UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR
INGENIERÍA INFORMÁTICA

SISTEMA PARA LA MAQUETACIÓN DE COMPONENTES JSF

JSFCOMPOSER

Jorge Villalobos Beato
David Díez Cebollero
06/05/2009

Índice de contenidos

Índice de figuras

Índice de tablas

Glosario de términos

ActionScript	Lenguaje de script utilizado por Adobe Flash.
AJAX	Utilización conjunta de diversas tecnologías, Javascript y XML, para conseguir realizar peticiones asíncronas al servidor.
API	Del inglés *Application Programming Interface*, se refiere a la interfaz externa que proporciona un determinado programa para ser invocado por otros.
Aplicación web	Aplicación informática cuyo medio de presentación es un navegador web.
Asíncrono	Algo que se lleva a cabo sin obligar a su creador a esperar hasta que este termine.
Bean	Objeto Java de almacenamiento de información volátil en aplicaciones web J2EE.
Blank	Se utiliza frecuentemente para referirse a aplicaciones web de ejemplo de incorporación y uso de una determinada librería, de forma que contienen el esqueleto mínimo.
Contenedor de servlets	Servidor web capaz de manejar el ciclo de vida de servlets.
Cross-browser	Algo que mantiene su aspecto y funcionalidad de un navegador a otro.
***Frames*/Marcos**	Separaciones físicas realizables en una página web, de manera que una página dividida en marcos referencia a su vez a un conjunto de páginas.
***Framework*/Marco de trabajo**	Librería o herramienta para facilitar el desarrollo en una tecnología o entorno concreto.
GPL	GNU *General Public License* es un tipo de licencia que acompaña al software libre y especifica qué el código fuente de dicho software puede ser copiado, modificado y distribuido libremente.
HTML	*HyperText Markup Language*, Lenguaje de Marcado de Hipertexto. Es el lenguaje en el que se escriben las páginas web.
Internacionalización	Proceso por el cual se posibilita que un recurso sea accesible o visible en varios idiomas.
IVA	Impuesto sobre el Valor Añadido, es el impuesto que recae sobre el consumo en España.
J2EE	*Java Enterprise Edition*, es la versión de Java para implementar aplicaciones web.
Java	Lenguaje de programación desarrollado por Sun Microsystems.
JavaCC	Generador de *parsers* escrito en Java.
JavaScript	Lenguaje de script que utilizan las páginas web.
JSF	JavaServer Faces, estándar de Sun Microsystems para la capa de presentación de las aplicaciones web J2EE.
JSP	JavaServer Pages, páginas web usadas en la capa de presentación de las aplicaciones web J2EE que incluyen procesamiento de código

	en la parte servidora.
Kickstart	Similar a *blank*.
Lógica de negocio	Capa de un sistema que se encarga de realizar las tareas de negocio, es decir, aquellas para las que el sistema fue desarrollado en primera instancia.
LOPD	Ley Orgánica de Protección de Datos de carácter personal aplicada en España a todos los datos sensibles almacenados o tratados en soportes informáticos.
Material fungible	Material de oficina no inventariable: papel, cartuchos para impresora, bolígrafos, clips, etc.
MD5	Función criptográfica de resumen de datos.
MVC	Modelo-Vista-Controlador, patrón de diseño web.
Open-source	Código abierto, indica que el código fuente de una aplicación puede ser visto por cualquiera.
Parser	Aplicación que analiza un lenguaje cualquiera y realiza operaciones con los elementos que se van identificando y procesando.
PHP	Lenguaje de programación para páginas web interpretado en la parte servidora.
Salario bruto	Salario que recibe un trabajador de la empresa en la que trabaja. A este salario han de restársele los porcentajes que se lleva el Estado.
Servidor web	Aplicación especial instalada en un ordenador que permite que recibir y procesar peticiones de clientes remotos o locales de recursos locales.
Servlet	Clase Java capaz de procesar una determinada petición web y generar el resultado correspondiente.
Singleton	Patrón de diseño gracias al cual sólo existe una instancia de la clase que lo implemente en todo el sistema.
SQL	*Structured Query Language*, Lenguaje Estructura de Consultas. Es el lenguaje más utilizado para realizar consultas en clientes de Bases de Datos.
UML	*Unified Modeling Language*, Lenguaje de Modelado Unificado. Es un lenguaje de modelado ampliamente utilizado para describir varios aspecto de una aplicación, como puede ser funcional, de comportamiento, estructural, etc.
UTF-8	Sistema de codificación de caracteres.
WAR	Web-ARchive, fichero comprimido que contiene una aplicación web J2EE.
XHTML	EXtended HTML, lenguaje diseñado para remplazar a HTML. Proporciona las mismas funcionalidades y estructura pero es mucho más estricto al tener estructura XML.
XML	EXtensible Markup Language, lenguaje para transporte y almacenamiento de datos.
XSD	*XML Schema Definition*, Definición de Esquema XML. Describe la estructura de un fichero XML.

1 Introducción

El presente documento describe el trabajo acometido durante la realización del Proyecto de Fin de Carrera "Sistema para la maquetación de componentes JSF". Dicho proyecto se encuentra enmarcado en el contexto del desarrollo de aplicaciones web. La función de esta memoria es la de explicar la planificación, diseño, desarrollo y seguimiento del mismo.

Este primer capítulo pretende servir como introducción al trabajo realizado, para lo cual se encuentra dividido en varias secciones. En primer lugar, se explicará el planteamiento del problema encontrado. A continuación, se comentarán los objetivos que se desean conseguir. Después, se discutirá la metodología seguida para llevar a cabo el proyecto. Por último, se expondrá la estructura de la presente memoria.

1.1 Planteamiento del problema

Para poder explicar el planteamiento del problema es necesario aclarar en primer lugar el contexto del mismo. En el desarrollo de aplicaciones web se utilizan a menudo *frameworks* o marcos de trabajo para ayudar en la lógica de negocio, el acceso a datos, la presentación de la información, etc. Aquellos marcos que facilitan la presentación de los datos son conocidos como "marcos para la vista", haciendo referencia a la parte concreta que resuelven del patrón de diseño web MVC**[14]** (Modelo-Vista-Controlador). Uno de los *frameworks* más utilizados hoy en día para la vista es JSF**[4]** (JavaServer Faces), el cual está basado en el uso de componentes para presentar datos y funcionalidad encapsulada en la parte cliente. En la actualidad, existen numerosas librerías de componentes JSF, cada una con su propio conjunto de componentes, su forma de configuración y modos de uso. En este contexto, se pueden identificar una serie de inconvenientes, los cuales tienen como resultado un problema principal: la pérdida de tiempo ocasionada por el proceso de aprendizaje de configuración y uso de las librerías de componentes JSF. Estos inconvenientes serían, entre otros, la falta de documentación, de centralización de catálogos de componentes y de aplicaciones en blanco o *kickstart* para ayudar al desarrollo.

1.2 Objetivos

El objetivo final del proyecto desarrollado es el de facilitar la elaboración de una página JSF, es necesario que sea sencillo componer una página JSF con los componentes deseados. Como objetivos secundarios al proyecto se destacan los siguientes:

- Agilizar el proceso de selección, inclusión y configuración de componentes JSF; para ello, se necesita facilitar al máximo el manejo de dichos componentes.

- Permitir la incorporación de librerías genéricas de componentes al conjunto inicial de librerías incluidas. Se requerirá la generalización absoluta en cuanto a las librerías que pueden ser importadas dentro del sistema.

- Ser accesible vía web, *Cross-browser*, portable y configurable: el acceso al sistema debe ser realizado a través de un navegador web, siendo la aplicación compatible con los más utilizados actualmente. Se necesita posibilitar su despliegue en diferentes servidores de aplicaciones, siendo completamente independiente del sistema operativo del mismo. Además, ha de ser posible cambiar tanto la apariencia de la aplicación como el conjunto de librerías disponibles para la composición de la página JSF.

1.3 Metodología

Una metodología es una forma, manera o método de llevar a cabo un proceso. En este caso el proceso sería la elaboración del proyecto "Sistema para la maquetación de componentes JSF". La forma o manera de realizarlo sería, *grosso modo*, la expuesta en la Figura 1.

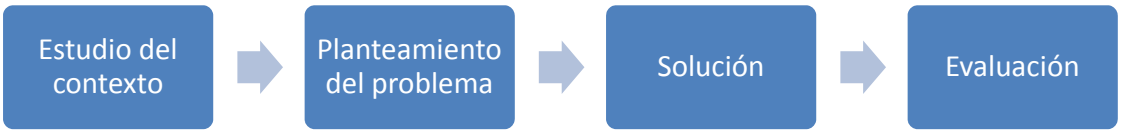

Figura 1. Etapas de la metodología seguida.

A continuación, se va a proceder a explicar en qué consiste y qué se ha realizado en cada una de las etapas mostradas en la Figura 1.

- Estudio del contexto: en esta fase se realiza un análisis del contexto en el que se enmarca el proyecto. Se llevó a cabo, por tanto, un estudio del desarrollo web, su historia y la tendencia que sigue. Se pasó a investigar los marcos de trabajo relacionados con la parte de la vista del patrón de diseño MVC, haciendo especial hincapié en JavaServer Faces y un amplio conjunto de librerías de componentes para él.

