte de los alumnos y docentes del centro.

Existen multitud de servidores Web como por ejemplo: Internet Information Server (IIS), Websphere o Cherokee. Pero sin ninguna duda, el más utilizado actualmente es Apache HTTP Server, y muchos de los servidores de aplicaciones actuales se basan en él.

2.2.1 Apache Tomcat

En este apartado detallaremos Apache Tomcat, por su amplio uso y Websphere, ya que el entorno laboral del que formo parte, se utiliza este servidor web para las implantaciones del software. Se realizará una comparativa entre ambos para ver las principales ventajas que aporta cada uno de ellos y comprender su potencial.

2.2.1.1 Introducción y desarrollo de versiones

Apache Tomcat (también llamado *Jakarta Tomcat*) funciona como un contenedor de servlets. Implementa las especificaciones de los *servlets* y de *Java Server Pages* (JSP) de *Sun Microsystems*, es decir, se trata de un servidor web con soporte de JSPs y servlets.

En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día esta percepción ha cambiado y Tomcat es empleado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de una máquina virtual Java.

Como cualquier producto, cuenta con diversas versiones existentes, echando la vista atrás, mencionar, que las primeras distribuciones de Tomcat fueron las versiones 3.0.x. Las más recientes son las definidas como 7.x, y soportan las últimas especificaciones de Servlets y JSPs. Se incluye a

continuación una lista de las distintas versiones con algunas de sus características más reseñables (Fuente: [18]).

Tomcat 3.x (distribución inicial)

- Implementado a partir de las especificaciones Servlet 2.2 y JSP 1.1
- Recarga de servlets
- Funciones básicas HTTP

Tomcat 4.x

- Implementado a partir de las especificaciones Servlet 2.3 y JSP 1.2
- Contenedor de servlets rediseñado como *Catalina*
- Motor JSP rediseñado con *Jasper*
- Conector *Coyote* (Servirá para comunicar el Servidor Web Apache con Tomcat)
- Java Management Extensions (JMX), JSP y administración basada en Struts

Tomcat 5.x

- Implementado a partir de las especificaciones *Servlet 2.4 y JSP 2.0*
- Recolección de basura reducida
- Análisis rápido JSP

Tomcat 6.x

- Implementado de *Servlet 2.5 y JSP 2.1*
- Soporte para *Unified Expression Language 2.1*
- Diseñado para funcionar en *Java SE 5.0* y posteriores
- Soporte para Comet a través de la interfaz *CometProcessor*

Tomcat 7.x

- Implementado de *Servlet 3.0 JSP 2.2 y EL 2.2*
- Mejoras para detectar y prevenir "fugas de memoria" en las aplicaciones web.
- Limpieza interna de código.
- Soporte para la inclusión de contenidos externos directamente en una aplicación web.

En el proyecto se ha utilizado la versión 7. Puede consultar los detalles de instalación y configuración en el ***Anexo 1***.

Tomcat se caracteriza por su alta modularidad como también por su agilidad en el tratamiento de peticiones. Además se dispone de varios módulos que pueden añadir funcionalidades.

2.2.1.2 Arquitectura y ficheros de configuración

Tomcat se constituye como una jerarquía anidada de componentes, donde un mismo tipo de elemento puede aparecer en múltiples lugares de la jerarquía. Ésta se organiza según una estructura de carpetas del modo en que aparece en la Figura 3.

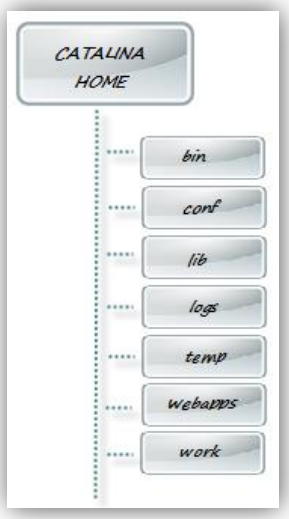


Figura 3 Arquitectura de Tomcat

- */bin* - Contiene los elementos necesarios para el arranque o cierre del servidor, y otros scripts. Se trata del directorio donde se encuentran el ejecutable de Tomcat, scripts y pre compilador de JSP (*Jasper*), entre los más importantes se encuentran:

  - *catalina.bat*: Fichero que alberga los parámetros del arranque. Permite tener un control más profundo sobre la ejecución de Tomcat, ya que se puede ejecutar en primer plano, en modo de depuración. De hecho, el script startup.bat ejecuta catalina.bat y le pide que se ejecute en segundo plano.
  - *startup.bat*: Script de arranque. Se emplea normalmente para ejecutar Tomcat como proceso en segundo plano.
  - *shutdown.bat*: Script de parada.

- */conf* - Ficheros XML y los correspondientes DTD para la configuración de Tomcat. Desde esta ubicación se realiza la configuración global para todas las aplicaciones. Los ficheros más destacados que encontramos bajo este directorio son:

  - *server.xml*: Fichero de configuración principal de Tomcat. Proporciona la configuración inicial para los componentes de Tomcat y especifica la estructura de Tomcat, lo que significa, permitir que Tomcat arranque y se construya a sí mismo ejemplarizando los componentes especificados en *server.xml*.
  - *catalina.policy*: Políticas de seguridad.
  - *catalina.properties*: para modificar la estructura de directorios.
  - *web.xml* – Descriptores de despliegue de aplicación web *(Web Application Deployment Descriptor).*
  - *context.xml* – Opciones de configuración específicas de Tomcat.
  - *tomcat-users.xml* – Base de datos de usuarios y contraseñas. Permite la creación de usuarios/contraseñas y roles.

- */lib*: Directorio que alberga las distintas librerías necesarias y donde podemos añadir las propias para disponer de funciones extra. Dentro de esta ruta están ubicados los ficheros .jar y las clases accesibles a Tomcat y a todas las aplicaciones web. Esto incluye los JAR de producto y las interfaces de programación de aplicación Servlet y JSP (API). Éste es el lugar

en que se colocan los JAR compartidos por las aplicaciones Web o los JAR de JDBC para establecer conexiones.

- */logs* - Contiene el historial de registros del servidor, incluyendo los logs de Catalina y de las aplicaciones. Este directorio almacena los ficheros (log) de registro que se generan en tiempo de ejecución: *Catalina.{yyyy-mm-dd}.log, localhost.{yyyy-mm-dd}.log*.
- */temp*: Ruta de almacenamiento temporal para la máquina Java. Principalmente es gestionado por el servidor para manejar archivos temporales.
- */webapps* - Directorio donde se alojan las aplicaciones web. Si se está utilizando un WAR descomprimido, es necesario colocarlo en este directorio, cuando esto se hace, entonces Tomcat lo desplegará. Cuando se despliega un fichero WAR completo mediante la aplicación de gestión de consola o el *Tomcat Client Deployer*, el fichero WAR también acabará en este directorio.
- */work* - Almacenamiento temporal de ficheros y directorios. Es generado automáticamente por Tomcat, y se trata del lugar donde Tomcat sitúa los ficheros intermedios (como las páginas JSP compiladas) durante su trabajo. Si borramos este directorio mientras se está ejecutando Tomcat no podremos ejecutar páginas JSP.

2.2.1.3 Configuración de Aplicaciones Web

Para que Tomcat pueda interpretar los componentes de nuestra aplicación y ejecutarlos cuando sean requeridos, es necesario incorporar estos componentes en el contenedor. Este proceso recibe el nombre de "despliegue de la aplicación". Se deben tener en cuenta dos aspectos básicos para la realización del mismo: la estructura de directorios que la aplicación debe respetar y un fichero especial denominado descriptor de despliegue.

Para cada aplicación Web que se desee instalar en Tomcat, se debe generar un nuevo directorio bajo el directorio webapps (normalmente recibe el nombre de la aplicación) de la instalación de Tomcat. Este nuevo directorio será específico para la nueva aplicación y define su contexto. A partir de aquí la estructura es la mostrada en la Figura 4.

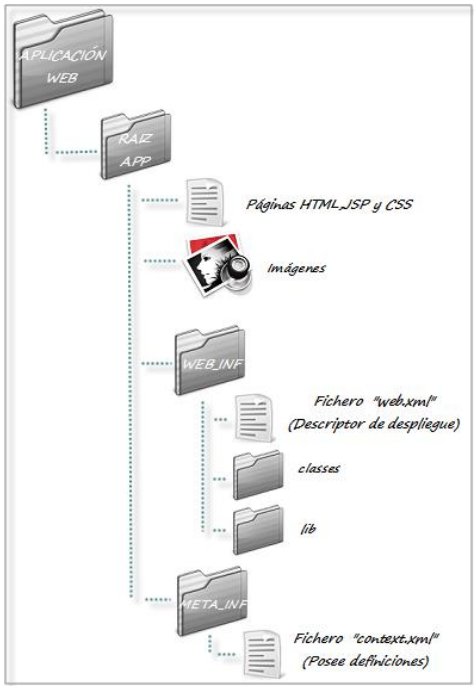


Figura 4 Estructura de una APP Web

- /*Nombre App* (Carpeta raíz): Contiene la parte pública de la aplicación: documentos, HTML, JSP, Hojas de estilo CSS, código JavaScript, imágenes, etc. Los ficheros que se incorporen en este directorio podrán ser accedidos directamente a través de la Web por los usuarios. Una buena práctica es organizarla en varios subdirectorios (para los documentos HTML, CSS).

- /*WEB-INF*: Contiene la parte privada de la aplicación, que no podrá ser accedida directamente por los clientes. Aquí se debe ubicar el descriptor de despliegue de la aplicación (web.xml), las clases compiladas (.class), las bibliotecas de clases (lib) y cualquier otro archivo que no se desee hacer público.

- /*WEB-INF/classes*: Aquí se encuentran los ficheros compilados tales como *servlets* o *beans*, de las clases utilizadas por la aplicación Web.

- /*WEB-INF/lib*: En él se colocan otras bibliotecas de clases adicionales (comprimidas con jar) que utilice tu aplicación.

- /*META_INF*: Suele contener el archivo "Context.xml" que contiene las definiciones necesarias para la aplicación web. Se suele generar automáticamente por los *IDEs* como Eclipse o Netbeans (Si no existe, Tomcat usa los valores por defecto).

El despliegue de aplicaciones web en Tomcat se hace mediante empaquetados con extensión .war. Un fichero WAR (*Web Archive*), no es más que el directorio completo de la aplicación comprimido en formato .jar, de hecho, se construye ejecutando el comando jar de Java, o con herramientas alternativas como *Ant* o *Maven*. Tales ficheros son compatibles con cualquier plataforma de ejecución de servlets y JSPs.

2.2.2 Websphere

IBM WebSphere Application Server (**WAS**, servidor de aplicaciones WebSphere) ofrece opciones para un entorno de ejecución de servidor de aplicaciones Java más veloz con fiabilidad y flexibilidad mejoradas para la creación y ejecución de aplicaciones. WAS está construido a partir de tales como J2EE,XML, y Servicios Web.

Esto funciona con varios servidores web incluyendo *Apache HTTP Server*, *Microsoft Internet Information Services(IIS)*, *IBM HTTP Server para i5/OS*, y también *IBM HTTP Server* para el sistema operativo *AIX/Linux/Microsoft Windows*.

Puede ser empleado para dar soporte a entornos compuestos por un único servidor y configuraciones de tamaño medio, al igual que para despliegues de gran tamaño que necesitan agrupación en clúster de nivel web en múltiples instancias de servidor de aplicaciones.

Las principales características que ofrece son:

- Aumentar la productividad del desarrollador con estándares abiertos y amplias modelos de programación, incluidas opciones ligeras para despliegues web.
- Desplegar y gestionar aplicaciones y servicios sin la restricción de tiempo, ubicación o tipo de dispositivo.
- Incluye el perfil *Liberty*, un perfil de servidor de aplicaciones web dinámico.
- Mejorar la seguridad y el control mediante gestión integrada y herramientas administrativas.

2.2.3 Tomcat vs Websphere

Apache Tomcat e *IBM Websphere Application Server* son dos productos muy diferentes que no obstante surgen con frecuencia en las mismas conversaciones.

En este módulo, vamos a revisar algunas de las funcionalidades en las que estos servidores de aplicaciones se superponen, así como algunas que las distinguen, para terminar indicando cual ha sido la decisión seleccionada, por considerarla la mejor opción para nuestro escenario.

Originalmente desarrollado y lanzado en la misma época, a finales de 1990, y que procediendo de la misma especificación Java Servlet, Apache Tomcat y WebSphere Application Server, han crecido en dos direcciones totalmente dispares. Mientras que Tomcat se ha mantenido como un contenedor de Servlets ligero y de código abierto, Websphere se ha convertido en un gran servidor de aplicaciones basado en pila. Se trata de un componente que permite interoperar los productos de IBM bajo la misma marca, incluyendo IDEs, motores de integración de datos y más funcionalidades.

Cuando llega el momento de decantarse por uno u otro, hay un montón de opinión para todos, pero gran parte de la discusión puede ser reducido a una serie de puntos centrales [19]:

- **Servidores de aplicaciones y servlets Contenedores**

En un comienzo, Websphere y Tomcat eran muy diferenciados por las tecnologías Java que empleaban, pero los nuevos desarrollos en el paisaje de Java han generado un acercamiento entre ambos en términos de funcionalidad Java. Por tanto, los parámetros para decantarnos por uno u otro producto están cada vez más centrados en torno a otros factores.

- **La simplicidad, flexibilidad, rendimiento y valor**

Los límites entre contenedores de servlets y servidores de aplicaciones se han vuelto cada vez más borrosa en los últimos años. Un experto equipo de arquitectos y desarrolladores consideraría factible desarrollar una infraestructura basada en Tomcat sabiendo que proporcionaría funcionalidades que reflejan muchos de los ofrecidos por Websphere utilizando marcos, extensiones y herramientas personalizadas. Pero, cuando la tecnología no es un límite firme, ¿cuáles son los otros factores de diferenciación entre Websphere y Tomcat? Todo se reduce a la mejor solución para el trabajo, es decir, en elegir la solución adecuada para nuestras necesidades.

En última instancia, la elección entre Websphere y Tomcat es una elección entre dos tipos diferentes de riesgos.

- Cuando se alojan las aplicaciones en Websphere, se están aceptando un conjunto de riesgos conocidos, incluidos los gastos potencialmente prohibitivos y alta sobrecarga de rendimiento. Pero aun enfrentándonos a estos riesgos, se obtienen una serie de beneficios a cambio, que incluye una plataforma conocida con estabilidad comprobada, un gran conjunto de herramientas de administración GUI, y un contrato que incluye garantías de servicio.

- Por el contrario, al alojar las aplicaciones en Tomcat, se estaría eligiendo una solución que podría disminuir los riesgos proporcionados por la elección de Websphere, así como ofrecernos algunos posibles beneficios adicionales, tales como la reducción de costes, un mayor rendimiento, un menor costo y una mayor flexibilidad para el desarrollador.

Sin embargo, mientras que algunos de los beneficios de alojar la aplicación en Websphere, tales como la estabilidad administrativa, también pueden llegar a ser los beneficios de una infraestructura de Tomcat, otros, como el apoyo de un gran proveedor, o la garantía de servicio, más complejos de lograr, y algunos de ellos, como la posibilidad de comprar todos sus servicios de un solo proveedor,

será prácticamente imposible de lograr. Además, podría incluso conllevar uno o más de los siguientes riesgos: la necesidad de producir una documentación propia y la necesidad de apoyar nuestra propia infraestructura, o la compra de apoyo.

Pero, ¿si hay tantas incógnitas, porque casi todos los sitios basados en Java, (más del 60%) se ejecutan en Tomcat?

La respuesta es que para la mayoría de los desarrolladores, son múltiples las ventajas de emplear Tomcat frente a Websphere, tales como la libertad de usar el IDE a su elección, en lugar de IDEs de propiedades de Websphere, los tiempos de reinicio de servidor rápidos que ahorran mucho tiempo en el desarrollo, y la flexibilidad para ampliar la red del modo más beneficioso para el caso de uso.

El hecho del asunto es que la gran mayoría de las aplicaciones web no requieren la mayoría de las características que ofrece Websphere, ni necesitan el "valor añadido" de marca o reputación. Se suele necesitar un alto rendimiento, infraestructura simple, de bajo costo que funcione correctamente y aporte facilidad de crecimiento a corto plazo. En un gran número de casos, una infraestructura Tomcat bien diseñada puede satisfacer todas estas necesidades.

Ahora que hemos hablado de los aspectos técnicos del debate Websphere frente Tomcat, pasemos a un tema más pragmático: el precio.

- **Precio y Licencias**

El punto más decisivo para la elección entre Tomcat y Websphere es el coste. Mientras que Websphere es un producto patentado astronómicamente caro, Tomcat es una tecnología gratuita y de código abierto.

Websphere cuenta con licencias de pago, pero, la enorme inversión inicial necesaria para conseguir una infraestructura Websphere es un factor importante en la generación de lo que se conoce como "los proveedores de tecnología" - cuando una empresa se ve obligado a seguir empleando el software de un proveedor específico, o comprar productos de extensión integrados de esa compañía en lugar del de un competidor, únicamente por el gran desembolso realizado en productos de ese proveedor.

Hay dos modos de pensar acerca de este fenómeno. Una empresa con solvencia económica (como el encontrado en el entorno laboral, el banco Santander), podría considerar interesante el hecho de que pueden obtener la totalidad de sus soluciones a través de un único proveedor - para ellos, el elevado precio es simplemente el costo de utilizar una solución bien establecida, y si alguna vez necesitasen migrar a un servicio diferente, tendría el amparo económico para financiar el traslado. Por otro lado, un número mucho mayor de empresas vería este aspecto como perjudicial, no sólo desde un punto de vista de costes, sino también desde un punto de vista de las operaciones que conlleva.

Para las empresas en pleno auge que no tienen la estimación de la complejidad a la que llegará su arquitectura, Tomcat es una apuesta segura, ya que permite obtener una iteración muy estable de la infraestructura y funcionamiento con una pequeña inversión inicial.

- **Popularidad**

A continuación podemos ver la popularidad de los servidores de aplicaciones (APPS), en el que destaca visiblemente Tomcat frente al resto. Figura 5 obtenida de una comparativa realizada en 2014 de diversos servidores de aplicaciones. (Fuente: [20]).

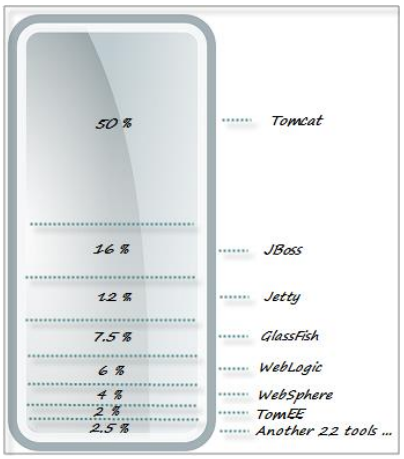


Figura 5 Aplicaciones más usados en 2014

En resumen, con respecto a los puntos analizados, hemos optado por implantar el servidor de aplicaciones Tomcat 7 que se incorpora en el *bundle* de Liferay.

Este servidor ya se encuentra personalizado y configurado para funcionar correctamente con el CMS. Por otro lado este contenedor de servlets ya nos proporciona una solución a los requisitos definidos que debe poseer el sistema que tenemos, por lo que se ha decidido desechar el resto de alternativas.

Por este motivo se cree oportuno adoptar esta solución, principalmente por el ahorro en tiempo de configuraciones necesarias para su implantación.

2.3 Gestión de Bases de Datos Relacionales

Un *sistema de gestión de bases de datos (SGBD)* es un conjunto de programas que facilitan el almacenamiento, modificación y extracción de información en una base de datos, además de proporcionar herramientas para agregar, eliminar, editar y analizar los datos.

Los SGBD deben permitir:

- *Definición de base de datos*: Especificar tipos, estructuras y restricciones de datos. Deberá proporcionarse un mecanismo que se encargue del control de la concurrencia de acceso y un mecanismo de recuperación frente a fallos.
- *Construcción de la BBDD*: Almacenamiento de los datos en algún medio controlado por el propio SGBD.
- *Manipulación la base de datos*: Debe posibilitarse a los usuarios el almacenamiento de datos, su uso, a través de la realización de consultas así como la actualización de la información sin necesidad de conocer su estructura física.

➢ *Objetivos*

Existen algunos objetivos que deben cumplir los SGBD y se recogen en el esquema de la Figura 6.

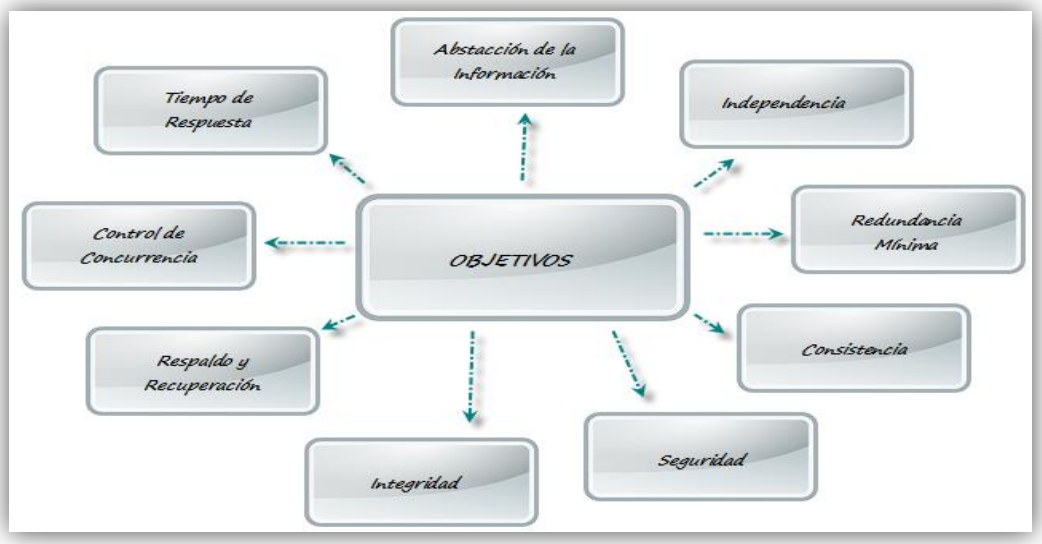


Figura 6 Principales Objetivos SGBD

- ***Independencia***. Es sin duda uno de los mayores beneficios de emplear SGDB. Los niveles de abstracción son útiles para explicar la independencia de datos: Capacidad de editar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

  Así, las aplicaciones están "aisladas" y no deben preocuparse de cómo se encuentran los datos estructurados y almacenados.

  - *Independencia lógica de los datos:* Proporcionan protección ante modificaciones en la estructura lógica de los datos.

  - *Independencia física de los datos*: Proporcionan protección ante modificaciones en las estructuras físicas de los datos.

- ***Redundancia mínima***: Para que una base de datos sea efectiva es necesario reducir en la medida de lo posible las redundancias, es decir, las repeticiones que puedan llevar a error para evitar problemas de inconsistencia entre las distintas versiones de los mismos datos. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de contenidos duplicados.

- ***Consistencia***. En aquellos casos en los que no se ha logrado eliminar la redundancia, se requerirá vigilar que la información duplicada se actualice de modo coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

- ***Seguridad e integridad.*** Es fundamental que los SGBD garanticen que la información almacenada se encuentra segura. La seguridad de los datos implica protegerlos frente a operaciones indebidas que puedan poner en peligro su definición, existencia, consistencia o integridad, con independientemente del usuario que lo produzca. Esto se logra mediante mecanismos que permiten gestionar el acceso y actualización de los mismos sin necesidad de modificar o alterar el diseño del modelo de datos; definido de acuerdo a los requisitos del sistema o aplicación software.

- ***Respaldo y Recuperación.*** Un SGBD proporciona las herramientas necesarias para la conservación de copias de seguridad de cada fichero como prevención ante posibles caídas del sistema.

- ***Control de Concurrencia.*** En la mayoría de los sistemas, lo más habitual es que muchos usuarios accedan a una base de datos, bien sea para recuperar información o para almacenarla. La ejecución concurrente de varios programas de usuario es necesaria para tener un buen rendimiento del SGBD. La concurrencia de acciones de diferentes programas de usuario podría conducir a inconsistencias en la base de datos.

  El SGBD asegura que estos problemas no surjan ya que los usuarios trabajarán con el sistema como si de un sistema monousuario se tratase.

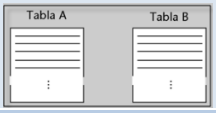
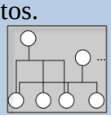
- ***Tiempo de respuesta***. Es deseable minimizar el tiempo que el SGBD tarda en retornar la información solicitada y en almacenar los cambios realizados.

- ***Abstracción de información***: Los SGBD enmascaran a los usuarios detalles acerca del almacenamiento físico de los datos. De este modo, la volumetría de las base de datos son transparentes para los usuarios.

➢ *Clasificación de los SGBD:*

Esta clasificación está basada en el modelo de datos en que está basado el SGBD. Los modelos más habituales se muestran en la Tabla 1.

Tabla 1 Clasificación de SGBD

MODELO LÓGICO	NÚMERO DE PUESTOS	CANTIDAD DE USUARIOS	ÁMBITO DE APLICACIÓN
Jerárquico: Datos organizados en estructura conocida como "árbol". En este, cada nodo puede tener un único "padre". 	***Centralizado:*** Los datos se encuentran en un único PC.	***Monousuario:*** Únicamente atienden a un usuario a la vez. 	***Propósito General***
Relacional: Los datos son almacenados en un conjunto de tablas. 	***Distribuido:*** La base de datos real y el propio software del SGBD pueden estar distribuidos en varios sitios conectados por una red.	***Multiusuario:*** Atienden a varios usuarios al mismo tiempo. 	***Propósito Específico***
De Red: Los datos se organizan como un grafo: las colecciones de son los nodos y las relaciones entre estos se representan a través de conjuntos. 			

La mayoría de los SGBD comerciales actuales se constituyen en el modelo relacional. El modelo Jerárquico y el de Red han caído más en desuso porque requieren del conocimiento previo por parte del usuario de la estructura física de la base de datos a la que se accede, mientras que el modelo relacional lo evita. Se define el modelo relacional como *declarativo* (se especifica los datos a obtener) y los modelos de red y jerárquico son *navegacionales* (se especifica el modo de obtener los datos).

Actualmente, los modelos de datos suelen ser centralizados y orientados a múltiples usuarios.

2.3.1 SGBD libres

En la actualidad, existen muchos SGBD. Algunos de software libre son: *PostgreSQL, SQLite Apache Derby*. Otros SGBD que ofrecen alguna versión reducida de libre disposición son: *MySQL, y Oracle*. Se van a desglosar *PostgreSQL* y *MySQL* y se va a realizar una breve comparativa entre ambos.

❖ *PostgreSQL*

Sistema de gestión de base de datos relacional orientada a objetos de software libre.

Algunas de sus principales características son, entre otras:

- ☑ ***Alta concurrencia.*** Mediante un sistema denominado *MVCC* (Acceso concurrente multiversión, por sus siglas en inglés) *PostgreSQL* permite que mientras un proceso agrega datos en una tabla, otros puedan paralelamente acceder a la misma sin producirse bloqueos .
- ☑ ***Soporte para transacciones distribuidas.*** Permite a *PostgreSQL* integrase en un sistema distribuido formado por varios recursos (por ejemplo, una base de datos PostgreSQL, una cola de mensajería IBM MQ JMS) gestionado por un servidor de aplicaciones donde el éxito de la transacción global es el resultante del éxito de las transacciones locales.

- ☑ ***Funciones con retorno.*** *PostgreSQL* soporta funciones que retornan "filas", donde la salida puede tratarse como un conjunto de valores que pueden ser analizados del mismo modo que una fila obtenida a través de una consulta o *Query*.

❖ *MySQL*

MySQL (My Structured Query Language o lenguaje de consulta estructurado) es un sistema gestor de bases de datos.

La virtud fundamental y clave de su éxito es que se trata de un ***sistema de libre distribución y de código abierto.***

A pesar de carecer de algunas características avanzadas disponibles en otros SGBD del mercado, es una opción atractiva para la mayoría de aplicaciones precisamente por su sencillez de uso y tiempo reducido de puesta en marcha. Esto, unido a su libre distribución bajo licencia GPL le otorga un alto grado de estabilidad y un rápido desarrollo.

Algunas de las principales ventajas y desventajas que ofrece se recogen en la Figura 7.

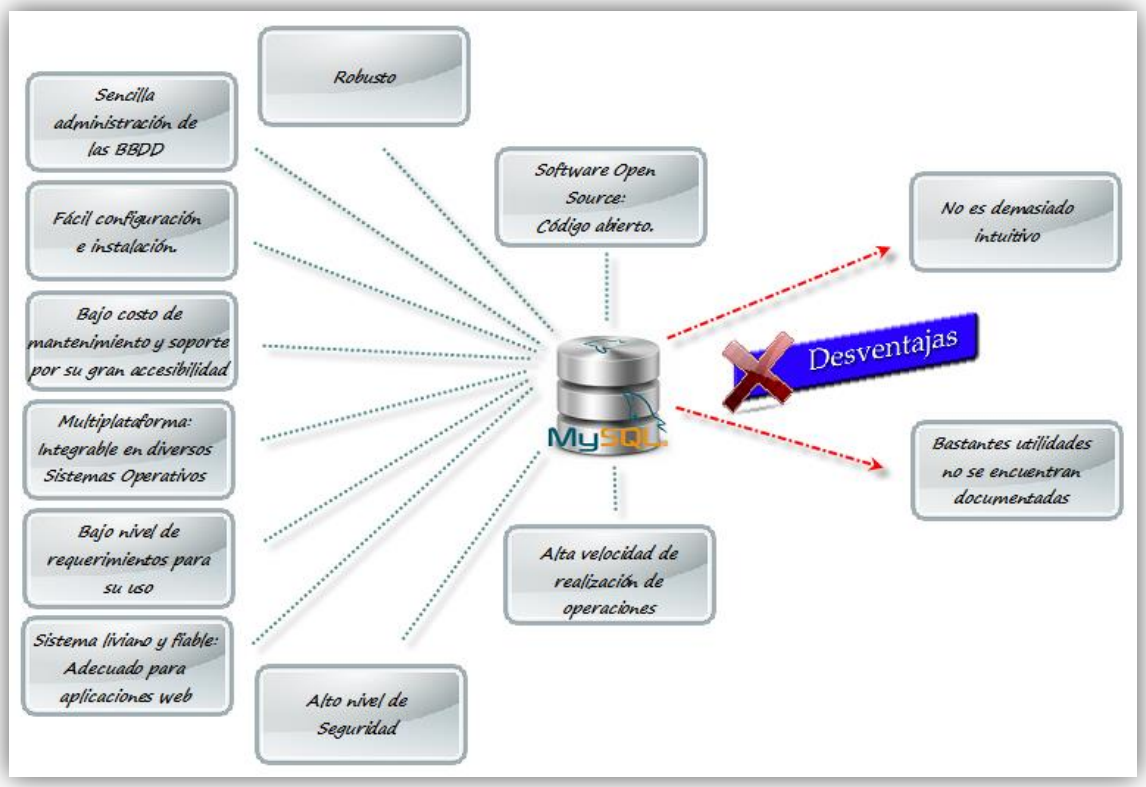


Figura 7 Ventajas e inconvenientes uso MySQL

❖ *PostgreSQL vs MySQL*

Si nos centramos en la usabilidad, apenas existen diferencias en cuanto a funciones que aporta cada tecnología. Pero, en el primero de ellos existen más funciones y resulta más flexible a la hora de realizar conexiones a la base de datos que MySQL.

Este último supera a PostgreSQL en todo lo referente a velocidad, tanto en consultas pequeñas como en las de mayor volumetría (para un elevado número de inserciones con datos aleatorios, en el mismo equipo, MySQL requiere aproximadamente un 25% menos de tiempo). En cambio PostgreSQL destaca en otros campos: tanto a la hora de mantener la integridad referencial, a la hora de usar *triggers* como en los campos de estabilidad y seguridad, a la hora de utilizarlo para desarrollos complejos en cuanto a facilidad de empleo. MySQL tiene más funciones predefinidas, aunque no permite crear tipos de datos definidos por el usuario como si lo hace PostgreSQL. Pero, si nos centramos en el caso que nos atañe, al tratarse de una aplicación web existe baja concurrencia en la edición de datos y en cambio el entorno requiere muchas operaciones de lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones.

Y si nos fijamos en la instalación, tanto en Windows como en Linux el premio es para PostgreSQL.

En resumen ¿Cual es mejor? Es complejo decantarse.

Por las características de cada uno, la conveniencia de usar uno u otro será determinada por el ámbito del uso que se le dará al mismo más que por las características intrínsecas que poseen. PostgreSQL es quizá la opción más indicada para aplicaciones que requieran de mayor seguridad (datos bancarios o sanitarios) mientras que MySQL se lleva la palma en cuanto a velocidad y bajo consumo de recursos, haciéndolo más adecuado para almacenar datos tan críticos, y por ello el seleccionado.

2.3.2 Hibernate

Hibernate es una herramienta de Mapeo Objeto Relacional (ORM (*Object-Relational Mapping*)) y servicio de consultas para Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

En otras palabras, "Hibernate es un Framework que agiliza la relación entre una aplicación y la base de datos".

Hibernate proporciona un lenguaje de consultas muy poderoso llamado "Hibernate Query Language" (HQL [21]). Se trata de un lenguaje muy similar al SQL estándar, con la diferencia de que es completamente orientado a objetos (se emplean nombres de clases y sus atributos en lugar de nombres de tablas y columnas), por lo que presenta un puente elegante para poder usar elementos como herencia, polimorfismo y asociaciones.

Hibernate también permite expresar consultas utilizando SQL nativo. Soporta todos los sistemas gestores de bases de datos SQL y se integra sin restricciones con los más populares servidores de aplicaciones J2EE y contenedores web.

Vamos a conocer las características clave:

- ***Persistencia transparente.*** Hibernate puede operar proporcionando persistencia de modo transparentes en entornos orientados a objeto para el desarrollador.

- ***Modelo de programación natural.*** Soporta el paradigma de orientación a objetos de una manera natural: herencia, polimorfismo, composición y el framework de colecciones de Java.

- ***Sin necesidad de mejorar el código compilado (bytecode).*** No es necesaria la generación de código ni el procesamiento del bytecode en el proceso de compilación.

- ***Escalabilidad extrema.*** Hibernate posee un alto rendimiento, tiene una caché de dos niveles y puede ser incorporado en un *clúster*.

- ***Lenguaje de consultas HQL.*** Este lenguaje proporciona una independencia del SQL de cada base de datos, tanto para el almacenamiento de objetos como para su recuperación.

- ***Soporte para transacciones de aplicación.*** Permite transacciones largas (aquellas que requieren la interacción con el usuario).

2.4 Spring

Spring es un *framework* de aplicaciones Java/J2EE. El objetivo central de Spring es permitir que los objetos de negocio y de acceso a datos sean reusables, no atados a servicios J2EE específicos de forma que estos objetos pueden ser reutilizados tanto en entornos J2EE (web o EJB) como en entornos de pruebas.

Spring proporciona:

- Una potente gestión de configuración basada en JavaBeans, aplicando los principios de Inversión de Control (*IoC*). Esto hace que la configuración de aplicaciones sea rápida y sencilla.
  Esta factoría de beans puede ser usada en cualquier entorno, desde applets hasta contenedores J2EE. Estas definiciones de beans se realizan en lo que se llama el contexto de aplicación.

- Una capa genérica de abstracción para la gestión de transacciones, haciendo simple la demarcación de transacciones sin tratarlas a bajo nivel. Se incluyen estrategias genéricas para *JTA* y un único JDBC *DataSource*. El soporte de transacciones de Spring no está atado a entornos J2EE.

- Una capa de abstracción JDBC que ofrece una significativa jerarquía de excepciones, lo que simplifica el manejo de errores, y reduce la cantidad de código necesario.

- Integración con Hibernate, en términos de soporte a implementaciones *DAO* y estrategias con transacciones. Todo ello cumpliendo con las transacciones genéricas de Spring y la jerarquía de excepciones DAO.

2.5 Spring MVC

Una vez desarrollado el portlet simple con el IDE de desarrollo de Liferay podemos usar varios frameworks java para desarrollar portlets tales como *Struts, Spring* [22] *o Ice Faces.*

Spring brinda un patrón MVC (*Model View Controller*) para aplicaciones Web bastante flexible y altamente configurable, pero esta flexibilidad no le quita sencillez, ya que se pueden desarrollar aplicaciones sin tener que realizar configurar tediosas.
Para esto se puede utilizar muchas tecnologías ya que Spring ofrece soporte para *JSP, Struts y Velocity*, entre otros.
El módulo Web MVC de Spring presenta algunas similitudes con otros frameworks existentes en el mercado, pero las siguientes características lo convierten en único:

- Spring hace una clara división entre controladores, modelos de *JavaBeans* y vistas.
- El MVC de Spring está basado en interfaces y es bastante flexible.
- Spring no obliga a utilizar JSPs como única tecnología para la vista (View).
- Los controladores son configurados de la misma manera que los demás objetos en Spring, a través de IoC.
- Las capas Web (*Web tiers*) son más sencillas de probar que en otros *frameworks*.
- Simplifica el desarrollo de aplicaciones J2EE al evitar el uso de EJBS.

Spring es un framework modular que cuenta con una arquitectura dividida en capas, entre las que se encuentra la capa DAO que se detallará en puntos posteriores.

Para más detalle véase: [23]

2.6 Sistemas de Gestión de Contenidos Web (CMS)

2.6.1 Definición de CMS

Un Sistema de gestión de contenidos Web, CMS (*Content Management System*) es una aplicación usada para crear, modificar, gestionar y publicar contenido digital multimedia en diversos formatos de modo colaborativo. El gestor de contenidos genera páginas web dinámicas interactuando con el servidor web para definir la página web bajo petición del usuario, con el formato predefinido y el contenido extraído de la base de datos del servidor. Esto permite gestionar, bajo un formato estandarizado, la información del servidor, disminuyendo el tamaño de las páginas para descarga y reduciendo el coste de gestión del portal con respecto a un sitio web estático en el que cada cambio de diseño debe ser realizado en todas las páginas web que lo compongan.

En la Figura 8 vemos un detalle del esquema de trabajo seguido por un CMS.

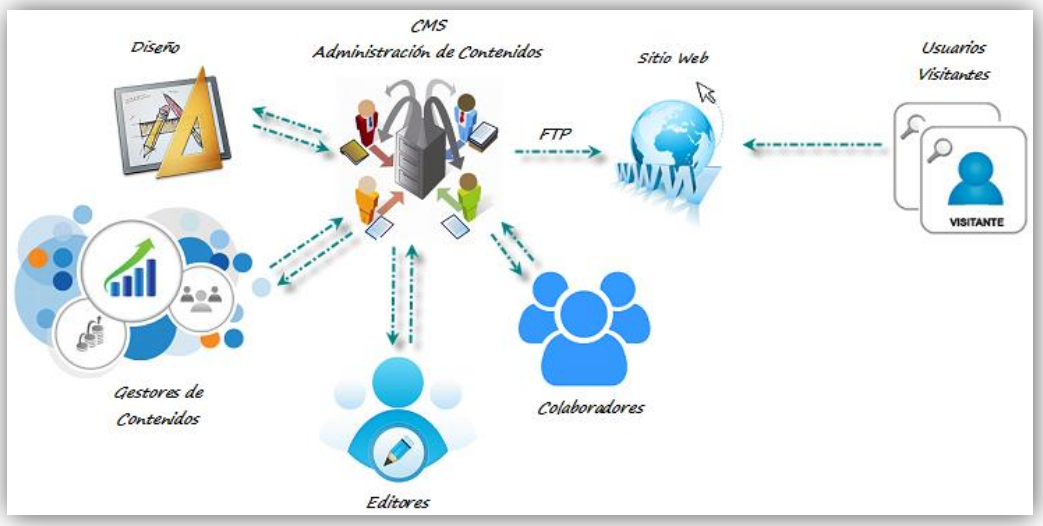


Figura 8 Esquema de trabajo de un CMS

Existen CMS de código libre y otros bajo otros tipos de licencia más restrictivos.

Un CMS tiene dos funciones principales:

- ***Creación, gestión y mantenimiento de contenidos***, provee una serie de herramientas para que la creación de contenidos sea tan sencillo como completar un formulario, y haya, además, un único origen de datos para todos ellos.
- ***Presentación de esos contenidos***, facilita la publicación de contenidos en múltiples formatos a partir de una sola fuente, y añade metadatos a los mismos, para facilitar la navegación en múltiples facetas (temporal y por categorías).

Por estos motivos, los CMS son herramientas ideales para ser empleados en la educación virtual pues posibilitan la creación de entornos *extranets* o *intranets* que pueden ser actualizados, brindando la posibilidad de crear información por usuarios con conocimientos básicos de informática, con

independencia total del personal técnico que la diseña, lo que abre una gran posibilidad para desarrollar una aplicación donde usuarios de todo tipo, en este caso profesores de cualquier especialidad, puedan acceder a un sitio Web y crear información, contenidos referidos a sus intereses sin necesidad de controlar código HTML.

En resumen, existen muchos gestores de contenidos en el mercado, algunos de ellos gratuitos y otros con licencia, pero todos comparten una idea en común: permitir que un usuario pueda gestionar él mismo el contenido de su Web sin la necesidad de conocimientos de programación avanzados. Esta tecnología tiene el potencial para crear un portal web donde el contenido sea dinámico, facilitando la publicación al usuario uno de los propósitos que se tiene con este proyecto.

2.6.2 Estudio de CMS

Existen cientos de soluciones disponibles en el mercado para los gestores de contenidos web. Para el objeto de este estudio se ha realizado un filtro previo y sólo se han comparado los mejores a fecha de la realización del análisis.