- Planteamiento del problema: es la etapa en la que se localiza un problema determinado a partir del contexto estudiado. En el caso de este proyecto, el problema es la pérdida de tiempo en el aprendizaje de la configuración y uso de los componentes de una librería JSF dada.

- Solución: la solución a desarrollar se determinó en base al problema localizado. Se llegó a la conclusión de que implementar un meta-maquetador (maquetador de componentes que utiliza estos mismos en su propia construcción) era la mejor opción, de tal forma que se pudiese demostrar la potencia de JSF. En concreto, el meta-maquetador está desarrollado como una aplicación web. La solución ideada se describe en su totalidad en el capítulo 4.

- Evaluación: en esta fase se prueba que la solución propuesta resuelve correctamente el problema planteado. Para esto se utilizan una serie de métodos y se analizan los resultados obtenidos.

1.4 Estructura del trabajo

Este trabajo se encuentra estructurado en 7 capítulos y 4 anexos, los cuales son descritos a continuación:

- Capítulo 1. Introducción: en este presente capítulo se ha tratado de mostrar una visión general del proyecto, el problema que lo origina, sus objetivos, metodología y estructura.

- Capítulo 2. Estudio del problema: en el siguiente capítulo se explica la situación actual en el entorno del proyecto. Se discute el estado de la cuestión y se define el problema localizado.

- Capítulo 3. Gestión del proyecto software: en este capítulo se discute todo lo relacionado con la gestión de este proyecto, desde el alcance del mismo hasta su evaluación final, pasando por la gestión de riesgos, recursos, análisis y diseño de la solución, etc.

- Capítulo 4. Solución: se encarga de describir la solución propuesta, cómo se ha llevado a cabo el proceso de desarrollo y qué ha sido obtenido al final.

- Capítulo 5. Evaluación: se describe el proceso de evaluación seguido para verificar que la solución desarrollada resuelve satisfactoriamente el problema planteado y se realiza un análisis de los resultados arrojados por dicha evaluación.

- Capítulo 6. Conclusión: este capítulo ofrece una conclusión final sobre el proyecto realizado y la forma en la que resuelve el problema original. También proporciona líneas futuras de ampliación del mismo.

- Capítulo 7. Bibliografía: aquí se incluyen todas las referencias encontradas a lo largo de la memoria, con la información correspondiente sobre la cita reseñada.

- Anexos: se recogerá como anexo la siguiente información: control de versiones (versiones de la presente memoria que fueron siendo generadas, revisadas y aprobadas o rechazadas) y seguimiento del proyecto de fin de carrera (planificación seguida a lo largo de todo el proyecto).

2 Estudio del problema

En este capítulo se va a realizar el estudio del problema. El capítulo tiene como objetivo analizar el entorno en el que se enmarca el presente proyecto, estudiar el contexto del problema y, finalmente, concretar el problema de interés.

2.1 El contexto del problema

El contexto en el que se sitúa el problema planteado y, por tanto, el presente proyecto, es el del desarrollo web. Este tipo de desarrollo es comúnmente confundido con el término *diseño web* por lo que se proponen definiciones para ambos términos:

> **Definición 1.** Diseño web.
> *Área de la creación de un sitio web en el que se diseñan las interfaces de usuario, estilos y demás aspectos de la presentación del mismo. No supone programación de ningún tipo ni en el cliente ni en el servidor.*

> **Definición 2.** Desarrollo web
> *Creación de un sitio web desde el punto de vista funcional: qué hace y cómo lo hace. Se encarga de proporcionar toda la funcionalidad accesible desde la parte cliente, así como todos los datos que requiere la parte de presentación. Consiste casi en su totalidad en programación, tanto en el cliente como en el servidor.*

A fin de concretar el conocimiento sobre el desarrollo web, a continuación se realizará un repaso a la historia del mismo.

Historia del desarrollo web

El nacimiento del desarrollo web tal y como se ha introducido en la sección anterior estuvo altamente condicionado por una mejora significativa de la parte servidor de los sitios web. Se va a introducir en esta sección cómo eran estos sitios web en sus comienzos, de qué forma evolucionaron cuando dicha mejora tuvo lugar y hacia qué están evolucionando hoy en día.

Inicialmente, un servidor web sólo contaba con la funcionalidad de devolver recursos estáticos, esto es, páginas HTML, imágenes, etc. De esta forma, todos los contenidos proporcionados por sitio web tenían que ser introducidos manualmente en el código mismo de la página, con el consiguiente retraso en la actualización de los mismos y la necesidad de un *webmaster* para llevarlo a cabo. No es necesario decir que la calidad de presentación de la información era realmente limitada, tal y como lo era también la navegación. Posteriormente, la tecnología tanto en la parte cliente como en la parte servidor fue mejorando considerablemente, dando lugar a navegadores capaces de mostrar más contenidos y a servidores web capaces de interpretar peticiones y realizar ejecución de programas o procesamiento de lógica antes de

devolver los recursos solicitados al cliente. Es en este punto donde nace el desarrollo web tal y como se ha introducido. Se va a proceder a introducir algunas de esas mejoras en ambas partes.

- Cliente:
  - Sun Java applets **[15]**: se introdujeron animaciones con una cierta lógica que se podía ejecutar en la parte cliente. Esto supuso en su día un cambio revolucionario, pues podían realizarse animaciones más complejas que simples GIFs en el navegador, así como introducir pequeños programas para que el usuario interactuase con ellos. Sin embargo, esta tecnología estaba algo limitada en cuanto a funcionalidades y tampoco proporcionaba demasiada potencia.
  - Macromedia Flash **[16]**: el avance revolucionario en la parte cliente fue la aparición de las películas o animaciones Flash, las cuales imitaban en cierta manera a los applets Java pero gozaban de mayor velocidad, rendimiento y potencia. Consiste, básicamente, en un conjunto de frames, los cuales contienen gráficos y textos, sobre los que se puede ir navegando hacia delante y hacia atrás, saltando en el tiempo, como si de una película se tratase. Aunque inicialmente era algo complejo realizar una animación "condicional" con esta tecnología, hoy en día una película Flash puede constituir una aplicación en sí misma, debido a la potencia y posibilidades que ofrece en sus últimas versiones, siendo capaz de comunicarse con otras películas, páginas, recursos, etc., al tiempo que proporciona su propio lenguaje de programación (ActionScript) para el procesamiento de toda la lógica necesaria. Una de las pocas desventajas del uso de esta tecnología es la necesidad de instalar un plugin en el navegador para poder visualizar películas Flash.
- Servidor:
  - CGI **[17]**: *Common Gateway Interface*, Interfaz de Entrada Común en español, es una tecnología que posibilitó la ejecución de programas residentes en el servidor web a partir de llamadas desde la parte cliente. Esto supuso un avance revolucionario, puesto que ahora lo que devolvía el servidor no era siempre un recurso estático, sino que dependiendo de una entrada previa generaba la salida correspondiente en función de su lógica interna. Al mismo tiempo, se posibilitaba acceso a bases de datos en el servidor y demás servicios que este prestase. Sin embargo, estos programas suponían tremendos agujeros de seguridad para los sitios web, al tiempo que, según avanzaba la historia de la web, no eran capaces de proporcionar toda la funcionalidad que el mercado requería.
  - Servidores de aplicaciones web: pronto se vio la necesidad de proporcionar una mejor gestión de la lógica de la parte servidor de un sitio web, por lo que surgieron los servidores de aplicaciones web. Estos programas viven en el servidor y reciben reenviadas las peticiones hechas por el cliente al servidor web. En ese punto, el servidor web (independientemente de la tecnología/lenguaje que utilice) realiza una serie de operaciones para saber qué hacer con la petición recibida, a qué recursos acceder y qué información devolver al servidor web, para

que este se la pase al cliente. Con este gran avance se consigue dotar a los sitios web de un tremendo poder sobre la navegación, lógica de negocio, acceso a datos, etc.; todo ello resultando en una mejor y más potente experiencia web para el usuario. Un ejemplo de estos servidores de aplicaciones web sería Apache Tomcat, un contenedor de Servlets (tecnología servidor de Sun), el cual posibilita todo lo anteriormente contando utilizando código Java.

Tendencia actual

Por otra parte, se van a introducir ahora varias áreas hacia las que se dirige el desarrollo web hoy en día.

- Patrones de diseño **[18]**: la posibilidad de utilizar código propio en la parte servidor utilizando servidores de aplicaciones web abre las puertas para la aplicación de soluciones comúnmente aplicadas para problemas frecuentes, soluciones que fueron diseñadas y perfeccionadas por especialistas y que permiten a los programadores finales contar con procedimientos confiables en los que basar sus aplicaciones. Estas soluciones son los patrones de diseño, los cuales ya han sido ampliamente aplicados en aplicaciones de escritorio, pasando ahora al contexto de las aplicaciones web. Es necesario comentar la importancia del patrón MVC sobre las aplicaciones web actuales. Este patrón supone la evolución desde una visión clásica de dicha creación web en la que la página (HTML, PHP, etc.) se encargaba de controlar la lógica de negocio, la obtención de los datos y su posterior presentación, hasta una visión innovadora en la que este proceso se separa en sus partes constituyentes: Modelo, Vista y Controlador. La aparición de este patrón posibilita en gran medida la aparición de marcos de trabajo y patrones modulares y diseñados para resolver problemas concretos de cada parte del patrón MVC.