De forma genérica, para la selección inicial de los candidatos se han tenido en cuenta tres grandes criterios de selección:

- ☑ ***Código Libre (Open Source)***. En el campo de los gestores de contenidos Web las soluciones software libre han alcanzado sin duda un reconocimiento más que merecido y las soluciones privativas (no libres) no presentan, en general, ninguna ventaja adicional. El software libre no impone un coste de licencia, y cuenta con las grandes ventajas del código abierto: flexibilidad, seguridad y rapidez en las tareas de desarrollo y actualización.

- ☑ ***Cuota de uso y relevancia***. El uso de tecnologías con la mayor comunidad de usuarios posible ofrece importantes ventajas: mayor soporte disponible, mayor número de módulos de terceros desarrollados y garantía de evolución de la plataforma.

- ☑ ***Posicionamiento en el mercado***. Hemos considerado los las soluciones más populares en el mercado.

Debido a esta decisión únicamente se analizarán soluciones software libre, que aportan más beneficios al usuario que las soluciones privativas equivalentes.

La Figura 9 se reflejan las soluciones CMS de software libre con mayor reconocimiento en el mercado.

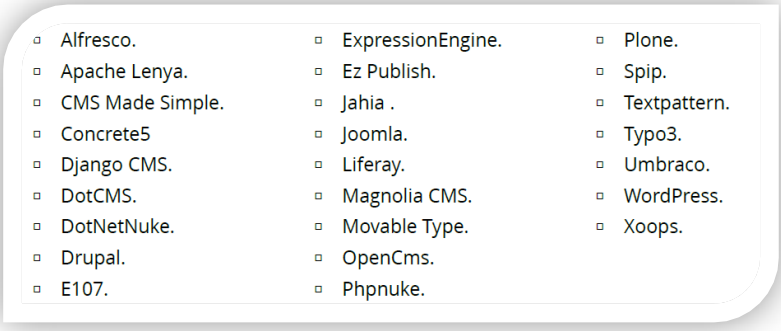


Figura 9 Soluciones CMS con mayor Reconocimiento

Atendiendo a los criterios generales mencionados y tras la investigación del marco referencial, se han seleccionado un grupo de cinco CMS candidatos.

- ☑ ***Drupal***
- ☑ ***Wordpress***
- ☑ ***Joomla***
- ☑ ***Liferay***
- ☑ ***Plone***

Antes de profundizar en el estudio, se debe tener en cuenta que todas las soluciones seleccionadas reúnen unos parámetros de calidad notablemente elevados.

2.6.3 Comparativas de las soluciones seleccionadas

Para realizar la comparativa de los diferentes gestores de contenido se han agrupado en una serie de aspectos diferenciados el conjunto de las funcionalidades y características que tienen este tipo de sistemas. A continuación se describen cada uno de estos aspectos y los puntos más significativos que se han considerado.

- ✓ ***Facilidad de instalación y administración***

  - Sencillez de implantación de portales (instalación y despliegue).
  - Usabilidad de las diversas vistas de administración.
  - Cantidad y calidad de documentación de administración disponible.
  - Soporte para la migración de versiones (actualización de la plataforma a una posterior versión).
  - Soporte visual para la gestión de módulos de terceros incluidos.
  - Soporte visual para la gestión de temas y estilos gráficos soportados.

- ✓ ***Facilidad de uso***

  - Facilidad de inclusión de imágenes y elementos multimedia embebidos en las páginas.
  - Existencia de un control de versiones sobre los contenidos creados para un control evolutivo.
  - Facilidad en la gestión de barras de menú, cabeceras y pies de páginas.
  - Soporte para la gestión de imágenes y documentos agregados al portal. Herramienta para buscar y eliminar recursos que han sido subidos pero ya no se encuentran en uso.
  - Existencia de URLS amigables.

- ✓ ***Potencia Estructural***

  - Facilidad y flexibilidad en la creación de *estructuras jerárquicas complejas* para definir los contenidos dentro del portal.
  - Creación de nuevos tipos de contenidos personalizados.

- Disponibilidad de funcionalidades típicas ya implementadas, como pueden ser: el buscador de contenidos o las redes sociales.

✓ ***Gestión de usuarios***

- Gestión de *usuarios*, *grupos*, *roles y permisos* sobre tipos de contenido y secciones del portal.
- Posibilidad de disgregar las tareas de creación, edición, revisión y publicación de los contenidos.

✓ ***Posibilidades de extensión e integración***

- Sistema para la creación de módulos que extiendan la funcionalidad básica del CMS.
- Disponibilidad de herramientas de desarrollo específicas del CMS que facilitan la creación de nuevos módulos y la extensión de las funcionalidades.
- Facilidad de integración con otros sistemas: posibilidad de uso de diferentes sistemas de persistencia; cohesión con diferentes sistemas de autenticación y autorización (LDAP); existencia de APIs para la conexión con otros sistemas mediante servicios web.

✓ ***Seguridad***

- Recoge aquellas características que posee el CMS, para protegerse frente a la seguridad, como por ejemplo verificación de email, granularidad de privilegios, autentificación LDAP, etc.
- La comunidad del CMS dispone de una metodología y plan precisos para el descubrimiento y reparación de vulnerabilidades y problemas de seguridad.

2.6.4 Valoraciones de los CMS

A continuación se presenta una valoración resumida de las ventajas e inconvenientes de cada uno de los CMS. Indicar que ningún CMS resultará claramente vencedor frente a otro, pues no existe la solución CMS ideal, sino que su idoneidad dependerá totalmente del escenario donde tenga que ser implantada. Sin embargo, sí se ofrecerá un análisis con las virtudes y desventajas más significativas de cada CMS y una puntuación final en forma de tabla.

Hay que reseñar que dentro del amplio espectro existente de gestores de contenidos web, los cinco preseleccionados forman parte del reducido grupo de los mejores CMS *OpenSource* disponibles actualmente. Por tanto, de los gestores analizados no hay ninguno que presente deficiencias significativas en algún campo.

De hecho, hay incluso ciertas características en las que resulta imposible realizar una confrontación entre los contendientes.

En el universo del software libre, los sistemas de gestión de contenidos (CMS) son multitud pero vamos a tratar de aportar una reseña de aquello que particulariza a los cinco seleccionados frente a los demás. Se ha intentado expresar las diferencias existentes entre cada uno para hacerlos encajar en un determinado tipo de portal.

➢ DRUPAL

Sistema de administración de contenidos Web especialmente versátil. Entre 2008 y 2009 Drupal se situó como uno de los CMS referentes del sector empresarial, siendo reflejado por *Gartner* durante varios años dentro del *cuadrado mágico de portales Web*.

A pesar de no ser de los más usados este gestor tiene numerosas ventajas respecto a sus competidores. Algunas de ellas se mencionan a continuación:

- Es el CMS que mejor relación presenta entre la *facilidad* y *usabilidad* de las tareas más típicas de un portal web y la potencia y flexibilidad para construir sitios complejos. Cuenta con una pantalla de administración accesible, lo que facilita mucho la construcción de páginas web(desarrollo amigable).

- Posee una gran capacidad para almacenar, organizar y gestionar grandes volúmenes de contenido.

- Dispone de *un entorno de personalización robusto*, tanto el contenido como la presentación pueden ser tratados de forma individual de acuerdo a unas preferencias seleccionadas por el usuario. La gestión de contenido se realiza como objetos independientes, de forma que puede realizarse un tratamiento individualizado de la información, facilitando su inclusión en cualquier página o permitiendo comentarios específicos sobre cada uno de ellos.

- Como el resto de CMS de su competencia cuenta con módulos con los que agregar cientos de funciones para hacer más completo este gestor y cumplir con las funciones que requeridas por los portales. Cuenta también con un soporte muy avanzado para la creación de nuevos tipos de contenido.

- Los mecanismos de actualización de contenidos son bastante sencillos, permite editar la mayor parte de los contenidos tanto desde el *frontend* como desde el *backend*.

- Otro componente que no podemos olvidar es la "estabilidad". Drupal es escalable sin esfuerzo y es estable incluso mientras proporciona servicio a miles de usuarios simultáneamente.

- El *rendimiento* y la *escalabilidad* son otras de sus señas de identidad: Cuenta con un sistema de cache avanzado, replicación de base de datos, mecanismos de control de congestión configurable para habilitar o deshabilitar módulos, balanceo de carga, etc.

- Desde el punto de vista de la *seguridad*, la gestión de permisos resalta por encima de cualquier otra características; ofrece un sistema muy avanzado y completamente personalizable tanto a nivel de rol como de páginas.

- La comunidad de desarrolladores es otro de los puntos fuertes de Drupal, ofreciendo un desarrollo dinámico y gran soporte.

Pero como cualquier producto, no todo van a ser ventajas. Entre las desventajas que tiene Drupal está que su gran flexibilidad hace de él un CMS que no es tan intuitivo como pueden ser Wordpress o Joomla y tampoco cuenta con la gestión de usuarios que sí ofrecer por ejemplo Plone. Drupal es enteramente un CMS de nivel empresarial. Se recomienda su uso en grandes proyectos donde la estabilidad, escalabilidad y potencia sean más sobresalgan frente a la facilidad de uso y la estética.

➢ WORDPRESS

Es uno de los CMS más conocidos, utilizados y descargados del mercado, el motivo se encuentran es la sencillez de uso e implantación, dirigido a sitios Web de tamaño reducido y de carácter personal, como por ejemplo: sitios corporativos y tiendas online.

Destaca por los siguientes aspectos:

- La personalización es sin duda uno de los puntos fuertes de este CMS, especialmente sencilla a través de la gran variedad de temas adaptables y extensiones.
- Destaca también la protección de la privacidad de los contenidos, a través de la definición de niveles de usuario, protección de contenidos por contraseña y filtros *anti spam*.
- Permite tener una monitorización del entorno, brindando la opción de generar estadísticas de acceso al sitio Web: número de visitantes, lugar de origen de las visitas, páginas visitadas y tiempo de acceso.
- Dispone de un buen soporte a través de abundante documentación y foros, no en vano es una de las comunidades más dinámicas en el contexto de los Sistemas de Gestión de Portales Web. Proporciona una gran librería de plugins.

Pero toda esta facilidad de uso tiene una serie de desventajas. Wordpress, es, a menudo conocido como un "mini CMS". Al contrario que Drupal, está más orientado a un perfil de administrador gráfico, antes que a un perfil de programador técnico, cosa que dificulta la gestión de arquitecturas de la información complejas, vistas elaboradas y el desarrollo de nuevos tipos de contenidos.

Wordpress es además el CMS más débil en cuanto al soporte de roles y *workflow* de publicación de contenidos.

➢ JOOMLA

Joomla es uno de los CMS más utilizados y mejor posicionado del mercado, en principio el proyecto está dirigido a proyectos de pequeña y media envergadura.

Los puntos más resaltables son:

- Tiene bastante potencial pero para poder sacarle partido se requiere cierto conocimiento y experiencia, ya que su máxima versatilidad se obtiene de la integración, adaptación y desarrollo de nuevos módulos.
- Sin lugar a dudas uno de los puntos fuertes de Joomla es su magnífica comunidad. Fruto de la gran participación de los usuarios, el sistema se encuentra en continuo avance frente a vulnerabilidad, nuevas funcionalidades y extensiones. Como *WordPress*, *Joomla* también tiene una fuerte comunidad de desarrolladores. La librería de plugins (llamadas 'extensiones' en Joomla) es muy extensa ya que está formada por muchos plugins gratuitos.
- Es un sistema que ofrece gran versatilidad a través de plantillas, extensiones y adaptaciones.
- Tiene disponibles una amplia variedad de extensiones a través de módulos de terceros, como por ejemplo carros de compra o funciones de comunidad. Las extensiones de Joomla

se dividen en cinco categorías – componentes, plugins, plantillas, módulos e idiomas. Cada una de ellas distinta en función, potencia y capacidad.

Sin embargo Joomla no alcanza la potencia que tienen Drupal y Plone en funcionalidades para la creación de estructuras de portal complejas. Joomla permite construir sitios con más estabilidad estructural y contenido que WordPress, y tiene una interfaz bastante intuitiva. Joomla es una buena opción para pequeñas y medianas tiendas de comercio electrónico, foros, blogs, o sitios con capacidades web estándar. Si quiere algo más potente para uso empresarial, es más recomendable el empleo de Drupal.

➢ LIFERAY

Liferay se sitúa entre los CMS referentes del sector empresarial. Liferay es con diferencia el CMS más utilizado en el mundo Java. Liferay es más que un CMS, es un framework para el desarrollo de aplicaciones Web formado por más de 60 portlets.

Durante varios años consecutivos se ha colocado en las primeras posiciones del cuadrante visionario de portales Web definido por Gartner; en 2011 aparece por primera vez dentro del cuadrante de líderes, destacando en los factores correspondientes a visión de futuro y capacidad para ejecutar dicha visión.

En octubre de 2013**.-** Liferay, Inc. primer suministrador mundial de plataformas web corporativas de código abierto, ha sido *de nuevo* posicionada por Gartner como "Líder" en su Cuadrante Mágico para Portales Horizontales como se observa en la Figura 10.

Es muy reseñable la siguiente cita:

"Por quinto año, la consultora independiente Gartner ha posicionado a Liferay como líder en el Cuadrante Mágico para Portales Horizontales".

Figura 10 Cuadrante Mágico 2014 para portales horizontales de Gartner. "Tomada de [24]"

Como gestor de contenidos, Liferay CMS, está dirigido a todo tipo de escenarios tanto portales corporativos como para el desarrollo de Intranets o nuevas aplicaciones que requieran ser integradas con los sistemas de una organización.

Como todo buen gestor de contenidos, Liferay permite administrar, integrar y publicar información de manera flexible, para ello ofrece todo tipo de funcionalidades.

- Dispone de un sistema de repositorio de documentos y archivos que permite clasificar e identificar los documentos.

- La creación y personalización de los sitios es sencilla, las páginas están formadas por contenido y aplicaciones.

- Incluye un sistema de gestión de Flujos de Trabajo o *Workflow* que permiten coordinar el proceso de la creación, revisión y publicación de contenidos, pudiendo definir reglas a nivel de usuarios y grupos.

- Dispone de múltiples extensiones adicionales que permite aumentar la funcionalidad original del sistema y se encuentran clasificadas según la versión de Liferay que se disponga. Java sobresale en tareas de *integración entre aplicaciones*, contando con una gran cantidad de librerías profesionales, *frameworks* y especificaciones de gran madurez.

- Cuenta además con una característica fundamental para algunos escenarios, como es un soporte muy avanzado para la gestión de varios portales en una misma instalación o servidor (*Multi Tenancy*).

La desventaja de Liferay es que su potencia tiene un coste claro en la complejidad de configuraciones y desarrollos llevados a cabo; en general los portlets de Java son más complejos de implementar que los módulos de otros CMS.

También, se puede decir que el soporte de comunidad es inferior para este CMS que para otros como puede ser Drupal, debido a su reciente aparición. La curva de aprendizaje inicial de Liferay es sensiblemente más elevada con respecto al resto.

➢ PLONE

El uso de Plone está orientado a cualquier tipo de escenario, pero se adapta especialmente bien a escenarios donde se requiere una mayor flexibilidad.

Es, junto con Liferay, el más potente de los CMS estudiados.

- Completa Gestión - Permite corregir o editar en tiempo real un contenido incompleto o erróneo. Dispone de un histórico de acciones que permite deshacer o restaurar contenidos previos. Incluye un motor de búsqueda completo y en tiempo real y ofrece la posibilidad de realizar copias de seguridad fiables, aunque existan usuarios trabajando en ese momento, lo cual puede resultar muy útil.

- Está desarrollado en el lenguaje de alto nivel Python. Este lenguaje es quizás uno de los más versátiles y potentes que existen en la actualidad, pero no está excesivamente implantado a nivel de empresas proveedoras, por lo que puede resultar complejo encontrar desarrolladores expertos en comparación con otros lenguajes más extendidos como Java.

- Ofrece un alto grado de control para soportar estructuras y flujos de trabajo complejos. Y junto con esto, tiene un conjunto de herramientas de administración y edición de gran usabilidad que facilitan el control de los contenidos.

En el lado negativo Plone cuenta con dos desventajas importantes. Primero que su potencia produce gran complejidad de aprendizaje y además, necesita de un entorno de instalación bastante potente para hacerlo funcionar.

Teniendo en cuanta las características ofrecidas por cada uno y las desventajas, hemos considerado adecuado para nuestro desarrollo el empleo de Liferay.

2.7 Sistema de Autenticación LDAP

Se ha considerado interesante dedicar un apartado al análisis de las técnicas de autenticación ofrecidas por el producto debido a la importancia que supone en el entorno que la gestión de la información sea securizada y que dependiendo del rol del usuario que acceda al sistema, se limite el acceso a los datos que visualice, cosa que incorpora dicho sistema.

Las propiedades más reseñables de un servidor LDAP son su óptima velocidad de lectura de datos frente a su pésima velocidad en los procesos de escritura y edición. Estas características lo convierten en un servidor apto para el almacenaje de información que no va a ser modificada en el sistema, pero que puede ser susceptible de sufrir muchas consultas como es el caso del registro de usuarios de la universidad que deberá ser consultado cada vez que un usuario acceda al portal, pero que permanecerá intacto una vez que un administrador de alta un nuevo registro.

Mencionar que Liferay ofrece diversos métodos de autenticación mostrados en la Figura 11: *General, LDAP, CAS, Facebook, NTLM, Open ID, Open SSO* y *SiteMinder y* cuenta con la configuración adaptada a cada uno de ellos como cabría esperar.

Figura 11 Métodos de autenticación Portal Liferay

Resaltar que la configuración de acceso con LDAP podrá ser modificada por el administrador del sistema para poder adaptar el sistema a otros entornos distintos de la UC3M y aportar mayor flexibilidad a este.

Como aclaración importante, indicar que no vamos a entrar a detallar la estructura de un sistema LDAP porque no se ha desarrollado un esquema para ello sino que el sistema se apoya en el existente en la UC3M. En el apartado referente al diseño de la aplicación ofreceremos mayor detalle de la gestión de usuarios realizada.

2.8 Framework Liferay: AlloyUI

AlloyUI [25] es el marco de interfaz de usuario para Liferay Portal, y emplea tres lenguajes de diseño web principales de la pila de aplicaciones para el usuario: HTML, CSS, JavaScript.
El primero de ellos se emplea para definir la estructura de la página, el segundo define la parte visual y el último ofrece la interacción de los elementos de la página.

Ahora que tenemos estos conceptos un poco más claros podemos decir que AlloyUI es un *metaframework* que proporciona una API o entorno de desarrollo estable e intuitivo para la generación de aplicaciones web con acceso a los tres niveles del navegador: estructura, comportamiento y estilo.

Nos hemos apoyado en este framework para crear la interfaz de usuario profesional, puesto que es sencillo de definir al aportar todo tipo de componentes como etiquetas de formularios personalizados, cuadros de texto y *radio buttons*.

Como en todos los campos, no se trata de la única librería existente para facilitar esta funcionalidad, existen muchas bibliotecas *JavaScript* en el mercado tales como *jQuery*, *AUI*, *YUI*, *ExtJs*, *Dojo y Prototype*. En el caso del desarrollo realizado, hemos conocido AUI, pero en el entorno empresarial se ha empleado Dojo.

La Figura 12 muestra algunos de los módulos que componen AUI y que son utilizados para la creación de portales [26].

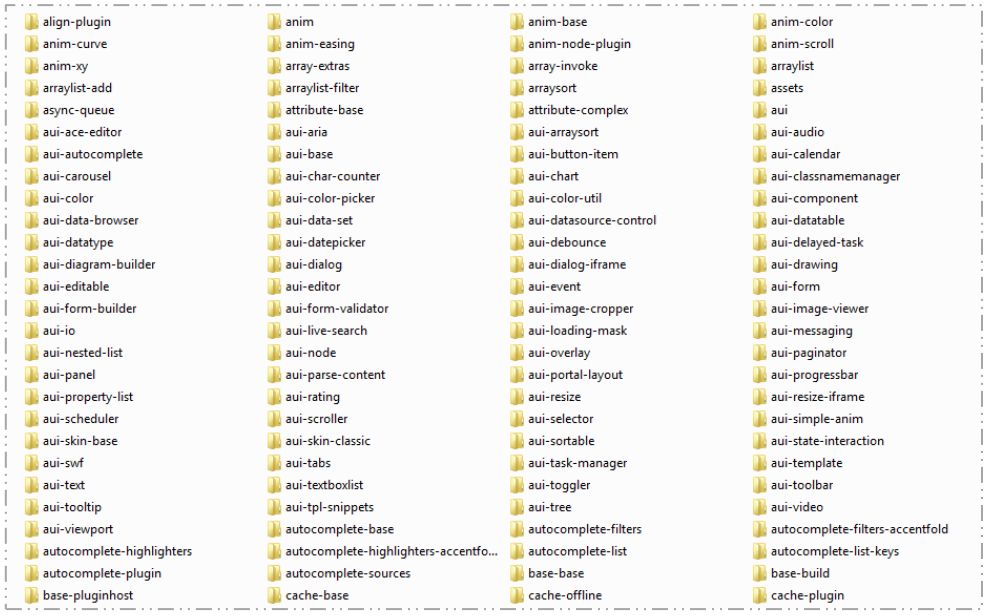


Figura 12 Módulos Librería JavaScript AUI

AlloyUI sigue una implementación modular basada en la carga del elemento requerido en tiempo de ejecución. A la hora de utilizarlo en una aplicación Web, se requiere agregar la línea mostrada en el Código 1 en nuestra página HTML o JSP.

```
<script src="http://cdn.alloyui.com/2.0.0/aui/aui-min.js"></script>
```

Código 1 Tag en JSP/HTML para emplear AlloyUI

Para su integración con Liferay, basta con agregar la línea mostrada en el Código 2 en nuestra JSP:

```
<%@ taglib uri="http://liferay.com/tld/aui" prefix="aui" %>
```

Código 2 Import AlloyUI en Liferay

Las etiquetas AUI se encuentran definidas en el fichero *liferay-aui.tld* y son interpretadas por las clases del paquete *com.liferay.taglib.aui*. Para poder utilizar AlloyUI en nuestros portlets, necesitamos definir Taglib para aui en las páginas JSP.

A diferencia de los componentes HTML originales, los elementos de los formulario comienzan con la cláusula ***aui***, tal y como puede apreciarse en el fragmento del Código 3.

```
<font face="Times New Roman" size="3" color="black"><i><strong>Nombre</strong></i></font></br>
   <aui:input  label="" name="nombre" type="text" maxlength="60">
       <aui:validator name="required" errorMessage="Campo Obligatorio"/>
   </aui:input>
```

Código 3 Elemento input empleando aui

Hay que tener en cuenta que en el momento en que se emplea el uso de esta librería, el atributo de nombre de entrada se anexará con el espacio de nombres del portlet, por lo tanto, las secuencias mostradas en el Código 4 serán equivalentes.

```
A) <aui:input name="nombre" type="text" value="" />

B) <input name="<portlet:namespace/>nombre" id="<portlet:namespace/>nombre"
   type="text" value="" />
```

Código 4 Definición Espacio de Nombres

En posteriores capítulos conoceremos más detalle del uso que hemos realizado en el portal del citado *framework*.

Capítulo 3

Portales Web – Liferay

A lo largo del capítulo, vamos a hacer una inclusión en el producto. Conoceremos las características básicas que lo definen, algunas de sus especificaciones técnicas, su arquitectura lógica. Se realizará una comparativa entre la especificación 6.1 y 6.2 y detallaremos los *Plugins* que lo forman.

3.1 Introducción

Liferay es un gestor de portales de código abierto escrito en lenguaje Java que presenta más de 60 portlets integrados en el núcleo, los cuales facilitan la puesta en marcha de un portal web. Surgió en el año 2000, en principio como solución para las organizaciones sin ánimo de lucro. Actualmente se encuentra en la versión 6.2, en la cual cabe resaltar que incorpora muchas funcionalidades nuevas aunque lo más destacable es su nuevo *diseño sencillo, práctico así como responsivo* y lo que es mejor, han incorporado un *pre visualizador* para se puedan probar los diseños en varios dispositivos sin necesidad de redimensionar la ventana. Podremos ver cómo queda nuestra web, con tan solo seleccionar el dispositivo final, pudiendo elegir entre *tablets* y *Smartphone*. También es bastante reseñable en esta versión la separación de las funciones de administración de portal de las funciones de administración del sitio, lo cual facilita la gestión del portal y también en esta versión de añadió una bandeja de reciclado para poder recuperar contenido que han sido borrado, cosa bastante útil frente a perdida de contenidos irrecuperables sufridas en versiones previas.

Este gestor en su versión 6 va acompañado de una SDK (*software development kit*) y un *plugin* de Eclipse que ayuda a implementar un portal totalmente a medida.

En definitiva, podemos decir que se trata de una solución todo-en-uno de código abierto para la creación de portales web; incorpora herramientas de comunicación, herramientas colaborativas, etc.

3.1.1 Definición de Portal

Un portal se define usualmente como una plataforma de software empleada para desarrollar aplicaciones y sitios web. Los portales actuales han incorporado multitud de características, lo que les convierte en una buena elección para desarrollar un amplio abanico de aplicaciones.

Sin embargo, no todos los entornos web cumplen con los requisitos que hoy día se demandan para que una plataforma pueda considerarse un portal. Los usos más demandados para un portal son los mostrados en la Figura 13.

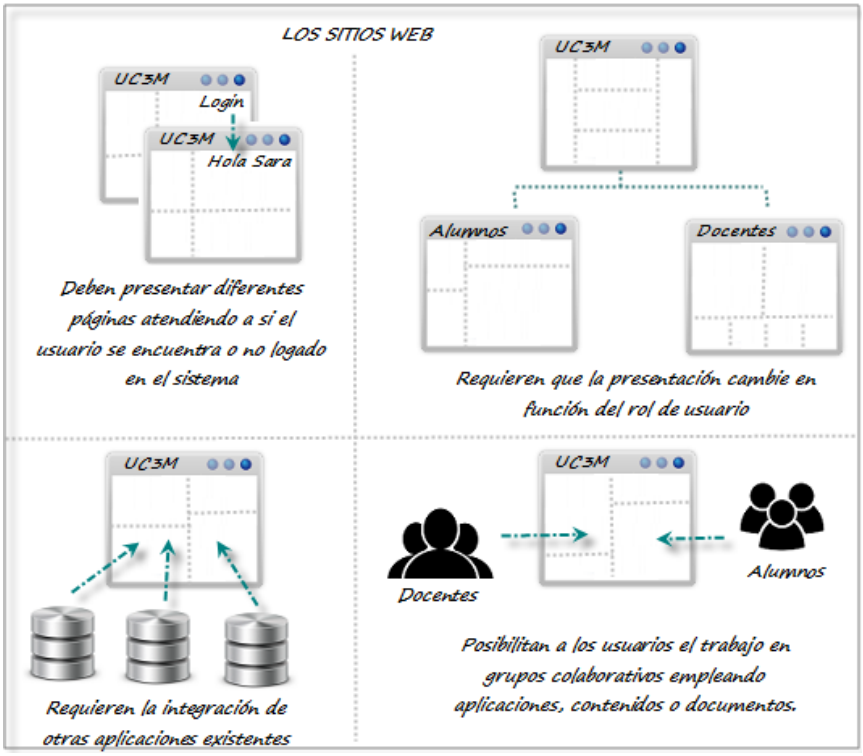


Figura 13 - Requisitos portales web

- Las plataformas de portal facilitan a los usuarios la construcción de páginas y sitios web mediante el ensamblaje de portlets, componentes o gadgets en una página de portal.
- Permiten combinar un tema, un conjunto de páginas, un sistema de navegación (barra de menús, etc.), y un conjunto de portlets y gadgets.
- Es muy útil ya que los administradores cuentan con la posibilidad de crear páginas sin tener que escribir código, simplemente reutilizando portlets y gadgets existentes.

En la Figura 14 mostramos los posibles orígenes de software en un portal.

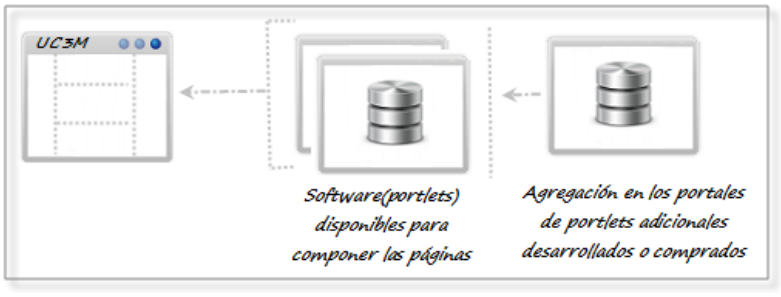


Figura 14 Esquema opciones de Portlets en el portal

- Los portales además simplifican el desarrollo de sitios web que muestran contenido diferente en función del tipo de usuario que accede a ellos. Permitir por tanto, construir sitios web que muestren un contenido distinto dependiendo de si el usuario está identificado en el sistema o no. Tomando como ejemplo un portal universitario como es el caso que nos atañe, podríamos hacer que ciertas páginas estuviesen visibles sólo para determinados usuarios del centro, como los profesores. De esta forma se pueden suministrar páginas

específicas para usuarios que cumplen alguna condición de perfil o rol en el centro educativo como observa en la Figura 15.

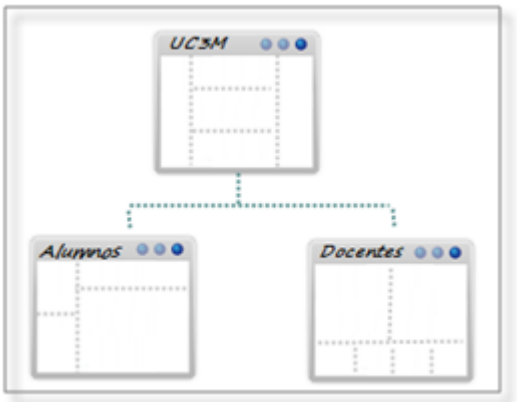


Figura 15 Opciones de visualización de un portal definida por Rol

- Los portales ofrecen a los usuarios la posibilidad de crear páginas, agregarles contenido a través de portlets, y deliberar quién tendrá acceso a ese contenido. Los miembros de un equipo pueden trabajar de forma colaborativa a través de las páginas de su comunidad, en nuestro caso, los docentes contarían con esta ventaja. Un esquema de este funcionamiento se recoge en la Figura 16.

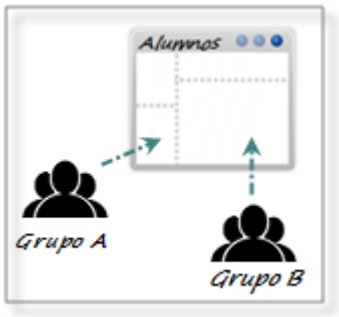


Figura 16 Páginas de Comunidad

- Es posible que una vez hayamos desarrollado nuestro sitio web deseemos que éste pueda se multi-idioma como se esquematiza en la Figura 17 o que se visualice en diversas plataformas (p.ej. *Smartphone*, *tablets*).

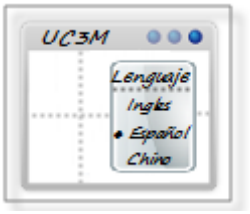


Figura 17 Idiomas distintos, múltiples dispositivos

- Los portales actuales deben también incluir un sistema de gestión del contenido web, incorporando además la característica de procedimientos de gestión del flujo de trabajo del mismo. De esta forma los contenidos pueden pasar por diferentes fases (creación, edición, validación, aprobación) como se muestra en la Figura 18, que, a su vez, pueden ser desempeñadas por usuarios de diversos departamentos dentro de una organización.

Figura 18 Gestión del contenido Web

- Otra característica muy reseñable es la capacidad de los portales para ser utilizados como *Repositorio de Documentos*. De igual modo que los contenidos web, los documentos pueden ser incorporados a un repositorio y puestos a disposición de los usuarios a través de la interfaz web del portal. En este caso, puede ser interesante tener un repositorio para albergar los contenidos teóricos y prácticos junto con los resultados de exámenes. Un ejemplo de ello se muestra en la Figura 19.

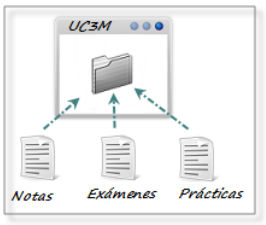


Figura 19 Portal Web como repositorio de Documentos

3.1.2 Funcionalidades del Portal (Características Básicas)

La galardonada plataforma Liferay Portal se diferencia de otras soluciones por su equilibrio óptimo entre funcionalidad, usabilidad, e innovación técnica. El producto ofrece de serie una gestión de contenidos, un entorno de colaboración y una plataforma para redes sociales fácilmente utilizable, gracias a su premiada interfaz de usuario.

Las características que lo hacen tan destacable son las ilustradas en la Figura 20.

Figura 20 Funcionalidades del portal

3.2 Arquitectura Lógica

La arquitectura del producto se puede disgregar en las capas mostradas en la Figura 21.

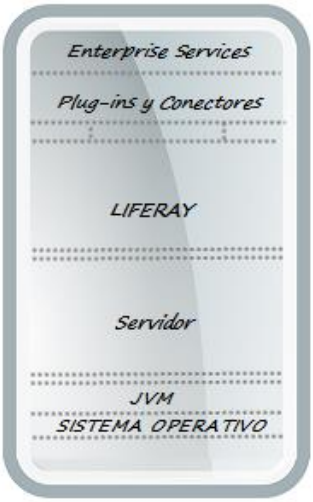


Figura 21 Arquitectura Lógica Liferay

Analizando las particiones que conforman la arquitectura de forma ascendente:

- *JVM* - Como se ha indicado en varias ocasiones en el presente documento, Liferay está basado en Java, así pues corre sobre un sistema operativo que tenga instalada una JVM.
- *Servidor* - Del mismo modo el servidor que habrá sobre la JVM proporcionará múltiples servicios como pueden ser JDBC, JMS y JSP/Servlets entre otros.
- *Liferay* - En la siguiente nivel, nos topamos con el núcleo de Liferay, dando cabida a los túneles que se emplearán para los portlets creados por el usuario, tecnologías y *frameworks* como *Hibernate*, *Spring* y adaptadores de distintos lenguajes (como *RUBY o PHP*) facilitando su integración [22].
- *Plug-ins y Conectores* - Justo por debajo encontramos los plug-ins de los distintos módulos de Liferay (detallados más adelante), el núcleo de administración y algunos conectores con los servicios de la capa superior.
- *Enterprise Services* - Se trata de la capa superior de servicios y componentes agrupados en taxonomías que apoyan y dan cuenta de las funciones empresariales como son la administración del portal, gestión de contenidos y gestión de usuarios.
  También están formados por componentes de servicio interrelacionados y características tales como personalización, colaboración y redes sociales.

3.2.1 Elementos de Arquitectura

Antes de sumergirnos en la interfaz de usuario para la incorporación y mantenimiento de diversos recursos del portal, lo mejor es ir conociendo los conceptos que Liferay utiliza para organizar un portal, detallando los componentes que lo forman:

- *Organizaciones*. La estructura del portal depende de una organización y por tanto, es necesaria la existencia de al menos una para cada portal.

- *Usuarios*. Existen roles (perfiles) definidos por cada tipo de usuario. Cada uno tiene asignados privilegios que indican el nivel de acceso a contenidos y el nivel de uso que puede realizar dentro del portal. Todo usuario tiene definido un perfil y pertenece al menos a una organización.

- *Páginas*. Se trata de la estructura básica de información. Se usan para visualizar los contenidos.

- *Portlet*. Son las aplicaciones que se agregan en las páginas y permiten visualizar los contenidos.

- *Contenidos*. Todo tipo de información susceptible de ser gestionada por Liferay.

La forma más sencilla para comprender la estructura es pensar que se dispone de usuarios y existen diferentes modos de agruparlos. Algunos de estos grupos siguen una jerarquía administrativa organizada mediante roles, y otras agrupaciones, podrán efectuarse por los propios usuarios.

Este modo tan flexible de organizar los conceptos de los portales se puede ilustrar como se muestra en la Figura 22.

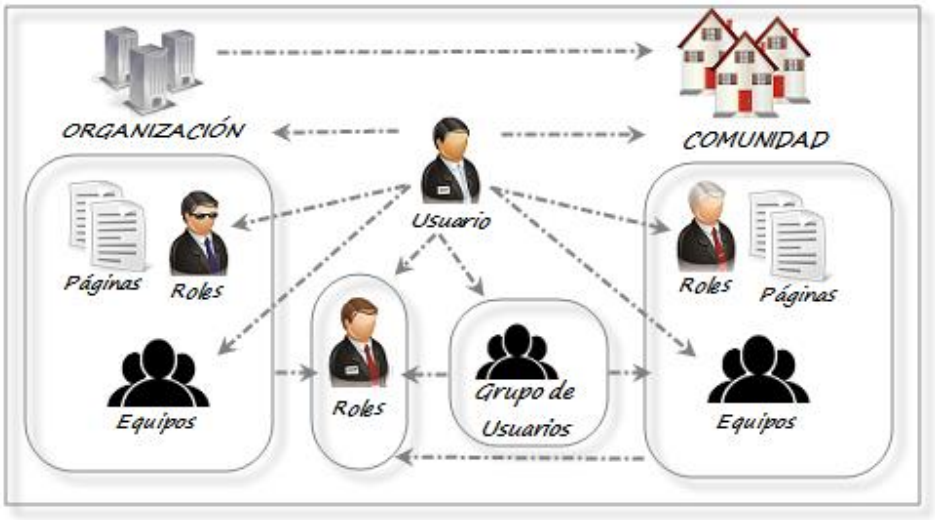


Figura 22 Arquitectura de un portal

En la ilustración, cada flecha se puede leer usando las palabras "puede ser un miembro de." Así que esto significa que las organizaciones pueden ser miembros de las comunidades, las comunidades pueden ser miembros de Roles, los usuarios pueden ser miembros de cualquier cosa, y así sucesivamente. Aunque esto parece muy complejo, proporciona un mecanismo poderoso para los administradores del portal a la hora de configurar los recursos del portal y la seguridad de una manera consistente y robusta.

3.3 Licencias

Liferay se presenta en dos ediciones en el mercado:

- *Liferay Portal Community Edition (CE):* Versión de código libre en la que cualquiera puede incorporar código. Esta licencia brinda la oportunidad de usar su producto, crear plugins y distribuirlos si fuese necesario sin coste alguno.

- *Liferay Portal Enterprise Edition (EE):* Versión de pago proporcionada por la compañía con una calidad del producto, certificada después de realizar sus pruebas y acompañada de soporte al cliente. Como para la mayoría de productos de este tipo, las licencias Enterprise son bastante caras y suelen estar sólo al alcance de proyectos con un presupuesto relativamente elevado.

En nuestro caso, trabajaremos con la versión 6.2 GA2 en el caso de trabajar con CE, pero en el entorno empresarial se ha tenido ocasión de trabajar con las versiones 6.1 GA2 CE y 6.1.2 GA2 en el caso de la EE.

3.3.1 Comparativa Liferay Portal CE & Liferay Portal EE

A la hora de pararnos a realizar una comparativa entre ambas licencias, se puede determinar que la EE es mucho más completa y robusta que la CE a nivel de calidad puesto que la contratación de una licencia EE lleva implícito un servicio de soporte con lo que se cuenta con la posibilidad, en caso de encontrar algún fallo en el producto o bug, de solicitar a Liferay que lo solvente o nos proporcione un soporte para mitigarlo. Liferay proporciona un portal en el que se pueden con las credenciales oportunas abrir tickets de incidencias y son atendidas por usuarios expertos. Por el contrario, si contamos con una licencia de tipo CE y se produce cualquier fallo en el producto deberá ser resuelto sin soporte oficial, resultado mucho más costoso su mantenimiento.

En la Tabla 2 podremos ver algunos de los aspectos que les diferencian en cuanto a software, soporte frente a resolución de problemas y a servicios disponibles.

	CE	EE
DISEÑO CASOS DE USO	Codigo Abierto, Desarrollo, Evaluación	Despliegue en Producción
SOFTWARE	EDICION COMMUNITY	EDICION ENTERPRISE
Herramientas para el Desarrollo	Eclipse IDE Plugin	Liferay Developer Studio
Aplicaciones, widgets y portlets incluidos	60 + portlets simples desarrollados	Cientas de las características incluidas para el flujo de trabajo, formularios web, integración de sistemas y más funcionalidades.
Plugins exclusivos de la empresa		✓ (Ver características adicionales a continuación)
Documentacion completa del producto	✓	✓
Innovaciones de la comunidad	✓	✓
Licencia	Licencia de Código Abierto	Licencia Comercial
Acceso al Código fuente del producto	✓	✓
Indemnización		✓
APOYO A LA RESOLUCIÓN DE INCIDENCIAS		
Ciclos de Prueba	Cálidad Característica	Empresa/Calidad de Producción
El rendimiento a nivel de Empresa y las pruebas de Seguridad		✓
Actualizaciones de Alertas		✓
Alertas de Seguridad y Parches		✓
Actualizaciones de Parches		✓
Service Packs Consolidados		✓
Correcciones de Emergencia (Hot Fixes)		Oro/Plata
Acceso a portal de Clientes		✓
Apoyo basado en la Web		hasta 24x7 (un dia de negocio de tiempo de respuesta)
Soporte Telefónico		hasta 24x7 (tan bajo como el tiempo de respuesta de 1 hora)
CARACTERÍSTICAS ADICIONALES		
Supervisión del Rendimiento		✓
Analítica		✓ Parcial
Auditoría		✓
Integración Motor de reglas		✓
Canales de comunicación agrupados avanzada		✓
Integración Motor de informes		✓
Edición de Terracota liferay		Disponible para su compra
Mayor réplica de memoria caché		✓
Edición de servidor Tcat liferay		Disponible para su compra
Sincronización liferay	✓ Un Sitio	✓ Múltiples Sitios
Diseñador de flujo de trabajo		✓
Formularios de flujo de trabajo Liferay		✓
SERVICIOS DISPONIBLES		
Entrenamiento	Disponible para su compra(Conenido basado en EE)	Disponible para su compra
Liferay Kick Start Program		Disponible para su compra
Liferay Go Live Program		Disponible para su compra
Desarrollo Patrocinado		Disponible para su compra
Instalación		Disponible para su compra
Comentarios Código		Disponible para su compra
Optimización del rendimiento		Disponible para su compra
Comentarios Arquitectura		Disponible para su compra
Actualización / Servicios de Migración		Disponible para su compra
Certificados Partners Consulting		✓
Soluciones Certificación de terceras partes		✓

Tabla 2 Comparativa Liferay Portal CE & Liferay Portal EE. Tomada de [27]

3.4 Tipos de Servidores de Aplicaciones

En el momento de instalar un servidor Liferay se debe determinar sobre qué tipo de servidor vamos a desear que corra la aplicación.
Liferay se distribuye bajo varios tipos de servidor. A continuación se muestran en la Figura 23 los más conocidos junto con algunas de sus versiones más características:

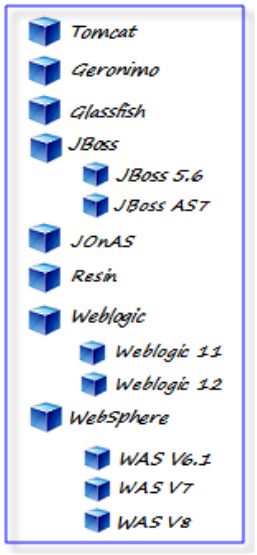


Figura 23 Tipos de Servidores de Aplicaciones

Para el desarrollo del proyecto, hemos empleado *Tomcat* ya que es sencillo de instalar, y fácil de implementar y por tanto, se trata del más extendido, pero a nivel empresarial, se ha empleado *Websphere 7.0* y se ha podido comprobar la complejidad que presenta a la hora de configurarlo para obtener un óptimo rendimiento.