- Marcos de trabajo web: la siguiente evolución de los patrones de diseño y de los problemas frecuentes en la web origina la creación de los marcos de trabajo web. Un marco de trabajo web representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. Engloba el trabajo perfeccionado y revisado de una serie de programadores, en el cual se han utilizado patrones de diseño, otros *frameworks* y mucho trabajo propio, dando lujar a una caja negra, o no tan negra si se quiere, que encapsula un conjunto de herramientas y librerías gracias a las cuales se puede levantar una robusta, completa y compleja aplicación web, sin necesidad de dedicar horas y horas a su programación. Todo ello gracias a la reutilización de código útil y de calidad. Normalmente, los marcos de trabajo web proveen funcionalidades como acceso a base de datos, plantillas, manejo de sesiones, generación de páginas, validación de formularios, control de lógica de negocio, localización de recursos, etc.

- AJAX **[19]**: significa *Asynchronous Javascript and XML*, Javascript y XML asíncrono en castellano. Es un grupo de tecnologías de desarrollo web que existían con

anterioridad pero usadas de manera diferente. En el caso de AJAX, Javascript y XML se utilizan de forma conjunto para realizar una petición el servidor web desde la parte cliente sin tener que actualizar toda la página mostrada en el servidor cuando la respuesta llegue, sino que se actualiza sólo aquella parte deseada. Esto consigue abrir las puertas al desarrollo de aplicaciones web mucho más ricas, capaces de asemejarse a las clásicas aplicaciones de escritorio. Hoy en día multitud de sitios web están incorporando AJAX a sus aplicaciones web para conseguir resultados más dinámicos y potentes y una mejor experiencia web del usuario final.

2.2 El estado de la cuestión

En esta sección se va a hacer un mayor hincapié en el contexto concreto de la cuestión a tratar: el desarrollo de aplicaciones web utilizando marcos de trabajo, en especial aquellos encargados de la vista en el modelo MVC.

Como ya se introdujo en la sección anterior, el patrón de diseño MVC consiste en una separación de las diferentes capas básicas que constituyen toda aplicación web. En primer lugar se puede encontrar el Modelo (M), encargado de encapsular el acceso a datos y la lógica de negocio de la aplicación. En segundo lugar, se tiene la Vista (V), la cual se encarga de presentar correctamente los datos que recibe, sin preocuparse de cómo se consiguen o de dónde proceden. En último lugar se tendría el Controlador (C), responsable de relacionar correctamente las peticiones de datos de la vista al modelo, la navegación entre diferentes vistas, etc.

A continuación se van a describir los marcos de trabajo más utilizados en la actualidad para implementar correctamente el patrón MVC. De cada uno de ellos se proporciona, de manera adicional, una captura de su sitio web. Tras la descripción de estos marcos, se realizará una comparativa de todos ellos. Estos son los marcos, separados según la parte del patrón MVC que resuelven:

- Marcos encargados de la Vista:
  - Velocity: sistema de plantillas basado en Java, alternativa a JSP.
  - Tapestry: divide las páginas de la aplicación en una serie de áreas y componentes.
  - Struts Tiles: se basa en plantillas reutilizables y altamente configurables.
  - JavaServer Faces: se centra en el uso de componentes para formar una página.
- Marcos encargados del Controlador:
  - Struts: recomendado para aplicaciones medias-grandes, sistema de navegación basado en acciones.

- Spring Framework: potente marco de trabajo que incluye Inversión de Control/Inyección de Dependencias.
- JBoss Seam: permite ensamblar complejas aplicaciones utilizando simples POJOs (Plain Old Java Objects). Se integra perfectamente con muchas otras tecnologías.
- Stripes: continúa donde Struts termina, mejorando determinados aspectos y detalles de su predecesor.

- Marcos encargados del Modelo:
  - Hibernate: potente servicio de persistencia altamente configurable y orientado a objetos.
  - TopLink: integra persistencia y transformación a objetos mientras que conserva la atención principal en el problema de dominio.

Marcos encargados de la Vista

En cuanto a los marcos de trabajo utilizados en la parte de la Vista del modelo anteriormente descrito, existen numerosas soluciones. Unas se basan en componentes para construir la vista final, mientras que otras recurren al uso de plantillas para generarla. A continuación se van a introducir algunos de los más comúnmente utilizados.

Velocity

Velocity **[20]** es un sistema de plantillas basado en Java. Permite usar un simple lenguaje de plantillas para referenciar objetos definidos en código Java. Cuando se utiliza para el desarrollo web, Velocity se encarga de desempeñar la función de vista en el patrón MVC. Velocity separa el código Java de las páginas web, haciendo el sitio web mucho más sostenible durante todo su tiempo de vida y proporcionando una alternativa a los ficheros JSP (JavaServer Pages).

Las capacidades de Velocity van más allá de su potencial en la web. Por ejemplo, puede ser utilizado para generar código SQL, PostScript y XML. Puede ser usado también de forma aislada para generar código fuente e informes o como componente integrado en otros sistemas. Proporciona, por ejemplo, un servicio de plantillas para el marco de desarrollo web Turbine, resultando juntos en un motor de vista que facilita el desarrollo web de aplicaciones que realmente siguen el modelo MVC.

Gracias a Velocity Tools se pueden extender las funcionalidades de Velocity. Incluso la propia sintaxis de las plantillas puede ser cambiada y extendida, dado que es procesada en un *parser* generado por JavaCC **[21]**.

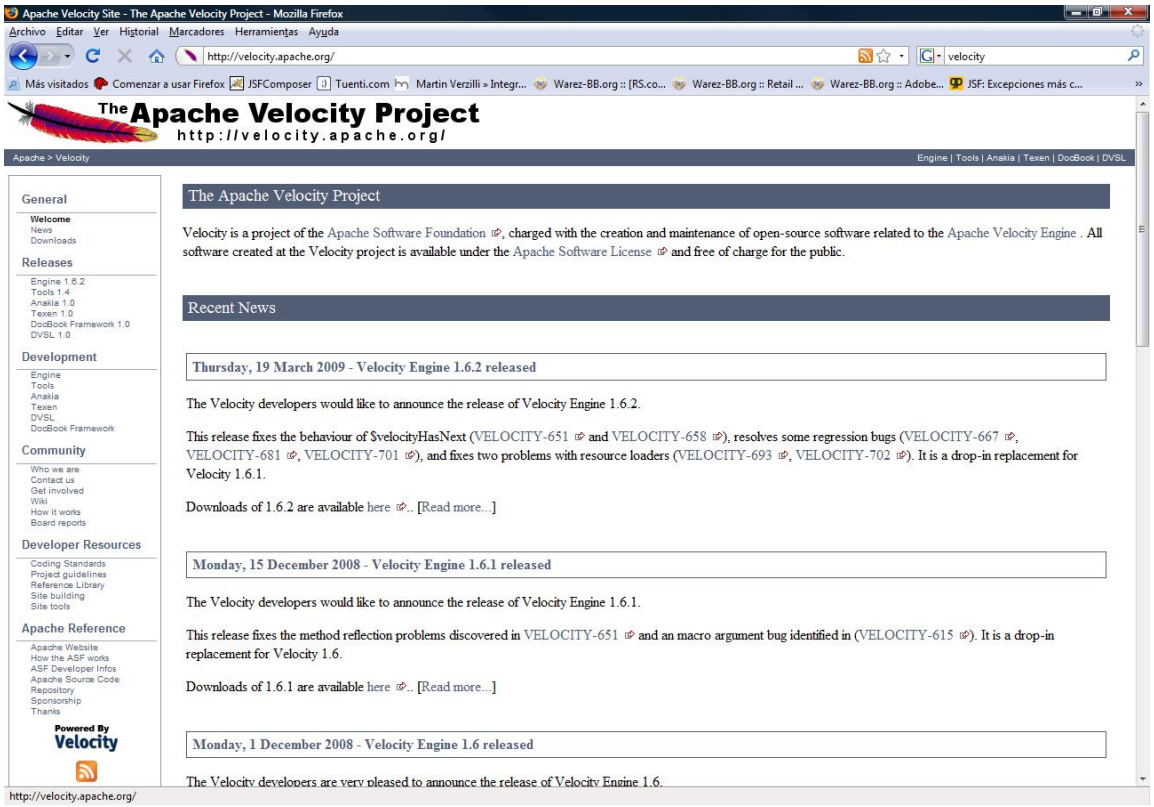

Figura 2. Marco de trabajo Velocity.

Tapestry

Tapestry **[22]** es un marco de desarrollo web para crear aplicaciones web dinámicas, robustas y altamente escalables. Complementa el API estándar de Java Servlet y, por ello, funciona sobre cualquier contenedor de servlets que siga dicho estándar.

Tapestry divide una aplicación web en un conjunto de páginas, cada una construida a partir de componentes. Esto proporciona una estructura consistente, permitiendo al marco de desarrollo asumir la responsabilidad de aspectos clave como la construcción de URLs, localización/internacionalización e informe de excepciones. Desarrollar aplicaciones web con Tapestry involucra crear plantillas HTML con simple código HTML y combinando dichas plantillas con pequeñas porciones de código Java.