3.5 Versiones de Portal

Existen diversas versiones de portal, y como es lógico, cada nueva versión de Liferay incorpora nuevas funcionalidades en varios ámbitos del portal, aunque normalmente los cambios suelen ser más profundos en alguno de los módulos funcionales de este.

3.5.1 Comparativa entre versión 6.1 y 6.2

En la Tabla 3 se describen los principales ámbitos mejorados por cada una de las versiones, destacando aquellos que hacen una mayor aportación en cada una de las versiones.

Tabla 3 Comparativa entre Versiones de Portal

	LIFERAY 6.0	LIFERAY 6.1	LIFERAY 6.2
Gestión de Contenidos	✓	✓	✓
Gestión Documental		✓	✓
Colaboración		✓	✓
Gestión de Sitios Web		✓	✓
Navegación	✓		✓
Desarrollo	✓	✓	✓
Plataforma	✓	✓	✓
Usabilidad	✓		✓
Movilidad			✓
Auditoría	✓		
SEO y Posicionamiento	✓	✓	✓
Rendimiento		✓	✓

Vamos a realizar una enumeración las mejores funcionales que se han añadido en la versión 6.2 frente a la anterior, 6.1 con el fin de reseñar las causas que han tomado peso para su elección.

- *Diseño Responsivo (Responsive Design)*

Mediante el uso de la aproximación de desarrollo gráfico *Boostrap*, todo el funcionamiento del portal es adaptable a diferentes tamaños de pantalla según el dispositivo de conexión que empleen los usuarios que se conectan. Esta característica, mostrada en la Figura 24 es muy interesante por el auge que han tenido en la actualidad los dispositivos portables.

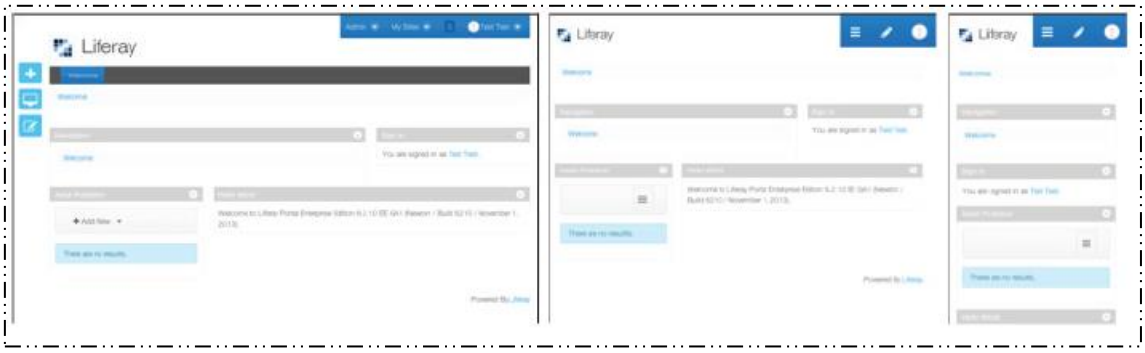


Figura 24 Propiedad Diseño Sensible Liferay V6.2

- *Papelera de reciclaje*

En la última versión, como ya se ha mencionado, se puede recuperar un contenido borrado por error ya que este no se elimina definitivamente si no que se almacena en la papelera de reciclaje.
Es posible definir un criterio para eliminar el contenido tras un determinado período de forma que el sistema borre aquella información más antigua sin necesidad de una supervisión constante.

- *Mejoras de Usabilidad:*

Existencia de un nuevo *Look & Feel*: El diseño, como se puede apreciar por alguien que haya empleado la versión 6.1 es mucho más actual y simple adaptándose a los diferentes terminales existentes. Presenta, como se observa en la Figura 25, una gama de colores más atractiva y un portal de administración más intuitivo.

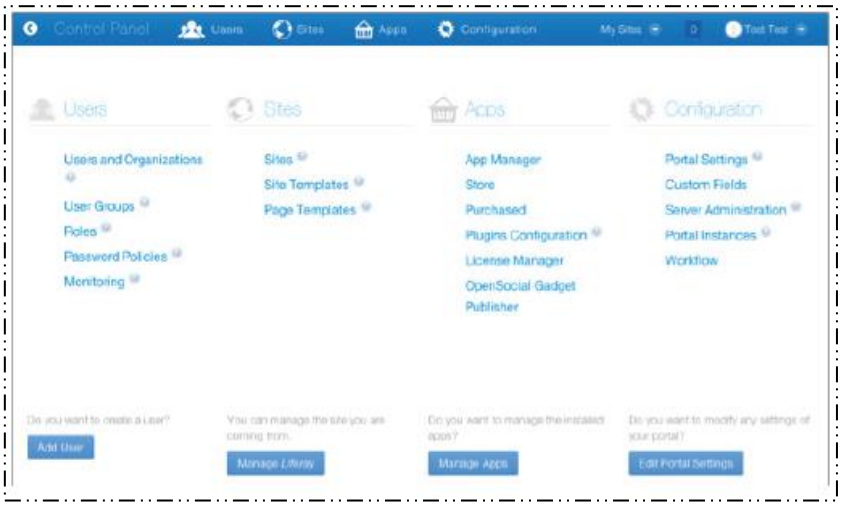


Figura 25 Mejoras de Usabilidad V6.2 Liferay Portal

- *Experiencia mejorada:*

Se han modificado las interacciones de los usuarios para hacerlas más intuitivas y rápidas y poder reducir así uno de los grandes *hándicaps* que tenía este CMS, la alta curva de aprendizaje del portal.

- *Contenidos y documentación*

En esta nueva versión es posible agrupar los contenidos en carpetas y subcarpetas para permitir una fácil clasificación y recuperación de la información lo que facilita enormemente la gestión de archivos. Para un entorno como el docente en el que se maneja tal cantidad de documentación, este punto puede resultar trivial.

- *Rendimiento*

La versión 6.2 incorpora cambios en el código fuente destinados a mejorar el rendimiento y escalabilidad del sistema. Este punto es realmente importante en los portales, ya que el rendimiento suele ser siempre un talón de Aquiles en todas las aplicaciones.

Por todas estas razones; por el hecho de ser la última versión y por todas las mejoras que aportan con respecto a la versión anterior, nos hemos decantado por ella para el desarrollo del proyecto.

3.6 Módulos/Plugins

Liferay es un Gestor de Contenidos y, como tal, consta de una parte interna (Sistema) y una externa (Contenidos).

Liferay separa sus herramientas en 5 bloques o *plugins*: *Portlets, Hooks, Layouts, Templates, y Theme* y también ofrece la posibilidad de extender del núcleo del producto empleando el *plugin Ext* que no detallaremos en el presente documento.

- La tecnología se despliega en servidores web y sólo es accesible por el equipo de desarrollo y de sistemas. En esta zona interna se define los *temas* de Liferay de los cuales hablaremos unos párrafos más abajo. Un tema no es más que la plantilla o esqueleto sobre el que se construyen todas las páginas de los portales. Incluye aspectos visuales (colores, disposiciones, tamaños de fuentes, iconos, etc.), es decir, los componentes de estilo que se suelen definir en plantillas CSS y aspectos funcionales (menús, comportamiento). Los temas son comunes a todos los portales y pueden ser comunes o diferentes entre páginas.

  También se definen a este nivel elementos de arquitectura los *portlet/widgets* que son módulos de programación comunes también a todos los portales.

- Los Contenidos, por su parte, se gestionan de forma externa dónde se desarrollan todas las actividades pertenecientes a la administración de un determinado portal.

Vamos a conocer más en detalle cada uno de los *plugins*/módulos descritos.

3.6.1 Portlets

Se trata de los componentes básicos y a la vez principales de programación en Liferay. Funcionan como módulos independientes y cada uno tiene una funcionalidad distinta. Un portlet es un componente modularizado que permite integrarse en un portal web siguiendo unos estándares. Son manejados por el contenedor de *Servlets* correspondiente al servidor de aplicaciones sobre el que esté desplegado y este se encarga de gestionar su ciclo de vida. Es importante indicar que cualquier código generado en un portlet no afectará nunca al código nativo del portal.

Liferay dispone de numerosos *Portlet* nativos pero otorga a su vez la posibilidad de crear tantos como se requieran para dar solución a todas nuestras necesidades o modificar la funcionalidad de los nativos empleando *Hooks*. En el entorno laboral, se hace uso de un *Hook* para extender funcionalidades de *framework*.

Estos componentes se pueden ver como sencillas aplicaciones *auto contenidas* regidas por estándares (*JSR 168 y JSR 286*). Esto hace que cualquier *portlet* pueda ser adaptable en cualquier portal que soporte dichos estándares, lo que facilita la reutilización de código.

Un portlet puede instanciarse o definirse varias veces en un portal web(según haya sido configurado en su fichero de propiedades) en el lugar que más se desee dentro del portal. Por este motivo cada portlet dispone de sus propias preferencias, lo que lleva a la posibilidad de configurar el mismo portlet de varias maneras sin necesidad de programarlo tantas veces como diferentes configuraciones necesitemos. Siempre configurándolo a través de la interfaz gráfica para hacerlo más usable.

Una aplicación Portlet es una extensión de una aplicación Web JEE que se empaqueta en un fichero de extensión WAR (*Web application Archive*), genera contenido dinámico para un portal y están desarrollados en código Java, cosa que los asemeja en gran medida a los S*ervlets*. Pero, a diferencia de estos, los portlet no pueden ser invocados de forma directa, es decir, un *Servlet* se puede invocar a través de su dirección web pero un portlet solo podrá ser llamado empleando la dirección web de la página que lo contiene, nunca realizando una petición directa sobre este. Otra diferencia es que los Servlets son capaces de generar páginas o documentos completos, mientras los portlets son partes del contenido [28].

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 posee el fichero XML que sirve como descriptor de despliegue para toda aplicación Web (*web.xml*) donde se define su estructura.

A continuación se puede observar la estructura típica de una aplicación Portlet. Como se puede apreciar en la Figura 26, la estructura no dista en exceso de la de una aplicación Web tradicional.

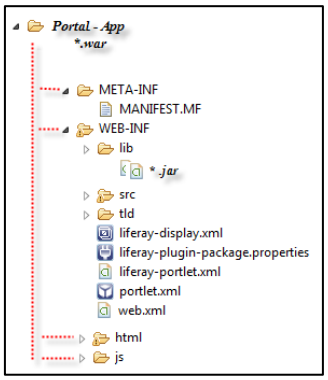


Figura 26 Estructura de una aplicación Portlet

Los entornos de desarrollo *NetBeans* y *Eclipse* cuentan con un plugin denominado *Portal Pack* que facilita ligeramente el desarrollo de Portlets, en este caso, se ha hecho uso del segundo ya que a pesar de estar menos desarrollado que el de *NetBeans*, permite la 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, cosa que nos es suficiente.

3.6.1.1 Arquitectura Lógica

La Figura 27 muestra el funcionamiento y arquitectura lógica de los portlets.

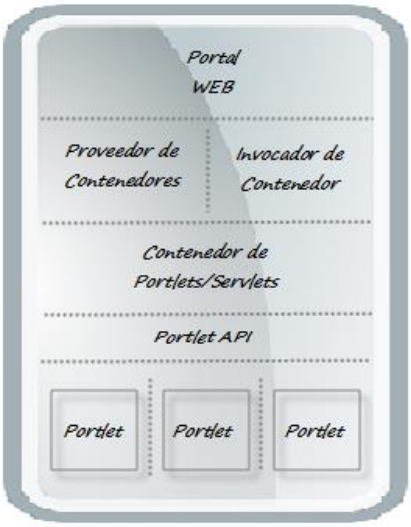


Figura 27 Arquitectura Lógica de un Portlet

Para entenderlo un poco mejor, señalar que, cuando se realiza una petición desde el portal web, este hace una llamada al contenedor de portlets (existen varios contenedores como *JetSpeed* o ***Liferay*)** por cada portlet esperando recibir su contenido a través del *Container Invoker*. Por aclarar, un contenedor de Portlets representa la interfaz entre los Portlets y el portal Web. A su vez el contenedor de portlets replicará la llamada a los portlets haciendo uso del API de *Portlets*.

Los Portlets se instalan en un contenedor de Portlets. Típicamente, 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 el ciclo de vida:
  - Inicializa y destruye los Portlets.
  - Recibe las peticiones del usuario y las traslada al Portlet.

El contenedor finalmente devolverá la información de los portlets gracias al *Container provider* o proveedor de contenidos.

3.6.1.2 Métodos que Implementa

Cada portlet debe implementar una interfaz de portlet o extender de una clase que la implemente. Usualmente lo que se hace es extender a '*javax.portlet.GenericPortlet'*.

Ejemplo de ello es el mostrado en la Figura 28.

Figura 28 Interfaz que implementa un Portlet

- *Modos del Portlet (formas de funcionamiento)*

Se pueden considerar dos tipos de agrupaciones para los métodos que encontraremos en un portlet:

- *Estándares*: *View* (tarea principal del Portlet), *Edit* (configuración de preferencias del Portlet) y *Help* (ayuda del Portlet).
- *Modos a medida* (creados por el desarrollador).

En nuestro caso, únicamente hemos empleado los métodos estándar.

3.6.1.3 Ciclo de vida

- Cuando el contenedor de portlet crear un instancia del portlet, al primer método que invoca es a *init* (…). Este método se invoca una sola vez.
- Una vez que el portlet recibe una solicitud (*request*) de representación se invoca al método *render* (…).
- *ProcessAction ()* se invoca como respuesta a una solicitud de acción, es decir, se usa para notificar al portlet que el usuario ha realizado una acción en un portlet.

- Por último se invoca el método *destroy* () para indicar el fin del ciclo de vida de un portlet.

En la Figura 29 se muestra gráficamente el citado ciclo de vida.

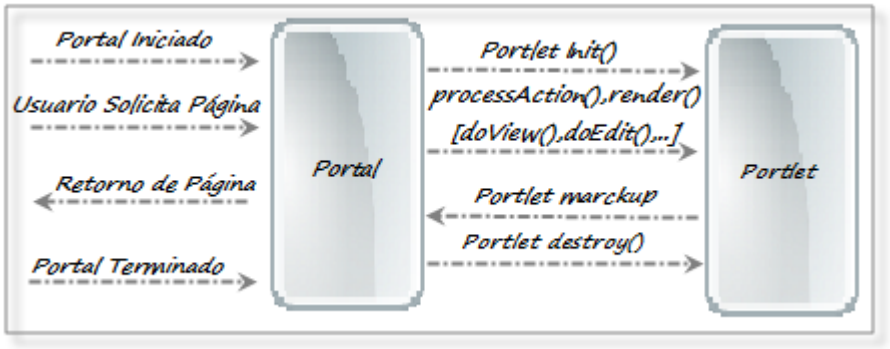


Figura 29 Ciclo de vida de un Portlet

Vamos a explicar con más detalle los tipos de peticiones que pueden existir.

- *Procesamiento y manejo de las peticiones del Portlet:*

Podemos encontrarnos con dos tipos de peticiones dentro de un portlet:

- *Peticiones de acción (action request)*: Peticiones que modifican el estado del portlet o que causan una redirección pero no producen visualización. Se procesan redefiniendo el método *processAction*().

- *Petición de renderización (render request):* Peticiones que solicitan la visualización del portlet. Se invoca al método render (…). Realizan el procesamiento de peticiones de renderización (producen el contenido que se muestra al usuario).

*Una petición de acción sobre un portlet siempre va seguida de una petición de renderizado sobre ese portlet. [29]

3.6.2 Themes

Cuando nos enfrentamos a un proyecto de desarrollo sobre *Liferay* no solamente tenemos que considerar el desarrollo de las características funcionales del portal sino también el de las características graficas del portal, para ello, empleamos los temas, puesto que es el módulo de *Liferay* encargado de gestionar la apariencia del portal web.

El tema es la plantilla o esqueleto sobre el que construyen las páginas del portal e incluye tanto aspectos visuales (colores, tamaños, disposiciones) como aspectos funcionales (menús, comportamiento, barra de administración) y se desarrollan principalmente empleando tres tecnologías: *CSS, Javascript-Jquery y Velocity.*

Lo más recomendable y por tanto, el modo en que se ha seguido, ha sido desarrollando este módulo partiendo de las bases que ofrece la SDK al generar un módulo de este tipo. Dentro de este módulo, los ficheros se encuentran organizados en carpetas, agrupados según la tecnología que contienen.

Entre estos directorios se encuentran:

- *_diffs* - Directorio donde se debe replicar la estructura base y modificar los fichero que se quieran adaptar, puesto que Liferay al compilar el módulo de tema, consulta la carpeta '*_diffs'* y replica los cambios que esta contenga al resto de carpetas. Esta carpeta prevalece ante el resto.
  Este es el directorio donde trabajaremos normalmente para añadir nuestros diseños (*css, js*) o imágenes.
- *css*: Lugar donde se ubicarán todos los ficheros de estilos.
- *images*: Lugar donde se encontrarán las imágenes.
- *js*: Es donde se ubicarán los *Javascript*.
- *templates*: Es donde se ubicarán las plantillas *Velocity* del tema.
- *WEB-INF/liferay-look-and-feel.xml:* Fichero donde se define el identificador del tema en Liferay y la versión de Liferay mínima a partir de la que está disponible el componente. En el Código 5 vemos los cambios realizados en la configuración.

```
<?xml version="1.0"?>
<!DOCTYPE look-and-feel PUBLIC "-//Liferay//DTD Look and

<look-and-feel>
    <compatibility>
        <version>6.2.1+</version>
    </compatibility>
    <theme id="temaPrueba" name="TemaPrueba">
        <template-extension>ftl</template-extension>
        <settings>
```

Código 5 Configuración liferay-look-and-feel.xml

- *WEB-INF/liferay-plugin-package.properties*: Como se observa en el Código 6, se trata del dichero donde se definen las características del paquete, como son el autor, el tipo de licencia y su versión.

```
name=TemaPrueba
module-group-id=liferay
module-incremental-version=1
tags=
short-description=
long-description=
change-log=
page-url=http://www.liferay.com
author=Sara Ostos Lobo
licenses=LGPL
liferay-versions=6.2.0+

#required-deployment-contexts=\
#    resources-importer-web

resources-importer-developer-mode-enabled=true
```

Código 6 Configuración liferay-plugin-package.properties

Hemos desarrollado un tema de prueba siguiendo un tutorial de la red por conocer la dificultad y francamente, debe reconocerse que crear la estructura inicial es relativamente sencillo con el SDK ofrecido por Eclipse pero al no haber agregado funcionalidad adicional no vamos a considerarlo en el documento.

3.6.3 Layouts

La parte de visualización de los portales, está dividida en el componente mencionado en el apartado anterior y en los *plugins* conocidos como *layouts*. Este módulo se emplea para subdividir el cuerpo de las páginas y poder configurar la disposición de los portlet dentro de estas, organizándolos en filas y/o columnas de ancho configurable según la combinación que se diseñe. *Liferay Portal* ofrece disposiciones por defecto que proveen un gran abanico de posibilidades para la confección de páginas y como en el resto de componentes, también se pueden agregar nuevas disposiciones personalizadas.

Cada página puede tener un único *layout*, pero existe la opción de reutilizar el mismo en múltiples páginas.

Es importante diseñar los *layout* y el tema en paralelo puesto que la estructura definida en el layout será maquetada posteriormente por el tema. De esta forma un layout puede ser utilizado por varios temas y un mismo tema puede emplearse en varios layout. Estos componentes, se desarrollan con *HTML y Velocity*.

Los layouts se encargan de la disposición del contenido en las páginas como se ve en la Figura 30 y el tema se encarga de definir los componentes como la cabecera, navegación principal y *footer(pie de página)*.

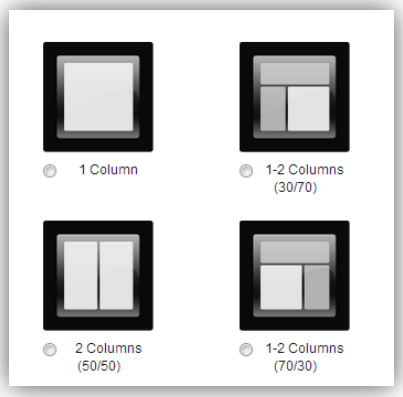


Figura 30 Ejemplo Layouts

3.6.4 Hook

Los *Hooks* (“gancho”) [30] son herramientas poderosas de liferay que nos permite alterar el funcionamiento del ‘*core’* del producto en nuestro portal, es decir, sirven para extender el producto, por ejemplo para agregando funcionalidades extras en el panel de administración. El mayor potencial que ofrecen es el de permitir de modo elegante modificar el comportamientos del portal sin necesidad de recompilar todo el código.

Se trata de componentes desarrollados en lenguajes como JAVA, HTML, JSP o XML y como se ha indicado, son utilizados para añadir determinada funcionalidad. Concretamente, puede definirse *hooks* englobados en los siguientes tipos:

- Sobrescribir ficheros JSPs del portal.
- Monitorizar la edición o creación de las entidades de Liferay.
- Sobrescribir servicios del Portal.

- Modificar propiedades del '*portal.properties'* para la configuración del servidor.
- Cambiar la configuración de modo dinámico.
- Sobrescribir las traducciones.
- Insertar eventos en el ciclo de vida de Liferay.
- Poder realizar acciones ante determinados eventos como pueden ser:
    - *Application Startup Events (application.startup.events)*
    - *Login Events (login.events.pre, login.event.post)*
    - *Service Events (servlet.service.events.pre, servlet.service.events.post)*

Interesante también recalcar que los cambios realizados en estos componentes se harán visibles en el portal de forma inmediata una vez lo despleguemos y que si posteriormente decidimos desinstalarlo volveríamos al estado inicial previo a su aplicación sin necesidad de realizar ningún cambio adicional.

Debe quedar claro que los Hooks permiten modificar funcionalidades de producto pero claro está que por motivos de seguridad no permiten realizar modificaciones de cualquier tipo, por ejemplo, el núcleo de Liferay no puede ser accedido con dichos elementos, aun así, esto tampoco es imposible, ya que podría hacerse con los llamados Ext Plug-ins. Ya que aumentan la complejidad de la instancia de Liferay, y por tanto, son bastante críticos, sólo se recomienda emplear un plugin Ext en caso de estar seguro de que no existe otro componente que permita lograr la funcionalidad deseada. Los plugins Ext (extensiones) son similares a los Hooks, pero distan en tres grandes puntos:

- Permiten modificar cualquier parte del código fuente, es decir, proporcionan un control total, de ahí la criticidad de su uso.
- Requiere el reinicio del servidor tras su despliegue para que surta efecto.
- Una vez instalado es muy difícil reinvertir los cambios para devolver el portal al estado original. Puede ocasionar que porciones de código dejen de funcionar correctamente y puede ocasionar problemas ante la casuística de migración entre versiones del producto.

En este caso de estudio no hemos implementado componentes ni de tipo *hook* ni *Ext* pero creíamos interesante mencionarlos.

Capítulo 4

Requisitos y Casos de Uso

En el desarrollo de este capítulo se definen detalladamente los requisitos funcionales obtenidos durante sesiones de trabajo con el tutor del proyecto. A continuación, se detallan los diagramas de casos de uso obtenidos a partir de estos requisitos de usuario.

4.1 Requisitos Funcionales

4.1.1 Acceso

Para acceder al gestor de notas será necesario contar con un usuario y una contraseña de modo que sólo los usuarios autenticados puedan acceder a la información. Inicialmente, los usuarios existentes serán importados en el producto obteniéndolos del propio LDAP de la universidad, pero se podrán dar de alta nuevos usuarios por parte de un usuario administrador.

En caso de olvido de contraseña se podrá restablecer la contraseña introduciendo el nombre de usuario y se enviará un email a la dirección de correo que se hubiese empleado en el registro.

4.1.2 Usuarios y grupos. Permisos

Se tendrán dos tipos de usuarios: Alumnos y Profesores.

Los profesores contarán a su vez con el rol de administradores por facilitar la gestión pero en un entorno productivo, lo conveniente es separar estos roles.

Se podrán formar grupos de usuarios y/o administradores para facilitar la gestión de la aplicación, puesto que según su rol, tendrán acceso a unos contenidos u otros dentro del portal.

4.1.3 Creación / Edición de los tipos de contenido

Todos los tipos de contenido se podrán crear y editar mediante un formulario.

Para futuros trabajos, sería interesante ofrecer la posibilidad de crear nuevos tipos de contenidos mediante la carga de un fichero de Excel o de documentos adicionales.

4.1.4 Búsquedas

Se podrán realizar búsquedas de los alumnos por cualquiera de sus campos. Las búsquedas serán auto recursivas, por tanto, no es necesario el campo concreto a buscar, con un fragmento será suficiente para obtener resultados. Se pueden analizar los datos mediante la inserción de texto en la barra del buscador o mediante una ayuda que se despliega al pulsar en el botón de configuración situado junto al buscador y que permite buscar considerando la coincidencia de todos los parámetros completados o solo de alguno. Más adelante se detallará lo mencionado en este párrafo.

4.2 Especificación de Casos de Uso

Como ya se ha indicado, el sistema cuenta con dos tipos de usuarios o actores principales: alumnos y profesores. Mencionar también que como cualquier sistema, existirá el rol administrador, que, a efectos, puede ser considerado como una extensión del rol profesor, con privilegios añadidos que le permita parametrizar la aplicación (En este caso, permitirá crear una página personalizada según el rol y personalizar contenidos según la titulación a la que pertenecen los alumnos). Cada rol permite al usuario el acceso a ciertas tareas. Esto lo vamos a esquematizar mediante los denominados diagramas de casos de uso. Vamos a especificar los diagramas para el perfil de profesor/administrador y para el caso de los alumnos.

En aras de claridad, vamos a dividir el comportamiento que proporciona el sistema desde el punto de vista del usuario realizando diversos diagramas para el perfil profesor/administrador así como para el de alumno.

4.2.1 Acceso Al Sistema / Login de Usuario

El usuario que desee acceder al sistema, ya sea docente o alumno, deberá introducir su usuario y contraseña para poder realizar alguna de las acciones disponibles en el sistema. Según sea su perfilado (definido atendiendo a la categoría a la que pertenece), tendrá la opción de visualizar un *Site* con unos contenidos u otro. En la Figura 31 vemos el proceso de acceso al portal.

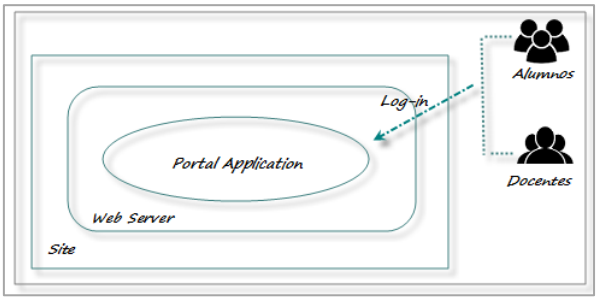


Figura 31 Casos de Uso: Acceso al Sistema

Dentro del portal, existe un portlet denominado "*Sign in*" proporcionado por Liferay que nos permite realizar la autenticación. El detalle del diagrama de uso del proceso de autenticación es el que aparece en la Figura 32.

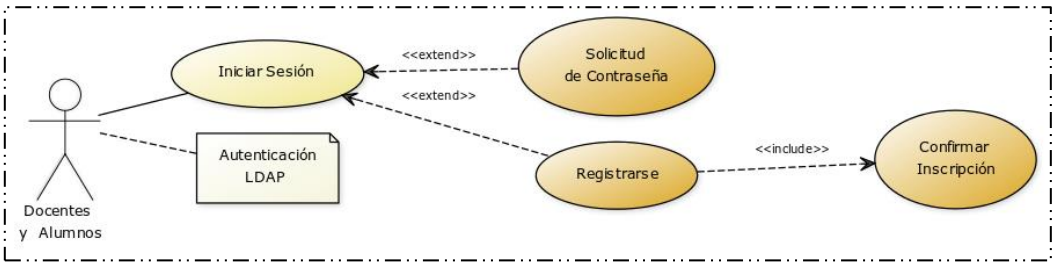


Figura 32 Casos de Uso: Autenticación

4.2.2 Perfil Alumno

Como se aprecia en el esquema inferior, la función del alumno en el sistema es simple ya que únicamente tiene acceso a un widget. La idea es que su participación sea rápida y sencilla para no incurrir en una complejidad que pueda desmotivar la utilización de la plataforma. Podrá acceder a un formulario a través del cual, con una breve serie de datos personales (como son el DNI y NIA, asignatura, año y convocatoria), podrá obtener sus calificaciones, que deberán haber sido cumplimentadas previamente por el profesor correspondiente. Previamente a estas acciones, se requerirá la autenticación en el sistema, para asegurar la veracidad de la identidad de los participantes.

Este caso de uso se ilustra en la Figura 33.

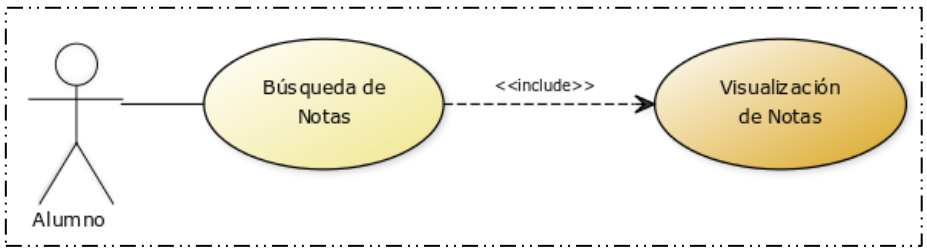


Figura 33 Casos de Uso: Perfil Alumno

4.2.3 Perfil Profesor

El perfil de profesor es algo más complejo, puesto que contará con más funcionalidades. Podrá gestionar asignaturas, acceder a las mismas o eliminarlas del sistema. De igual modo, podrá añadir, modificar o eliminar los registros de sus alumnos. Como última opción, podrá asignar las notas correspondientes a los alumnos existentes y obtener un listado de todos los alumnos que gestiona, así como buscar un alumno concreto. El acceso a un alumno, en este caso, es significativamente distinto al acceso al listado de alumnos, pues permite una búsqueda por cualquier parámetro deseado. De nuevo, se requerirá una autenticación en el portal antes de poder realizar cualquiera de estas acciones.

El conjunto de acciones se describen en la Figura 34.

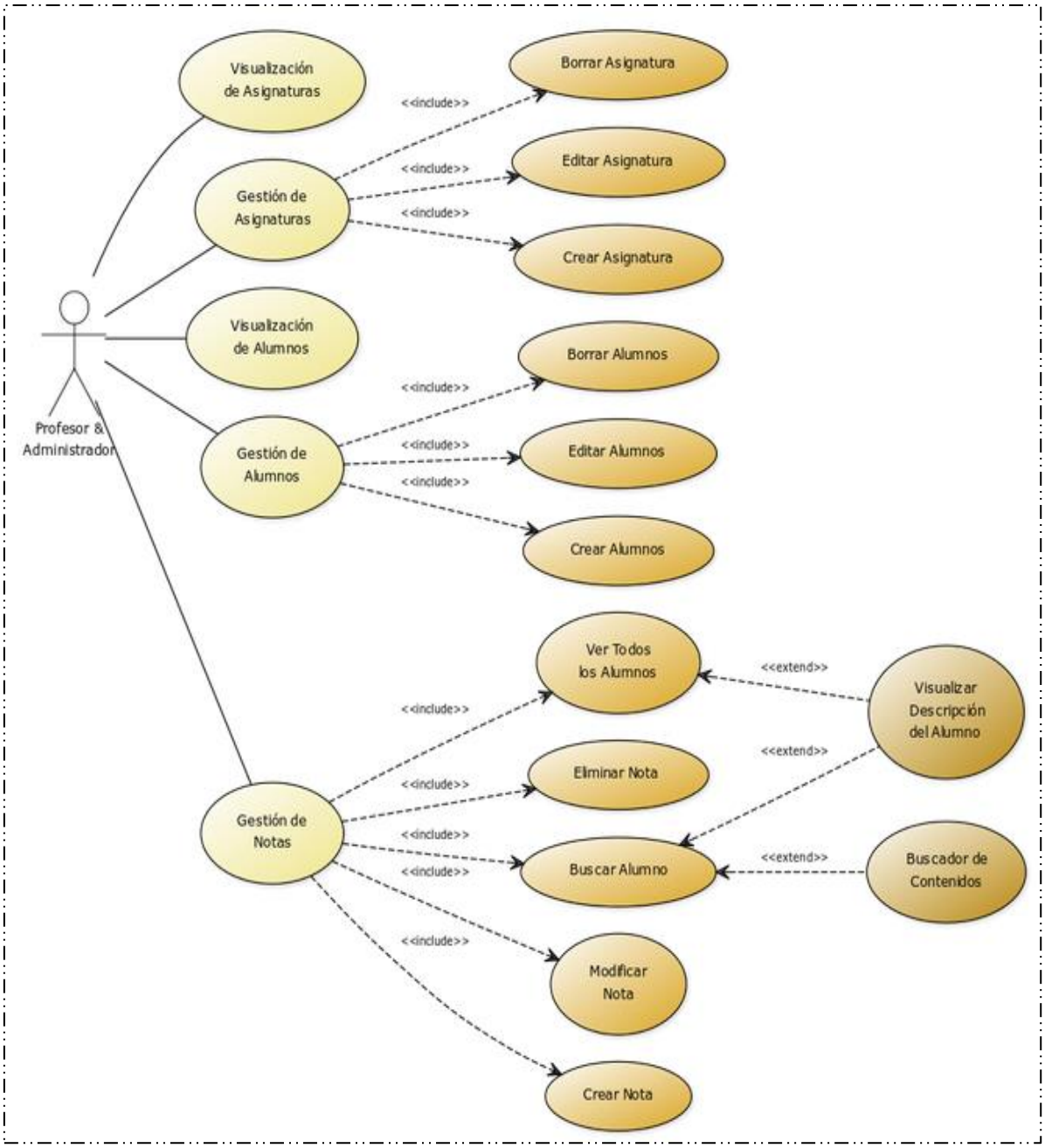


Figura 34 Casos de Uso: Perfil Profesor

4.2.4 Perfil Administrador

Como se ha indicado, el perfil administrador, tiene funcionalidades extra con respecto al perfil Profesor.

Estas pueden verse en la Figura 35.

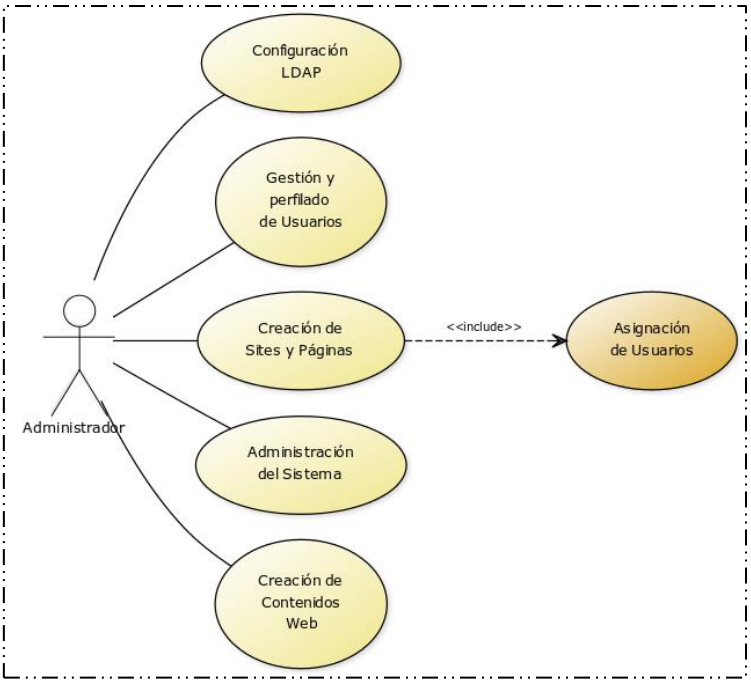


Figura 35 Casos de Uso: Perfil Administrador

Capítulo 5

Diseño de la Aplicación

En este capítulo pasamos a conocer el sistema ofreciendo una visión global de la funcionalidad así como de su organización. A continuación se ilustrarán los componentes necesarios y la interacción entre los mismos para lograr los objetivos requeridos. Como es comprensible, todo esto no sería posible implementarlo sin contar con un entorno de trabajo, el cual vamos a mencionar brevemente en el primer apartado. Se detallará también el modelo de datos definido para la aplicación.

5.1 Entorno de Trabajo

Para entender esta implementación explicaremos el entorno de trabajo al que vamos a someter a Liferay Portal.

El proyecto ha sido montado en un pc local y los requisitos fundamentales de dicho equipos así como el software que se ha necesitado se resume en el siguiente listado:

- *Sistema operativo*: *Windows 7 Enterprise Service Pack 1*.
- *Portal Web*: Instalación del bundle de *Liferay 6.2 CE ga2*
- *Servidor Web: Apache Tomcat* 7.0.42
- Instalación del *JDK 1.7.0_51* y configuración de la variable de entorno *JAVA_HOME*.
- *Base de datos*: *MySQL* 5.6. Opción más que recomendable por la multitud de soporte que ofrece esta BBDD y por tratarse de una de las versiones gratuita más popular del mercado.
- *Entorno de Desarrollo*: Eclipse Kepler Service Reléase 2.
- *Liferay IDE 2.1.0.201403281241-ga1* para el desarrollo de las funcionalidades en Eclipse. Con esta herramienta los desarrolladores pueden crear, empaquetar, desplegar y probar portlets dentro de Eclipse. Destaca su facilidad de integración con varios servidores de portal, permitiendo realizar despliegues de portlets en servidores remotos y locales, en nuestro caso de modo local empleando Tomcat.

5.2 Arquitectura general de la aplicación

En este caso, se puede asimilar nuestra aplicación a una aplicación Web convencional y como tal, su arquitectura no dista de la de cualquier APP Web, es decir, cuenta de un cliente o navegador Web, un servidor Web y una base de datos desde la que se recopila la información a visualizar.

Para este tipo de desarrollos, existen diversos patrones de diseño que ayudan al desarrollo de las mismas. Para la solución se ha optado por un diseño MVC (Modelo-Vista-Controlador), que abstrae las tareas de interacción del usuario y la representación de la información. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la segmentación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

El resultado es el diseño esquemático que se muestra en la Figura 36. En ella se aprecian los diversos bloques que componen la aplicación.

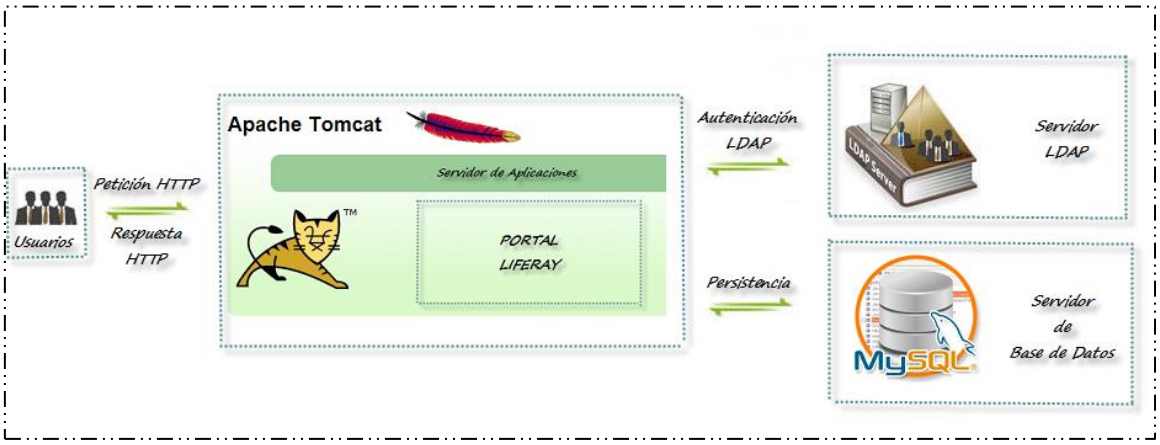


Figura 36 Arquitectura de la Aplicación

Según se observa en el diagrama, la aplicación cuenta con diferentes módulos, cada uno de los cuales desarrollará una funcionalidad específica dentro de la misma.

Pasamos a detallar brevemente el proceso definido:

- *Bloque Usuarios:* Van a realizar el papel de cliente dentro de nuestro sistema, y serán quienes generen la petición HTTP a través de un navegador web contra el servidor de Tomcat.
- *Petición/Respuesta HTTP:*Puesto que estamos desarrollando una aplicación Web, accederemos a ella mediante la URL del servidor de aplicaciones. En nuestro caso*:* http*://localhost:8085* y se obtendrá la respuesta de la página desde el servidor Apache.
- *Servidor de Aplicaciones:* Este bloque, es sin duda el más importante, puesto que la funcionalidad recae sobre él. Es el encargado de gestionar las peticiones generadas por los usuarios y devolver la interfaz correspondiente del portal, es decir, se encargará de la gestión de los recursos.
- *Servidor LDAP:* Gracias a este módulo se pueden agregar usuarios en el sistema para realizar el control de autenticación en el portal. Se trata de un módulo muy importante en la aplicación.
- *Servidor MySQL:* Este módulo interactúa con el servidor de aplicaciones en caso de que el sistema lo requiera.

Es interesante matizar que la división en bloques mostrada no se corresponde con una división física puesto que en nuestro entorno todos se encuentran ubicados en el mismo equipo, a excepción del servidor LDAP, que en este caso está ubicado en la Universidad Carlos III. No obstante, lo más normal en la realidad es encontrarse con un entorno mucho más distribuido donde cada bloque se encuentre en lugares distintos.

Es muy importante mencionar también que la arquitectura del sistema se basa en un diseño multicapa, cosa muy incorporada en las aplicaciones de gestión ya que permiten manejar el tratamiento de los datos persistentes, la lógica de negocio y la interfaz del usuario de un modo independiente.
Esto es interesante ya que permite reducir las dependencias entre capas facilitando el mantenimiento del sistema, dado que las modificaciones efectuadas en una parte del mismo sólo afectarán a la capa concreta.

5.3 Mapa de Sitio Web

Un mapa de sitio web (o mapa de sitio o mapa web) es una lista de las páginas de un sitio web accesibles por parte de los usuarios. Se puede desarrollar de varios modos, en este caso, se va a esquematizar el portal como dos sitios que listan las páginas de una web (ya realizada), organizadas comúnmente de forma jerárquica con el fin de ayudar a los lectores a conocer las páginas de nuestro sitio web y los portlets disponibles en cada una de ellas.

Resulta interesante este tipo de esquema porque son de gran ayuda a la navegación por ofrecer una vista general del contenido de un sitio con un simple vistazo como se corrobora en la Figura 37.

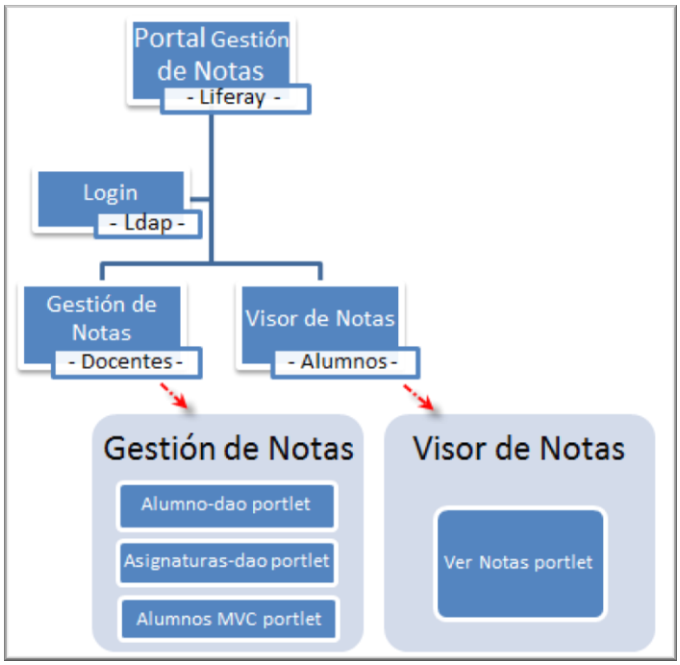


Figura 37 Mapa de Sitio Web

5.4 Modelo de Datos

Para la obtención de los datos en la implementación del portal creado, se ha hecho uso de una base de datos MySQL. Se necesita conocer cómo va a estar estructurada la información para sus consultas

y modificaciones. El fin del presente PFC era el desarrollo de una aplicación sencilla, y por lo tanto se ha generado un modelo de datos también sencillo. En nuestro caso, existe una tabla por cada portlet desarrollado pero podría invocarse desde un mismo widget a varias tablas del modelo. Aclarar que para realizar el acceso a las tablas de la base de datos, se han implementado clases de tipo "*Beans*", es decir, clases que contienen un constructor de la clase, los atributos privados de ésta y todos los métodos "*Get*" y "*Set*" de los atributos para poder acceder a ellos y editarlos.

En la Figura 38 se engloban el conjunto de tablas que forman la aplicación y los campos que las componen, según los objetos que se han considerado apropiados. Se muestra por cada uno de los objetos, su identificador único (PK), el uso de claves foráneas (FK) para relacionar tablas y la relación existente entre los diferentes objetos.

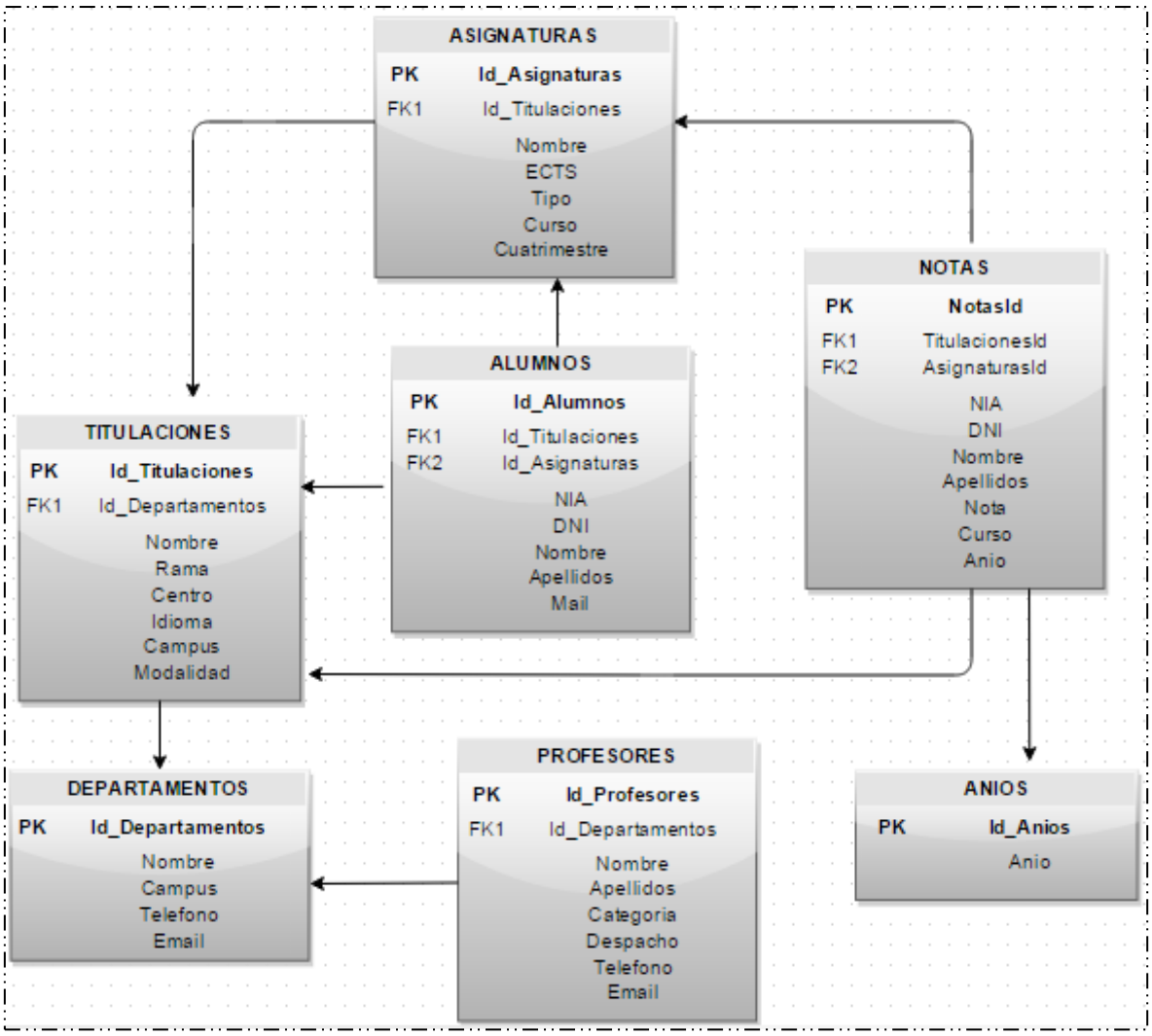


Figura 38 Modelo de Datos

5.5 Gestión de Usuarios: LDAP

Como ya es sabido, nuestro sistema va a incorporar *LDAP* para llevar a cabo la gestión de usuarios y adaptar los contenidos según su perfilado.

El portal deberá tener una carga de usuarios sincronizada con el servicio *LDAP* (*Lightweight Directory Access Protocol (Protocolo Ligero de Acceso a Directorios)*) de la universidad UC3M. Este servicio/protocolo, a través del CMS Liferay agregará y sincronizará los usuarios de la organización en la base de datos del producto.
Se dispondrá de una propiedad de configuración en el fichero "*portal-ext.properties*" que sincronizará el *LDAP* con Liferay cada cierto tiempo configurable.

Se ha considerado útil el uso de esta herramienta porque, además del ágil tratamiento de datos que soporta, aporta una gran flexibilidad en la gestión de usuarios, e incluso se trata de un sistema independiente y estándar que permite, en caso de necesitad, compartirlo con otras aplicaciones y servicios.

❖ *Integración del protocolo LDAP en Liferay*

La conexión de LDAP en un CMS Liferay puede realizarse siguiendo dos métodos: agregando las oportunas propiedades en el fichero de configuración del portal ("*portal-ext.properties*") o empleando las facilidades que nos ofrece la interfaz gráfica del producto. En este caso, se ha empleado el segundo método por su sencillez gráfica, pero nos hemos apoyado en el primero para configurar un parámetro inaccesible a través de la consola.

Estas propiedades son:

- *ldap.import.interval=10*: Con este parámetro, cada 10 minutos se sincronizarán los datos de usuarios con el LDAP. No se ha considerado necesario un tiempo inferior por tratarse de un entorno de pruebas, pero en caso de un entorno productivo, este valor sería recomendable parametrizarlo a un valor cercano a 2 minutos.
- *ldap.import.method=group*: Con esta propiedad, nos aseguramos que solo se importarán los usuarios que estén asignados a un grupo en LDAP.

Para más información sobre las propiedades configurables en el fichero de propiedades, véase [31]

La configuración oportuna deberá ser realizada por un usuario con perfil administrador, ya que se trata de una información crítica que no debe ser accedida por cualquier otro rol. La configuración de Liferay para la autenticación mediante el protocolo sugerido, permite importar tanto los usuarios como los grupos de estos y una vez finalizada, los usuarios existentes en el sistema LDAP podrán autenticarse ante Liferay.

Para mayor detalle sobre el modo de configurarlo, véase el ***Anexo III***.

Un ejemplo de resultado obtenido se encuentra en la Figura 39.

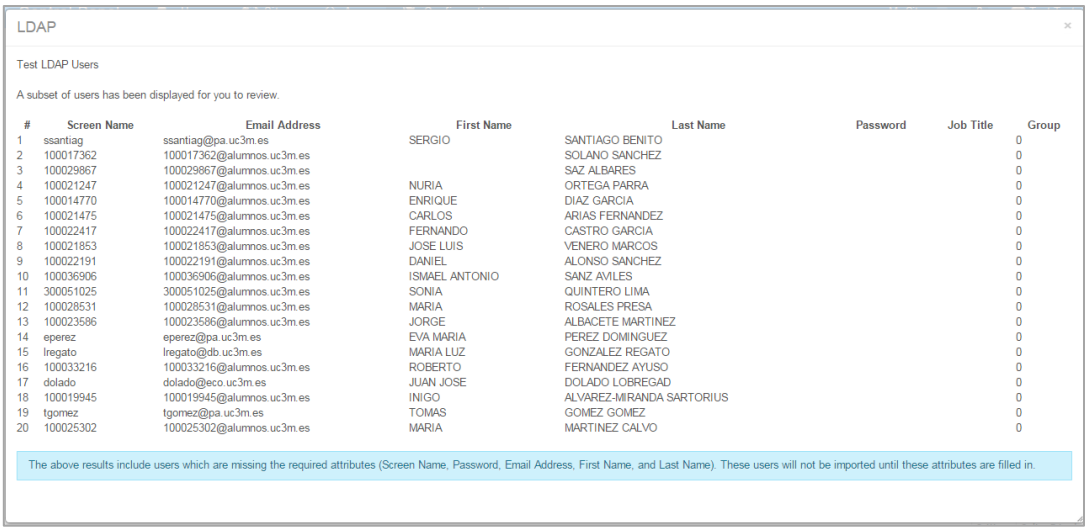


Figura 39 Usuarios LDAP Uc3m

Se trata de un proceso muy sencillo de gestión de usuarios, porque, una vez que se tengan, el administrador agrupará los usuarios por roles y creará un *Site* por cada tipo de perfilado, uno para el caso de los alumnos y otro para los docentes.

Capítulo 6

Implementación del portal para la gestión de Notas

Este capítulo es sin duda el más importante. En el vamos a adentrarnos en cada módulo desarrollado, conociendo el modo de implementación de estos. Se indicarán varios métodos de gestión de los datos desde los portlets y profundizaremos en las clases fundamentales que constituyen las piezas del producto que nos atañe.

6.1 Entorno de Aplicación del sistema

Antes de comenzar a desarrollar el software creado, recordar que los portlets, elementos van a estar presentes a lo largo de todo este capítulo, son componentes modulares de interfaz de usuario gestionados y visualizados dentro de las página de nuestro portal.

Típicamente, siguiendo la metáfora de escritorio, una página de un portal se visualiza como una colección de ventanas de portlet no solapadas, donde cada una de estas muestra un portlet. Por lo tanto un portlet se asemeja a una aplicación web que está hospedada en un portal. Dentro del portal, encontraremos diversos elementos: Theme, Portlets y Layout.

Por mostrarlo de modo gráfico en la Figura 40:

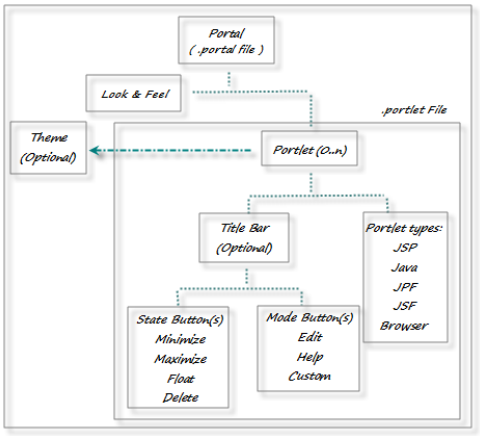


Figura 40 Componentes de un portal Liferay

Como venimos indicando a lo largo del documento, el sistema diseñado se compone de 4 portlets que contendrán la gestión de los datos y un layout para la gestión de dichos portlets dentro de las páginas, ubicados en dos *Sites* diferentes según los consumidores del portal. Para la apariencia de ambas páginas hemos empleado un Theme público. En cada página podemos aplicar un tema de apariencia diferente, pero en este caso, hemos mantenido el mismo tanto para el perfil de docente como para el de alumnado.

Antes de adentrarnos en los detalles, es importante reseñar que hay dos modos de desarrollar portlets, uno consiste en emplear la clase ***GenericPortlet*** y el otro en ***MVC Portlet***. En el presente estudio se han querido emplear ambas para conocer al máximo las capacidades del producto y poder valorar la dificultad de cada una de ellas.

Y una cuestión que puede surgir es, *dónde radica la principal diferencia entre ambos métodos de implementación*.

- *GenericPorlet* es parte de la especificación *JSR286*, esto significa que si el portlet se crea extendiendo de esta clase y no emplea ningún código específico *Liferay*, el portlet será portable a través de contenedores de portlets.

  La clase *GenericPortlet* proporciona una implementación predeterminada para la interfaz de portlets, ofreciendo una clase abstracta para tener subclases que permiten la creación de portlets. Estos componentes, normalmente se ejecutan en servidores multiproceso, así que no hay que perder de vista que un portlet debe manejar las solicitudes de modo concurrente y prestar especial atención a la gestión de la sincronización al existir accesos a recursos compartidos. Los recursos compartidos incluyen datos en memoria y acceso a objetos externos tales como archivos, conexiones de base de datos o conexiones de red.

- Por otro lado, *Liferay* dispone de un framework denominado *Liferay MVC*. Se trata de un framework de desarrollo de portlets que permiten definir plugins compatibles con los estándares *JSR 168 y 286*. Este marco de trabajo tiene muchas características y utiliza las API de *Liferay* para el desarrollo de portlets de un modo sencillo y fácil de usar.

Generalmente, se emplea la primera opción cuando el desarrollo requiere manejar más código, pero cuando usamos *Liferay MVC* se logrará reducir un esfuerzo de desarrollo. Cuando la comparativa se realiza en prospectiva de código, *MVC* tiene muy buen mecanismo para el manejo del ciclo de vida de los portlets.

6.2 Portlet Gestión de Alumnos

El portlet "Alumno-dao" permite la gestión de alumnos, como su nombre indica, facilitándonos la creación, edición y borrado de estos de modo sencillo e intuitivo.

La interfaz de usuario y la gestión del portlet es muy simple, ya que se realiza a través de una tabla de datos que indica el listado de alumnos que hay disponibles en el sistema. Esta tabla será personalizada por cada profesor en su S*ite* privado. En el caso de que un docente imparta clases en varias titulaciones, podrá duplicar este portlet para visualizar los alumnos de cada una de ellas por separado.

6.2.1 Funcionalidad

Al invocar la página que contiene el widget, se mostrarán todos los registros existentes en la tabla de alumnos de nuestra base de datos. Dicho volcado se visualiza en formato tabla para su posterior gestión así como botones con las opciones disponibles sobre los datos: Añadir un nuevo registro, modificar uno existente o eliminar una fila de la tabla.

- Si la opción seleccionada por el usuario es la de añadir un nuevo alumno al sistema, deberá rellenar un formulario con los campos: NIA, DNI, Nombre, Apellidos, Titulación, Asignatura y Mail. Una vez se pulse el botón de aceptar, aparecerá el listado con el nuevo registro agregado.
- Si se selecciona la opción de edición de una fila, sea cual sea el valor a editar, aparecerá un formulario con los datos existentes y el usuario tendrá la potestad de editarlos según su necesidad.
- Como última opción sobre este portlet, se podrá eliminar un registro deseado simplemente seleccionando la opción de borrado sobre la fila sobrante.

Para más detalle consulte el ***Anexo V***.

6.2.2 Diagrama de Clases

En la *figura 44* se muestra el diagrama de clases que forman el portlet de gestión de alumnos. En ella se han agrupado las clases por empaquetado, según la disposición desarrollada. Indicar al lector que las relaciones entre clases pueden obtenerse a partir de los atributos de cada una de estas.

Con el fin de mejorar la legibilidad del diagrama, para el caso de la clases Alumno, Titulación y Asignatura, se ha obviado la mención explícita de todas las operaciones que contiene ya que éstas consisten únicamente en el constructor y métodos de acceso a cada atributo, definidos en el gráfico como (*Getters* y *Setters*).

Como aclaración de la Figura 41, indicar que la clase *ConnectionPool*, se apoya en un fichero de propiedades denominado "connection-pool", en el que pasamos por parámetros los datos de conexión de nuestra base de datos y son manejados desde dicha clase.
De este modo, si surge cualquier cambio en el nombre de la base de datos o en los valores de autenticación, solo requerirá el cambio en el fichero de propiedades, resultado transparente para el resto del sistema.

Para esclarecer la interacción entre las clases, se presenta en el siguiente apartado un diagrama de secuencia con la actividad.

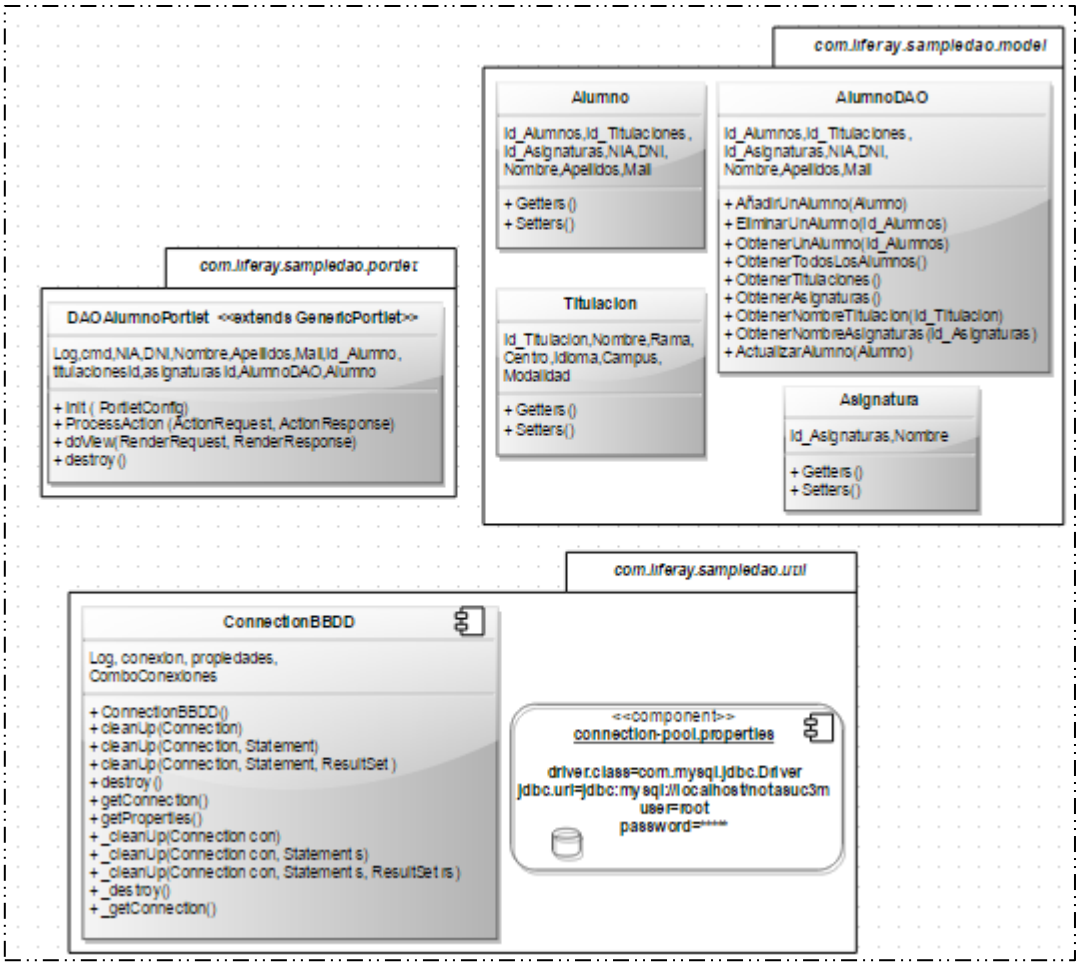


Figura 41 Modelo de Clases Portlet Alumno-DAO

6.2.3 Diagrama de Secuencia

En la siguiente página se muestra la Figura 42 que representa la interacción entre las clases desarrolladas.

Se recoge en el mismo gráfico el proceso seguido según la petición realizada, ya se trate de (*DoView*, *Add*, *Edit o Delete*). Se indican los métodos invocados en las peticiones así como una descripción de los elementos devueltos.

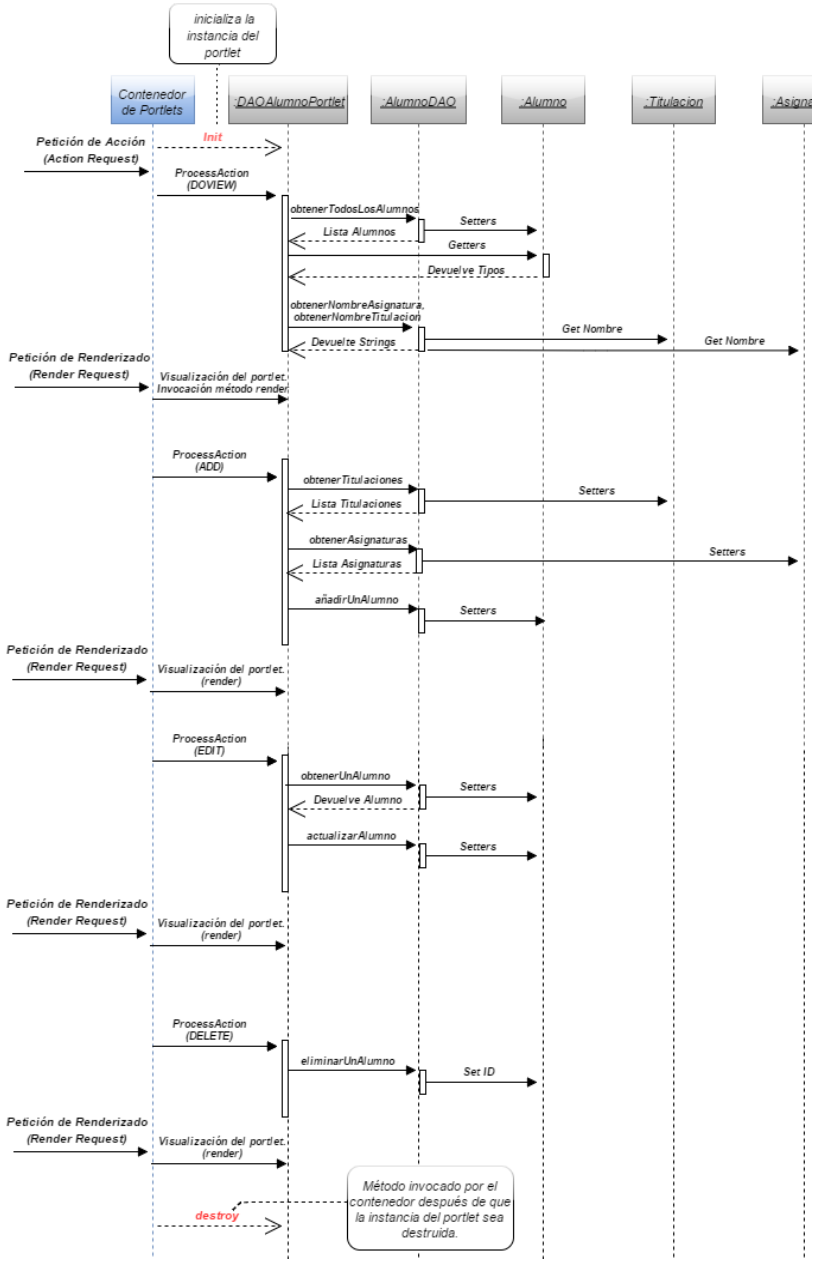


Figura 42 Diagrama de Secuencia portlet gestión Alumnos

Con respecto al diagrama de secuencia, hay ciertos métodos que son invocados durante el ciclo de vida del portlet directamente por el contenedor, detallar con respecto a ellos los siguientes aspectos:

- El primer método que se invoca de un portlet es, como hemos reflejado en la figura anterior, es el método *init* (). Se llama cuando el *portlet* es instanciado por el contenedor para inicializarlo antes de que el portal envíe alguna petición al mismo. Es decir, contiene la lógica que prepara al portlet para recibir peticiones. Vemos un ejemplo de implementación en el Código 7.

```
public void init(PortletConfig portletConfig) throws PortletException {
    super.init(portletConfig);
}
```

Código 7 Ejemplo método Init

❖ Suponiendo que el portlet inicia con éxito, cuando los usuarios acceden al portal (y tienen el portlet deseado en su página) se invoca el método *doView* () para representar el contenido del portlet. Un ejemplo de ello se muestra en el Código 8.

```
public void doView(RenderRequest renderRequest, RenderResponse renderResponse) throws IOException, PortletException {

    PortletContext portletContext = getPortletContext();

    PortletRequestDispatcher portletRequestDispatcher = portletContext.getRequestDispatcher("/view_alumno.jsp");

    if (portletRequestDispatcher == null) {
          _log.error("/view_alumno is not a valid include");
    }
    else {
        try {
            portletRequestDispatcher.include(renderRequest, renderResponse);
        }
        catch (Exception e) {
            _log.error(e, e);

            portletRequestDispatcher = portletContext.getRequestDispatcher("/error.jsp");

            if (portletRequestDispatcher == null) {
                _log.error("/error.jsp is not a valid include");
            }
            else {
                portletRequestDispatcher.include(
                    renderRequest, renderResponse);
            }
        }
    }
}
```

Código 8 Ejemplo método doView ()

En este caso, en este método se muestra el contenido de la página "*view_alumnos.jsp*" y en caso de que haya algún error, se redirige a la página "*error.jsp*" como se representa en el Código 9.

view_alumno.jsp | AlumnoDAO.java | DAOAlumnoPortlet.java | Alumno.java

```
1  <%@ taglib uri="http://java.sun.com/portlet_2_0" prefix="portlet" %>
2  <%@ taglib uri="http://liferay.com/tld/aui" prefix="aui" %>
3  <%@ taglib uri="http://liferay.com/tld/portlet" prefix="liferay-portlet" %>
4  <%@ taglib uri="http://liferay.com/tld/theme" prefix="liferay-theme" %>
5  <%@ taglib uri="http://liferay.com/tld/ui" prefix="liferay-ui" %>
6  <%@ page import="com.liferay.portal.kernel.util.Constants" %>
7  <%@ page import="com.liferay.portal.kernel.util.HtmlUtil" %>
8  <%@ page import="com.liferay.portal.kernel.util.ParamUtil" %>
9  <%@ page import="com.liferay.sampledao.model.Alumno" %>
10 <%@ page import="com.liferay.sampledao.model.Titulacion" %>
11 <%@ page import="com.liferay.sampledao.model.Asignatura" %>
12 <%@ page import="com.liferay.sampledao.model.AlumnoDAO" %>
13 <%@ page import="java.util.List" %>
14 <%@page import="com.liferay.portal.kernel.util.ListUtil"%>
15 <%@ page import="javax.portlet.WindowState" %>
16
17
18 <h2><font face="Book Antigua" color="black">Gestion de Alumnos</font></h2>
19
20 <liferay-theme:defineObjects />
21 <portlet:defineObjects />
22
23
24 <%
25     WindowState windowState = renderRequest.getWindowState();
26 %>
27
28 <form action="<portlet:actionURL />" method="post" name="<portlet:namespace />fm">
```

Código 9 Ejemplo redirección JSP

Es habitual que al comenzar a realizar el desarrollo de los portlets de aplicaciones surja la necesidad de enviar formularios y mostrar los datos en páginas JPS o incluso de almacenar los datos en la propia base de datos y como no podía ser de otro modo, en este caso no fue diferente, estas necesidades aparecieron.

Solamente indicar que en nuestra aplicación web usaremos formularios en formato HTML para publicar o transmitir los datos al servidor y nos apoyaremos del IDE de Liferay para la creación de los componentes. Vamos a proceder a explicar cómo se han gestionado los datos a través de formularios en conjunción con el método de acción y como estos han sido presentados en la página JSP. Explicaremos a su vez los detalles más relevantes dentro de su implementación.

En este portlet, a diferencia de otros que detallaremos posteriormente, se ha implementado toda la parte visual en una única página JSP y se ha realizado la ejecución de unas partes u otras dentro del documento a través de una constante, elemento en el que nos apoyaremos para disgregar el fichero en diversas partes según la acción requerida.

- Adentrándonos en el contenido de "*View_alumnos.jsp*", observaremos que en su cabecera, al inicio de su implementación, se definen el listado de librerías de etiquetas de portlet mostradas en el Código 10. Estas son lo equivalentes a los "*Imports*" empleados en Java, por tanto, nos permitirán emplear funciones ya desarrolladas en nuestros portlets.

```
<%@ taglib uri="http://java.sun.com/portlet_2_0" prefix="portlet" %>
<%@ taglib uri="http://liferay.com/tld/aui" prefix="aui" %>
<%@ taglib uri="http://liferay.com/tld/portlet" prefix="liferay-portlet" %>
<%@ taglib uri="http://liferay.com/tld/theme" prefix="liferay-theme" %>
<%@ taglib uri="http://liferay.com/tld/ui" prefix="liferay-ui" %>
<%@ page import="com.liferay.portal.kernel.util.Constants" %>
<%@ page import="com.liferay.portal.kernel.util.HtmlUtil" %>
<%@ page import="com.liferay.portal.kernel.util.ParamUtil" %>
<%@ page import="com.liferay.sampledao.model.Alumno" %>
<%@ page import="com.liferay.sampledao.model.Titulacion" %>
<%@ page import="com.liferay.sampledao.model.Asignatura" %>
<%@ page import="com.liferay.sampledao.model.AlumnoDAO" %>
<%@ page import="java.util.List" %>
<%@page import="com.liferay.portal.kernel.util.ListUtil"%>
<%@ page import="javax.portlet.WindowState" %>
```

Código 10 Imports Liferay

- A continuación, utilizando la biblioteca de etiquetas JSP nativo, se crean los contenidos con la información de los alumnos. Nada más invocar al portlet, se ejecuta las porciones de Código 11 y 12 albergadas en el contenido de la página.

```
<form action="<portlet:actionURL />" method="POST" name="fm">
<aui:input name="<%=Constants.CMD%>" type="hidden" value="" />
<aui:input name="Id_Alumno" type="hidden" value="" />
<input onClick="self.location = '<portlet:renderURL><portlet:param
name="<%=Constants.CMD%>" value="<%=Constants.ADD%>"/></portlet:renderURL>';"
type="button" value="Nuevo Alumno" /><br /><br />

<table border="1">
        <tr>
           <td bgcolor="#B5E3FC"><strong>NIA</strong></td>
           <td bgcolor="#B5E3FC"><strong>DNI</strong></td>
           <td bgcolor="#B5E3FC"><strong>Nombre</strong></td>
           <td bgcolor="#B5E3FC"><strong>Apellidos</strong></td>
           <td bgcolor="#B5E3FC"><strong>Titulacion</strong></td>
           <td bgcolor="#B5E3FC"><strong>Asignatura</strong></td>
           <td bgcolor="#B5E3FC"><strong>Mail</strong></td>
           <td bgcolor="#B5E3FC"><strong>Accion</strong></td>
</tr>

<%
 List<Alumno> ListaAlumnos = AlumnoDAO.obtenerTodosLosAlumnos();

   for (int i = 0; i < ListaAlumnos.size(); i++) {
        Alumno alumno = (Alumno)ListaAlumnos.get(i);
%>

    <tr>
       <td bgcolor="#E0F4FC"><%=alumno.getNIA()%></td>
        <td bgcolor="#E0F4FC"><%=alumno.getDNI()%></td>
        <td bgcolor="#E0F4FC"><%=alumno.getNombre()%></td>
        <td bgcolor="#E0F4FC"><%=alumno.getApellidos()%></td>
```

Código 11 Ejemplo Formulario

```
                <td bgcolor="#E0F4FC">
                <%=AlumnoDAO.obtenerNombreTitulacion(alumno.getTitulacionesId())%></td>
                <td bgcolor="#E0F4FC">
                <%=AlumnoDAO.obtenerNombreAsignaturas(alumno.getAsignaturasId())%></td>
                <td bgcolor="#E0F4FC"><%=alumno.getMail()%></td>
                <td bgcolor="#E0F4FC">
                        <input onClick="self.location = '<portlet:renderURL>
                        <portlet:param name="<%= Constants.CMD %>" value="<%=
                        Constants.EDIT%>"/>
                        <portlet:param name="Id_Alumno" value="<%=
                        String.valueOf(alumno.getAlumnosId())%>"/>
                        </portlet:renderURL>';" type="button" value="Editar"/>

                        <input onClick="self.location = '<portlet:renderURL>
                        <portlet:param name="<%= Constants.CMD %>" value="<%= Constants.DELETE
                        %>"/>
                        <portlet:param name="Id_Alumno" value="<%=
                        String.valueOf(alumno.getAlumnosId())%>"/>
                        </portlet:renderURL>';"  type="button" value="Eliminar" />
                </td>
        </tr>
      <% } %
</table>
<% } %>
</form>
```

Código 12 Continuación Ejemplo Formulario

Su propósito consiste en producir una página Web con una tabla que muestre el contenido de la tabla Alumnos de la base de datos, con botones para añadir, editar y eliminar nuevos alumnos a la base de datos como se muestra en la Figura 43.

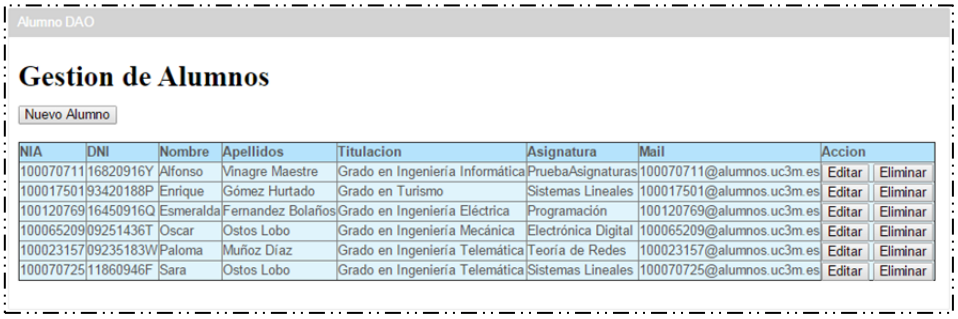


Figura 43 Portlet Gestión de Alumnos

- Con respecto al código implementado, destacar los siguientes aspectos:

En el elemento "action" del formulario, tenemos que insertar la *URL* de acción para invocar a los métodos del ciclo de vida de los *portlets*. Existen dos: *ActionURL* que ejecutará el método "*processAction*" y entonces se ejecutará el método de renderizado y el *RenderURL* que únicamente invocará el método "render" En este caso, hemos empleado el primer caso.

- ***Manejo captura de datos de formulario en el método processAction***

Si revisamos el método nombrado, vemos en el Código 13 el modo en que se realiza la obtención de los valores de los parámetros, que serán utilizados en la jsp.

```
public void processAction(ActionRequest actionRequest, ActionResponse actionResponse) throws IOException, PortletException {

    cmd = ParamUtil.getString(actionRequest, Constants.CMD);

    // Obtenemos los valores de los atributos
    Id_Alumno = ParamUtil.getLong(actionRequest, "Id_Alumno");
    titulacionesId = ParamUtil.getLong(actionRequest, "titulacionesId");
    asignaturasId = ParamUtil.getLong(actionRequest, "asignaturasId");

    NIA = actionRequest.getParameter("NIA");
    DNI = actionRequest.getParameter("DNI");
    Nombre = actionRequest.getParameter("nombre");
    Apellidos = actionRequest.getParameter("apellidos");
    Mail = actionRequest.getParameter("mail");
```

a)

b)

Código 13 Ejemplo formulario método ProcessAction

Vamos a obtener datos de entrada de formulario o valores de los parámetros petición de las siguientes maneras:

- ***ParamUtil.getType (actionRequest, "inputName/requestParameterName")*** - Liferay dispone la clase "*ParamUtil*" que ofrece diversos métodos y en este caso, a emplear el remarcado para capturar los valores de los parámetros de petición o formar valores de entrada. En este caso, debemos hacer un casting al tipo de dato primitivo deseado (*String, Long, Double*, etc.). En el caso concreto, únicamente hemos definido variables de tipo Long con este formato.
- ***actionRequest.getParameter ("inputName/requestParameterName")*** - Emplearemos este método para obtener los valores en el método de acción de la clase portlet.
- ***ParamUtil.getParameterValues (--)*** - Existe un tercer modo, para el caso concreto en que un usuario pueda seleccionar varias opciones simultáneamente pero no lo hemos empleado en nuestro proyecto.

❖ ***Gestión BBDD***

En base a cada opción seleccionada, vamos a mostrar la porción de código que se ejecutará:

- ***Insertar un Registro en la Base de Datos (Nuevo Alumno)***

A la hora de crear un nuevo alumno, se deberá rellenar un formulario con los datos correspondientes. El modo de implementarlo es el mostrado en el Código 14 y 15.

```
<aui:model-context bean="<%=alumno%>" model="<%=Alumno.class%>" />

    <aui:fieldset>

     <% List<Titulacion> ListaTitulaciones = AlumnoDAO.obtenerTitulaciones();
        if (ListaTitulaciones!= null){
     %>
        <font face="Times New Roman" size="3"
        color="black"><i><strong>Titulacion</strong></i></font>

      <aui:select label="" name="titulacionesId" >
        <% for (Titulacion titulaciones : ListaTitulaciones) { %>
          <aui:option
               value="<%=titulaciones.getTitulacionId()%>"><%=titulaciones.getNombre()%>
           </aui:option>
        <% } %>
      </aui:select>
     <%} %>

     <% List<Asignatura> ListaAsignaturas = AlumnoDAO.obtenerAsignaturas();
        if(ListaAsignaturas!= null){
     %>
```

Código 14 Implementación "Nuevo Alumno"

```
        <font face="Times New Roman" size="3"
        color="black"><i><strong>Asignatura</strong></i></font>

      <aui:select label=""  name="asignaturasId" >
        <% for (Asignatura asignaturas : ListaAsignaturas) { %>
         <aui:option
              value="<%=asignaturas.getAsignaturasId()%>"><%=asignaturas.getNombre() %>
         </aui:option>
        <% } %>
      </aui:select>
     <%} %>

        <font face="Times New Roman" size="3"
        color="black"><i><strong>NIA</strong></i></font></br>
        <aui:input  label=""  name="NIA" type="text" maxlength="9"/>
        <font face="Times New Roman" size="3"
        color="black"><i><strong>DNI</strong></i></font></br>
        <aui:input label=""  name="DNI" type="text" maxlength="9"/>
        <font face="Times New Roman" size="3"
        color="black"><i><strong>Nombre</strong></i></font></br>
        <aui:input label=""  name="nombre" type="text" maxlength="25"/>
        <font face="Times New Roman" size="3"
        color="black"><i><strong>Apellidos</strong></i></font></br>
        <aui:input label=""  name="apellidos" type="text" maxlength="50"/>
        <font face="Times New Roman" size="3"
        color="black"><i><strong>Mail</strong></i></font></br>
        <aui:input label=""  name="mail" type="text" maxlength="40"/>
</aui:fieldset>

<aui:button name="saveButton" type="submit" value="Guardar" />

<portlet:renderURL var="cancelURL" windowState="normal">
     <portlet:param name="jspPage" value="/view_alumno.jsp"/>
</portlet:renderURL>
<aui:button name="cancelButton" type="cancel" value="Cancelar" onClick="<%=cancelURL%>"/>
```

Código 15 Continuación Implementación "Nuevo Alumno"

- Para poder gestionar los campos en nuestra jsp, hemos empleado el taglib que nos proporciona AUI lo que nos evita realizar ningún esfuerzo adicional en el código, solamente usando en nuestro formulario el componente "<aui:model-context bean …>". Se ha utilizado el *framework AUI* proporcionado por el producto para el desarrollo de formularios, ya que facilita la construcción dinámica y flexible de los componentes web así como su validación. Simplemente son *tags* como los de *HTML*, precedidos por los caracteres "aui".
- A continuación he reseñado dos partes de código en los que se mostrarán, dos combos de datos con los nombres de las titulaciones y asignaturas existentes.
- Este framework, permite validaciones en los campos de los formularios muy cómodas, en este caso, podemos definir el tamaño máximo de caracteres que se pueden insertar en los campos de texto.
- En la última sección, se muestra el uso de *renderURL* con la redirección, en caso de cancelación a la misma página. De igual modo podría indicarse una jsp diferente.