La distribución oficial incluye aproximadamente cincuenta componentes, incluyendo desde simples componentes de salida de texto hasta complejas tablas de datos y navegación por árboles. Además, Tapestry está pensado para ser completamente escalable, pudiendo enfrentarse a pequeñas aplicaciones y a grandes proyectos formados por cientos páginas individuales desarrollados por grandes y diversos equipos.

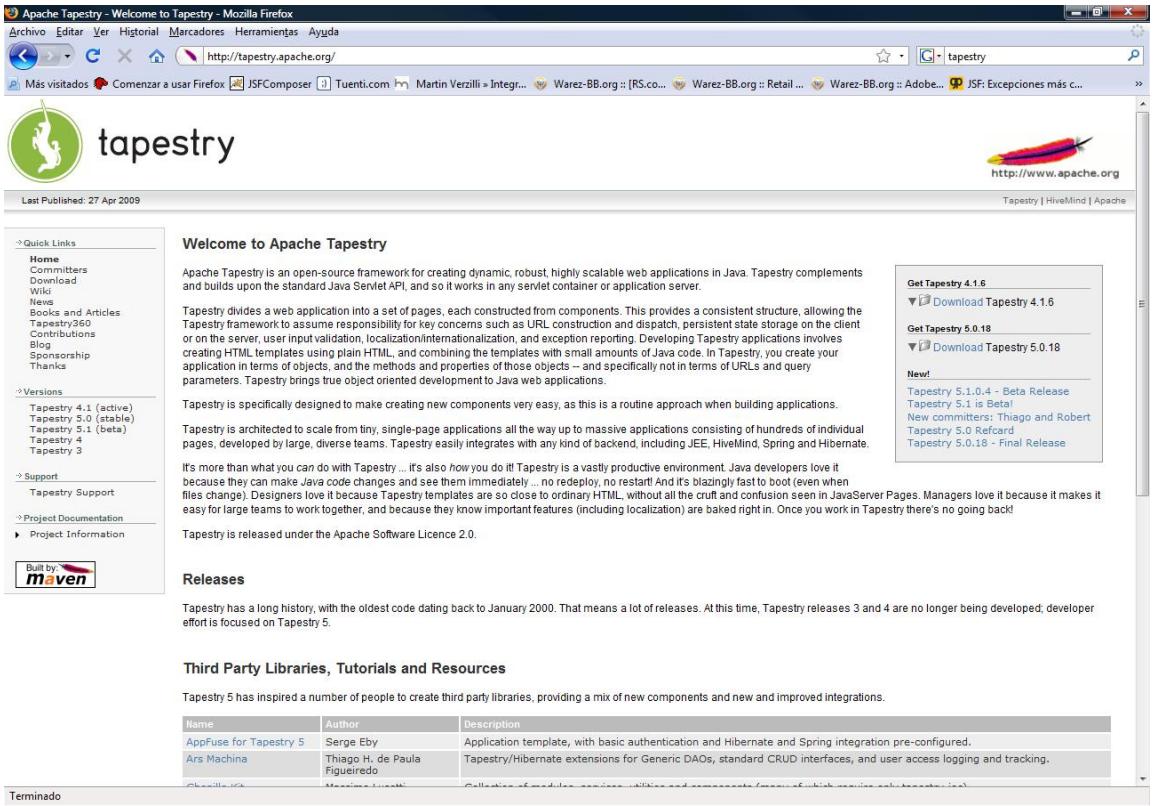

Figura 3. Marco de trabajo Tapestry.

Struts Tiles

El marco de desarrollo Struts Tiles **[24]** se basa en un sistema de plantillas. Estas pueden ser definidas de manera centralizada en un fichero XML, directamente en ficheros JSP o programáticamente. Struts Tiles se integra dentro del ampliamente conocido marco de trabajo Struts, utilizado en la parte del Controlador. Es por esto por lo que las plantillas pueden ser programadas en los propios *Actions* de Struts.

Las plantillas definidas con Struts Tiles son completamente reutilizables, tanto dentro de una aplicación web concreta como entre diferentes aplicaciones. Además, la última versión de Tiles (Apache Tiles 2) consigue independizarse del marco de trabajo Struts, con lo que las posibilidades de reutilización de estas plantillas aumentan considerablemente al ser posible integrarlas con diferentes marcos encargados del Controlador.

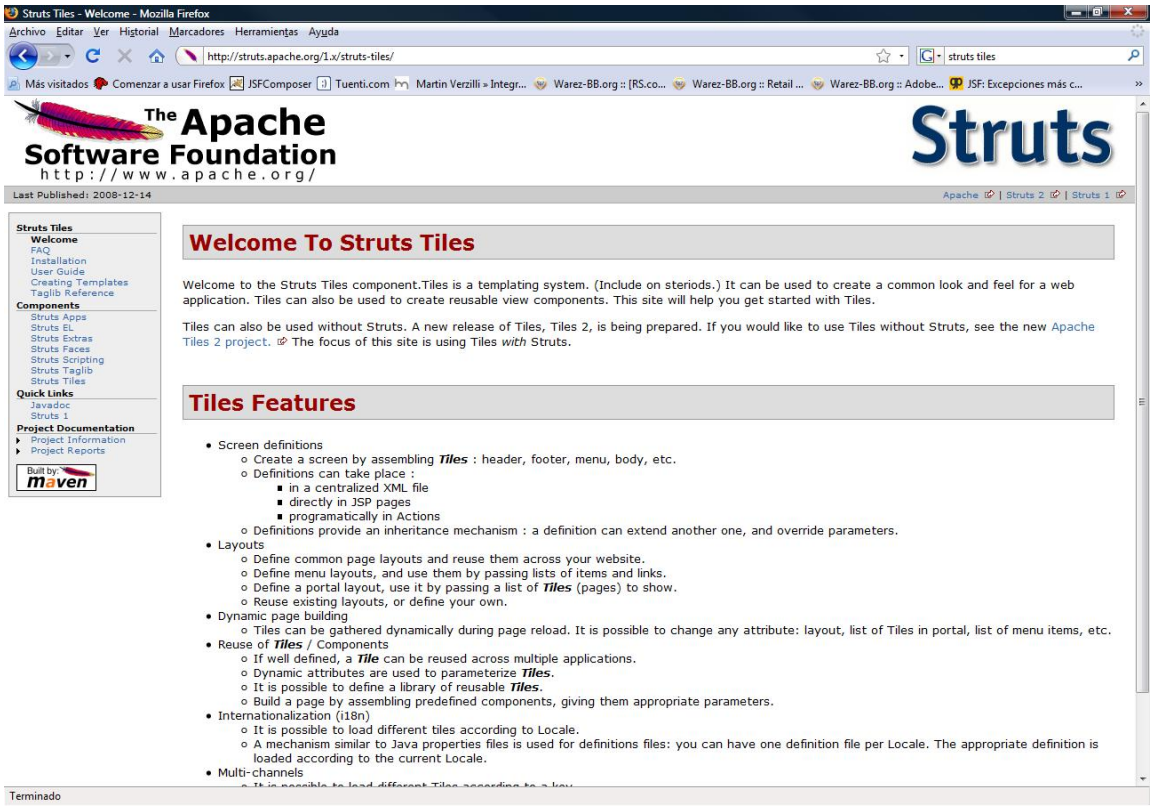

Figura 4. Marco de trabajo Struts Tiles.

JavaServer Faces

JavaServer Faces, JSF de ahora en adelante, es un estándar de Sun Microsystems que especifica cómo debe ser la parte de la Vista de una aplicación web, haciendo que esté basada en componentes. De reciente nacimiento, ha supuesto un gran avance en la manera de concebir la presentación de la información y la interacción con el usuario en la parte cliente.

JSF se basa en el uso de componentes. Estos componentes tienen tanto parte cliente, código HTML resultante de su *renderizado* y código JavaScript asociado, como parte cliente, *beans* manejados que sirven para almacenar y procesar la información recogida o mostrada por dichos componentes.

Es necesario hacer hincapié en la estructura interna de JSF y los productos relacionados existentes en el mercado. JSF consta de dos partes principales: la implementación mínima del estándar de Sun y los componentes JSF listos para usar que se pueden incorporar a la misma. Por un lado, se pueden encontrar implementaciones del estándar perteneciente a Sun Microsystems, como pueden ser Apache MyFaces **[26]** (Figura 5) y la suya propia, Sun JSF RI **[27]** (Figura 6). Por

otro lado, es posible hallar marcos de trabajo que sólo incluyen componentes JSF para incorporar a una determina aplicación web que ya utilice una determina implementación del estándar. Ejemplos de este segundo caso serían JBoss RichFaces **[28]**, IceSoft Technologies IceFaces **[30]**, Vedana Rich Client Faces **[31]**, Oracle ADF Faces **[32]**, Apache MyFaces Tomahawk **[33]**, etc.

Además, JSF también permite definir la navegación entre las diferentes vistas o páginas de que consta una aplicación web, utilizando para ello etiquetas de navegación.