1. *Editar un Registro en la Base de Datos (Editar)*

En este caso, se ejecuta la misma sección de código pero previamente se obtienen los datos del alumno empleando la sentencia del Código 16.

```
Alumno alumno = AlumnoDAO.obtenerUnAlumno(Id_Alumno);
```

Código 16 Implementación Opción "Editar Alumno"

De modo visual, se recuperarán los valores de la fila seleccionada y podremos editar el que deseemos, en el ejemplo, la asignatura. Salvando el cambio veremos el registro editado en la página principal.

2. Eliminar un Registro en la Base de Datos (Eliminar)

Si lo que se realiza, sin embargo es un borrado, veremos de nuevo un formulario con los datos del alumno que vamos a eliminar y al aceptar veremos que el registro desaparece de la tabla.

- El método ***processAction***() se invoca cuando el usuario envía una de las formas que presta el portlet (*doView* () o el método de *doEdit* (), en este caso, solo hemos implementado la primera opción). De modo más aclaratorio, indicar que este método se invoca siempre que un usuario envíe cambios a un portlet, ya que este método está destinado a procesar la acción realizada por un usuario. Aplicado a nuestro desarrollo, según la elección realizada por el usuario sobre los datos (Añadir nuevo registro, editar o eliminar un registro), vamos a almacenar dicha opción en una constante, proporcionada por el *Kernel* de *Liferay* y según el valor de está, se va ejecutarse una porción de código del método *processAction*() como podemos ver en las siguientes porciones de código.

  a) *Nuevo Alumno*: En este caso, editamos las propiedades del objeto alumno con los valores introducidos por el usuario a través de un formulario y posteriormente lanzamos la *query* contra nuestra tabla para insertar el registro en *BBDD* tal y como se muestra en el Código 17.

```
if (cmd.equals(Constants.ADD)) {

        alumno = new Alumno();

        alumno.setTitulacionesId(titulacionesId);
        alumno.setAsignaturasId(asignaturasId);
        alumno.setNIA(NIA);
        alumno.setDNI(DNI);
        alumno.setNombre(Nombre);
        alumno.setApellidos(Apellidos);
        alumno.setMail(Mail);

        AlumnoDAO.añadirUnAlumno(alumno);
}
```

Código 17 Porción de Código Ejecutado en processAction() al agregar "Nuevo Alumno"

  b) *Edición de Alumno*: A la hora de editar un alumno, primero lo recuperamos, posteriormente cambiamos los parámetros para registrar el que haya sido introducido por el usuario y actualizamos los resultados.

```
if (cmd.equals(Constants.EDIT)) {

        alumno = AlumnoDAO.obtenerUnAlumno(Id_Alumno);

        alumno.setTitulacionesId(titulacionesId);
        alumno.setAsignaturasId(asignaturasId);
        alumno.setNIA(NIA);
        alumno.setDNI(DNI);
        alumno.setNombre(Nombre);
        alumno.setApellidos(Apellidos);
        alumno.setMail(Mail);

        AlumnoDAO.actualizarAlumno(alumno);
}
```

Código 18 Porción de Código Ejecutado en processAction() al agregar "Edición de Alumno"

c) *Borrado de Alumno*: En este caso, se invoca al método de base de datos que realiza la sentencia "*DELETE*" sobre la tabla de alumnos a través del método descrito en el Código 19.

```
if (cmd.equals(Constants.DELETE)) {
        AlumnoDAO.eliminarUnAlumno(Id_Alumno);
}
```

Código 19 Porción de Código Ejecutado en processAction() al agregar "Edición de Alumno"

- El método *render*() se invoca cuando el portlet se vuelve a redibujar en el portal.
- Al finalizar el ciclo de vida, el contenedor de portlets se encarga de invocar al método *destroy*() para destruirlo cuando ya no se requiera su uso o en caso de que se apague el servidor.

Con respecto a los métodos que no forman parte de la especificación, sino que han sido desarrollados por nosotros, reseñar que la clase AlumnoDAO posee métodos con las consultas SQL(*Select, Insert, Update, Delete*..) contra las tablas existentes en nuestro modelo e interactúa con la clase ConnectionPool para manejar la apertura, gestión y cierre de conexiones con la base de datos.

Cada componente utilizará operaciones proporcionadas por los DAO de la capa de acceso a datos para almacenar datos en la base de datos.

Para más detalle consulte el ***Anexo V***.

6.2.4 Gestión de Base de Datos: DAO

El principal objetivo de esta capa consiste en la gestión de los datos del sistema que tengamos en nuestra base de datos. Para cada acceso, se define en esta capa una abstracción que permita gestionar su persistencia. A cada una de estas abstracciones se le conoce como *DAO* [32](*Data Access Object*).

El diseño realizado en esta capa ha querido preservar tres objetivos fundamentales:

- Realizar la gestión de datos de modo transparente para el resto del sistema.
- Ocultar al resto del sistema la tecnología de base de datos empleada (*MySQL u Oracle*).
- Realizar el acceso a la base de datos con la tecnología seleccionada de modo transparente al resto del sistema (*JDBC u Otros*).

Estos requisitos facilitan cambios en el tipo de repositorio de datos utilizado, en la tecnología de base de datos concreta y en la tecnología utilizada para el acceso a base de datos, consiguiendo una mayor gestionabilidad. Para alcanzar estos objetivos, se ha aplicado el patrón de diseño *Data Access Object (DAO)*.

- ***Data Access Object (DAO)***

Ante la necesidad que suele presentarse en los sistemas para acceder a BBDD mediante JDBC (*Java Database Connectivity*), como en nuestro caso, u otra librería de acceso a BBDD o incluso de recuperar información mediante W*eb Services* externos, surge una alternativa para encapsular el acceso a los datos y la forma de conseguirlos. Para el cliente debe ser transparente la obtención de

las conexiones, o de las consultas que se hacen, y para que esta abstracción sea posible se deben encapsular los resultados en objetos.

Este componente se encarga de disgregar la lógica de acceso a datos de la lógica de negocio y presentación. Esto lo realiza, abstrayendo y encapsulando el acceso a la fuente de datos de un sistema de información. Por tanto, facilita la edición del mecanismo de persistencia proporcionado por la aplicación.

El fragmento mostrado en Código 20, representa un método implementado en la clase "AlumnoDAO" el cual estará accesible por el resto de clases:

```
public static void eliminarUnAlumno(long Id_Alumnos) throws SQLException {


    try {

        con = ConnectionBBDD.getConnection();
        ps = con.prepareStatement("DELETE FROM Alumnos WHERE Id_Alumnos = ?");

        ps.setLong(1, Id_Alumnos);
        ps.executeUpdate();

    }
    finally {

        CerrarConexionBBDD();
    }
}
```

Código 20 Ejemplo Método Acceso Base de Datos

6.2.5 Configuración

A la hora de definir un portlet, es necesario editar ciertos ficheros de configuración. Vamos a conocer los más importantes y detalles de su contenido. Se tratan de ficheros que encontraremos bajo el directorio /WEB-INF/ de nuestro proyecto.

- ☑ ***liferay-display.xml*** – En su interior detallaremos la categoría bajo la que encontraremos el portlet una vez desplegado para adicionarlo a nuestra página.
  Como se muestra en la Figura 54, encontraremos en la categoría "*simple*" el portlet denominado "*Alumno DAO*"

```
<?xml version="1.0"?>
<!DOCTYPE display PUBLIC "-//Liferay//DTD Display 5.2.0//EN" "http://www.liferay.com/dtd/liferay-display_5_2_0.dtd">

<display>
    <category name="category.sample">
        <portlet id="Alumno DAO"></portlet>
    </category>
</display>
```

Figura 44 Fichero "liferay-display.xml"

  Si analizamos su contenido, veremos que nuestro portlet lo encontraremos en el portal en la categoría simple y se llamará Alumno DAO. El valor insertado en el campo Id debe coincidir con el definido en el fichero Portet.xml. Hemos definido todos los portlets bajos la misma categoría, pero según su funcionalidad pueden en conjuntos.

- ☑ ***portlet.xml*** – Se conoce con este nombre al descriptor de despliegue. Se trata de un fichero en el que se indicarán las características del portlet que se está manejando.

- ☑ ***liferay-portlet.xml*** – Consiste en una extensión del anterior y en su interior se definirán propiedades avanzadas del portlet.

6.3 Portlet Gestión de Asignaturas

El portlet "*Asignaturas-dao*" permite como su nombre indica, la gestión de alumnos, facilitándonos la creación, edición y eliminación de estos de forma simple, al igual que en el caso anterior. La gestión es igual que el portlet anterior porque se ha empleado la misma forma de desarrollo. De igual modo, se visualizarán los datos en una tabla y será adaptable por cada profesor en su *site* privado.

6.3.1 Funcionalidad

Al inicio, el sistema mostrará las asignaturas existentes en formato tabla para su posterior gestión así como botones con las opciones disponibles sobre los datos: Añadir un nuevo registro, modificar uno existente o eliminar una fila de la tabla.

- Si la opción seleccionada por el usuario es la de añadir una nueva asignatura, deberá rellenar un formulario con los campos: Departamento, Titulación, Nombre, ECTS, Tipo, Curso, Cuatrimestre. Una vez se pulse el botón de aceptar, aparecerá el listado con el nuevo registro incorporado.
- Si se selecciona la opción de edición de una fila, aparecerá un formulario con los datos existentes y el usuario tendrá la potestad modificar el que desee.
- Como última opción sobre este portlet, se podrá eliminar un registro seleccionando la fila sobrante.

6.3.2 Diagrama de Clases

En la Figura 45 se muestra el diagrama de clases que componen el portlet de gestión de asignaturas. En ella se han agrupado nuevamente las clases por empaquetado, según la disposición desarrollada.

Las clases Asignatura, Titulación y Departamento, se componen de los métodos de acceso a cada atributo, *Getters* y *Setters*.

Para esclarecer la interacción entre las clases, se presenta en el siguiente apartado un diagrama de secuencia con la actividad.

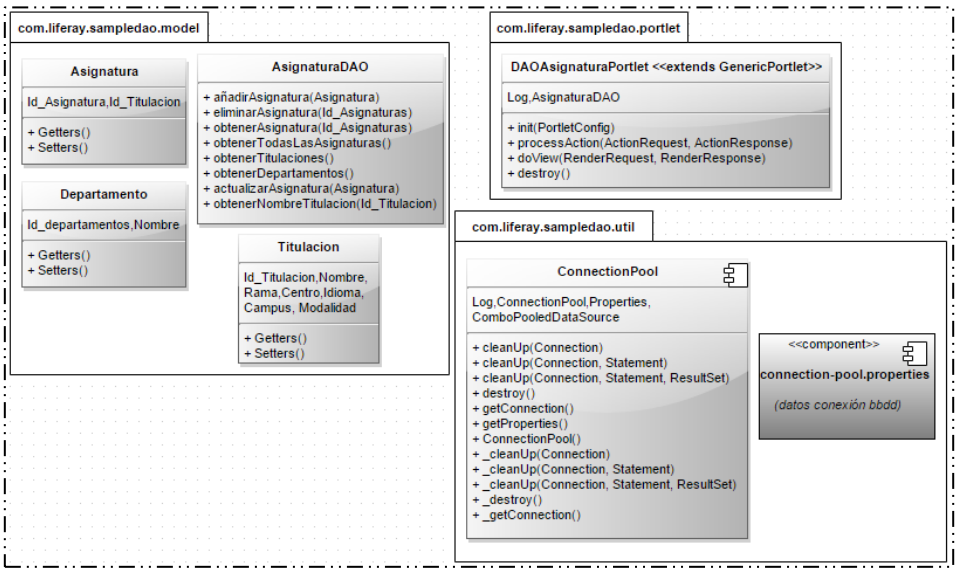


Figura 45 Diagrama de Clases: Gestión de Asignaturas

Para este portlet no vamos a entrar en más detalle porque se ha implementado del mismo modo que el anterior y también se accede a la base de datos a través del objeto DAO.

6.4 Portlet Gestión de Notas

El portlet "*AlumnoMVC-portlet*" permite realizar la gestión de notas del alumnado, facilitándonos la creación, edición y eliminación de estos de forma simple, a través de formularios, al igual que podremos realizar búsquedas a través de un buscador si queremos realizar un filtrado de los resultados. En este caso, vamos a pasar a detallar el modelo de desarrollo que ya cambia con respecto a los dos plugins anteriores. Los datos serán adaptable por cada profesor en su *Site* privado también en este caso.

6.4.1 Funcionalidad

Al inicio, el sistema mostrará un menú de opciones que permitirán: Crear una Nota, Modificar Nota, Buscar Alumno, Eliminar Nota y Ver todos los Alumnos. En este caso hemos empleado una navegación entre diversas JSPS ilustradas en la Figura 46 según la opción deseada por el mayor nivel de complejidad que en casos previos, entonces, se va a mostrar un esquema de las páginas existente para agilizar la comprensión del lector de la funcionalidad y se va a explicar las cosas más reseñables del contenido de cada una de ellas.

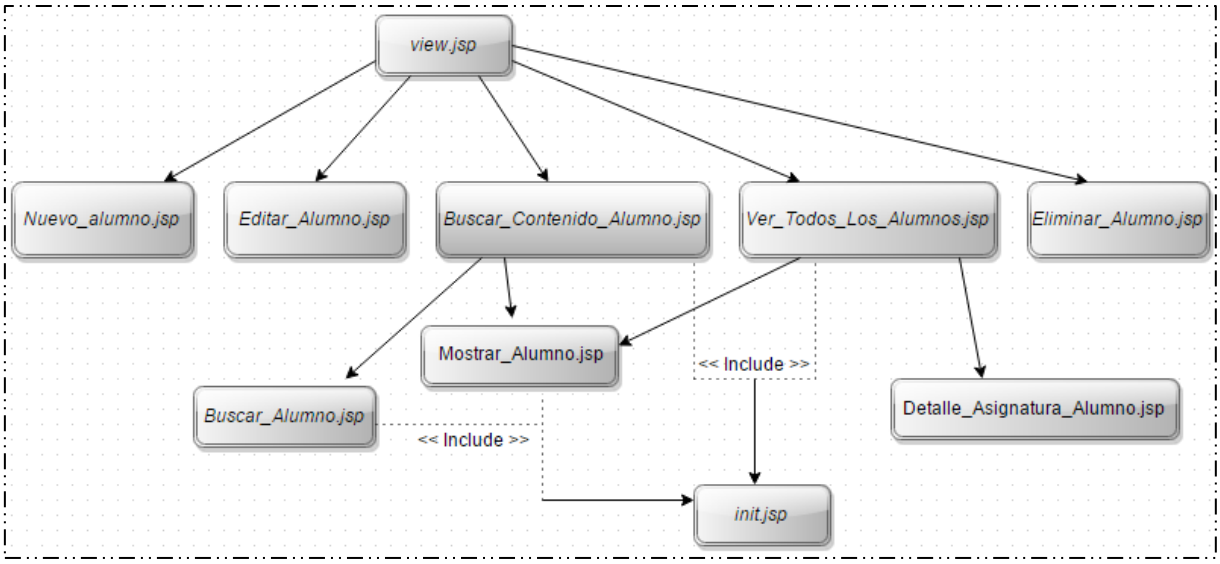


Figura 46 Páginas portlet Gestión de Notas

- ***View.jsp*** - En el modelo de desarrollo empleado en este portlet (Spring MVC), se indica en el fichero "portlet.xml" la primera jsp y paquete del portlet que se visualizará y ejecutará al invocarle, en este caso, view.jsp se albergará bajo el directorio "/html/jsps/" de nuestro proyecto web. En la Figura 47 se detalla un ejemplo de esta configuración.

```
<?xml version="1.0"?>

<portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd" xmln
    <portlet>
        <portlet-name>AlumnoMVC</portlet-name>
        <display-name>AlumnoMVC</display-name>
        <portlet-class>com.paq.liferaymvc.AlumnoMVCPortletAction</portlet-class>
        <init-param>
            <name>view-template</name>
            <value>/html/jsps/view.jsp</value>
        </init-param>
```

Figura 47 Ejemplo Fichero "*portlet.xml*"

En el fichero JSP encontraremos la configuración oportuna para crear el menú de opciones inicial. En este caso, se ha empleado el *tag "<portlet:renderURL>"* en el que definiremos una variable en cuyo *path* agregaremos la ruta física donde se encuentra la jsp y se creará una referencia a través de *"<a href">* que apuntará a dicha variable para que a raíz de la opción elegida, se redirija a una página u otra. He aquí una porción del Código 21 descrito.

```
<portlet:renderURL var="NuevoAlumno">
    <portlet:param name="mvcPath" value="/html/jsps/Nuevo_alumno.jsp"/>
</portlet:renderURL>

<a href="<%=NuevoAlumno.toString()%>">
    <font face="Times New Roman" size="3" color="black">
        <i>Crear Nota</i>
    </font>
</a><br/>
```

Código 21 Definición de Menú de Opciones

Al pulsar la opción "Crear Nota", pasaríamos de la página *view.jsp* a *Nuevo_alumno.jsp*.

- ***Nuevo_alumno.jsp*** - En este desarrollo, vamos a acceder a los datos a través de servicios, más conocido como "*Service builder*" que se detallará en un punto concreto. Para tener acceso en este modelo a los métodos expuestos desde las clases java, deberemos importar los paquetes de las clases deseadas en nuestras páginas JSPS como se puede ver en la figura del Código 22.

```
<%@page import="com.paq.dbservice.service.AlumnoLocalServiceUtil"%>
<%@page import="com.paq.dbservice.model.Alumno"%>
```

Código 22 Import Service Builder

Con esta implementación, tendremos acceso a los atributos y métodos de estas clases.

En esta página, tendremos un formulario con los campos: Nombre, Apellidos, NIA, DNI, Titulación, Año, Asignatura, Convocatoria, Nota y Curso. Todos serán campos de tipo "input" menos los campos titulación, año y asignatura que serán "combos" rellenos con valores obtenidos de consultas SQL y el campo Curso que será de tipo "radio button" puesto que solo será seleccionable un curso cada vez. Mostramos un ejemplo del campo Curso.

```
<b><font face="Times New Roman" size="3" color="black"><i>Curso</i></font></b><br/>
<input type="radio" name="<portlet:namespace/>Curso" value="1"><font face="Times New Roman" size="3" color="black">Primero</font><br>
<input type="radio" name="<portlet:namespace/>Curso" value="2"><font face="Times New Roman" size="3" color="black">Segundo</font><br/>
<input type="radio" name="<portlet:namespace/>Curso" value="3"><font face="Times New Roman" size="3" color="black">Tercero</font><br/>
<input type="radio" name="<portlet:namespace/>Curso" value="4"><font face="Times New Roman" size="3" color="black">Cuarto</font><br/><br/>
```

Código 23 Ejemplo Código Radio Button JSP

- Se empleará el tag " *<portlet: namespace />* " en los atributos de entrada "name" para que estos sean accesibles desde el método de acción del portlet, ya que, de cualquier otro modo, serán ignorados, es decir, nos permitirá gestionarlos para su edición. Este comportamiento se puede observar a partir de la versión 6.2 de Liferay.

  Esta etiqueta proporcionará un nombre único a los elementos de la página y estos elementos están asociados a los respectivos portlets, esto evitará conflictos de nombres en la página. En nuestro caso, no se trata de algo crítico, ya que solo tenemos un portlet, pero pueden existir diversos y pueden producirse colisiones de nombres.

En este caso, hemos desarrollado métodos de acción según la opción seleccionada, por ello, como vemos en el Código 24 , al pulsar el botón de guardar una vez completado el formulario, se ejecutará el método "*addAlumno*" que estará albergado en la clase "*AlumnoMVCPortletAction*"

```
<portlet:actionURL var="addAlumnoActionURL" windowState="normal"
name="addAlumno"></portlet:actionURL>

<input type="submit" name="addAlumno" id="addAlumno" value="Guardar"/>
```

Código 24 Invocación del método addAlumno() de la clase "AlumnoMVCPortletAction"

Este método, a diferencia de los desarrollados anteriormente porque se emplean como ya se ha mencionado servicios, por tanto, se dispone de muchos métodos accesibles pero ya predefinidos que podemos invocar pero no debemos implementar.

Si nos adentramos en el método "*AddAlumno*", hay cosas novedosas a detallar.

I. Al comienzo se recuperan los parámetros haciendo uso del método "*ParamUtil.getTipoPrimitivo*". Un ejemplo se ve en el Código 25.

```
float Nota = ParamUtil.getFloat(actionRequest, "Nota");
int Curso = ParamUtil.getInteger(actionRequest, "Curso",1);
String NIA = ParamUtil.getString(actionRequest, "NIA");
```

Código 25 Método empleado para recuperar Parámetros

II. A continuación se invoca el método "*CounterLocalServiceUtil.increment();" mostrado en el Código 26* para generar una nueva *primary key* antes de realizar la inserción un nuevo registro.

```
long AlumnoId = CounterLocalServiceUtil.increment();
```

Código 26 Método empleado para crear clave primaria

III. Generamos un alumno y lo inicializamos través de un objeto persistente con el id generado en el paso II. La implementación se recoge en el Código 27.

```
Alumno alumno = null;
alumno = new AlumnoImpl();
alumno = AlumnoLocalServiceUtil.createAlumno(AlumnoId);
```

Código 27 Creación de Alumno e inicialización

IV. Rellenamos sus parámetros a través de los Setters de la clase Alumno y pasándole por parámetro los valores obtenidos de la JSP en el paso I como se ve en el Código 28.

```
alumno.setNota(Nota);
alumno.setCurso(Curso);
alumno.setNIA(NIA);
```

Código 28 Agregar contenido a los Parámetros

V. Añadimos el objeto persistente a la tabla de base de datos de Notas con la sentencia del Código 29 que invoca al servicio de igual nombre(addAlumno).

```
alumno = AlumnoLocalServiceUtil.addAlumno(alumno);
```

Código 29 Inserción del objeto en la base de datos

VI. Por último, mostramos un mensaje en el propio portlet informando del éxito o fracaso de la inserción. Para ello se emplea el Código 30:

```
if (alumno != null) {
    SessionMessages.add(actionRequest.getPortletSession(),"Add-OK");
    log.info(" - Nueva nota insertada OK - ");

} else { //Si algo falla
    SessionErrors.add(actionRequest.getPortletSession(),"Add-KO");
    log.error(" - Error insertando Nueva Nota - ");
}
```

Código 30 Mensaje Inserción Registro Base de Datos

Que devolverá un resultado captado desde la JSP en el Código 31.

```
<% if(SessionMessages.contains(renderRequest.getPortletSession(),"Add-OK")){%>
    <liferay-ui:success key="Add-OK" message="¡Nueva Nota insertada Correctamente!" />
<%} %>
<% if(SessionErrors.contains(renderRequest.getPortletSession(),"Add-KO")){%>
    <liferay-ui:error key="Add-KO" message="¡ERROR insertando Nueva Nota!" />
<%} %>
```

Código 31 Gestión del resultado de la transacción

Donde se observa que si el método contiene el valor "*Add-OK*", el usuario verá el texto *"¡Nueva Nota insertada Correctamente!"* por pantalla mientras que si contiene el valor "*Add-KO*", el usuario verá *"¡ERROR insertando Nueva Nota!"* y será consciente del resultado de la transacción.

- ***Editar_Alumno.jsp*** – En este caso, se muestra un combo con los *IDs* de notas existentes. Una vez seleccionado uno, se va mostrando en la parte inferior los datos dela nota a la que corresponde, por lo tanto el usuario no tiene que conocer previamente los *IDs*.

Una vez elegido uno, tendrá los valores originales cargados en cada campo y podrá seleccionar el/los que desee.

- ***Buscar_Contenido_Alumno.jsp y Ver_Todos_Los_Alumnos.jsp*** - Estas dos vistas, junto emplean un componente presente en las librerías de Liferay con cierta complejidad denominado "*search container*" al que hemos dedicado la siguiente sección.
- ***Eliminar_Alumno.jsp*** – Contamos con la opción de eliminar un registro en el que se obtendrá la información de la nota a borrar a través de su *ID*.
- ***Init.jsp*** – Por último mencionar una jsp, que no contiene código referente a la vista, solo tiene las importaciones de taglib y la inicialización de atributos para su posterior uso. Se emplea para no recaer en redundancia de código de importaciones.

6.4.2 Uso de Liferay-ui:Search-Container

Este elemento permite mostrar listados de objetos, y representarlos en filas de una tabla. En nuestro caso, vamos a representar objetos de persistencia que han sido previamente creados mediante *Service-Builder*. Las tablas que hemos definido no cuentan con ordenación. Vamos a detallar su implementación:

1. En primer lugar hemos generamos una URL que permita redibujar la página. Esto es posible a través del tag mostrado en el Código 32.

```
<liferay-portlet:renderURL varImpl="BuscarAlumnoURL">
    <portlet:param name="mvcPath" value="/html/jsps/Buscar_Contenido_Alumno.jsp" />
</liferay-portlet:renderURL>
```

Código 32 Repintado de la página

2. A continuación definimos un formulario de método GET, al contrario que en los anteriores caso que eran POST y vamos a definir los parámetros a obtener. Esto se ve en el Código 33.

```
<aui:form action="<%=BuscarAlumnoURL%>" method="get" name="alumnoFormulario">
    <liferay-portlet:renderURLParams varImpl="BuscarAlumnoURL" />
    <liferay-portlet:renderURL varImpl="iteratorURL">
        <portlet:param name="Nombre" value="<%= Nombre %>" />
        <portlet:param name="Apellidos" value="<%= Apellidos %>" />
        <portlet:param name="Nota" value="<%= String.valueOf(Nota) %>" />
        <portlet:param name="Titulacion" value="<%= Titulacion %>" />
        <portlet:param name="Curso" value="<%= String.valueOf(Curso) %>" />
        <portlet:param name="Asignatura" value="<%= Asignatura %>" />
        <portlet:param name="NIA" value="<%= NIA %>" />
        <portlet:param name="DNI" value="<%= DNI %>" />
        <portlet:param name="mvcPath" value="/html/jsps/Buscar_Contenido_Alumno.jsp" />
    </liferay-portlet:renderURL>
```

Código 33 Método GET

3. Definimos ahora el contenedor en sí mediante la cláusula indicada en el Código 34.

```
<liferay-ui:search-container
    displayTerms="<%= new DisplayTerms(renderRequest) %>"
    emptyResultsMessage="No-hay-Registros"
    headerNames="Titulacion,Nombre,Apellidos,Nota,Curso,Asignatura,NIA,DNI"
    delta="4"
    iteratorURL="<%= iteratorURL %>"
    >
```

Código 34 Creación Contenedor

Existen varios atributos, entre ellos, *emptyResultsMessage* que indicará el mensaje a mostrar al usuario en caso de no haber registros, el parámetro *headerNames* que se corresponde con las columnas de la tabla de Notas, el campo delta que muestra el número máximo de filas que tendrá la tabla de resultados (será el parámetro empleado para realizar la paginación de resultados) e *iteratorURL* donde apuntaremos a la URL generada en el punto anterior, para, en el caso de que haya paginación, se repinte la página.

4. Tenemos a continuación el tag que emplearemos para realizar esta búsqueda por cada objeto que obtengamos en la jsp *Buscar_Alumno*, mostrado en el Código 35.

```
<liferay-ui:search-form
    page="/html/jsps/Buscar_Alumno.jsp"
    servletContext="<%= application %>"
/>
```

Código 35 Ejemplo <liferay-ui:search-form>

5. En este punto realizamos la gestión de resultados. En el tag *"<liferay-ui:search-container-results>"* encontraremos el código perteneciente al buscador. A la hora de realizar la búsqueda, se puede insertar o no un valor en el buscador. En caso de no incluir ninguno y pulsar únicamente el botón de búsqueda, se obtienen el total de registros existentes y sus valores. En caso contrario, se hará un macheo del campo insertado y se filtran los resultados únicamente que contengan el texto citado. Para que el contenedor se consciente de la totalidad de páginas diferentes, hemos de pasarle la variable ***total*** que contiene el tamaño del listado.

6. En este momento, definiremos otra de las etiquetas más importantes, como el mostrado en el Código 36.

```
<liferay-ui:search-container-row
    className="Alumno"
    keyProperty="alumnoId"
    modelVar="alumno">
```

Código 36 Ejemplo tag <liferay-ui:search-container-row>

En ella se identifica la clase Java a partir de la que se genera la lista, incluida en la propiedad *className* y se identificará una de sus propiedades como la principal, en este caso, se tratará de su clave primaria en la propiedad *keyProperty* y definiremos una variable llamada *alumno* para realizar el acceso a cada uno de los objetos de cada iteración empleando el parámetro *modelVar*.

7. Por último, definiremos las columnas que compondrán nuestra tabla con el siguiente esquema, reiteradas por cada propiedad del bean que deseemos mostrar. La implementación se realiza con el Código 37.

```
<liferay-ui:search-container-column-text
    href="<%= rowURL %>"
    name="Titulacion"
    property="titulacion"
/>
<!-- name es el nombre de la columna -->
<liferay-ui:search-container-column-text
    href="<%= rowURL %>"
    name="Nombre"
    property="nombre"
/>
```

Código 37 Ejemplo uso <liferay-ui:search-container-column-text>

Cada una de ellas contará con el nombre de la misma, definido en el parámetro *name*, y será el mostrado en la cabecera de la tabla y con el parámetro *property* que identificará una propiedad del bean que se está iterando y se encargará de mostrar el valor de dicha propiedad.

8. Para concluir, únicamente nos faltaría cerrar la tabla y el formulario con la porción de Código 38.

```
<liferay-ui:search-iterator searchContainer="<%=searchContainer %>" />
    </liferay-ui:search-container>
</aui:form>
```

Código 38 Cierre del formulario y tabla

6.4.3 Diagrama de Clases

Una vez detallada la parte visual y conocido el modelo de gestión de los datos, vamos a mostrar el diagrama de clases que componen el portlet que nos atañe.

Antes de mostrar los diagramas, nos surge una duda, *¿Por qué se generan tantas clases?*

Pues bien, el gestor de servicio sigue una filosofía de diseño llamado acoplamiento débil. A nivel macro, está formado por tres capas: modelo, servicio, y las capas de persistencia. Este acoplamiento se define como débil debido a que la aplicación está diseñada de tal modo que se puedan realizar modificaciones sobre la capa de persistencia produciendo el mínimo o ningún impacto en el resto de capas.

Si lo analizamos a bajo nivel, veremos que cada capa se implementa utilizando interfaces Java y múltiples implementaciones de estas. En lugar de tener una clase que defina el modelo, la capa *Service Builder* [33] ha generado un sistema de clases que incluyen una interfaz de entrada, una clase abstracta *EntryBaseImpl* para realizar la gestión, y una clase *EntryImpl*, esta última personalizable. Esto otorga la flexibilidad de poder cambiar nuestro modelo, al tiempo que nos permite generar el código adicional que deseemos. Por eso y como mayor potencial, podemos decir de esta capa permite genera la parte de código que de otro modo es un proceso realmente tedioso de definir y ofrece da la libertad de personalizar al gusto.

Sin más preámbulo y debido a su complejidad no vamos a definir todos los métodos que forman las clases sino que vamos a realizar una división de clases según la funcionalidad que aportan en el sistema.

- ***Modelo***

El esquema de iteración entre los componentes se muestra en la Figura 48.

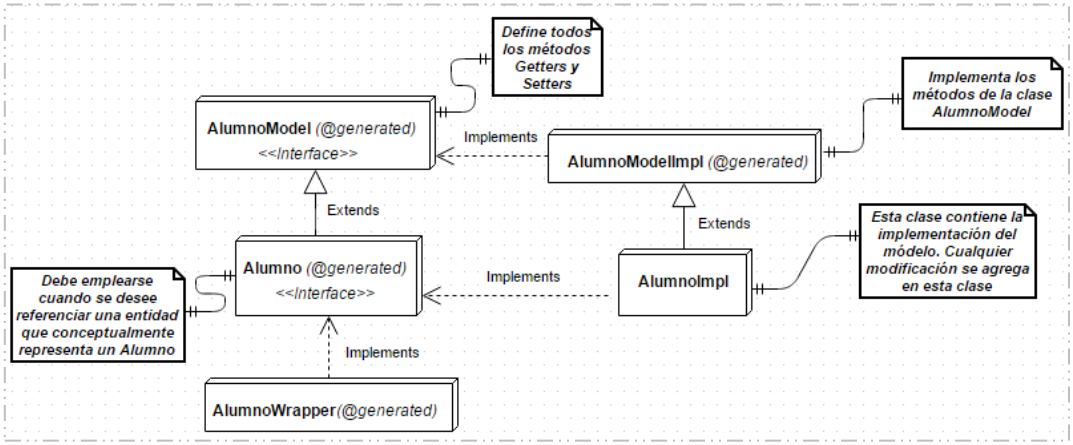


Figura 48 Clases que componen el Modelo: Portlet Gestión de Notas

- ***Persistencia***

La relación entre los elementos se detalla en la Figura 49.

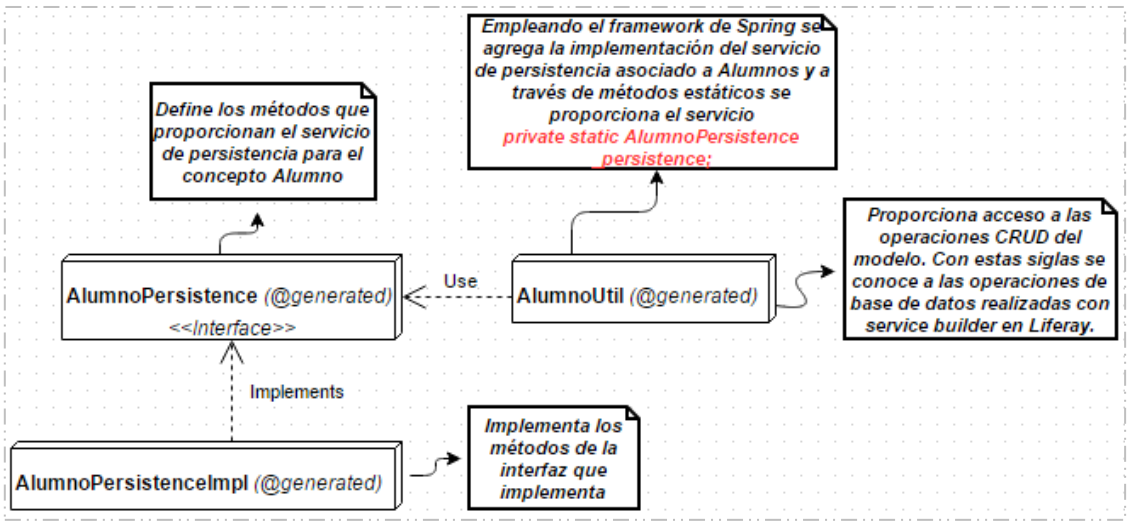


Figura 49 Clases que componen la persistencia: Portlet Gestión de Notas

- ***Servicio Local de Negocio***

El esquema de relación entre los elementos se muestra en la Figura 50.

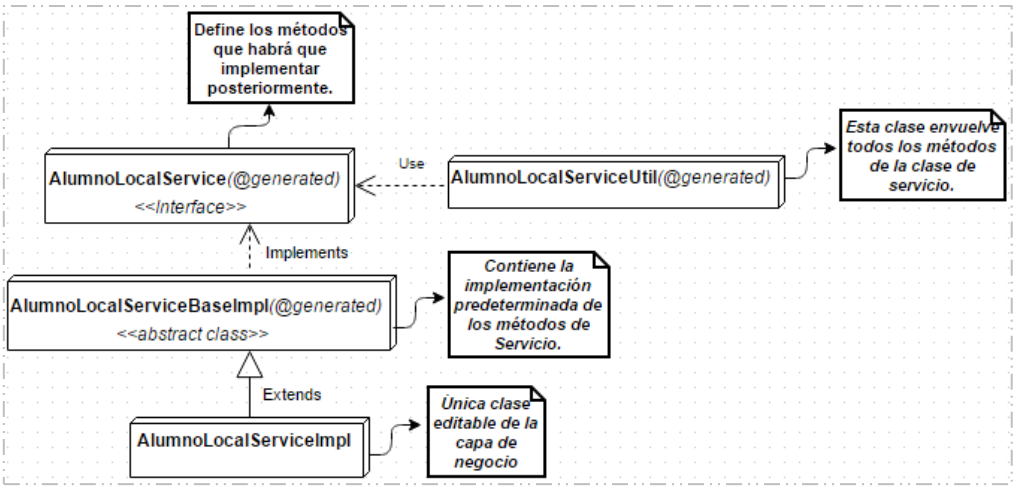


Figura 50 Clases que componen el Servicio Local de Negocio: Portlet Gestión de Notas

- ***Servicio Remoto de Negocio***

La iteración entre los componentes que componen el servicio se muestra en la Figura 51.

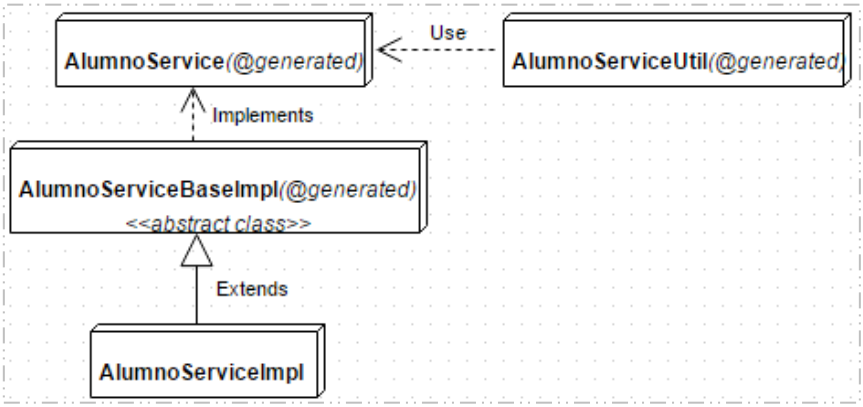


Figura 51 Clases que componen el Servicio Remoto de Negocio: Portlet Gestión de Notas

En resumen; de todas las clases mencionadas, únicamente se pueden modificar: *AlumnoLocalServiceImpl*, *AlumnoServiceImpl y AlumnoImpl*.

La principal diferencia entre los servicios locales y los remotos radica en que los primeros definen servicios que se encontrarán accesibles en un entorno local o en el entorno de la aplicación y podrá ser invocado mediante código java, mientras que los segundos van a ser accesibles desde el exterior por ejemplo en el caso de querer exponer nuestros datos a otras plataformas como servicio web o a aplicaciones móviles y será invocado mediante servicios web. En nuestro caso hemos generados ambos pero únicamente empleamos los servicios locales.

Después de mencionar las clases generadas automáticamente, es importante detallar el diagrama de la clase *action* de nuestro portlet de la que si detallaremos los métodos que la conforman indicado en la Figura 52.

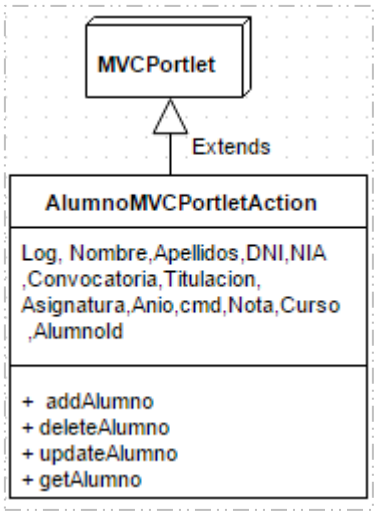


Figura 52 Diagrama Clase AlumnoMVCPortletAction

Y la clase *"AlumnoCompararNombre"* que únicamente hará una comparativa entre dos objetos por el campo nombre y los ordenará de forma ascendente o descendente.

En el presente proyecto, cada una de las opciones que podemos realizar desde el menú se corresponde con una opción *CRUD* ya que ejecuta acciones sobre el modelo de datos. Hemos incorporado un apartado donde detallaremos como se ha realizado cada una de estas operaciones sobre la Base de Datos.

6.4.4 Gestión de Base de Datos: Liferay Service Builder

Existen varias maneras de desarrollar portlets que utilicen la base de datos de Liferay, pero en este caso nos hemos apoyado en un componente de Liferay IDE: *Service Builder (Gestor de Servicio)*.