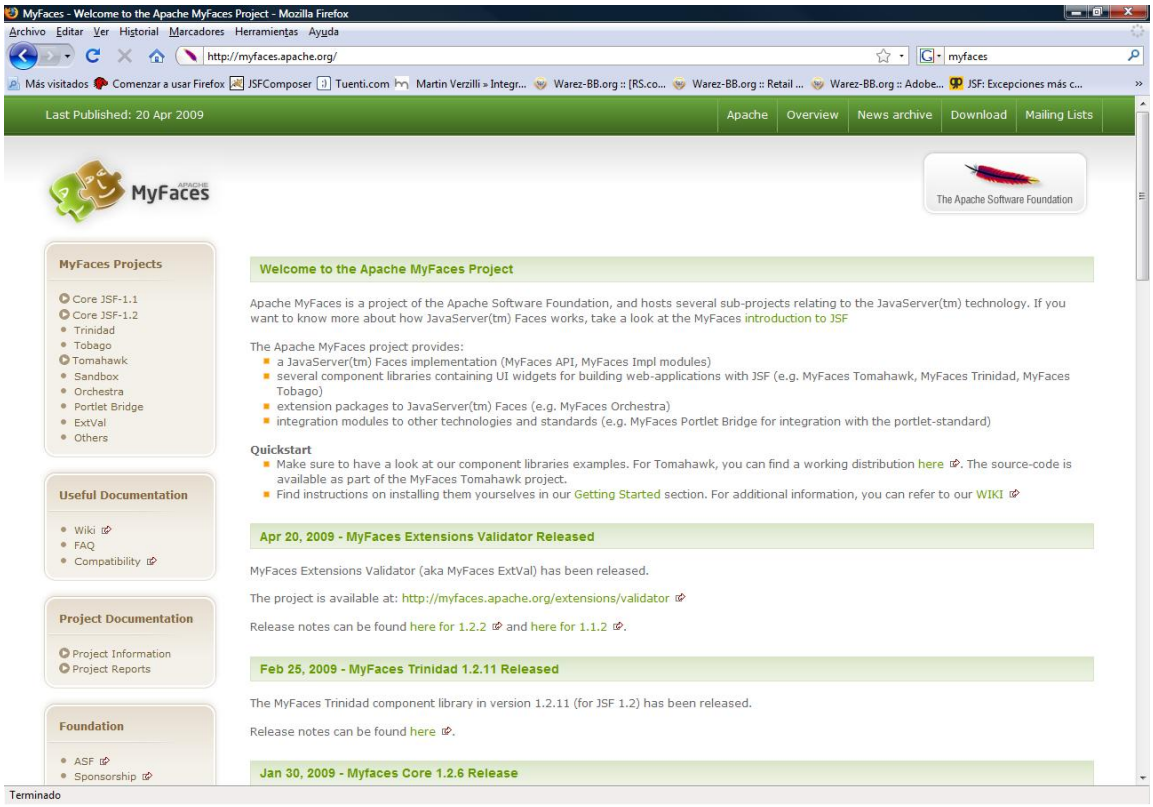

Figura 5. Marco de trabajo JavaServer Faces: implementación de MyFaces.

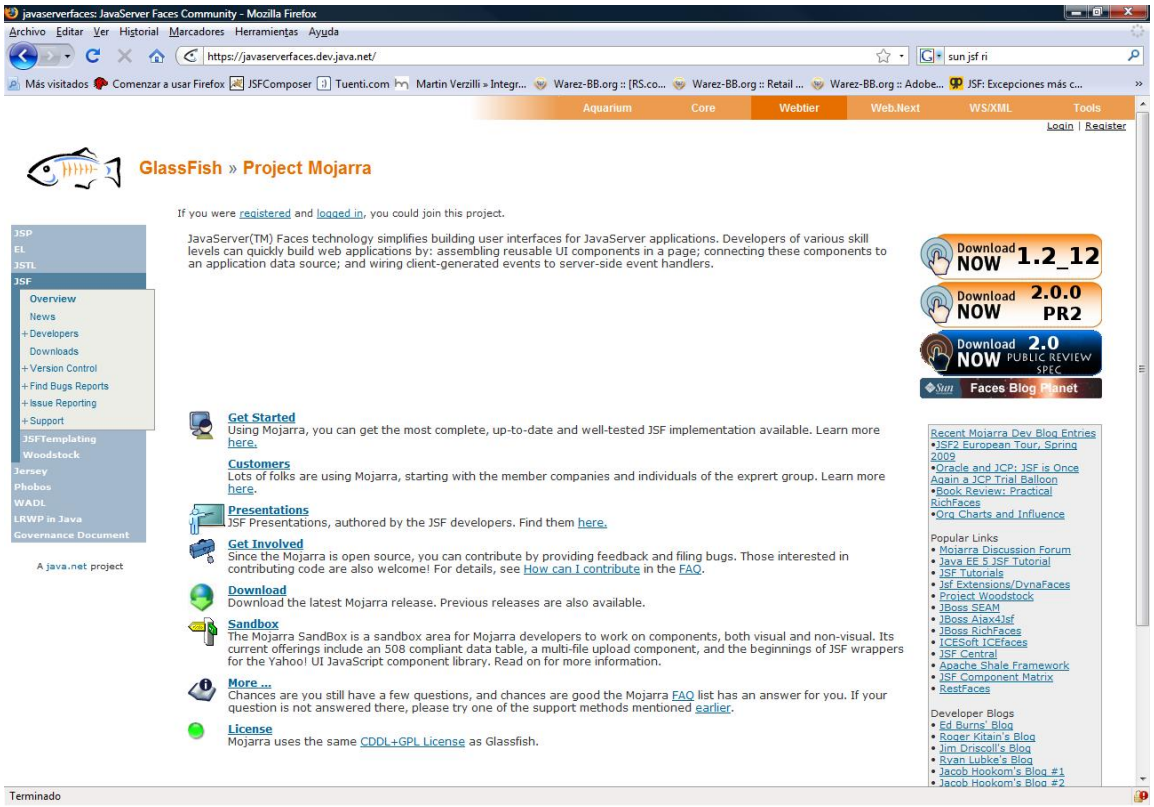

Figura 6. Marco de trabajo JavaServer Faces: implementación de Sun.

Marcos encargados del Controlador

Respecto a los marcos de trabajo utilizados en la parte del Controlador, se encuentran Struts, Spring Framework, JBoss Seam y Stripes, los cuales son descritos a continuación.

Struts

Struts **[23]** es un marco de desarrollo web altamente establecido en lo que a aplicaciones web se refiere. Es un referente considerable a la hora de crear este tipo de aplicaciones. Este marco de desarrollo debería ser la elección por defecto a la hora de acometer un proyecto de este tipo.

Se encarga de toda la arquitectura del patrón MVC, sirviendo principalmente de controlador pero encargándose de modelo y vista, si bien puede delegarlas en otros marcos de desarrollo. La configuración básica pasar por la edición del fichero struts.xml para agregar nuevos Actions a los que referenciar desde los ficheros JSP.

El marco de desarrollo web Struts está, principalmente, recomendado para aplicaciones medias-grandes, dado que para las más simples o pequeñas exige un proceso de configuración realmente innecesario.

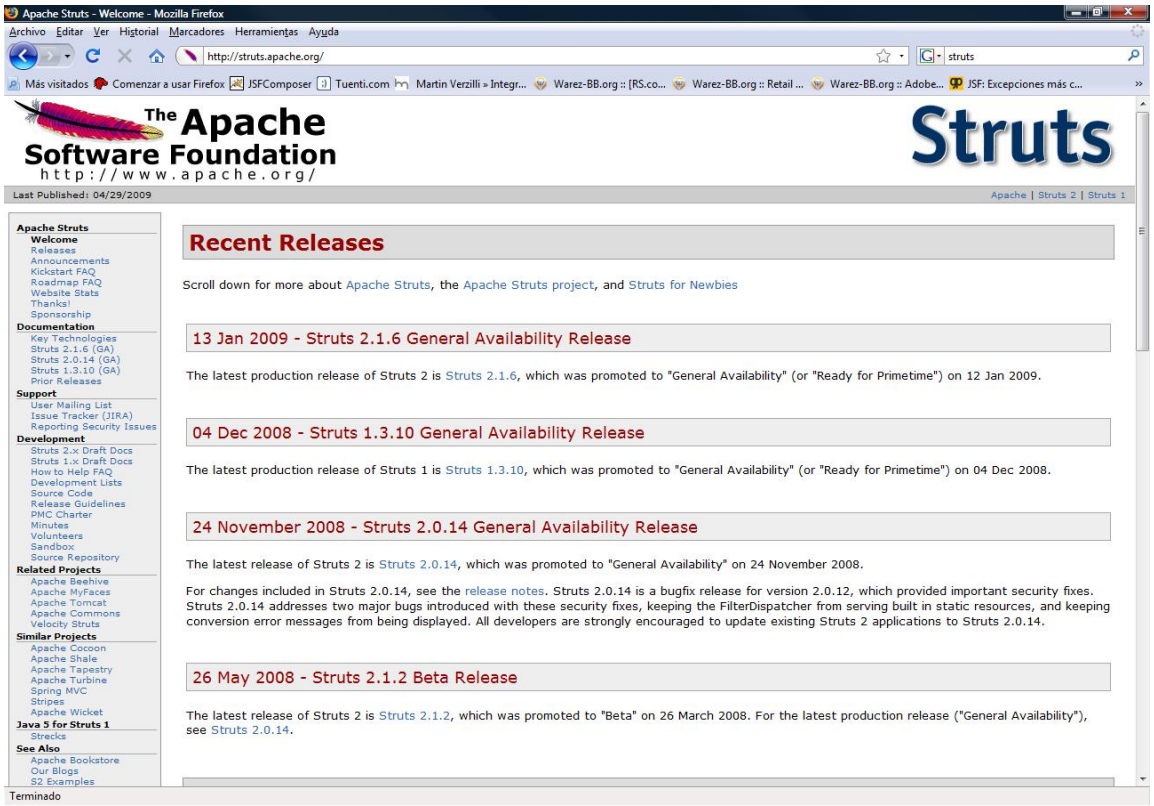

Figura 7. Marco de trabajo Struts.

Spring Framework

Como gran competidor de Struts en la actualidad, puede encontrarse Spring Framework **[34]**. Su principal punto innovador y revolucionario es la Inversión de Control o Inyección de Dependencias, gracias a la cual puede reconfigurarse una aplicación media-grande, incluso desde el punto de vista de mapping o interconexión de componentes, simplemente editando sus numerosos ficheros de configuración. Precisamente este es también uno de sus puntos débiles, pues para poder sacar provecho de esta novedosa técnica es necesario aprender una cantidad considerable de conocimientos sobre dichos ficheros de configuración, lo que repercute directamente en la curva de aprendizaje.

Sin embargo, la Inversión de Control posibilita la interconexión de Spring con una gran multitud de marcos de desarrollo encargados de la vista o modelo, debido a la facilidad de reconfiguración inherente a esta Inyección de Dependencias.

Spring Framework también trata de facilitar la tarea del programador mediante la simplificación de la programación de beans y servlets. Su esfuerzo se canaliza a través de clases singleton y una clara orientación al uso de interfaces sobre el de clases.

Por tanto, Spring Framework, tras un periodo de aprendizaje, consigue una capacidad de integración excelente y de reconfiguración muy buena.

Figura 8. Marco de trabajo Spring.

JBoss Seam

JBoss Seam **[39]** es un potente y novedoso marco de desarrollo web pensado para crear aplicaciones de nueva generación (Web 2.0) unificando e integrando tecnologías como AJAX, JSF, EJB3, Java Portlets y Business Process Management.

Seam ha sido diseñado para eliminar la complejidad en la arquitectura de una aplicación web. Posibilita que los desarrolladores ensamblen complejas aplicaciones web con simples POJOs (Plain Old Java Objects), componentes de interfaz de usuario y XML.

Utiliza JavaServer Faces o cualquiera de sus variantes para renderizar la vista y puede integrarse con Hibernate para controlar la parte del modelo.

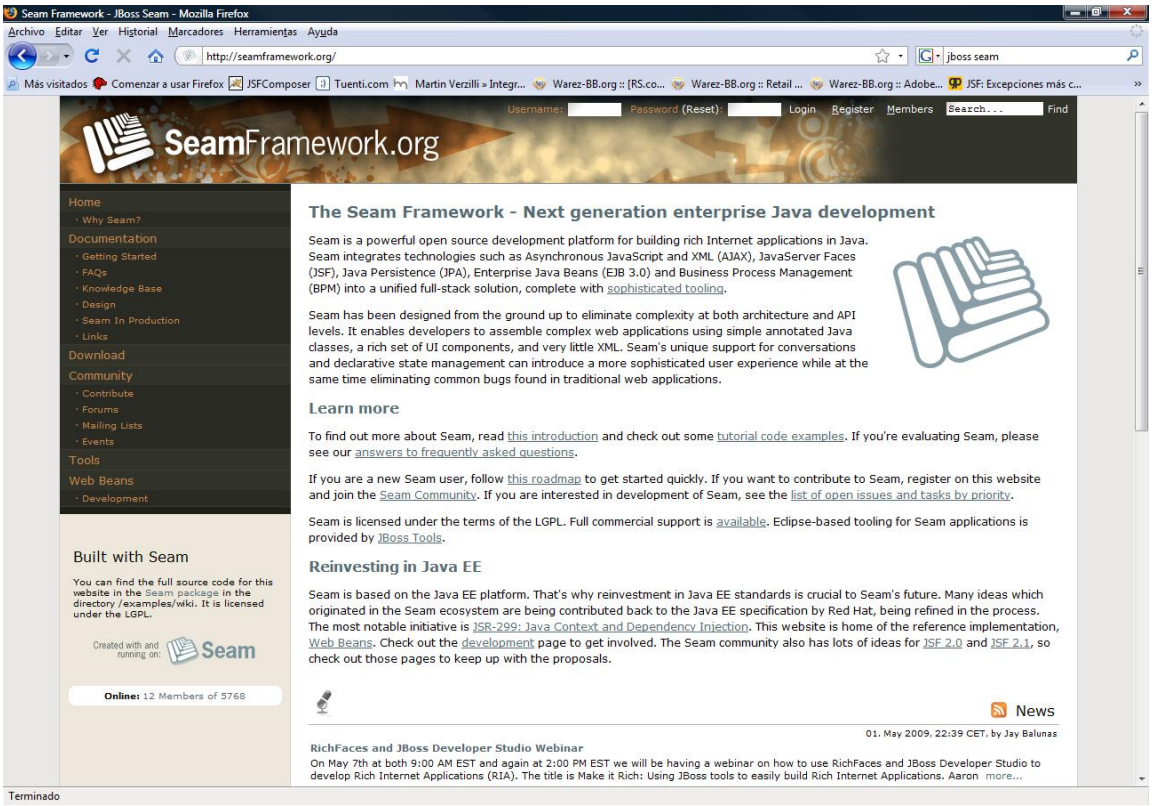

Figura 9. Marco de trabajo Seam.

Stripes

Stripes **[40]** es una versión de Struts simplificada. Conserva parte de la estructura del segundo pero facilita un sinfín de pequeñas tareas y errores o formas no demasiado correctas de hacer las cosas que los desarrolladores de Struts no consideraron como tal. Entre otros, cabe citar problemas de validación, necesidad de explicitar los actions en el fichero de configuración, etc.

Además, Stripes también aporta un conjunto nuevo de funcionalidades bastante interesante: transparencia en la subida de ficheros, manejo de errores y excepciones, validación, pruebas unitarias, manejo de estados y la posibilidad de integrar AJAX de forma sencilla en los componentes existentes para la parte de la vista.

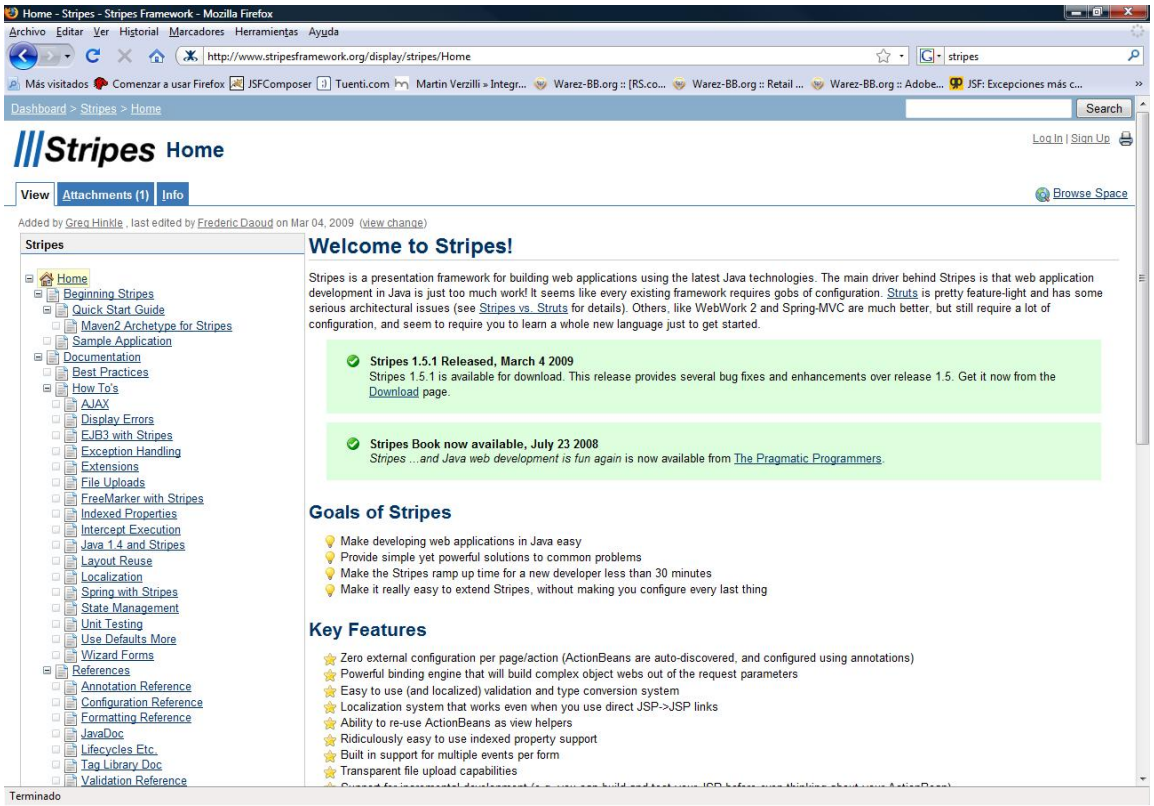

Figura 10. Marco de trabajo Stripes.