Consiste en una herramienta que permite generar de modo automático, en base a modelos construidos por *Liferay*, las clases e interfaces necesarias para crear una capa de servicio y poder manejar la persistencia de base de datos. Este elemento generará la mayor parte del código común necesario para implementar búsquedas, creaciones, actualizaciones y operaciones de borrado en la base de datos, lo que le permite centrarse en los aspectos de mayor nivel de diseño de servicios.

Service Builder genera automáticamente el código necesario para permitir el acceso a los servicios remotos utilizando *SOAP, JSON y Java RMI*.

- ***Definición del Modelo de Datos***

El primer paso para utilizar este componente consiste en definir las clases del modelo y sus atributos en un archivo denominado "*service.xml*". En este caso, con solo definir la estructura de nuestro modelo de datos en este archivo se genera automáticamente toda la configuración para que nuestro portlet pueda interactuar con la base de datos, lo cual es potencialmente atractivo frente a las conocidas conexiones JDBC.

Como bien hemos dicho, debemos describir nuestra tabla o conjunto de tablas en el fichero indicado, definido en el directorio "/WEB-INF/" de nuestro proyecto.

- Por entender su contenido:

En la cabecera nos topamos con el paquete donde se almacenará el código fuente ("*com.paq.dbservice*") y el namespace, *LF* bajo el que se ha definido la tabla de nombre Alumno, por tanto, en nuestra base de datos "alumnosuc3m" tendremos una tabla de nombre *LF_Alumno* que tendrá la información relativa a las notas de los alumnos de nuestro portal.

A continuación tenemos la definición de la entidad indicada, en este caso únicamente hemos definido una pero se pueden incluir las que sean necesarias. Hemos definido una entidad que obtiene el parámetro "*local-service=true*" y "*remote-service=true*", ya que vamos a definir entidades para su interacción con la base de datos.

Por cada entidad tenemos la definición de cada columna con su tipo de dato Java correspondiente. Es importante, al igual que en MySQL, que cada entidad tenga definida siempre una clave primaria (primary="true"), en este caso *AlumnoId*.

Y por último contamos con un parámetro de ordenación que será aplicado a la entidad una vez que se recupere desde base de datos. En este caso consistirá en una ordenación por clave primaria en orden ascendente.

Una vez definida la estructura, y utilizando Service Builder, tendremos generada la capa de servicio con la que podremos interactuar con nuestra base de datos. Este proceso genera una gran cantidad de clases y empaquetados de modo automático que conoceremos en el siguiente apartado en el que veremos el diagrama de clases que componen el portlet. Únicamente indicar que a este patrón de desarrollo se le conoce como *MVC*.

Un ejemplo de este fichero lo encontramos en el Código 39:

```
<service-builder package-path="com.paq.dbservice">
    <author>Sara Ostos Lobo</author>
    <namespace>LF</namespace>

    <entity name="Alumno" local-service="true" remote-service="true">
        <!-- PK fields -->
        <column name="AlumnoId" type="long" primary="true" />
        <column name="Titulacion" type="String"  />
        <column name="Asignatura" type="String" />
        <column name="Nombre" type="String" />
        <column name="Apellidos" type="String" />
        <column name="Nota" type="float" />
        <column name="Curso" type="int" />
        <column name="Anio" type="String" />
        <column name="Convocatoria" type="String" />
        <column name="NIA" type="String" />
        <column name="DNI" type="String" />
        <!-- Order -->
        <order by="asc">
            <order-column name="AlumnoId" />
        </order>
        </entity>

</service-builder>
```

Código 39 Contenido Fichero "Service.xml"

❖ ***Creación del Servicio***

Una vez creado el servicio a través de la opción "*Build Services* " o gestor de servicio en *Eclipse* se generarán una serie de paquetes que contienen la capa de modelo, servicio y persistencia. Vamos a aportar más detalle en la siguiente sección una vez que conozcamos el diagrama de clases que forman este componente.

6.4.5 Operaciones CRUD(Interacción BBDD)

Mayormente, emplearemos métodos relacionados con la clase "***Util***" para realizar las operaciones de interacción con nuestra base de datos. Estas ejecuciones se realizarán en la clase "***AlumnoMVCPortletAction***" de la cual vamos a explicar la implementación y funcionalidad de sus métodos.

☑ ***Insertar/Crear Nuevo Contenido***

Cuando añadimos una nueva nota a nuestra tabla, se va a ejecutar el método "***addAlumno***" cuya ejecución se puede englobar en los siguientes pasos:

En primer lugar, obtendremos los valores de los parámetros a insertar por medio del método, ya conocido, "*ParamUtil.getTipoPrimitivo(actionRequest, "Atributo a Recuperar")"*. A continuación deberemos crear una nueva clave primaria única para el registro. Esto lo llevaremos a cabo a través del método *"CounterLocalServiceUtil.increment()"*. Posteriormente, crearemos un objeto persistente, con el método "*AlumnoLocalServiceUtil.createAlumno(AlumnoId)" y* lo rellenamos con los valores obtenidos al comienzo, empleando los métodos *Setters* de cada atributo.(P.ej: *alumno.setTitulacion(Titulacion)).*

Una vez rellenado nuestro objeto, nos quedará añadirlo en la base de datos, cosa que haremos con la sentencia: "*AlumnoLocalServiceUtil.addAlumno(alumno)".*

Por último, devolveremos los datos en la JSP correspondiente a la operación que estamos ejecutando. En esta operación, se realiza mediante la porción de código: "*actionResponse.setRenderParameter("mvcPath","/html/jsps/Nuevo_alumno.jsp");"*.

☑ ***Eliminar un Registro***

En el caso del borrado, se va a ejecutar el método *"**deleteAlumno**".* Su implementación va a ser sencilla, ya que únicamente necesitamos la clave principal para proceder a la eliminación del registro. Existe dos modalidades de borrado: ya sea el borrado del objeto, o únicamente del registro. En nuestro caso, hemos implementado la segunda opción invocando al método *"AlumnoLocalServiceUtil.deleteAlumno(AlumnoId);"*.

Una vez eliminado, se devolverá el objeto persistente eliminado en la pantalla correspondiente; *"actionResponse.setRenderParameter("mvcPath","/html/jsps/Eliminar_Alumno.jsp");"*.

☑ *Actualizar un Registro*

Para actualizar un registro usaremos el método *"**updateAlumno**"* en el que se define la siguiente secuencia de implementación. Primero obtendremos el valor del identificador a editar mediante la cláusula:

"AlumnoId = ParamUtil.getLong(actionRequest, "AlumnoId");".

Seguidamente obtendremos el objeto existente a través de esta clave primaria recuperada inicialmente, mediante el código, *"AlumnoLocalServiceUtil.getAlumno(AlumnoId)"*, para, a continuación ingresar la nueva información actualizada en el objeto persistente, en caso de que este exista, con el fin de poder actualizar el registro en la base de datos, usando el método: *"AlumnoLocalServiceUtil.updateAlumno(alumno);"*. Una vez que se actualicen los datos, se devolverá el objeto persistente resultante.

En este caso, se devolverán los datos en la JSP de edición:

"*actionResponse.setRenderParameter("mvcPath","/html/jsps/Editar_Alumno.jsp");"*.

☑ *Mostrar Registros*

Existen diferentes métodos para obtener los registros de nuestra tabla. Se pueden añadir métodos personalizados a través del fichero Service.xml pero no es nuestro caso. En nuestro caso, el código que obtiene los registros existentes se realiza en las JSPS de búsqueda

(*Buscar_Contenido_Alumno.jsp* y *Ver_Todos_Los_Alumnos.jsp*), no a través de un método de nuestra clase portlet como en los casos anteriores.

La porción de Código 40 generará el método de búsqueda y permitirá buscar el registro basado en la columna insertada en la barra del buscador.

```
<liferay-ui:search-container-results>
    <%

        DisplayTerms displayTerms =searchContainer.getDisplayTerms();
        displayTerms =searchContainer.getDisplayTerms();

                if (displayTerms.isAdvancedSearch()) {
                    total = AlumnoLocalServiceUtil.getBusquedaAlumnosCount(Titulacion,Nombre,Apellidos,Nota,Curso,
                                                           Asignatura,NIA,DNI,displayTerms.isAndOperator());

                    searchContainer.setTotal(total);
                    searchContainer.setResults(AlumnoLocalServiceUtil.getBusquedaAlumnos(Titulacion,Nombre,Apellidos,
                                                          Nota,Curso,Asignatura, NIA,DNI,displayTerms.isAndOperator(),
                                                          searchContainer.getStart(), searchContainer.getEnd(),
                                                          new AlumnoCompararNombre()));

                }else {
                    String searchkeywords = displayTerms.getKeywords();
                    String numbesearchkeywords = Validator.isNumber(searchkeywords) ? searchkeywords : String.valueOf(0);
                    total = AlumnoLocalServiceUtil.getBusquedaAlumnosCount(Titulacion,Nombre,Apellidos,Nota,Curso,Asignatura,
                                                          NIA,DNI,displayTerms.isAndOperator());
                    searchContainer.setResults(AlumnoLocalServiceUtil.getBusquedaAlumnos(searchkeywords,searchkeywords,searchkeywords,
                                                           Float.parseFloat(numbesearchkeywords),Integer.parseInt(numbesearchkeywords),
                                                           searchkeywords,searchkeywords,searchkeywords,false,searchContainer.getStart(),
                                                           searchContainer.getEnd(), new AlumnoCompararNombre()));
                }
    %>
    </liferay-ui:search-container-results>
```

Código 40 Implementación Buscador

En el código vemos la parte del "***if***" que se corresponde al caso en que no se inserte ningún valor en el campo de búsqueda, en cuyo caso se devolverán todos los registros y en el caso del "***else***" del bucle, se devolverán los registros que cumplan con la búsqueda.

6.4.6 Ficheros de Configuración

Liferay, realiza el manejo de sus servicios apoyándose en dos tecnologías: Hibernate Framework y Spring Framework. En nuestro caso hemos empleado la segunda opción.

El empleo de estas tecnologías implica a su vez la agregación de una serie de ficheros de configuración que encontraremos bajo el directorio "*src/META-INF*". Vamos a proceder a detallar los más críticos.

- ***Hibernate-spring.xml*** – Este documento define el modo en que *Spring* maneja *Hibernate* en la capa de datos.
- ***Infraestructure-spring.xml*** – Aquí se encuentra definida la fuente de datos para el producto.
- ***Portlet-spring.xml*** – Se definen los paquetes en los que se encuentran los paquetes bajo los que están implementadas las clases para el correcto funcionamiento de los servicios.
- ***Portlet-model-hints.xml*** – Este fichero posibilita la edición del comportamiento por defecto una vez se generen las columnas de nuestra tabla.

6.5 Plantilla de Diseño/Layout 

Por hacer una división entre los *portlets* descritos, disponibles en el perfilado de docentes, vamos a mostrar el desarrollo de una plantilla o *layout* y posteriormente explicaremos el *portlet* de consulta de notas de un alumno.

6.5.1 Funcionalidad

Como hemos mencionado a lo largo del documento, un componente de tipo *layout* no es más que un *portlet* empleado para gestionar la disposición de los *portlets* dentro de nuestra página. Reseñar que en un portal podemos tener tantas plantillas de diseño como nos sean necesarias.

Hemos desarrollado una disposición 1-4-1. Lo nombramos Columnas 1 4 1 porque existirá en la primera fila una única columna, en la segunda la segunda fila 4 columnas, y en la inferior, otra única también.

6.5.2 Configuración

En el proyecto encontraremos ficheros que definen la propia estructura de la plantilla así como ficheros de configuración.

- ❖ Ficheros de Diseño

Este tipo de componente, únicamente se define mediante tres ficheros:

- *[NombreApp].png* – Se trata de la *r*epresentación en miniatura de la plantilla que se visualiza en el portal para que el usuario tenga visibilidad del resultado de la disposición.

- *[NombreApp].tpl* – Contiene la estructura HTML de la plantilla.

- *[NombreApp].wap.tpl* – Y por último nos encontramos con una *p*lantilla adicional para dispositivos móviles ya que WAP son las siglas de *"protocolo de aplicaciones inalámbricas".*

Vamos a ver el contenido del segundo fichero para entender lo sencillo que resulta definir una estructura en liferay. Basta con agregar una línea por cada parcela que deseemos y desplegar nuestro componente.

```
<div class="columns-1-4-1"id="content-wrapper">
   <table class="lfr-grid"id="layout-grid">
      <tr id="column-center">
         <td class="lfr-column"id="column-1" colspan="4" valign="top">
         $processor.processColumn("column-1")
         </td>
      </tr>
      <tr id="column-center">
         <td class="lfr-column primera"id="column-2" colspan="2" valign="top">
         $processor.processColumn("column-2")
         </td>
         <td class="lfr-column primera"id="column-3" colspan="2" valign="top">
         $processor.processColumn("column-3")
         </td>
         <td class="lfr-column primera"id="column-4" colspan="2" valign="top">
         $processor.processColumn("column-4")
         </td>
         <td class="lfr-column primera" id="column-5" colspan="2" valign="top">
         $processor.processColumn("column-5")
         </td>
      </tr>
      <tr id="column-center">
         <td class="lfr-column segunda"id="column-6" valign="top">
         $processor.processColumn("column-6")
         </td>
      </tr>
   </table>
</div>
```

Código 41 Definición Fichero "columns_1_4_1.tpl"

Para correlacionarlo con la imagen, vemos que el código se puede dividir en tres bloques, pertenecientes a lo visto de modo gráfico. En la parte superior tenemos la definición de la primera celda, justo debajo definimos cuatro más y en la parte inferior del documento agregaremos otra celda.

Es importante tener en cuenta que los Valores contenidos en el campo Id de cada columna, debe coincidir con el pasado por parámetro en el método *processColumn*.

Este método, asociado a la variable *processor* crea un área donde poder arrastrar el portlet.

❖ Ficheros de Configuración

Además de los tres archivos específicos de la plantilla, el proyecto posee dos archivos de configuración de Liferay.

- *liferay-layout-templates.xml*: Especifica el nombre de la plantilla y la ruta (*path*) de los archivos que la componen.
- *liferay-plugin-package.properties*: Describe el *plugin* del proyecto(la versión a partir de la cual es compatible, el nombre del autor que lo ha desarrollado, el tipo de licencia, etc..).

A diferencia del resto de componentes, los layouts no contienen clases java.

6.6 Portlet Visor de Notas

Llegados a este punto, ya conocemos los portlets que están accesibles para un usuario con perfil docente, y vamos a conocer el que tendrán accesibles los alumnos para conocer sus calificaciones.

6.6.1 Funcionalidad

Esta funcionalidad está recogida en el portlet definido como "*VerNotas-portlet*" y en él, el alumno contará con un formulario en el que seleccionará los campos: NIA y DNI que serán cruciales para autentificar al alumno, así como el año, la convocatoria, y la asignatura que desee visualizar. Una vez seleccionados será redirigido a otra jsp en la que se mostrará una tabla con el resultado de la búsqueda.

Se trata de una interfaz sencilla que obtiene nuevamente los datos de nuestra base de datos *MySQL*.

Vamos pues a conocer la arquitectura de clases que se han desarrollado para tal fin.

6.6.2 Diagrama de Clases

La Figura 53 muestra el diagrama de clases que definen el portlet de visor de Notas de un Alumno. En ella se han agrupado las clases por empaquetado, nuevamente, según la disposición desarrollada.

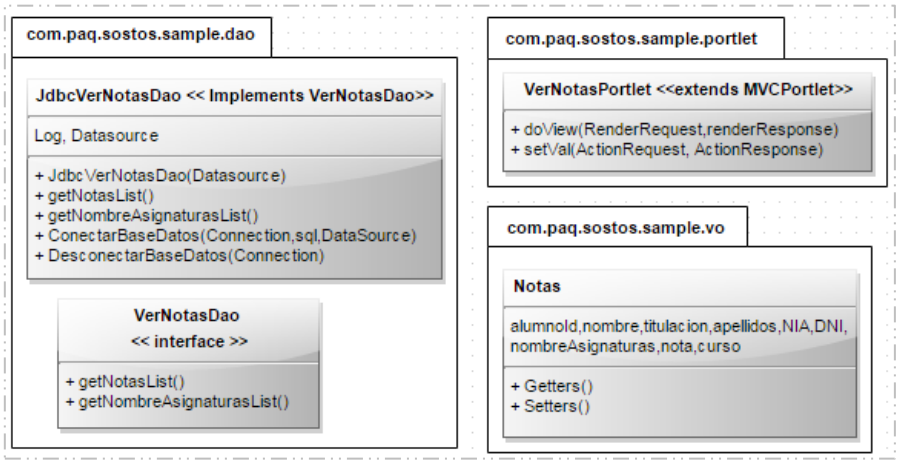


Figura 53 Diagrama de Clases - Portlet Visor de Notas

En este widget es interesante el modo en el que se obtienen los valores de la base de datos, ya que desde la página de visualización(JSP), solicitamos los datos asociados a una variable y desde nuestra clase java hacemos la operativa para almacenar en dicha variable los resultados de nuestra consulta.

Para esclarecer la interacción entre las clases, se presenta en el siguiente apartado un diagrama de secuencia con la actividad.

6.6.3 Diagrama de Secuencia

Por resumir el funcionamiento del widget, existe una JSP desde la que se introducen los parámetros de búsqueda. En esta visualización aparece un formulario compuesto con varios combos de datos. Algunos de ellos se rellenan desde la pantalla porque son contenidos estáticos y otros, como el nombre de la asignatura se obtienen de la base de datos.

Una vez rellenos, pulsando el botón de búsqueda, nos redirige a otra página en la que se realiza la verificación de los datos insertados en la primera visualización para mostrar una tabla con la nota requerida.

Vamos a presentar en la Figura 54 el diagrama de la secuencia seguida por el portlet para la obtención y representación de los datos al usuario, con el fin de esclarecer la interacción entre los objetos que definen la aplicación.

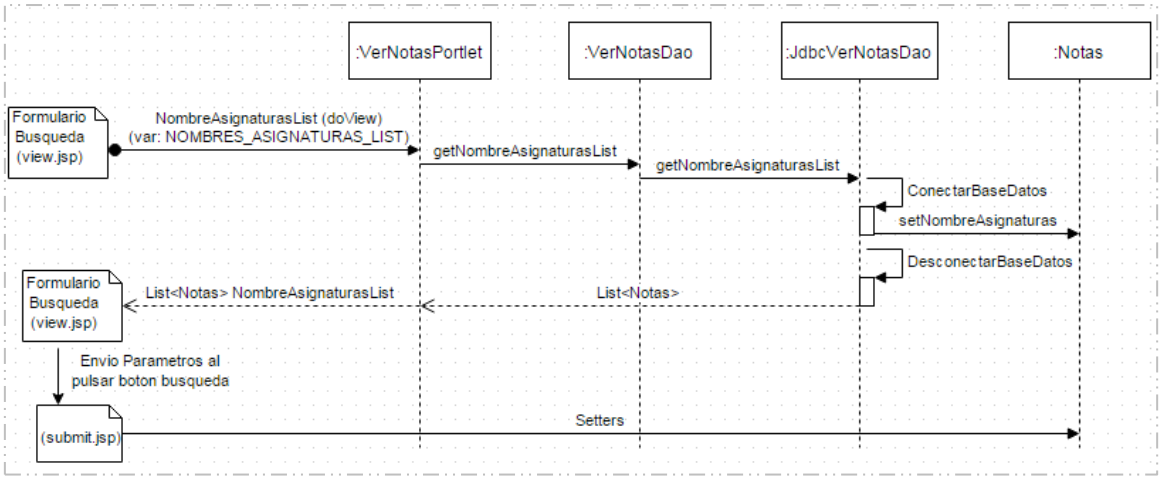


Figura 54 Diagrama de Secuencia - Portlet Visor de Notas

Capítulo 7

Pruebas y Evaluación

Una vez finalizado el desarrollo de nuestra aplicación, se dispuso a una comprobación completa de todas sus funciones, revisando su funcionamiento y respuesta ante distintas entradas de datos por parte del usuario. También se pensó en distintas situaciones conflictivas que podrían generarse, tales como formularios vacíos o ingesta de datos incoherentes, y se comprobó que estaban controladas.

Para probar que el entorno estaba correctamente definido, vamos a detallar el Plan de pruebas realizado, desglosado por acciones así como por rendimiento.

7.1 Pruebas de Integración

En la Tabla 4 se indican las pruebas realizadas, agrupadas según el causante de las acciones y el resultado obtenido.

Tabla 4 Resultados Pruebas de Integración

Descripción	Resultado
Acciones Comunes	
Autenticación	✓
Cierre Sesión	✓
Accesibilidad a sitio según perfilado de Usuario (Alumno/Profesor)	✓
Acciones Sobre Alumnos	
Buscar Nota	✓
Acciones Sobre Profesores	
Alta / Edición / Borrado Alumno	✓
Alta / Edición / Borrado Asignatura	✓
Creación / Búsqueda / Borrado / Listado Notas	✓
Administrador Portal	
Creación / Gestión Usuarios, Roles, Sitios y Grupos	✓
Configuración conexión LDAP	✓
Desactivación creación cuenta login usuario	✓

7.2 Pruebas de Carga

Con el fin de analizar la eficiencia de nuestro portal, se ha llevado a cabo la medición del tiempo de respuesta de las opciones que ofrece el sistema, dividiendo estas según el perfilado del usuario de acceso.

También hay que tener en cuenta que los tiempos para una misma acción no son constantes, puesto que por ejemplo la primera petición suele ser siempre más lenta, por ello, los datos indicados son el resultado de una media de 4 ejecuciones.

Para la medición mostrada se ha empleado un procesador: *Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz 2.50 GHz*

Los resultados obtenidos se muestran en la Tabla 6 y se dividen según las páginas seleccionadas. Hemos mostrado también el tiempo de carga de cada portlet.

Tabla 5 Resultados Pruebas de Carga

Página / Componentes	Ejec.1	Ejec.2	Ejec.3	Ejec.4	Tiempo	T. Medio(s)
Página Inicio Portal						
Bienvenido	0.23	0.03	0.06	0.03	0.35	0.09
Tablón De Anuncios	0.41	0.04	0.02	0.02	0.49	0.12
YouTube	0.40	0.22	0.05	0.05	0.72	0.18
Otros Recursos	0.07	0.06	0.20	0.21	0.54	0.14
Total Página Inicio	1.11	0.35	0.33	0.31	2.1	0.53
Página Alumno						
Ver Notas	0.04	0.02	0.02	0.01	0.09	0.02
Otros Recursos	0.68	0.39	0.37	0.51	1.95	0.49
Total Página Alumno	0.72	0.41	0.39	0.52	2.04	0.51
Página Profesor						
Alumno DAO	0.01	0.01	0.01	0.01	0.04	0.01
Alumno MVC	0.04	0.03	0.01	0.01	0.09	0.02
Asignaturas DAO	0.01	0.01	0.01	0.01	0.04	0.01
Otros Recursos	0.66	0.29	0.35	0.30	1.60	0.40
Total Página Profesor	0.72	0.34	0.38	0.33	1.77	0.44

7.3 Conclusiones

En líneas generales, puede decirse que los tiempos de respuesta no son excesivamente elevados.

No se puede perder la perspectiva de que los medidos son relativamente pequeños a lo que se obtendrían en un entorno productivo, puesto que tanto el servidor Apache Tomcat como la base de datos se encuentran albergados en una máquina local y esta coincide con la posición en la que se encuentra el cliente que interactúa con el sistema.

Pero también señalar que en cuanto a recursos, el tiempo es bastante real porque se han agregado widgets adicionales y temas de apariencia adicionales que también penalizan el rendimiento.

Para mejorar estos tiempo, por tratarse el rendimiento de un punto muy crítico, podrían seguirse recomendaciones de propiedades a agregar en el fichero "*portal-ext.properties*" para agilizar la carga de recursos CSS y Javascript, como agregar propiedades para deshabilitar filtros de funcionalidades que no sean estrictamente necesarias.

Capítulo 8

Conclusiones

En este capítulo se presenta un balance del proyecto implementado, analizando la tecnología empleada, detallando los conocimientos tanto aplicados como adquiridos, y presentando un análisis de los puntos que pueden ser interesantes para una ampliación futura.

El objetivo de este Proyecto de Fin de Carrera era la creación de una interfaz Web desarrollada sobre la plataforma Liferay, para proporcionar un acceso electrónico a la gestión y visualización de calificaciones de la universidad Carlos III por parte de alumnos y docentes dentro del proyecto de innovación docente descrito en los capítulos iniciales.

Tras el desarrollo del proyecto, se puede concluir que la aplicación cumple con los requisitos solicitados expuestos en el capítulo 1, y permite la adaptación Web de la estrategia requerida, siendo viable su acoplamiento dentro de un entorno docente real.

Para la implementación, es sabido que se han empleado las tecnologías JEE, Mysql y Apache Tomcat. Al ser todas de libre disposición, ha resultado sencilla su obtención y existe bastante soporte para ciertas incidencias que hemos ido encontrando en la configuración.

Se ha comprobado también el bajo tiempo de carga de los portales, permitiendo a usuario una respuesta ágil.

9.1 Valoración Tecnología Liferay

Con respecto a la utilización de Liferay como gestor de contenidos web, la experiencia ha sido muy grata, ya que ha resultado relativamente intuitivo de manejar, cuenta con una interfaz gráfica bastante atractiva y aporta, la versión utilizada (6.2), con respecto a la anterior, un apoyo para la visualización de portales en dispositivos móviles, ya sea tablets o teléfonos móviles, que tan presentes están en nuestra actividad diaria.

A lo largo del estudio, hemos averiguado, que esta tecnología nos proporciona una serie de herramientas potentes para poder gestionar sitios web. Para ello nos ofrece un CMS totalmente moldeable a cualquier sector de implantación.

He de indicar que me ha sorprendido gratamente su política de publicación de contenido y su sencilla integración con los visores de contenido empleando la opción de creación de estructuras y plantillas que facilitan la visualización de los contenidos.

En cuanto a la publicación, en entornos productivos, se ha podido verificar con un sistema denominado como "*Staging/Live*", que permite realizar una duplicación lógica de los contenidos de un entorno productivo para realizar modificaciones de modo paralelo sin afectar al usuario final, pudiendo posteriormente publicar dichas modificaciones para hacerlas visibles a todos los usuarios del portal.

Con respecto a la gestión de usuarios, también ha resultado sencillo el manejo, ya que es fácil configurar la conexión a en servidor *LDAP* y posteriormente, *Liferay* proporciona Roles y *WorkFlows* predefinidos, que marcaran la filosofía de trabajo o moldearlos a nuestro gusto.

Si nos centramos en el aspecto relativo al desarrollo del software, indicar que no ha sido especialmente engorroso por los conocimientos previos en el lenguaje Java, pero al ser una tecnología aun relativamente pionera no es excesiva la información de apoyo encontrada en la red.

En resumen, en líneas globales, Liferay es una buena solución para proyectos de cualquier tamaño e índole. Por lo tanto, el uso de este CMS para la gestión de notas en una red universitaria es acertado, aun sabiendo que como toda tecnología tiene sus contras.

9.2 Conocimientos Aplicados

En el marco de los conocimientos aplicados, cabe destacar la utilización previa del producto en el ambiente laboral, en su mayor parte como rol de administración, cosa que ha facilitado la creación de un portal propio.

Por otro lado, ha sido posible aplicar en el desarrollo del proyecto los conocimientos previos adquiridos en el puesto laboral que desempeño, lo cual me ha permitido, no solo utilizar una tecnología que ya conocía y me había resultado atractiva, si no profundizar y ampliar su estudio y análisis, pudiendo valorar su empleo bajo varios servidores de aplicaciones "*WebSphere*" y "*Apache Tomcat*" que de otro modo habría sido muy complejo conocer y verificar las implicaciones de cada uno de ellos.

Este es sin duda, el componente más importante que consideré para seleccionar este proyecto por el interés de ampliar conocimientos interesantes para mi carrera profesional.

9.3 Conocimientos Adquiridos

Como es de esperar, muchos han sido los conocimientos que he adquirido durante la realización del proyecto, pero, a continuación, detallaré las áreas más relevantes desde el punto de vista técnico:

- Instalación y administración de un portal Liferay.
- Alta y Gestión de Usuarios.
- Creación de sitios personalizados por grupos de usuarios.
- Conocimiento de las nuevas funcionalidades de la versión 6.2 de Liferay.
- Conocimiento de la integración con base de datos MySQL.
- Desarrollo de plugins de la tecnología.
- Gestión de contenidos web basados en *template* y plantillas.

A medida que he ido desarrollando la plataforma de comunicación, me han ido surgiendo nuevas necesidades, que han hecho que deba cubrir los diferentes perfiles profesionales que existen en una empresa de desarrollo de software. Es decir, he tenido que realizar las funciones propias de cada componente existente en un portal web (los clientes, los analistas funcional/técnico, el arquitecto y el desarrollador).

9.4 Líneas Futuras de Trabajo

En este proyecto se ha desarrollado para cubrir la funcionalidad deseada, y más estratégicamente, se ha querido emplear las diversas posibilidades de desarrollo de widgets que ofrece el producto. Se ha presentado una versión sencilla, pero las posibles líneas de trabajo podrían centrarse en mejorar la aplicación, ampliando las funcionalidades presentadas y haciendo uso de nuevas, según los entornos planteados. A continuación se identifican algunas de las posibles mejoras:

9.4.1 Funcionalidades adicionales

A la hora de centrarnos en la usabilidad del portal, podría ser interesante agregar nuevas funcionalidades: ampliando el modelo de datos, con el fin de llevar un registro por ejemplo de los exámenes realizados, las notas, así como las notas de prácticas, ordenadas por cuatrimestre y asignatura. También puede resultar interesante poder adaptar el sistema para entornos diferentes al docente, puesto que en otros sectores, será interesante la gestión de los datos.

Se propone también la posibilidad de ampliar el desarrollo de la aplicación para que los alumnos tengan la posibilidad de descargar sus calificaciones o un proceso para solicitar las tutorías y revisión de exámenes a los profesores en modo online.

9.4.2 Interfaz gráfica

Otra línea de desarrollo bastante necesaria podría consistir en mejorar la interfaz gráfica desarrollando un tema personalizado. Actualmente se han empleado estilos simples para los formularios, botones y demás componentes gráficos de la aplicación, pero el producto dispone de plantillas de aplicación específicas que extienden las opciones de visualización para poder potenciar la experiencia visual a los usuarios. Son bastante interesantes porque son fácilmente importables y no requieren ser desplegadas para su utilización por lo que los cambios serán instantáneos y no ocasionarán problemas en entornos productivos.

9.4.3 Documentos/Archivos Multimedia

La herramienta empleada en nuestro desarrollo, incluye un repositorio que nos permite almacenar documentos, archivos de audio/vídeo, imágenes y demás archivos multimedia en un mismo lugar. Permite la posibilidad de emplearlo como un repositorio web de documentos por todos los usuarios que accedan a nuestro portal, por un grupo específico o por un individuo concreto.

En el entorno en que estamos trabajando, por tratarse de un ámbito docente, resulta interesante poder utilizar repositorios de documentos, puesto que un profesor puede querer subir apuntes, enunciados de prácticas, listados de notas, listado de grupos de prácticas y no únicamente documentos, sino, videos de clases grabadas para su visualización online posteriormente, etc..

O incluso, puede ser interesante de cara a los alumnos disponer de repositorios en los que poder almacenar documentos de interés relacionados con las materias que está cursando.

Resulta bastante útil esta funcionalidad puesto que los usuarios disponen asimismo de sus propios repositorios de contenidos en los que se les ofrecerá la posibilidad de almacenar archivos en la nube.

9.4.4 Alojamiento Web en la Nube

A propósito de la gestión de datos en la nube y debido al auge de internet en los últimos tiempos, resulta importante hablar del conocido término "*Cloud Computing*" puesto que se trata de una tendencia tecnológica que se está convirtiendo en el futuro del alojamiento web.

Todo apunta a que en un corto espacio de tiempo, la mayoría de las aplicaciones estarán hospedadas en este sistema.

Vamos a mencionar las principales ventajas que posee el empleo de este sistema.

- ✓ Una de las cualidades más revolucionarias de su uso se basa principalmente en su accesibilidad. A diferencia del alojamiento web convencional en el que se contaba con servidores compartidos, en los que los datos se encontraban almacenados en un servidor dedicado, el nuevo modelo de alojamiento web en la nube se asienta sobre una red de servidores, de modo que presentan una alta inmunidad en el caso de existencia de fallos en alguno de ellos, el sitio no sufrirá consecuencias puesto que se balanceará el servicio con el resto de nodos otros servidores que formen la "nube", por lo que el usuario no experimenta cortes en el acceso a las páginas web. Esto hace que el tiempo de acceso de una web para los usuarios sea virtualmente del 100%.

- ✓ Otra ventaja de este tipo de servidores radica en su escalabilidad. Si en algún momento una web requiere una mayor cantidad de recursos/servicios por un acceso elevado de usuarios, el sistema no sufre limitaciones por su hardware, ya que puede emplear los recursos de la nube de servidores.

- ✓ En el alojamiento en la nube, existen diversas opciones de gestión de los recursos disponibles, y únicamente se abonan los que se utilizan, lo que hace que el coste sea relativamente moderado, resultando económico para los usuarios.

- ✓ Es adecuado para proyectos de cualquier volumetría, lo que lo hace apto para cualquier proyecto.

Puede ser interesante considerarlo para un futuro pero debe tenerse cuidado de no sufrir brechas de seguridad en el sistema.

9.4.5 Rendimiento

El entorno que hemos presentado, consiste en único nodo, pero lo normal es que en entornos productivos contemos con más de uno. Sería necesario replicar el software en todos ellos y proceder a su gestión para tener mayor seguridad frente a fallos en alguno de ellos.

En el diseño de un portal de alto rendimiento necesitamos gestionar una caché que nos permita minimizar el acceso a los recursos que consumimos y que optimice los accesos a nuestro sistema.

La cache es, sin duda, un aspecto muy importante para cualquier sistema que quiera alcanzar unas prestaciones adecuadas. En este ámbito, Liferay ofrece una integración sencilla con diferentes frameworks para tal efecto. En el caso del ámbito empresarial en el que se ha trabajado con la tecnología, se ha empleado "*ehcache*".

Se trata de un framework de cache distribuido bastante potente, ya que consiste en un proyecto *OpenSource* desarrollado en lenguaje Java. La configuración usa una cache en instancias locales. Esto significa que si se está usando un entorno clusterizado, cada nodo poseerá su propia caché.

9.4.6 Acceso a los datos

A la hora de realizar la gestión de datos para alimentar nuestros portlets, hemos considerado que se consumen desde una base de datos albergada en el mismo servidor que el producto, pero no siempre tiene porque ocurrir de este modo.

Como era de esperar, un producto tan maduro como Liferay proporciona los métodos necesarios para interactuar con servicios externos (por ejemplo: *Fatwire*) gracias a la utilización de ***web services***. De esta forma podríamos, por ejemplo, crear un cliente que agregara contenido web o gestionara los usuarios desde otra aplicación totalmente independiente a Liferay. Esto permite construir portales reutilizables que empleen en cada caso los servicios necesarios.

9.4.7 Colaboración y Redes Sociales

Una de las necesidades de la plataforma es mejorar el trabajo colaborativo entre los usuarios. Nuevamente, Liferay proporciona mecanismos sencillos para su realización. Algunas a tener en cuenta podrían ser:

- ☑ ***Wiki*** - Se ofrece la posibilidad de recopilar y documentar la información colectiva relevante.

  Cada comunidad de Liferay cuenta con su propia Wiki y un conjunto de personas autorizadas a realizar acciones sobre el mismo. Puede ser interesante, definir una wiki gestionada por el personal docente en el que se recojan los elementos más relevantes para el alumnado.

- ☑ ***Foros*** – De cara a potenciar la comunicación entre usuarios (entre alumno-alumno o alumno-profesor), puede resultar atractiva la incorporación en los portales de Foros.

  Liferay proporciona vistas de la actividad así como entradas recientes y ofrece a los usuarios la posibilidad de responder a los hilos a través del correo electrónico. Proporciona también, para estos componentes, una seguridad granulada del sistema de permisos y autorizaciones, evitando accesos indeseados y facilitando el acceso a diferentes niveles y con diversas acciones para cada participante.

- ☑ ***Mensajería Instantánea*** - Con el fin de potenciar la relación entre compañeros, puede ser cómodo agregar el servicio de mensajería instantánea en nuestra web. En este se mostrara una lista de usuarios conectados en ese momento.

- ☑ ***Calendario compartido*** **–** Puede resultar también de gran utilidad el empleo de un calendario común a los usuarios pertenecientes a una misma asignatura. En él, el docente podrá crear, administrar y buscar eventos, que pueden ser compartidos con otros sitios web para que los alumnos estén actualizados en todo momento. Es una funcionalidad ágil para indicar fechas de exámenes, calendario de prácticas, etc. Es posible también configurar recordatorios de eventos que envíen alertas a través de correo electrónico, mensajería instantánea o SMS lo cual hará que los alumnos estén en todo momento al corriente de las fechas relevantes.

Cada día, están más en auge las herramientas colaborativas de ahí su gran relevancia y el punto a considerar en mayor medida.

Capítulo 10

Anexos

Este capítulo contiene las instrucciones necesarias para llevar a cabo una instalación en local del entorno de desarrollo de la herramienta Liferay y el manual de usuario para la utilización del portal desarrollado.

Anexo I Instalación Entorno de desarrollo Local Liferay 6.2 (Tomcat)

Procederemos a detallar los pasos a seguir para instalar un portal Liferay en un servidor Apache Tomcat. Indicar que esta guía está orientada para el sistema operativo Windows.

Para la instalación de este entorno, en general se recomienda disponer de un equipo con sistema operativo Windows, Mac o Linux con al menos 4GB de memoria RAM y un procesador de 2.0GHz.

Una vez dispongamos de él, deberemos realizar los siguientes pasos:

1.- Instalación previa de herramientas

El primer paso para disponer de un entorno de desarrollo local de Liferay consiste en instalar una serie de herramientas necesarias para trabajar con el portal:

- Java JDK
- MySQL

A continuación se describe cómo realizar la instalación de cada una de estas herramientas indicadas.

❖ Java JDK

Se debe utilizar la versión Java JDK 6 y se recomienda desinstalar otras versiones que puedan estar en el equipo para evitar conflictos.

El Java JDK se puede obtener desde la dirección:
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Una vez finalizada la instalación se deben configurar las variables de sistema necesarias:

1. Desde el Panel de Control ir a Configuración Avanzada del sistema, y en opciones avanzadas pulsar sobre "Variables de Entorno". Opción mostrada en la Figura 55.

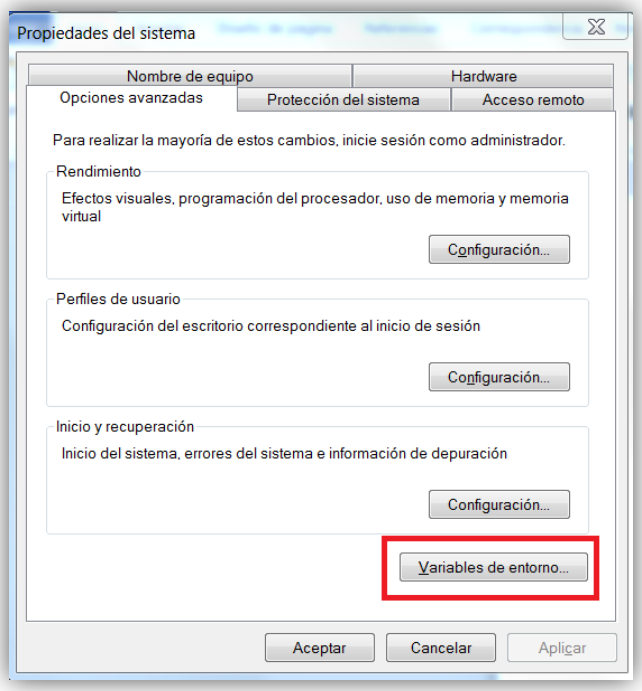


Figura 55 Variables de Entorno del Sistema

2. Dentro de las Variables de Entorno, crear una nueva Variable del Sistema (o modificarla si ya existe), como en el ejemplo indicado en la Figura 56.
    a. Nombre de variable: *JAVA_HOME*
    b. Valor: ruta a la instalación de la JDK (p.ej. c:\java\jdk6)
    c. Pulsar Aceptar.
    d. Comprobar que la nueva variable aparece en la lista de Variables del Sistema.

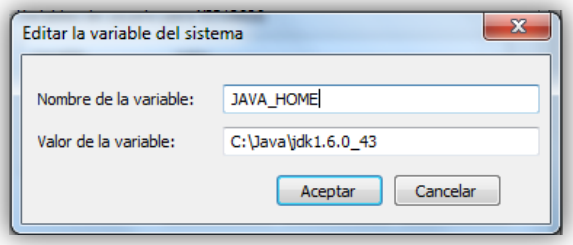


Figura 56 Configuración variable entorno - JAVA_HOME

3. Dentro de las Variables del Sistema buscar la variable Path, y pulsar en Editar como se ve en la Figura 57.
    a. Modificar su valor añadiendo al final del mismo la cadena ;*JAVA_HOME*.
    b. Pulsar Aceptar.

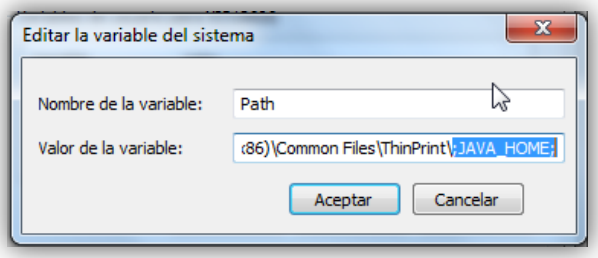


Figura 57 Configuración variable entorno – Path

4. Para comprobar la correcta configuración
    a. Abrir una consola ejecutando el comando *cmd.*
    b. Ejecutar: *java –version*
    c. Debe obtenerse una respuesta similar a la siguiente.