Marcos encargados del Modelo

A continuación se describen los marcos encargados del Modelo seleccionados.

Hibernate

Hibernate **[35]** es un potente y altamente eficiente servicio de persistencia objeto/relacional y de consulta. Este marco de desarrollo permite desarrollar clases persistentes siguiendo una especificación orientada a objectos, incluyendo asociación, herencia, polimorfismo, composición y colecciones. Hibernate permite expresar consultas tanto en su propio SQL portable (HQL) como en SQL nativo.

Al contrario que otras soluciones para la persistencia, Hibernate no oculta el poder de SQL y garantiza que los conocimientos y esfuerzos previos del programador en tecnología relacional sean tan válidos como siempre.

Hibernate es un proyecto profesional y un componente crítico de la suite de productos de JBoss, una división de Red Hat.

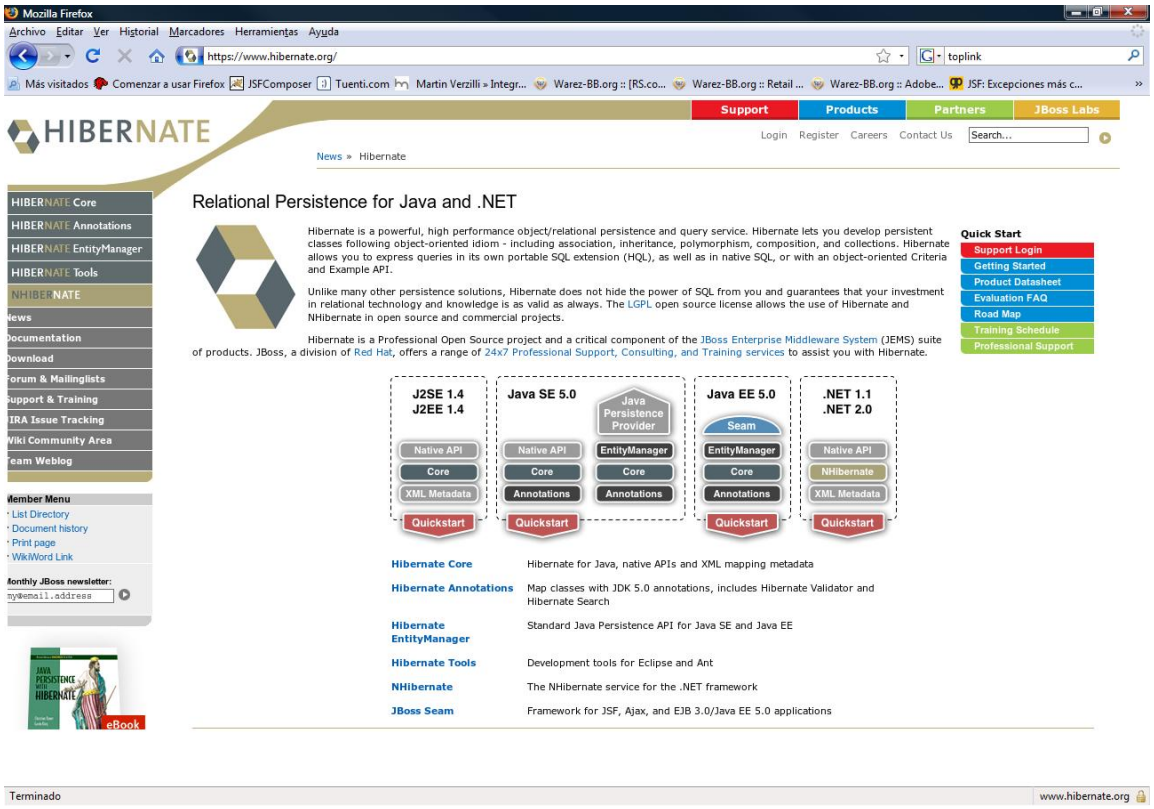

Figura 11. Marco de trabajo Hibernate.

TopLink

Con Oracle TopLink **[41]**, se puede integrar la persistencia y la transformación de objetos en una aplicación web, mientras se conserva la atención principal en el problema de dominio. Gracias a esto se consigue ventaja en eficiencia, flexibilidad y soluciones probadas sobre el terreno.

El uso de TopLink es apropiado para un amplio rango de aplicaciones J2EE y arquitecturas Java. Se puede usar para diseñar, implementar, desplegar y optimizar una capa avanzada, con persistencia y transformación de objetos que soporta una gran variedad de fuentes de datos y formatos, incluyendo relacional (JDBC), objeto-relacional (Oracle Database), Sistema de Información Empresarial (J2C) y XML.

TopLink incluye soporte para aquellos contenedores con persistencia manejada por contenedor (CMP) de una variedad de distribuidores, como Oracle Containers for J2EE (OC4J), IBM WebSphere application Server y BEA WebLogic Server.

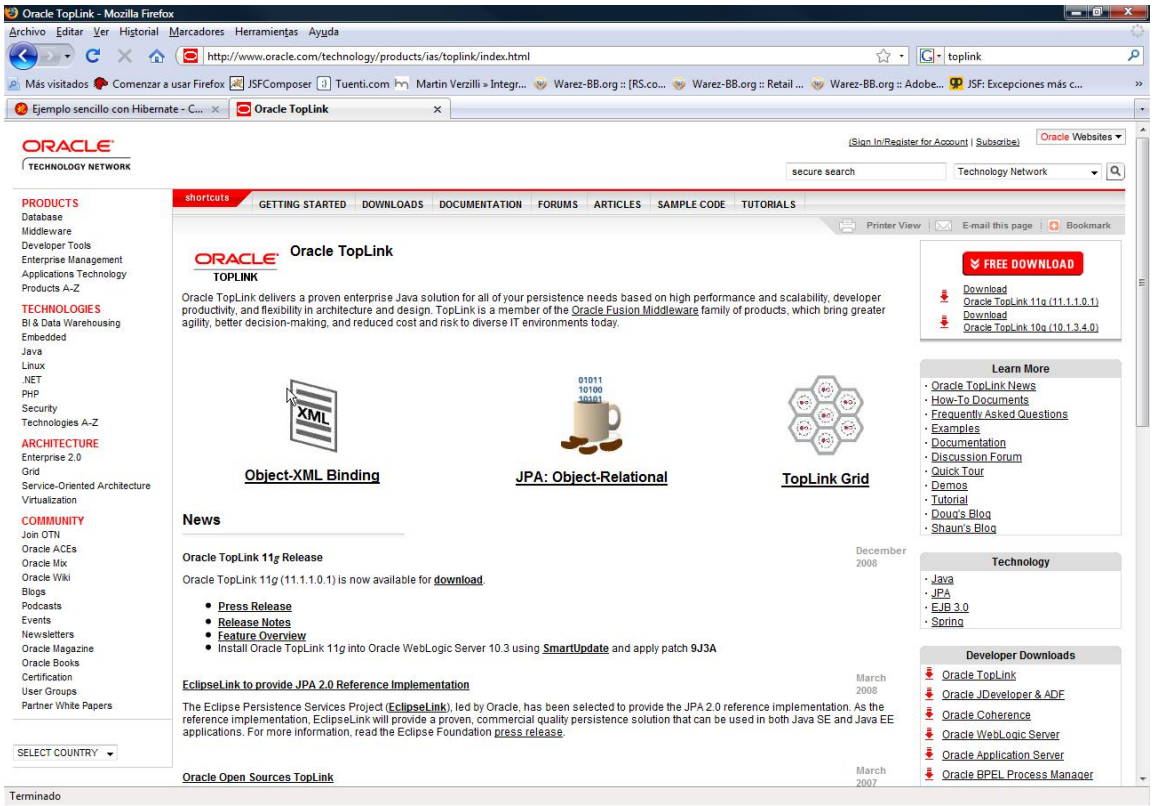

Figura 12. Marco de trabajo TopLink.

Comparativa de marcos de trabajo

En esta sección se van a comparar los marcos de trabajos que han sido introducidos. Esta comparativa es importante para entender la razón de que este proyecto utilice las tecnologías elegidas en lugar de utilizar otras.

Los criterios de comparación son los siguientes:

- Cantidad de documentación disponible: incluyendo recursos externos al marco o a su equipo de desarrollo. Se mide en valores numéricos del 1 al 5, siendo 1 poca documentación y 5 mucha.

- Facilidad de aprendizaje: lo que cuesta aprender a utilizar el marco desde cero, sin conocimientos previos. Se mide en valores numéricos del 1 al 5 también, siendo 1 poca facilidad (mucha dificultad) y 5 mucha.

- Facilidad de instalación y configuración: indica el nivel de facilidad a la hora de incorporar el marco de trabajo a un proyecto dado y configurar el mismo. También se mide del 1 al 5, siendo 1 poca facilidad (mucha dificultad) y 5 mucha.