Un ejemplo gráfico es el definido en la Figura 58.

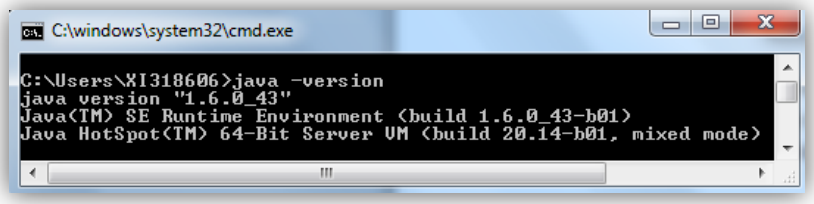


Figura 58 Validación Instalación Java JDK

❖ MySQL

A continuación se muestran los pasos a seguir para realizar la instalación de la base de datos MySQL en su versión 5.5

1. Descargar el software desde la *URL*: http://dev.mysql.com/downloads/mysql/5.5.html#downloads
2. Ejecutar el instalador.
3. Seleccionar el tipo de instalación personalizada (*Custom*).
4. Cambiamos la ruta de instalación a c:/mysql
5. Pulsamos siguiente hasta que nos aparece "*finish*", seleccionamos el *check* para que se nos abra la configuración del Msql Server al terminar.
6. Dentro de la configuración, una vez instalado el Mysql, seleccionamos "*Standard Configuration*" y pulsamos siguiente.
7. En la ventana que aparece, dejamos la configuración que aparece por defecto y pulsamos siguiente.
8. Ahora introducimos la contraseña (por ejemplo: *root*), pulsamos siguiente.
9. Solo nos queda pulsar "*execute*" para terminar.

La pantalla de configuración descrita de muestra en la Figura 59.

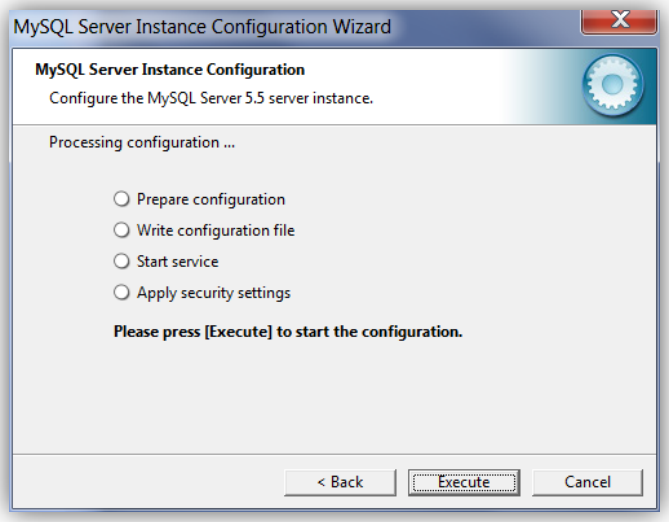


Figura 59 Instalación MySQL

10. Una vez configurado, añadimos en la variable de entorno Path la ruta donde se encuentra el *bin* del *mysql*: p.ej:C:\msql\bin como se ve en la Figura 60.

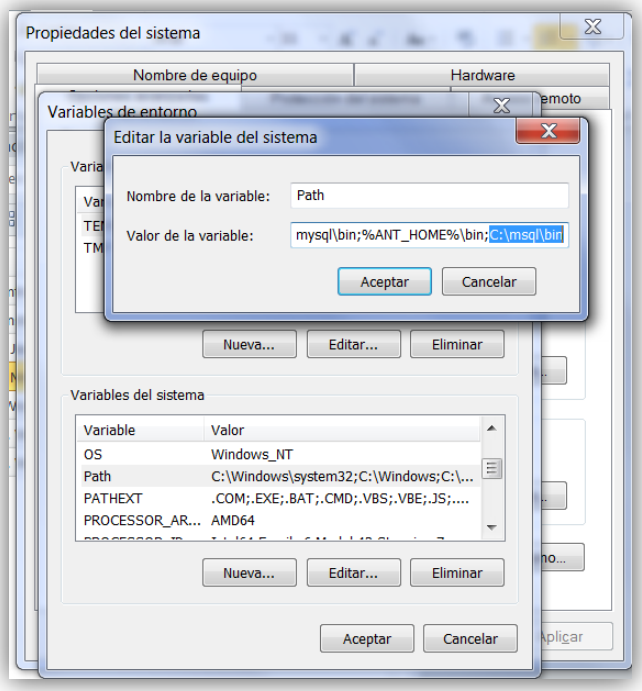


Figura 60 Configuración ruta MySQL - Variable entorno Path

11. Para acabar con la parte de Mysql solo nos queda crearnos la base de datos, para ellos solo tenemos que entrar en Mysql, desde la línea de comandos.
12. Una vez dentro de Mysql introducimos el siguiente comando: "*create database* ***lportal*** *character set utf8;*"

    (***lportal*** es el nombre por defecto del portal que crea *LFR Developer Studio* si queremos otro debemos crear la BD con el nombre que queramos).

13. Con esto ya tenemos preparado el *Mysql Server*.

2.- *Instalación paquete (bundle) Liferay con Tomcat*

❖ *Preparación de directorios*

Prepararemos una estructura de carpetas donde se ubicara nuestro proyecto Liferay Portal.

- Nos ubicamos en un disco del sistema y creamos una carpeta donde pondremos el ambiente de desarrollo de nuestro Portal, (*C:\Liferay)* y la cual llamaremos <Liferay-Home>.
- Obtendremos el paquete de instalación desde el sitio web oficial en la siguiente URL: http://www.liferay.com/downloads/
- Este archivo es del tipo *bundle. E*sto nos permite instalar el producto en los Sistemas Operativos más comunes, Windows, Linux etc... .Este tipo de empaquetado permite que Liferay Portal sea portable, puesto que una vez descomprimido tendremos dentro de esa carpeta todo lo necesario para que el portal funcione.
- Descomprimimos el paquete [*liferay-portal-tomcat-6.2-ce-ga2-20140319114139101.zip*] que contiene el proyecto Liferay por defecto sobre el cual trabajaremos y que ya viene con el servidor Tomcat incorporado.
- Tendremos una carpeta llamada *liferay-portal-6.2-ce-ga2* en cuyo interior se encuentra todo la configuración predeterminada de carpetas. El resultado de la descompresión se refleja en la Figura 61.

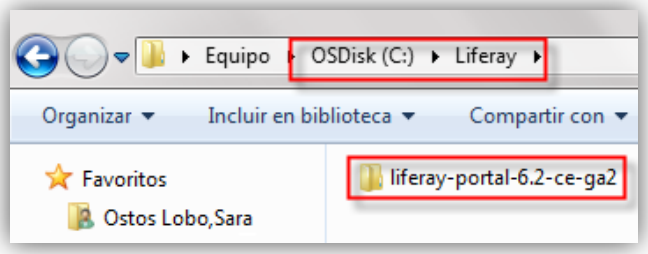


Figura 61 Carpeta descomprimida donde se ubica el portal

- Ingresando en el directorio observaremos los siguientes directorios:

  /data: En esta Carpeta se almacenaran las imágenes y documentos que subamos al portal.
  /deploy*:* Directorio donde se copiaran los *wars* de las aplicaciones que hagamos para ser ejecutado en tiempo de ejecución (*portlets*, temas, *layouts*, etc.).
  /websphere-deploy: Los *.war* de la carpeta deploy llegan a esta carpeta mediante un proceso que ejecuta Liferay
  /license: la licencia. En nuestro caso en esta carpeta no tendremos contenido puesto que estamos usando una versión sin coste.
  /logs: Directorio de *logs* de aplicación.
  /tomcat-7.0.42: Es el servidor *Apache Tomcat* pre-configurado para *Liferay* que usaremos para ejecutar la aplicación.

❖ *Preparación del Servidor*

Liferay puede ser desplegado en múltiples servidores de aplicación, se puede tener un Liferay sobre un servidor de aplicaciones previamente configurado (como el ejemplo) o configurar un servidor existente.

Para este caso usaremos el servidor Apache Tomcat en su versión 7.

Liferay viene por defecto con una aplicación de ejemplo instalada, es decir si levantamos el servidor sin hacer ningún cambio funcionara normalmente, incluso Liferay cuenta con una base de datos embebida en memoria llamada *hsql* que solo sirve para casos demostrativos no para ambientes de desarrollo y menos de producción, procederemos a levantar el portal tal y como está sin ninguna configuración externa.

Para arrancar el servidor Apache Tomcat nos ubicamos en la ruta donde está instalado [*tomcat-7.0.42*] y accedemos a la carpeta '*bin'*. En ella encontraremos un archivo llamado '*startup.bat'* el cual ejecutamos con lo cual el aplicativo comenzara a levantar.

Seguidamente, insertando en el navegador la dirección *http://localhost:8085* se cargará el aplicativo de ejemplo para probar las funcionalidades del portal. Validaremos que los datos de configuración son los correctos, seleccionamos el idioma si lo deseamos y el usuario a utilizar.

❖ *Configuración de propiedades del portal*

Liferay posee un archivo *properties* donde se definen muchos de sus atributos principales una de ellas el acceso a la base de datos, para ello debemos crear dicho archivo que se debe llamar '*portal-ext.properties*' y ubicarlo en la carpeta [*C:\Liferay\liferay-portal-6.2-ce-ga2*].

Este fichero se puede considerar como el fichero de configuración más importante del producto, por lo tanto, habrá que tener cuidado antes de modificar un valor. En él se sobrescribe los valores por defecto de aquellas propiedades que utiliza Liferay y que se desee modificar. Este fichero se define teniendo en cuenta el fichero con todas las propiedades que utiliza Liferay por defecto, se encuentra físicamente dentro del *.JAR*: *portal-imp.jar*, y se denomina '*portal.properties'*.

Una vez creado el fichero, hemos configurado la conexión a una base de datos local, donde el portal al instalarse por primera vez generará sus tablas y dependencias. Liferay soporta diferentes base de datos (*MySQL, Oracle, SQL Server, etc*.) para conectarse a cada una de ella se debe tener el driver y configuración particular.
Para ello debemos tener creado una base de datos exclusiva para el portal en el caso descrito, se ha seleccionado una *BBDD MySQL* y se ha denominado '*mysql*.

Editaremos el archivo "*portal-ext.properties*" y agregaremos la configuración necesaria para conectarse a la base de datos, en este caso la porción de Código 42.

```
# MySQL database configuration
jdbc.default.driverClassName=com.mysql.jdbc.Driver
jdbc.default.url=jdbc:mysql://localhost:3306/notasUC3M?useUnicode=true&characterEncoding=UTF-8&useFastDateParsing=false
jdbc.default.username=root
jdbc.default.password=root
```

Código 42 Propiedades de conexión base de datos local MySQL

Además de ello debemos copiar el driver correspondiente a la base de datos seleccionada en la carpeta de *jars* externos [\tomcat-7.0.42\lib\ext] como se detalla en la Figura 62.

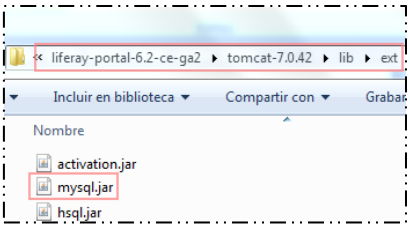


Figura 62 Ruta donde se alberga .jar de MySQL

Es importante recalcar que cualquier modificación en este fichero, requiere de un reinicio del servidor de aplicaciones para que se haga efectivo el cambio.

Realizando estas configuraciones, dispondríamos de un ambiente Liferay dispuesto para poder comenzar a desplegar los *plugins* desarrollados.

Anexo II Instalación Entorno de Desarrollo Eclipse

1.- Eclipse, Entorno de desarrollo integrado (IDE)

Para instalar Eclipse necesitamos descargarlo de la siguiente URL en la que veremos lo representado en la Figura 63.

http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/marsr

Figura 63 Descarga Entorno de Desarrollo Eclipse

Existirán varias opciones de descarga, dependiendo del sistema operativo en el que vayamos a instalarlo.

En este caso, el tipo de archivo que se descargará también es del tipo bundle por lo que, únicamente tendremos que descomprimirlo en la carpeta de instalación y ya lo tendremos disponible para comenzar a trabajar.

2.- *Plugins Adicionales para Eclipse*

Para facilitar el desarrollo de componentes para Liferay, es necesario adaptar Eclipse, para ello basta con instalar el plugin necesario.

Para su instalación emplearemos el Marketplace de Eclipse.” Help->Eclipse Marketplace…”. La ruta se muestra en la Figura 64.

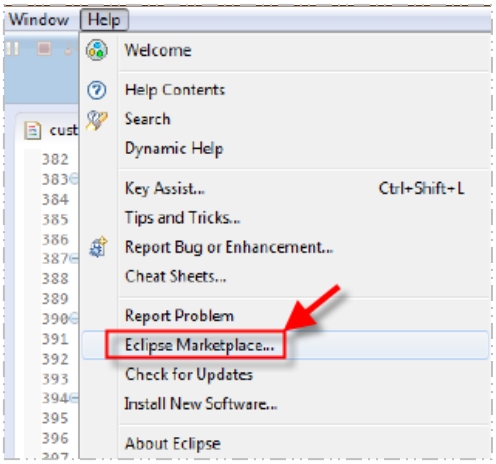


Figura 64 Eclipse Marketplace

Una vez accedamos, buscamos el plugin Liferay (Liferay IDE) y obtendremos los resultados existentes. Desde aquí tendremos la opción de instalarlo como se ve en la Figura 65.

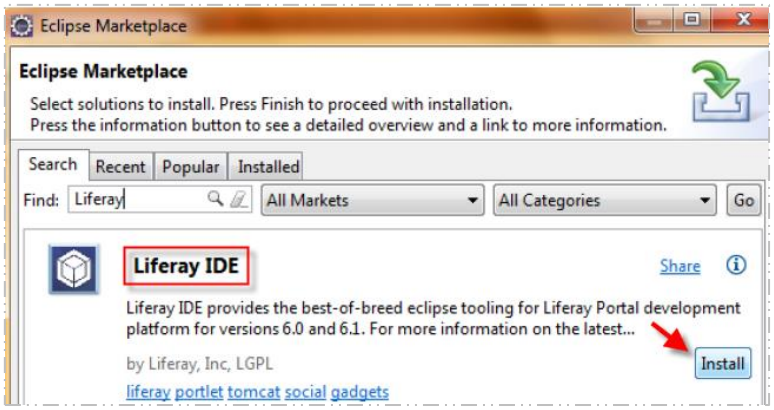


Figura 65 Plugin Liferay IDE – Eclipse

Debe tenerse en cuenta que para poder empezar a utilizarlo habrá que reiniciar Eclipse.

Anexo III Configuración LDAP

Como ya es conocido a estas alturas del documento, Liferay proporciona diversas alternativas de autenticación, entre ellas, LDAP.

Esta configuración se puede gestionar desde el ***Panel de Control*** > ***Configuración*** que se encuentra a nivel de Portal. En esta sección, seleccionamos la opción "***Autenticación***" como se muestra en la Figura 66.

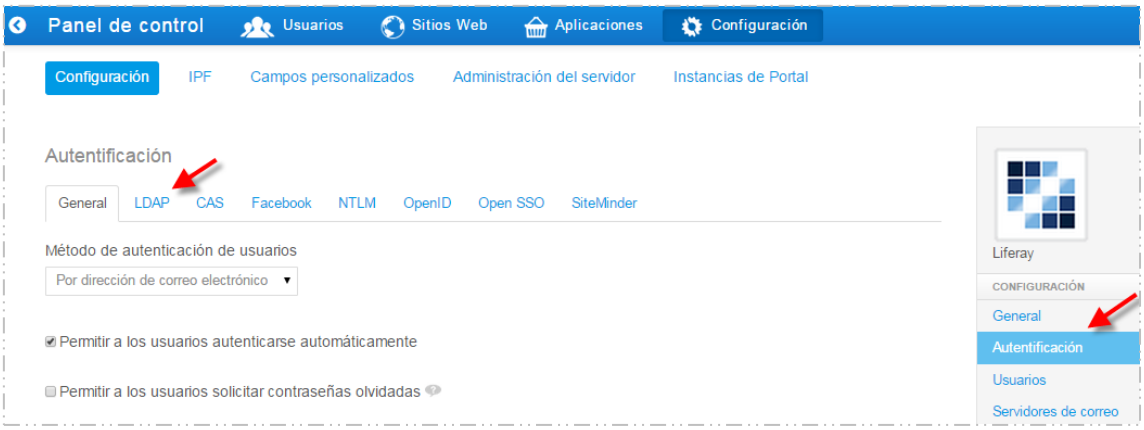


Figura 66 Opciones Autenticación Liferay

Seleccionando la pestaña "LDAP", mostrada en la Figura 67, accedemos a la página desde la que se podrá configurar de modo ágil y sencillo nuestra conexión contra un servidor de estas características.

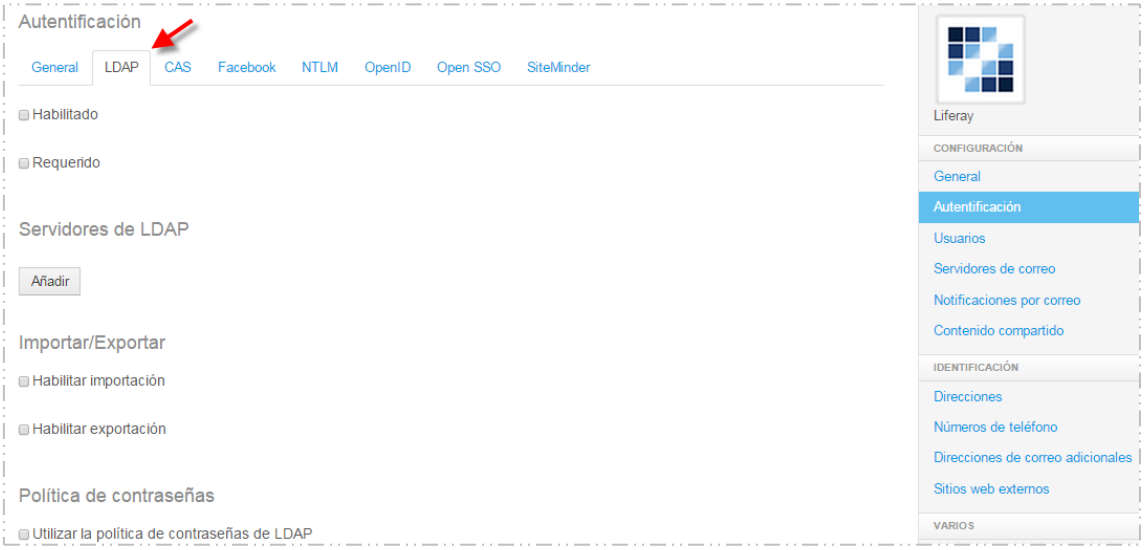


Figura 67 Pestaña configuración LDAP

Desde aquí habrá diversas opciones que se podrán elegir con respecto a los usuarios que se obtengan del servidor configurado, pudiendo habilitar la importación y exportación de estos así como emplear la política de logado únicamente con los usuarios de LDAP.
Existen dos modos de configuración de esta conexión, una es a través del fichero de propiedades del portal y otra es como se está detallando en este anexo. La ventaja fundamental de realizarlo a través del panel de control es que no requiere reinicio del servidor ante cualquier cambio que se realice.

Pues bien, para administrar la conexión, lo primero que debemos hacer es dar de alta el servidor LDAP con el que deseemos trabajar. Para ello, basta con pulsar el botón "Añadir" que aparece en la sección "Servidores de LDAP".

Una vez dentro, se deben adecuar los parámetros como corresponda para el servidor LDAP que se vaya a emplear, en nuestro caso se tratará como venimos comentando *LDAP*. Un ejemplo de esta pantalla se observa en la Figura 68.

Figura 68 Parámetros Conexión Servidor LDAP

Se debe indicar cadena de conexión a nuestro sistema a través de la máquina y puerto de LDAP, y aportar un usuario con los permisos necesarios para editar los directorios donde se almacenan los usuarios y grupos de usuario de Liferay y a continuación, se debe escoger la relación de los campos de usuario para realizar la importación de los usuarios, proceso que se realizará de igual modo para el caso de los grupos de usuarios.

El campo provider URL representa la *URL* del servidor *LDAP*, y como vemos se indica el tipo de protocolo (*LDAP*), la dirección IP o el nombre de dominio, en este caso, "*ldap.uc3m.es*" y el puerto de red del servidor *LDAP*, correspondiente al 389. El campo *BDN* indica el directorio que se consultará para la obtención de usuarios dentro de la estructura de directorios *LDAP*. En este caso, gente de la Universidad Carlos III situada en España.

En este escenario los usuarios delegan el proceso de autenticación y gestión al CMS correspondiente, pero la única diferencia de la del uso corriente es que Liferay no necesita consultar sus tablas de datos para determinar los datos de acceso, sino que se apoya en *OpenLDAP* para este fin.

Una vez configurados los parámetros de conexión, se podrán agregar las configuraciones para la búsqueda de *usuarios* y *grupos de usuarios*.

Una vez guardemos la configuración, obtendremos una ventana emergente con los usuarios que cumplan los requisitos indicados.

Anexo IV Creación Carrusel de imágenes AUI

Liferay, en su versión 6.2, cuenta con un sencillo sistema de gestión de contenidos sobre el que podemos apoyarnos para diseñar y publicar contenido en nuestra página web.

Para el diseño, existe la posibilidad de definid dos componentes definidos como estructuras y plantillas de entidades que sirven para crear diseños comunes y poder utilizarlo en las diferentes páginas que conforman el portal, únicamente modificando los datos de origen.

En este documento vamos a mostrar un ejemplo de creación de un carrusel de imágenes que hemos incorporado en la pantalla inicial de nuestro gestor de notas. Nos apoyaremos en el Carrusel AUI para diseñar las imágenes y usaremos una plantilla y una estructura para implementar el propio carrusel.

- ***AUI Carrusel***

Al emplear este componente, obtenemos las siguientes prestaciones:

- En caso de querer mostrar varias imágenes necesitamos una diapositiva con anchura y altura configurable.
- Se debe especificar el intervalo de tiempo con el que se rotarán las imágenes, por lo que debe ser de igual modo configurable.
- Cada imagen de diapositivas posee un hipervínculo. Al seleccionarla nos mandará a la página correspondiente, nuevamente configurable.

Pues bien, para poner en marcha nuestro carrusel de imágenes, vamos a tener que realizar los siguientes pasos:

1.- Diseñar una Estructura

Para proceder a la creación de este tipo de contenido, accederemos a través del panel de control a la ruta "*Admin > Site Administration*".

Una vez allí, se seleccionará desde el menú lateral el elemento "*Web Content*", que se encontrará bajo el panel "*Content*" como se ve en la Figura 69.

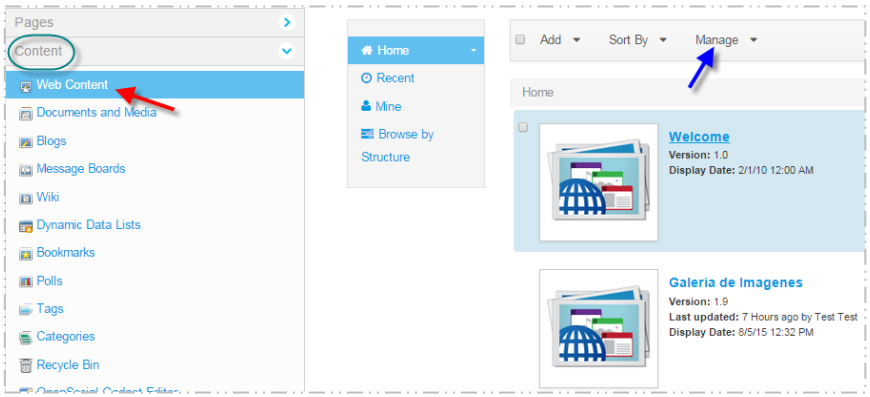


Figura 69 Agregar Contenido Web

Posteriormente, seleccionaremos la opción "*Manage* > *Structures*", a continuación accedemos a la opción "+ Add" y rellenaremos los campos de la estructura a generar. La pantalla a visualizar se muestra en la Figura 70.

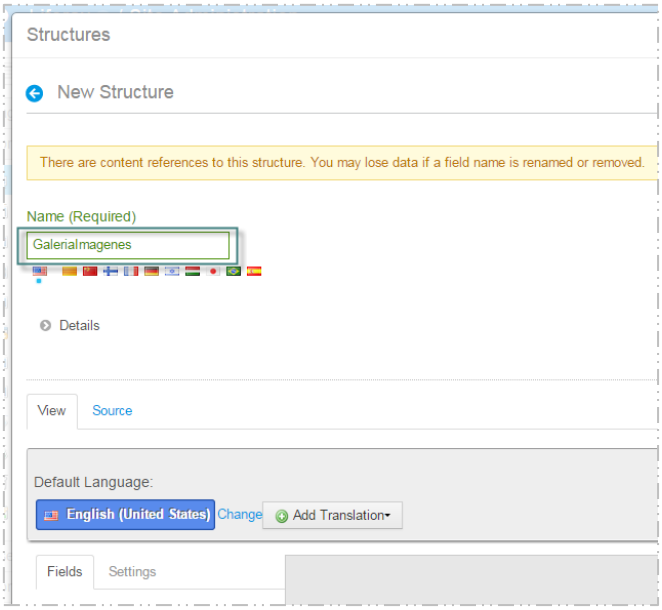


Figura 70 Creación de una Estructura

En la parte inferior de la ventana pop-up se puede ver la interfaz de usuario para la realización del diseño de la estructura mostrada en la Figura 71.

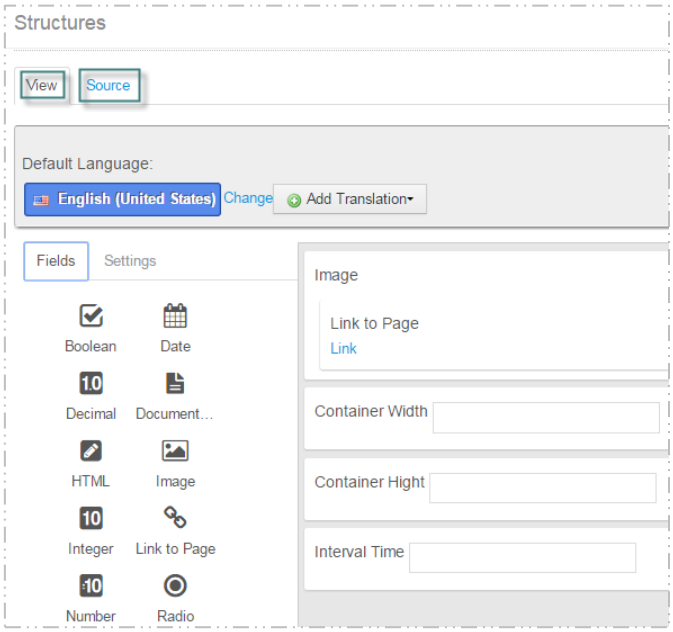


Figura 71 Interfaz de Usuario - Estructuras

Existen muchos elementos que podemos escoger en el lateral izquierdo para definir nuestro diseño pero también podemos generar la estructura a través del código fuente seleccionando la pestaña *view* o *source* respectivamente.

Una vez guardados los cambios contaríamos con nuestra plantilla.

2.- Crear una plantilla y asociarle una Estructura

Ahora necesitamos crear la plantilla y le asociaremos la estructura anteriormente creada. Volvemos nuevamente a la opción "M*anage"* y en este caso seleccionamos la opción "*Templates*" como se ve en la Figura 72.

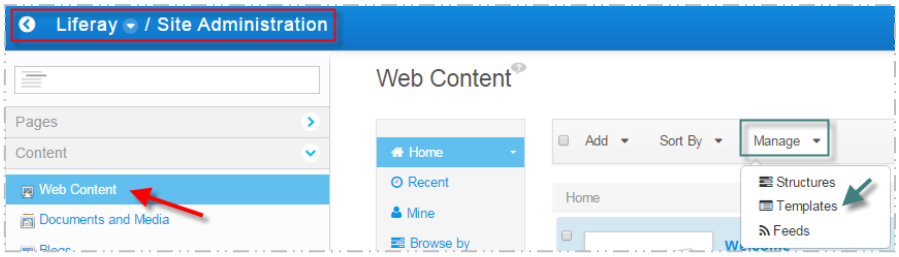


Figura 72 Selección componente Estructura

Una vez que seleccionamos la opción, en la ventana emergente pulsaremos el botón *"+ Add*" para crear una nueva plantilla, le otorgaremos un nombre identificativo y seleccionaremos la estructura creada en el punto previo.

De nuevo, en la parte inferior de la ventana emergente se puede ver el editor de plantillas y sus respectivos campos en la parte izquierda. En este caso, agregaremos el script correspondiente al componente AUI de tipo carrusel.

```
<script>
AUI({ filter: 'raw' }).use('aui-carousel', function(A) {
new A.Carousel({
intervalTime:${intervaltime.getData()},
contentBox: '#myCarousel',
activeIndex:0,
height:${containerhight.getData()},
width:${containerwidth.getData()}
}).render();

});
</script>
<#if images.getSiblings()?has_content>
<div id="myCarousel">
<#list images.getSiblings() as cur_images>

<#if cur_images_index==0>
<a href="${cur_images.imagelink.getData()}">
<div class="carousel-item" style="background:
url(${cur_images.getData()})width:${containerwidth.getData()}px;
height:${containerhight.getData()}px;" class="carousel-item carousel-item-active";">
</div>
</a>
</#if>
<a href="${cur_images.imagelink.getData()}"> <div class="carousel-item" style="background:
url(${cur_images.getData()});width:${containerwidth.getData()}px;
height:${containerhight.getData()}px;" class="carousel-item""></div></a>
</#list>

</div>
</#if>
```

Código 43 Ejemplo Script AUI Carrusel

Una vez guardados los cambios tendremos la plantilla disponible para su uso.
Para más detalle de los campos accesibles véase: [34]

3.- Creación de contenido web

En este paso vamos a crear un contenido web empleando la plantilla y estructura creadas en los dos pasos previos. En la pantalla de contenido web se puede ver la opción "+Add" y veremos el nombre de la plantilla que hemos definido. La opción la vemos en la Figura 73.

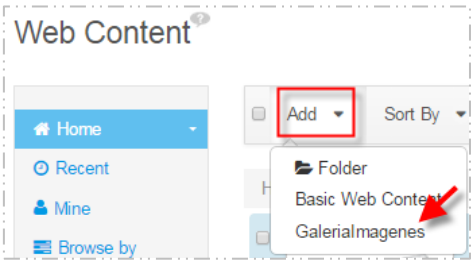


Figura 73 Creación de Contenido Web

Ahora se accederá a la pantalla de creación de contenido web con la plantilla seleccionada.

Sobre la base de la plantilla utilizada para llenar el contenido hemos empleado cinco imágenes diferentes como elementos de entrada a mostrar.

Hemos predefinido también la dimensión de las imágenes y el intervalo, en segundos, de oscilación entre una y otra.

4.- Uso del Contenido Web

Tras la realización de todos los pasos, debería estar disponible el contenido para su uso tal y como se muestra en la Figura 74. Usaremos la aplicación "*Web Content Display*" que encontraremos bajo el menú "*highlighted* (destacado)" para mostrar contenido web en las páginas.

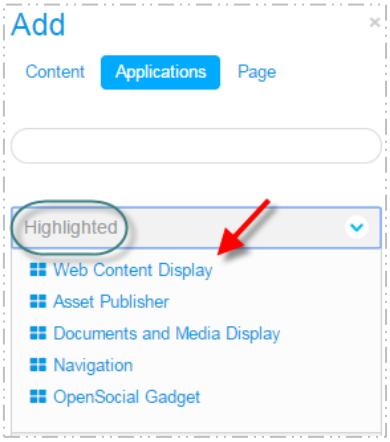


Figura 74 Uso de Contenido Web

En el portlet de visualización de contenido web se puede seleccionar la opción "*Select Web Content*" de la parte inferior para agregar nuestro contenido como se refleja en la Figura 75.

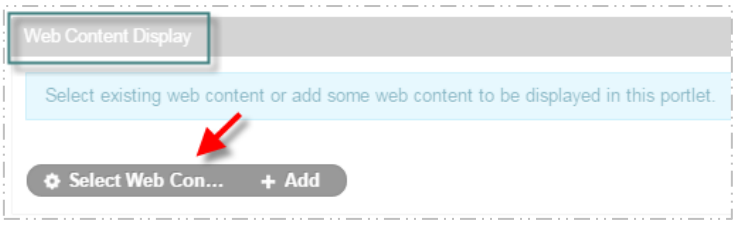


Figura 75 Configuración Widget - Web Content Display

Seleccionamos el contenido deseado como en la Figura 76.

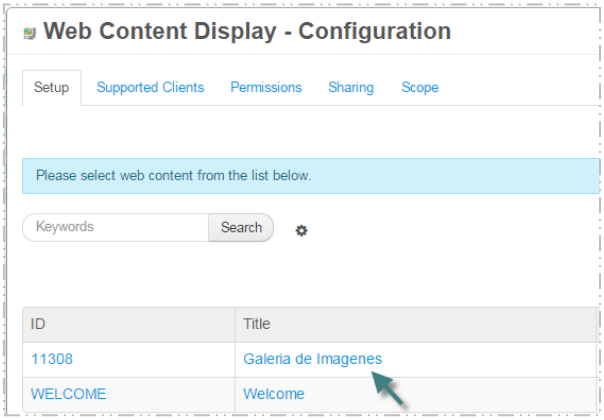


Figura 76 Selección Contenido

Una vez guardemos la selección, veremos el resultado de nuestro carrusel de Imágenes, similar al de la Figura 77.

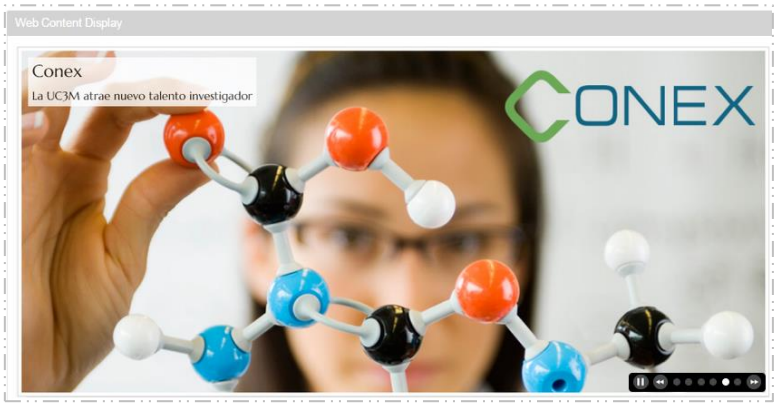


Figura 77 Resultado Carrusel de Imágenes

Anexo V Manual de Usuario

En este anexo se detallan los conocimientos mínimos que debe tener el usuario, ya se trate de un alumno o un profesor para el correcto funcionamiento del portal de Gestión de Notas.

Lo primordial que se debe conocer el usuario es que para acceder debe autenticarse en la aplicación. Podrá hacerlo a través de una cuenta de correo o con un usuario LDAP según lo desee. Una vez autenticado tendrá acceso a un sitio según su perfilado y en este podrá realizar diversas acciones.

Debido a que se trata de una herramienta poco convencional, el usuario deberá adquirir unos conocimientos básicos del uso de la herramienta.
Mentar que la administración mostrada no va a ser demasiado detallada porque se sobreentiende que en un sistema final habrá un administrador experto que poseerá los conocimientos necesarios para realizar las funciones de maquetación de los sitios, gestión de usuarios, etc...

Se han agregado Apps disponibles desde el "*Marketplace*" oficial de Liferay para crear páginas más similares a las que encontraríamos en un entorno productivo y para mostrar la sencilla integración de plugins en un portal.

1 .- Acceso al Portal

Para acceder a la aplicación, como si de cualquier otra aplicación web se tratase, basta con introducir la siguiente URL en un explorador: http://localhost:8085

Una vez la solicitemos, se presentará la página principal, mostrada en la Figura 78, compuesta por 5 portlets remarcados por separado.

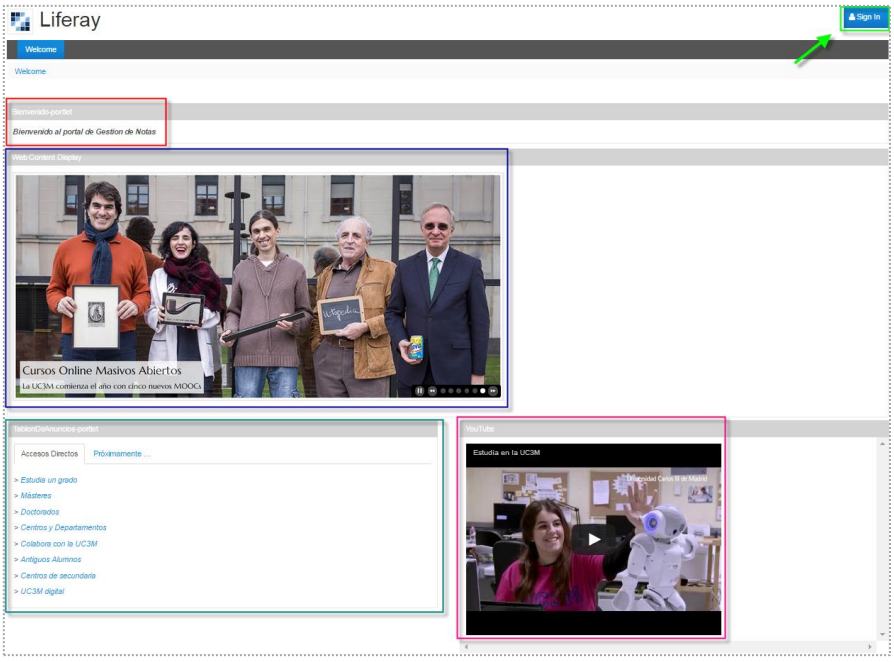


Figura 78 Portal Gestor de Notas

Esta página es común a todos los usuarios, ya se trate de alumnos o docentes.

En ella he desarrollado 3 plugins, descritos en orden:

- Bienvenido-portlet: Es el equivalente al "Hola-Mundo". En su cuerpo se muestra un texto estático de saludo.
- Web Content Display: Se trata de un contenedor de imágenes que van pasando sucesivamente. [*para más detalle véase* ***Anexo IV***]
- Tablón de anuncios: Compuesto por dos pestañas con enlaces a contenidos de interés general.

Adicionalmente se ha agregado un widget multimedia, "YouTube", que muestra un contenido configurado a través de su correspondiente *URL*. Este widget se ha obtenido del *Marketplace*.

2 .- Login de Usuario

En la parte superior derecha, encontramos el widget desde el que podremos realizar el logado en el portal mostrado en la Figura 79. Será necesario estar registrado previamente, en caso de no disponer de usuario y contraseña, se requerirá contactar con el administrador del portal.

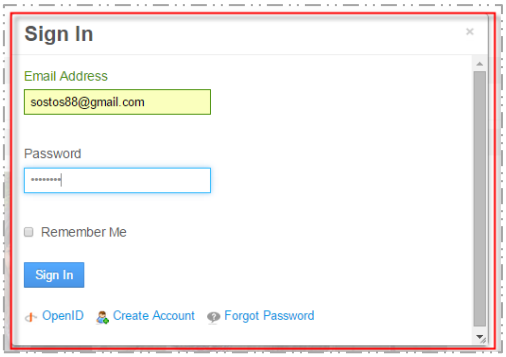


Figura 79 Widget Autenticación

La autenticación se realiza contra el servidor *LDAP* de la uc3m, pero pueden darse de alta nuevos usuarios, en mi caso, se ha creado un usuario de prueba con una cuenta de Gmail.

Cada usuario, tendrá un perfilado de roles, según pertenezca al grupo alumno o profesor y tendrá acceso a un contenido diferente dentro del portal.

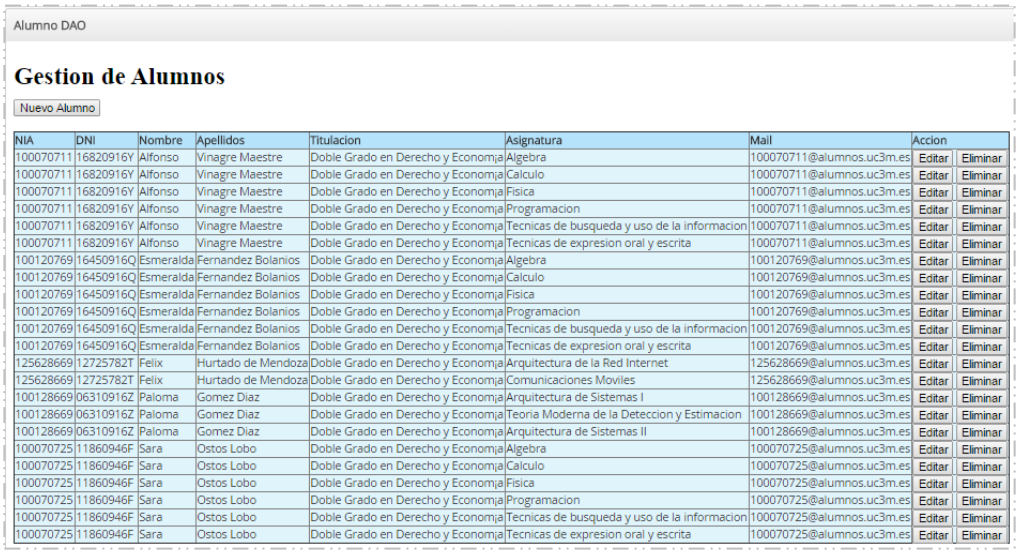
3 .- Contenidos Visibles por un Profesor

Todos los contenidos de obtienen a través de una *BBDD MySQL*.

Se han desarrollado diversos portlets que se van a ir detallando:

➢ *Gestión Alumnos*

Se visualizan los alumnos existentes por titulación, asignatura y curso y se podrán, tanto borrar, como editar sus datos como dar de alta uno nuevo. Un ejemplo se detalla en la Figura 80.

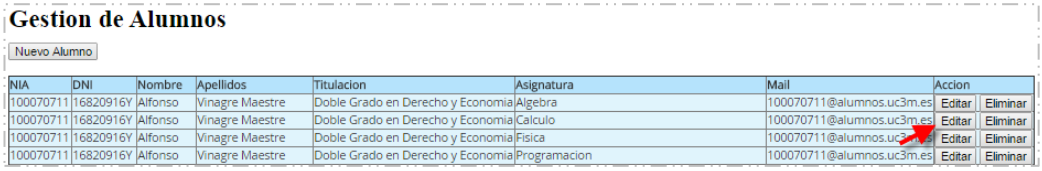


NIA	DNI	Nombre	Apellidos	Titulacion	Asignatura	Mail	Accion	
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Algebra	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Calculo	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Fisica	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Programacion	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Tecnicas de busqueda y uso de la informacion	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Tecnicas de expresion oral y escrita	100070711@alumnos.uc3m.es	Editar	Eliminar
100120769	16450916Q	Esmeralda	Fernandez Bolanios	Doble Grado en Derecho y Economia	Algebra	100120769@alumnos.uc3m.es	Editar	Eliminar
100120769	16450916Q	Esmeralda	Fernandez Bolanios	Doble Grado en Derecho y Economia	Calculo	100120769@alumnos.uc3m.es	Editar	Eliminar
100120769	16450916Q	Esmeralda	Fernandez Bolanios	Doble Grado en Derecho y Economia	Fisica	100120769@alumnos.uc3m.es	Editar	Eliminar
100120769	16450916Q	Esmeralda	Fernandez Bolanios	Doble Grado en Derecho y Economia	Programacion	100120769@alumnos.uc3m.es	Editar	Eliminar
100120769	16450916Q	Esmeralda	Fernandez Bolanios	Doble Grado en Derecho y Economia	Tecnicas de busqueda y uso de la informacion	100120769@alumnos.uc3m.es	Editar	Eliminar
100120769	16450916Q	Esmeralda	Fernandez Bolanios	Doble Grado en Derecho y Economia	Tecnicas de expresion oral y escrita	100120769@alumnos.uc3m.es	Editar	Eliminar
125628669	12725782T	Felix	Hurtado de Mendoza	Doble Grado en Derecho y Economia	Arquitectura de la Red Internet	125628669@alumnos.uc3m.es	Editar	Eliminar
125628669	12725782T	Felix	Hurtado de Mendoza	Doble Grado en Derecho y Economia	Comunicaciones Moviles	125628669@alumnos.uc3m.es	Editar	Eliminar
100128669	06310916Z	Paloma	Gomez Diaz	Doble Grado en Derecho y Economia	Arquitectura de Sistemas I	100128669@alumnos.uc3m.es	Editar	Eliminar
100128669	06310916Z	Paloma	Gomez Diaz	Doble Grado en Derecho y Economia	Teoria Moderna de la Deteccion y Estimacion	100128669@alumnos.uc3m.es	Editar	Eliminar
100128669	06310916Z	Paloma	Gomez Diaz	Doble Grado en Derecho y Economia	Arquitectura de Sistemas II	100128669@alumnos.uc3m.es	Editar	Eliminar
100070725	11860946F	Sara	Ostos Lobo	Doble Grado en Derecho y Economia	Algebra	100070725@alumnos.uc3m.es	Editar	Eliminar
100070725	11860946F	Sara	Ostos Lobo	Doble Grado en Derecho y Economia	Calculo	100070725@alumnos.uc3m.es	Editar	Eliminar
100070725	11860946F	Sara	Ostos Lobo	Doble Grado en Derecho y Economia	Fisica	100070725@alumnos.uc3m.es	Editar	Eliminar
100070725	11860946F	Sara	Ostos Lobo	Doble Grado en Derecho y Economia	Programacion	100070725@alumnos.uc3m.es	Editar	Eliminar
100070725	11860946F	Sara	Ostos Lobo	Doble Grado en Derecho y Economia	Tecnicas de busqueda y uso de la informacion	100070725@alumnos.uc3m.es	Editar	Eliminar
100070725	11860946F	Sara	Ostos Lobo	Doble Grado en Derecho y Economia	Tecnicas de expresion oral y escrita	100070725@alumnos.uc3m.es	Editar	Eliminar

Figura 80 Widget Gestión Alumnos

○ *Opción Editar*

En la Figura 81 encontramos un ejemplo de resultado obtenido.



NIA	DNI	Nombre	Apellidos	Titulacion	Asignatura	Mail	Accion	
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Algebra	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Calculo	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Fisica	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Programacion	100070711@alumnos.uc3m.es	Editar	Eliminar

Figura 81 Opción Editar - Widget Gestión Alumnos

Seleccionamos el registro a editar a través del formulario mostrado en la Figura 82.

Figura 82 Formulario Edición - Widget Gestión Alumnos

Cambiamos, por ejemplo la asignatura a “Informática grafica” como en la Figura 83 y guardamos el cambio.

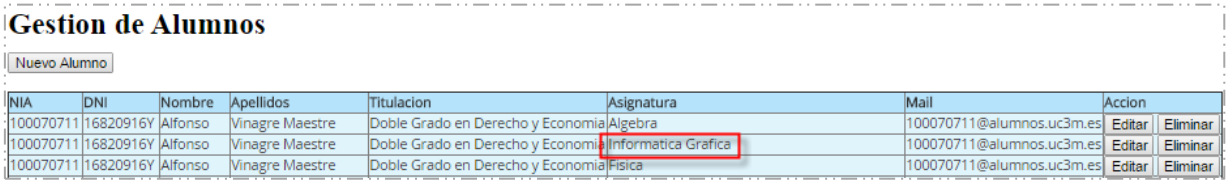


Figura 83 Ejemplo Edición Asignatura - Widget Gestión Alumnos

Al regresar vemos que se ha cambiado correctamente.

- ***Opción Eliminar***

En la Figura 84 recogemos un ejemplo de tabla de datos desde la que podemos eliminar un registro.

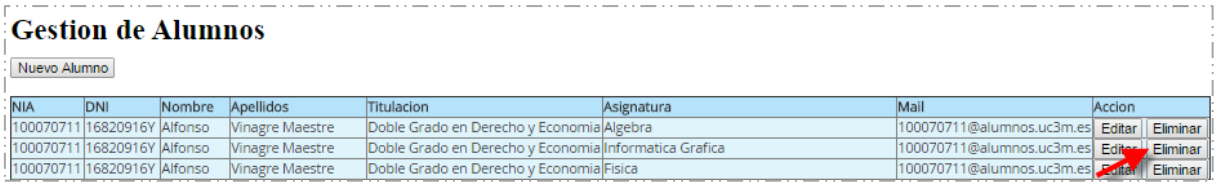


Figura 84 Opción Eliminar - Widget Gestión Alumnos

Seleccionamos el alumno a eliminar y nos aparece el formulario de la Figura 85.

Figura 85 Formulario comprobación - Widget Gestión Alumnos

Nuevamente nos aparecen los datos del usuario que vamos a eliminar para verificar que no eliminemos erróneamente y al guardar desaparecerá el registro como se observa en la Figura 86.

NIA	DNI	Nombre	Apellidos	Titulacion	Asignatura	Mail	Accion	
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Algebra	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Fisica	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Programacion	100070711@alumnos.uc3m.es	Editar	Eliminar

Figura 86 Resultado Borrado Registro - Widget Gestión Alumnos

- ***Opción Nuevo Alumno***

Pulsando el botón nos aparece el formulario de la Figura 87 a completar.

Figura 87 Formulario Alta Alumno - Widget Gestión Alumnos

Agregamos los datos, guardamos y veremos que se ha añadido el usuario en nuestra tabla. Esta comprobación se visualiza en la Figura 88.

NIA	DNI	Nombre	Apellidos	Titulacion	Asignatura	Mail	Accion	
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Algebra	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Fisica	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Programacion	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Tecnicas de busqueda y uso de la informacion	100070711@alumnos.uc3m.es	Editar	Eliminar
100070711	16820916Y	Alfonso	Vinagre Maestre	Doble Grado en Derecho y Economia	Tecnicas de expresion oral y escrita	100070711@alumnos.uc3m.es	Editar	Eliminar
100120769	16450916Q	Esmeralda	Fernandez Bolanios	Doble Grado en Derecho y Economia	Algebra	100120769@alumnos.uc3m.es	Editar	Eliminar
100120769	16450916Q	Esmeralda	Fernandez Bolanios	Doble Grado en Derecho y Economia	Calculo	100120769@alumnos.uc3m.es	Editar	Eliminar
100120769	16450916Q	Esmeralda	Fernandez Bolanios	Doble Grado en Derecho y Economia	Fisica	100120769@alumnos.uc3m.es	Editar	Eliminar
100120769	16450916Q	Esmeralda	Fernandez Bolanios	Doble Grado en Derecho y Economia	Programacion	100120769@alumnos.uc3m.es	Editar	Eliminar
100120769	16450916Q	Esmeralda	Fernandez Bolanios	Doble Grado en Derecho y Economia	Tecnicas de busqueda y uso de la informacion	100120769@alumnos.uc3m.es	Editar	Eliminar
100120769	16450916Q	Esmeralda	Fernandez Bolanios	Doble Grado en Derecho y Economia	Tecnicas de expresion oral y escrita	100120769@alumnos.uc3m.es	Editar	Eliminar
125628669	12725782T	Felix	Hurtado de Mendoza	Doble Grado en Derecho y Economia	Arquitectura de la Red Internet	125628669@alumnos.uc3m.es	Editar	Eliminar
125628669	12725782T	Felix	Hurtado de Mendoza	Doble Grado en Derecho y Economia	Comunicaciones Moviles	125628669@alumnos.uc3m.es	Editar	Eliminar
100128669	06310916Z	Paloma	Gomez Diaz	Doble Grado en Derecho y Economia	Arquitectura de Sistemas I	100128669@alumnos.uc3m.es	Editar	Eliminar
100128669	06310916Z	Paloma	Gomez Diaz	Doble Grado en Derecho y Economia	Teoria Moderna de la Deteccion y Estimacion	100128669@alumnos.uc3m.es	Editar	Eliminar
100128669	06310916Z	Paloma	Gomez Diaz	Doble Grado en Derecho y Economia	Arquitectura de Sistemas II	100128669@alumnos.uc3m.es	Editar	Eliminar
100044434	12292323T	Prueba	Prueba	Doble Grado en Estudios Internacionales y Ciencias Politicas	Inteligencia Artificial	prueba@gmail.com	Editar	Eliminar

Figura 88 Resultado Alta Alumno - Widget Gestión Alumnos

➢ *Gestión Asignaturas*

Visualmente es igual y permite las mismas opciones que el plugin de gestión de alumnos. En este plugin, también se permite modificar, agregar y eliminar registros de la tabla de asignaturas como se ve en la Figura 89.

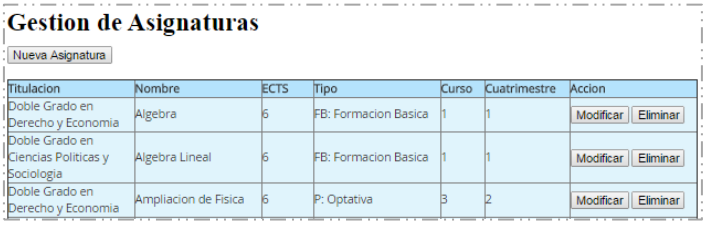


Figura 89 Widget Gestión Asignaturas

○ *Opción Modificar*

La opción de Modificar se puede seleccionar desde el botón señalado en la Figura 90.

Figura 90 Opción Modificar - Widget Gestión Asignaturas

Seleccionamos la fila a editar, en el ejemplo de la Figura 91 modificamos el campo "Tipo".

Figura 91 Formulario Edición - Widget Gestión Asignaturas

Cambiamos el Tipo de Asignatura y salvamos el cambio. Como se ve en la Figura 92, el cambio se ha realizado correctamente.

Titulacion	Nombre	ECTS	Tipo	Curso	Cuatrimestre	Accion
Doble Grado en Derecho y Economia	Algebra	6	FB: Formacion Basica	1	1	Modificar Eliminar
Doble Grado en Ciencias Politicas y Sociologia	Algebra Lineal	6	TF: Trabajo Fin de Grado	1	1	Modificar Eliminar

Figura 92 Resultado Edición - Widget Gestión Asignaturas

- ***Opción Eliminar***

Desde el menú inicial reflejado en la Figura 93 podemos seleccionar la opción de Eliminar un registro.

Gestion de Asignaturas

Nueva Asignatura

Titulacion	Nombre	ECTS	Tipo	Curso	Cuatrimestre	Accion
Doble Grado en Derecho y Economia	Algebra	6	FB: Formacion Basica	1	1	Modificar Eliminar
Doble Grado en Ciencias Politicas y Sociologia	Algebra Lineal	6	TF: Trabajo Fin de Grado	1	1	Modificar Eliminar

Figura 93 Opción Eliminar - Widget Gestión Asignaturas

Validamos que sea el registro deseado y guardamos. Veremos, como en la Figura 94 que el registro ha desaparecido.

Asignaturas DAO

Gestion de Asignaturas

Nueva Asignatura

Titulacion	Nombre	ECTS	Tipo	Curso	Cuatrimestre	Accion
Doble Grado en Ciencias Politicas y Sociologia	Algebra Lineal	6	TF: Trabajo Fin de Grado	1	1	Modificar Eliminar
Doble Grado en Derecho y Economia	Ampliacion de Fisica	6	P: Optativa	3	2	Modificar Eliminar

Figura 94 Resultado Borrado Registro - Widget Gestión Asignaturas

- ***Opción Nueva Asignatura***

Seleccionamos el botón señalado en la Figura 95 para dar de alta una nueva asignatura.

Figura 95 Botón Alta Asignatura

Rellenamos el formulario de alta como el mostrado en la Figura 96.

Figura 96 Formulario Alta Registro - Widget Gestión Asignaturas

Grabamos los datos y vemos el registro agregado como en el ejemplo de la Figura 97.

Doble Grado en Derecho y Economia	Proyectos, Normativa y Politica de Telecomunicaciones	6	O: Obligatoria	4	1	Modificar Eliminar
Doble Grado en Derecho y Economia	Prueba	3	FB: Formación básica	4	2	Modificar Eliminar
Doble Grado en Ciencias Politicas y Sociologia	Redes de Neuronas Artificiales	6	O: Obligatoria	4	1	Modificar Eliminar

Figura 97 Resultado Alta Registro - Widget Gestión Asignaturas

- ***Gestión Notas Alumno***

A través de un menú inicial mostrado en la Figura 98 se ofrece la posibilidad de realizar varias operaciones.

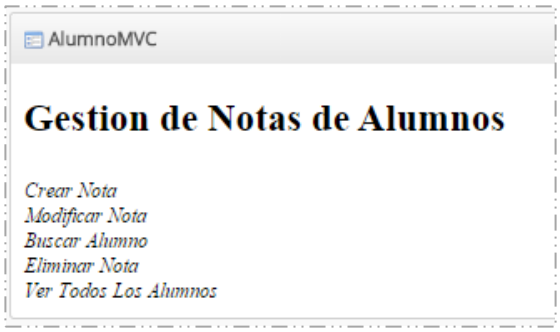


Figura 98 Menú Widget Gestión Notas Alumno

Vamos a detallar las pantallas que se visualizarían al acceder a cada una de ellas:

- ***Opción Crear Nota***

Opción mostrada en la Figura 99.

Figura 99 Opción Crear Nota - Widget Gestión Notas Alumno

Rellenamos todos los campos y pulsamos guardar para almacenar el nuevo registro en nuestra base de datos. Aparecerá una alerta indicando si ha sido satisfactoria o errónea la operación como la de la Figura 100.

Figura 100 Mensaje Resultado Operación

- ***Opción Modificar Nota***

Cuando intentamos editar una nota, se obtiene la pantalla de selección de identificador como la reflejada en la Figura 101.

Figura 101 Opción Modificar Nota - Widget Gestión Notas Alumno

Seleccionando un *id* se muestra el detalle del alumno junto con su nota. No es necesario conocer los identificadores de la tabla de antemano porque la carga se realiza en tiempo de ejecución. Un ejemplo de la pantalla mencionada se encuentra en la Figura 102.

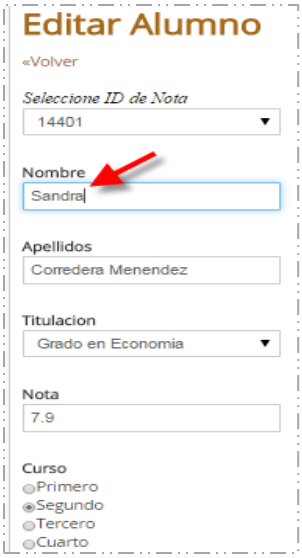
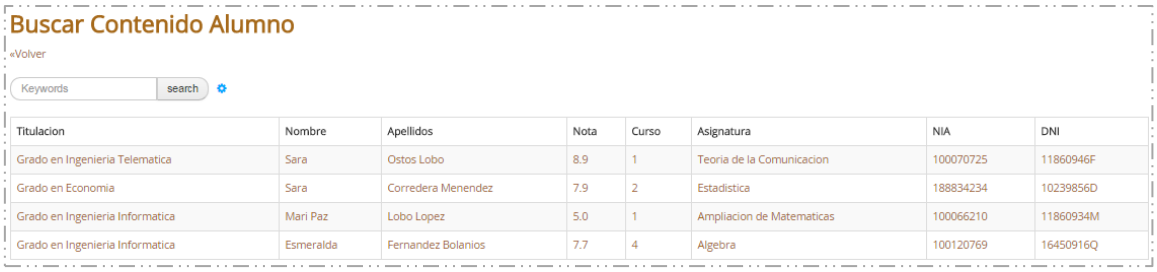


Figura 102 Formulario Edición - Widget Gestión Notas Alumno

Cambiamos un campo, por ejemplo el nombre. Se muestra una notificación al alumno indicando que el proceso ha funcionado correctamente o en caso contrario, si se produjese un error en el proceso de guardado en la parte superior de la ventana emergente.

- ***Opción Buscar Alumno***

Se presenta la Figura 103 con los registros de alumnos existentes, junto con su asignatura, curso y nota a la que pertenece.



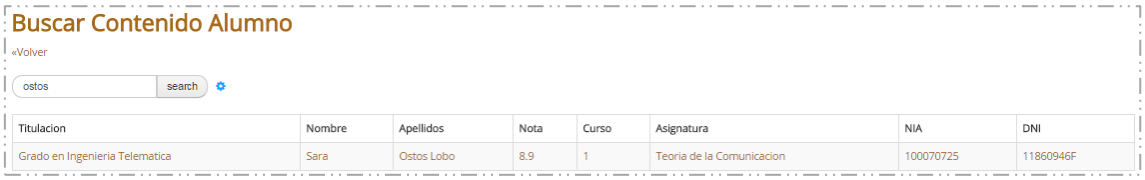
Titulacion	Nombre	Apellidos	Nota	Curso	Asignatura	NIA	DNI
Grado en Ingenieria Telematica	Sara	Ostos Lobo	8.9	1	Teoria de la Comunicacion	100070725	11860946F
Grado en Economia	Sara	Corredera Menendez	7.9	2	Estadistica	188834234	10239856D
Grado en Ingenieria Informatica	Mari Paz	Lobo Lopez	5.0	1	Ampliacion de Matematicas	100066210	11860934M
Grado en Ingenieria Informatica	Esmeralda	Fernandez Bolanios	7.7	4	Algebra	100120769	16450916Q

Figura 103 Opción Buscar Alumno - Widget Gestión Notas Alumno

Insertamos en el buscador un texto de búsqueda como por ejemplo el agregado en la Figura 104.

Figura 104 Buscador - Widget Gestión Notas Alumno

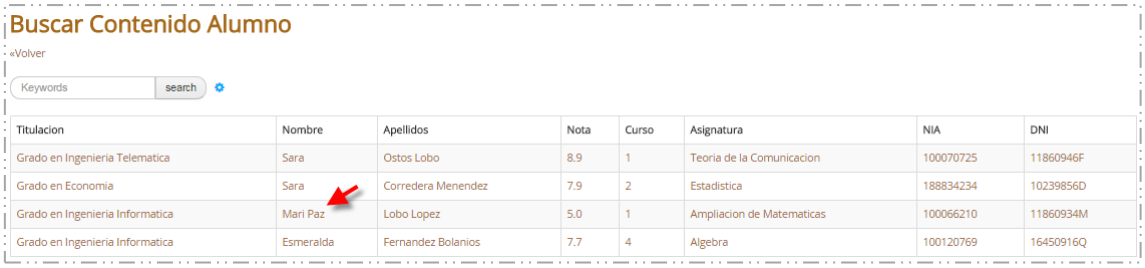
Search – Al pulsar el botón se realiza un filtrado obteniéndose el resultado de un alumno concreto. El registro obtenido será similar al de la Figura 105.



Titulacion	Nombre	Apellidos	Nota	Curso	Asignatura	NIA	DNI
Grado en Ingenieria Telematica	Sara	Ostos Lobo	8.9	1	Teoria de la Comunicacion	100070725	11860946F

Figura 105 Resultado Búsqueda - Widget Gestión Notas Alumno

Si pulsamos sobre uno de los campos, ya sea: nombre, apellidos, nota, curso o asignatura, vemos un detalle del alumno. En la Figura 106 seleccionamos el campo Nombre.



Titulacion	Nombre	Apellidos	Nota	Curso	Asignatura	NIA	DNI
Grado en Ingenieria Telematica	Sara	Ostos Lobo	8.9	1	Teoria de la Comunicacion	100070725	11860946F
Grado en Economia	Sara	Corredera Menendez	7.9	2	Estadistica	188834234	10239856D
Grado en Ingenieria Informatica	Mari Paz	Lobo Lopez	5.0	1	Ampliacion de Matematicas	100066210	11860934M
Grado en Ingenieria Informatica	Esmeralda	Fernandez Bolanios	7.7	4	Algebra	100120769	16450916Q

Figura 106 Detalle Búsqueda - Widget Gestión Notas Alumno

La pantalla resultante de detalle de alumno recoge los datos mostrados en la Figura 107.

Figura 107 Descripción de Alumno - Widget Gestión Notas Alumno

- ***Opción Eliminar Nota***

Para eliminar una nota, obtendremos la interfaz mostrada en la Figura 108.

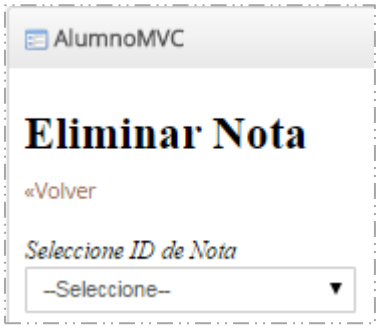


Figura 108 Opción Eliminar Nota - Widget Gestión Notas Alumno

Seleccionando el *identificador* se obtienen los datos del alumno a eliminar como en la Figura 109.

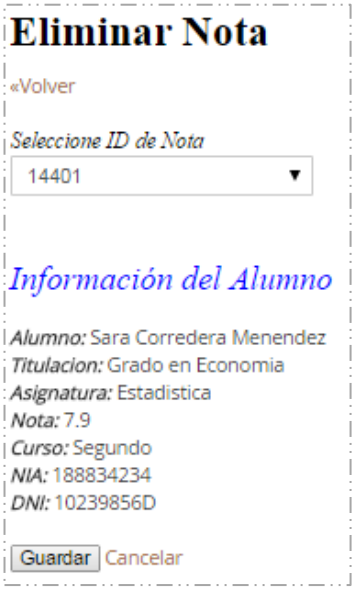


Figura 109 Información Alumno a Eliminar - Widget Gestión Notas Alumno

- ***Opción Ver Todos los Alumnos***

Si elegimos la opción de ver todos los contenidos, obtenemos la tabla resultante de la Figura 110. Los resultado se encuentran paginados de 5 en 5 resultados.

Listado de Alumnos

«Volver

Titulacion	Nombre	Apellidos	Nota	Curso	Asignatura Alumno	NIA	DNI
Grado en Ingenieria Telematica	Sara	Ostos Lobo	8.9	1	Teoria de la Comunicacion	100070725	11860946F
Grado en Economia	Sara	Corredera Menendez	7.9	2	Estadistica	188834234	10239856D
Grado en Ingenieria Informatica	Mari Paz	Lobo Lopez	5.0	1	Ampliacion de Matematicas	100066210	11860934M
Grado en Ingenieria Informatica	Esmeralda	Fernandez Bolanios	7.7	4	Algebra	100120769	16450916Q
Grado en Ingenieria Telematica	Alfonso	Vinagre Maestre	9.6	2	Teoria de la Comunicacion	100070711	16820916Y

Figura 110 Opción Ver Todos Los Alumnos - Widget Gestión Notas Alumno

Pulsando sobre el ***apellido*** o cualquier otro campo tendremos acceso a la descripción. El detalle de dicha información se puede ver en la Figura 111.

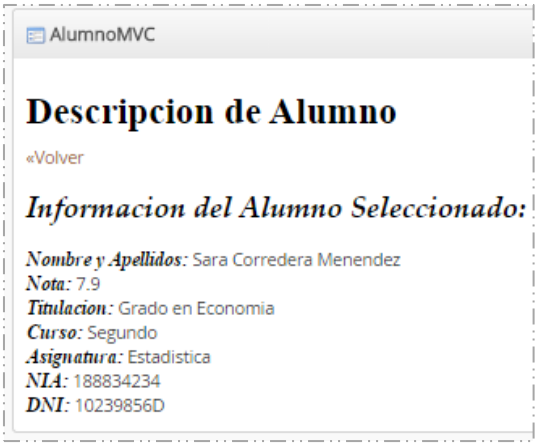


Figura 111 Descripción Alumno - Widget Gestión Notas Alumno

4 .- Contenidos Visibles por un Alumno

El *Site* que visualizará cualquier alumno será equivalente al de la Figura 112.

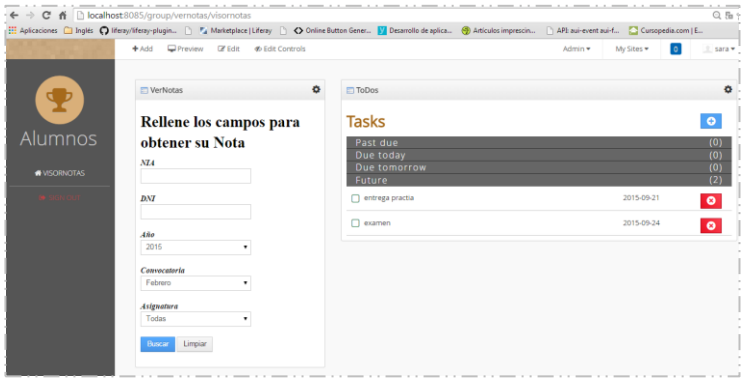


Figura 112 Sitio Web Alumno

Un alumno tendrá la opción de ver su Nota, para ello, únicamente deberá rellenar los campos debidamente y seleccionar el botón de búsqueda como se realiza en la Figura 113.

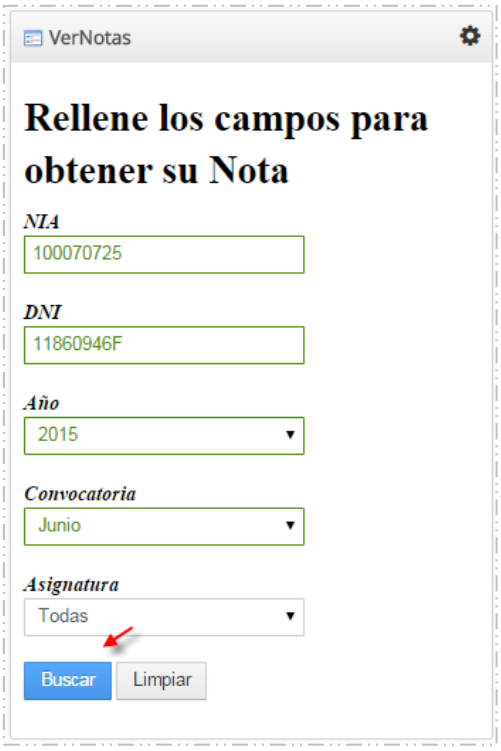


Figura 113 Formulario Búsqueda Nota

En el caso de que haya resultados que coincidan con los criterios de búsqueda, se obtendrá un resultado como el de la Figura 114.

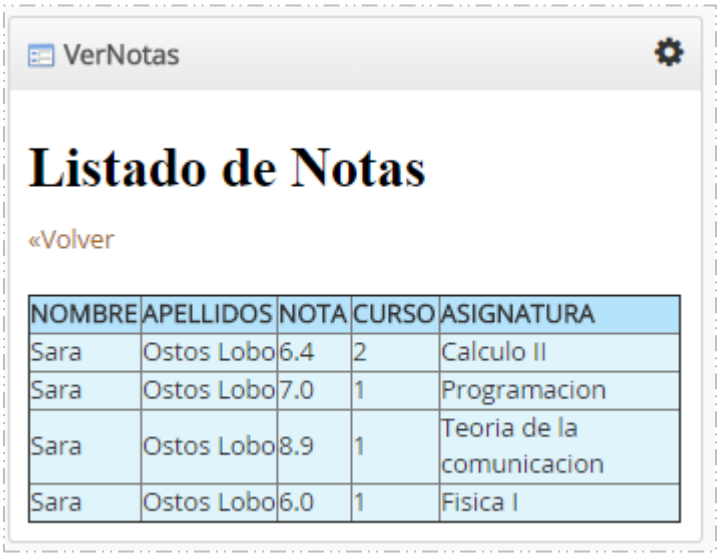


Figura 114 Resultado Búsqueda Nota

En caso contrario, se notificará al usuario la falta de resultados como se muestra en la Figura 115.

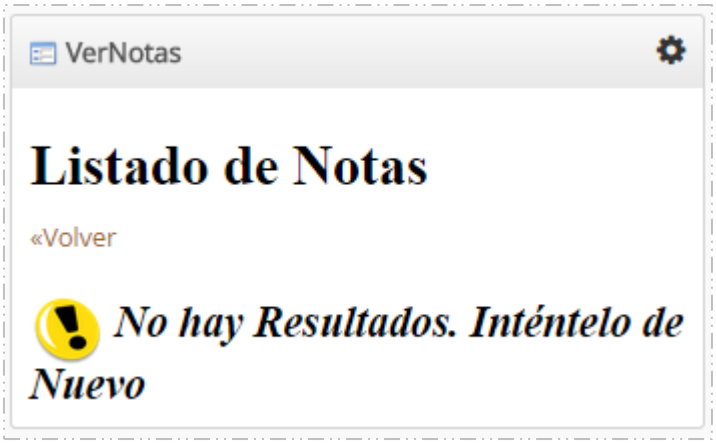


Figura 115 Búsqueda Nota Sin Resultados

Planificación y Presupuesto

En este capítulo se detallan las fases que se han llevado a cabo durante la realización del Proyecto y el coste que ha supuesto la realización del mismo.

Planificación Temporal

Para mostrar la planificación vamos a presentar un diagrama de Gantt. Esta herramienta nos permite modelar de forma cómoda la planificación de las tareas necesarias para la realización de nuestro proyecto. Primero presentaremos un resumen de las tareas que han formado el proyecto en la Figura 116.

Nombre	Fecha de inicio	Duración
Definición de Requisitos	16/02/15	7 Días
Documentación Inicial	25/02/15	2 Días
Diseño Modelo de Datos	27/02/15	9 Días
Planificación	12/03/15	4 Días
Análisis	18/03/15	23 Días
Objetivos del Sistema	18/03/15	5 Días
Usuarios Requeridos	25/03/15	1 Día
Requisitos de la Aplicacion	26/03/15	5 Días
Plugins Necesarios	2/04/15	10 Días
Estado del Arte	13/04/15	2 Días
Preparación Entorno	15/04/15	5 Días
Instalación Liferay	15/04/15	2 Días
Instalación MySQL	17/04/15	2 Días
Instalación Eclipse	21/04/15	1 Día
Aprendizaje Herramientas	22/04/15	30 Días
Administración	22/04/15	5 Días
Base de Datos MySQL	22/04/15	1 Días
Liferay Portal	23/04/15	4 Días
Desarrollo	29/04/15	70 Días
Plugin`s Liferay	29/04/15	45 Días
Portlets	29/04/15	37 Días
Layouts	18/06/15	8 Días
Spring MVC	29/06/15	4 Días
Integración Base de Datos	1/07/15	2 Días
Diseño Portal	15/06/15	18 Días
Contenido	15/06/15	12 Días
Interfaces	25/06/15	6 Días
Desarrollo	6/04/15	70 Días
Implementación	6/04/15	43 Días
Diseño Gráfico	3/06/15	12 Días
Codificación	16/06/15	15 Días
Pruebas	7/07/15	4 Días
Documentación	29/06/15	40 Días

Figura 116 Diagrama de Gantt (Planificación Temporal)

La planificación consta de aproximadamente 212 días. Se ha realizado una clasificación global de las tareas pero muchas de ellas podrían detallarse más si cabe (especialmente las de análisis, aprendizaje de herramientas y desarrollo).

Para cada tarea, se indica el coste de realizarlas medido en días.

En la Figura 117 vamos a mostrar en una línea de tiempo la duración correspondiente a las tareas descritas anteriormente:

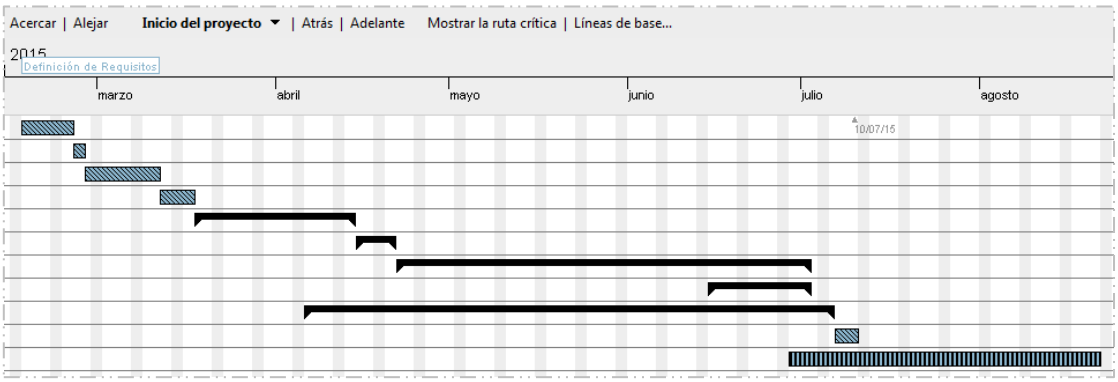


Figura 117 Duración Tareas Desarrollo Proyecto

Valoración Económica

Este apartado se intentará cuantificar el esfuerzo realizado dándole una proyección económica. No obstante, mencionar que hay ciertos costes adicionales que van más allá del precio del desarrollo de la aplicación web.

Para calcular el presupuesto, hay que considerar los costes de personal, los de material y los propios de desarrollo.

- ***Costes de Personal***

Se debe tener en cuenta, que el desarrollo deberá ser realizado por un Ingeniero Técnico, cuya tarifa se estima en 32 €/hora. Considerando que han sido necesarios, según lo visualizado en el diagrama de Gantt, 212 días y que en cada uno de ellos, se ha trabajado una media de 3 horas. Con estas consideraciones se obtienen los datos reflejados en la Tabla 7.

Tabla 6 Valoración Económica

FASE	DIAS	HORAS	COSTE
Análisis, Planificación y estudio de la documentación inicial	22	66	2112
Análisis y Diseño	41	123	3936
Implementación	105	315	10080
Pruebas de Aplicación	4	12	384
Documentación	40	120	3840
Coste Material (Portátil)			1000
COSTE TOTAL			***21352 €***

Bibliografía

[1] L. Inc., «Liferay,» [En línea]. Available: http://www.liferay.com/.

[2] Oracle., «MySQL,» [En línea]. Available: http://www.oracle.com/es/products/mysql/index.html.

[3] Oracle., «Differences between Java EE and Java SE,» [En línea]. Available: http://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html.

[4] H. Barrantes, «Diferencias entre j2ee/se/me,» [En línea]. Available: http://www.codigofantasma.com/blog/diferencias-entre-j2ee-j2se-j2me/.

[5] A. R. y. W. Sagástegui, «JVM,» [En línea]. Available: http://www.aprenderaprogramar.com/index.php?option=com_content&id=392:la-maquina-virtual-java-jvm-o-java-virtual-machine-compilador-e-interprete-bytecode-cu00611b&Itemid=188.

[6] Wikipedia, «API Java,» 24 Abril 2015. [En línea]. Available: https://es.wikipedia.org/wiki/API_Java.

[7] Oracle., «JDBC Introduction,» [En línea]. Available: https://docs.oracle.com/javase/tutorial/jdbc/overview/.

[8] Wikipedia, «Extensible Markup Language,» 30 Junio 2015. [En línea]. Available: https://es.wikipedia.org/wiki/Extensible_Markup_Language.

[9] Reichard, «Componentes J2EE,» [En línea]. Available: http://www.muskingum.edu/~reichard/J2EE/j2eetutorial/doc/Overview2.html.

[10] Oracle, «Java Transaction Service (JTS),» [En línea]. Available: http://www.oracle.com/technetwork/java/javaee/jta/index.html.

[11] Oracle, «JNDI,» [En línea]. Available: http://www.oracle.com/technetwork/java/jndi/index.html.

[12] Jbarrios, «Servicios J2EE,» 30 05 2003. [En línea]. Available: http://users.dcc.uchile.cl/~jbarrios/J2EE/node21.html.

[13] Oracle, «Introducing Java Portlet Specifications: JSR 168 and JSR 286,» 16 Marzo 2007. [En

línea]. Available: http://www.oracle.com/technetwork/java/index-raji-test-141933.html.

[14] «JSR 168: Portlet Specification,» [En línea]. Available: http://jcp.org/jsr/detail/168.jsp.

[15] E. d. p. 2.0, «JSR 286: Portlet Specification 2.0,» [En línea]. Available: http://jcp.org/en/jsr/detail?id=286.

[16] «JSR-286,» 5 Marzo 2008. [En línea]. Available: http://okham.blogspot.com.es/2008/08/jsr-286.html.

[17] j. c. fernandez, «Liferay con soporte a la nueva JSR-286,» 04 07 2008. [En línea]. Available: http://www.juancarlosfernandez.net/2008/07/liferay-con-soporte-la-nueva-jsr-286.html.

[18] Wikipedia, «Tomcat,» 8 Julio 2015. [En línea]. Available: http://es.wikipedia.org/wiki/Tomcat.

[19] mulesoft, «tomcat-websphere,» [En línea]. Available: https://www.mulesoft.com/tcat/tomcat-websphere.

[20] «APP Server Most Used,» 21 Mayo 2014. [En línea]. Available: http://blog.arungupta.me/jboss-wildfly-top-javaee-appserver-zeroturnaround-report/.

[21] Hibernate, «HQL: El Lenguaje de Consulta de Hibernate,» Red Hat Middleware, LLC., 2004. [En línea]. Available: https://docs.jboss.org/hibernate/orm/3.3/reference/es-ES/html/queryhql.html.

[22] C. Walls, Spring in Action, Manning, Third Edition Edition.

[23] R. E. P. T. Thomas Risberg, «Desarrollando una aplicacion Spring Framework MVC paso a paso,» [En línea]. Available: http://www.davidmarco.es/spring-mvc.

[24] Liferay, Octubre 2014. [En línea]. Available: https://www.liferay.com/about-us/awards/gartnermq-portals.

[25] AlloyUI, «Api AlloyUI,» [En línea]. Available: http://alloyui.com.

[26] M. Prince, «Introducción a AUI,» [En línea]. Available: http://www.liferaysavvy.com/2014/02/introduction-to-aui-java-script-in.html.

[27] Liferay, «Liferay Portal CE & Liferay Portal EE,» [En línea]. Available: http://catalogo.asac.as/documents/28537/0/LiferayPortalCE%26LiferayPortalEE.pdf/acd3749f-d950-45dc-aaa2-a884350631bd.

[28] K. Ramirez, «Portlets and Servlets: What's the difference?,» [En línea]. Available: https://portlet.dev.java.net/files/documents/1654/11398/tip4.html.

[29] J. L.C., «Aprendiendo sobre portlets,» [En línea]. Available: http://jesuslc.com/2013/02/18/aprendiendo-sobre-portlets/.

[30] Liferay, «Portal Hook Plugins,» [En línea]. Available: http://www.liferay.com/es/community/wiki/-/wiki/Main/Portal+Hook+Plugins.

[31] Liferay, «Portal Properties 6.0.5,» [En línea]. Available: https://www.liferay.com/es/community/wiki/-/wiki/Main/Portal+Properties+6.0.5#section-Portal+Properties+6.0.5-LDAP.

[32] Oracle, «Core J2EE Patterns - Data Access Object,» [En línea]. Available: http://www.oracle.com/technetwork/java/dataaccessobject-138824.html.

[33] Liferay, «Service Builder,» [En línea]. Available: https://www.liferay.com/es/documentation/liferay-portal/6.0/development/-/ai/service-build-2 .

[34] Liferay, «Advanced Content with Structures and Template,» [En línea]. Available: https://www.liferay.com/es/documentation/liferay-portal/6.2/user-guide/-/ai/advanced-content-with-structures-and-te-liferay-portal-6-2-user-guide-03-en.

[35] Wikipedia, «Características del producto,» 8 Julio 2015. [En línea]. Available: https://es.wikipedia.org/wiki/Tomcat.

[36] «Wikipedia,» 8 Julio 2015. [En línea]. Available: http://es.wikipedia.org/wiki/Tomcat.