La comparativa de los marcos de trabajo quedaría, por tanto, resumida en las tablas Tabla 1, Tabla 2 y Tabla 3, que están referidas, respectivamente, a los marcos encargados de la Vista, del Controlador y del Modelo:

	Velocity	Tapestry	Struts Tiles	JavaServer Faces
Cantidad de documentación	1	4	4	5
Facilidad de aprendizaje	2	4	4	4
Facilidad de instalación y configuración	4	5	5	5

Tabla 1. Comparativa de marcos de trabajo para la Vista.

	Struts	Spring	Seam	Stripes
Cantidad de documentación	5	5	3	5
Facilidad de aprendizaje	4	4	3	4
Facilidad de instalación y configuración	5	4	3	4

Tabla 2. Comparativa de marcos de trabajo para el Controlador.

	Hibernate	TopLink
Cantidad de documentación	5	4
Facilidad de aprendizaje	4	3
Facilidad de instalación y configuración	5	3

Tabla 3. Comparativa de marcos de trabajo para el Modelo.

2.3 La definición del problema

Una vez introducidos el contexto del problema y el estado de la cuestión se puede proceder a la definición del problema. El problema localizado viene dado por un uso frecuente y de nivel avanzado de diferentes librerías de componentes JSF. Este uso frecuente, junto al análisis realizado en las secciones anteriores, exponen el problema en cuestión.

El problema encontrado es la alta dificultad de aprendizaje y uso de librerías de componentes JSF. Esto se explica analizando el proceso de incorporación de una librería dada, el cual se compone de las etapas mostradas en la Figura 13.

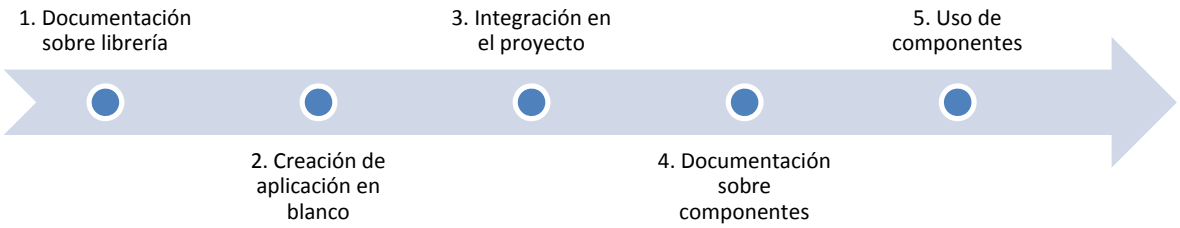

Figura 13. Etapas de la incorporación de una librería JSF.

1. Documentación sobre librería: en primer lugar es necesario documentarse sobre la librería en cuestión, estudiar si es compatible con la configuración actual del proyecto real, analizar las ventajas de su utilización, aprender cómo se configura e instala, etc.

2. Creación de aplicación en blanco: en segundo lugar es importante realizar la típica aplicación *blank* o *kickstart*, que es aquella que es totalmente carente de funcionalidad pero que consigue demostrar que la configuración mínima para utilizar la librería funciona correctamente y no ocasiona problemas al ser desplegada la misma.

3. Integración en proyecto: una vez probada con la aplicación en blanco, se procede a la integración de la librería en el proyecto. Para esto se traslada la configuración mínima de la etapa anterior a la configuración actual del proyecto, tomando las consideraciones oportunas para evitar que dichas modificaciones dejen el proyecto en un estado no operativo.

4. Documentación sobre componentes: el cuarto paso consiste en estudiar y analizar el conjunto de componentes de la librería que se desee incorporar al proyecto, prestando especial atención a la manera de configurarlos.

5. Uso de componentes: el paso final es incorporar satisfactoriamente estos componentes seleccionados, conociendo previamente cuáles son sus propiedades/atributos y cómo se deben incluir en el código fuente del proyecto.

Este proceso tiene numerosos defectos o efectos negativos causados por las etapas siguientes:

- Documentación: en primer lugar, se pierden horas y horas en documentarse apropiadamente para poder crear una aplicación de prueba o para siquiera ver algún componente funcionando. Además, no siempre existe toda la documentación deseada para una librería determinada ni los recursos online requeridos.

- Aplicación en blanco: en segundo lugar, se pierde un valioso tiempo de desarrollo implementando la aplicación en blanco, dado que hay que recrear un nuevo entorno con condiciones ideales y simplificadas que nada se asemejará al entorno final del proyecto.

- Uso de componentes: por último, todo ese proceso de documentación y creación de la aplicación *kickstart* no exime a los programadores finales de recurrir a métodos de ensayo y error para evaluar la compatibilidad o viabilidad de uso de los componentes deseados en el entorno del proyecto real.

Por todo esto se considera que la tarea de agregar una librería de componentes JSF a un proyecto existente es bastante ineficiente y debería ser optimizada.

3 Gestión de proyecto software

La gestión de proyectos es una herramienta muy útil a la hora de asegurar la correcta realización de un proyecto. Según Wikipedia **[8]**:

"La gestión de proyectos es la disciplina de organizar y administrar recursos de manera tal que se pueda culminar todo el trabajo requerido en el proyecto dentro del alcance, el tiempo, y coste definidos. Un proyecto es un esfuerzo temporal, único y progresivo, emprendido para crear un producto o un servicio también único."

En el caso del presente proyecto, se amplía el significado de los citados recursos para englobar también el conjunto de tareas en las que se descompone un proyecto. Adicionalmente, se incluye en este capítulo el análisis de los riesgos que pueden producirse durante la realización del proyecto.

3.1 Alcance del proyecto

El alcance del proyecto está directamente ligado a los objetivos marcados y a las condiciones impuestas. En esta sección se van a delimitar estos objetivos y condiciones, así como los recursos necesarios para cumplir ambos.

Definición del proyecto

Este proyecto tiene como objetivo desarrollar una herramienta de maquetación de componentes JSF. De esta forma, el usuario podrá componer una página JavaServer Faces de manera visual utilizando componentes de librerías que no tiene necesidad de conocer y sin emplear para ello línea alguna de código. Dicha herramienta de maquetación será accesible vía web, por lo que se facilita en gran manera el acceso a la misma, no siendo necesario ningún proceso de instalación en el equipo del usuario.

La información de entrada al sistema será el conjunto de librerías de componentes que el administrador considere oportuno dejar disponibles al usuario final. A su vez, la salida del proceso de maquetación realizado por este último será una página JSF en la que se encuentren codificadas las etiquetas correspondientes a los componentes dispuestos por el citado usuario, además de las librerías que hayan sido empleadas y los ficheros de configuración pertinentes.

De manera adicional, se le permite al usuario modificar a su elección determinados elementos del sistema, las librerías disponibles, información de salida, etc.

Este proceso de maquetación asistida se hace necesario debido a los inconvenientes del modelo de aprendizaje de librerías JSF que se sigue hoy en día. Esto ha sido debidamente explicado en la sección 2.3, "La definición del problema".

Estimación de tareas y recursos

En este apartado se va a exponer la estimación realizada en cuanto al coste del proceso completo de desarrollo de este proyecto. Los distintos tipos de gastos que se van a tener en cuenta son los siguientes:

- Personal: gastos relativos a los sueldos de los diversos empleados necesarios.
- Equipamiento: requisitos hardware y software para el desarrollo del proyecto.
- Material: referente a la infraestructura requerida.
- Transporte: gastos relacionados con los viajes asociados a reuniones con el cliente.
- Otros: conjunto de gastos que no que encajan en ninguna de las categorías anteriores.

Antes de comenzar con el desglose de cada categoría es necesario definir varios valores imprescindibles en toda planificación de desarrollo:

Horas laborables al día	8
Días laborables al mes	22
Fecha de inicio del proyecto	15/01/2009
Fecha de fin del proyecto	04/05/2009
Duración del proyecto	3,5 meses

Tabla 4. Definición de valores clave para la planificación.

Personal

Para realizar el cálculo del gasto de emplear los trabajadores necesarios se requiere conocer su salario anual y las horas trabajadas por cada uno, viniendo este dato de la planificación que más adelante se expondrá. Los roles necesarios son:

- 1 Jefe de proyecto
- 1 Analista
- 2 Programadores

Por tanto, este es el resumen de gastos por rol de trabajador:

Rol	Salario bruto anual (€)	Salario bruto mensual (12 pagas) (€)	Coste por hora (€/h)	Horas trabajadas (h)	Coste total (€)
Jefe de proyecto	55.000	4.583,34	26,05	219	5.704,95
Analista	42.000	3.500	19,89	534	10.621,26
Programador	36.000	3.000	17,05	552	9.411,6

Tabla 5. Coste por rol.

A continuación se muestra el desglose de horas trabajadas por cada empleado en cada una de las tareas que componen el desarrollo:

Tareas	Jefe de proyecto	Analista	Programador 1	Programador 2	Horas totales
Estudio preliminar					**34**
Planteamiento del problema y esbozo de la solución	12	6	0	0	18
Elaboración del Plan de proyecto	16	0	0	0	16
Estado de la cuestión					**85**
Estudio de las tecnologías disponibles	10	30	0	0	40
Estudio de las librerías de componentes JSF	10	35	0	0	45
Análisis del sistema					**105**
Definición de requisitos	30	20	0	0	50
Especificación de requisitos	35	20	0	0	55
Diseño					**223**
Diseño de la arquitectura	9	24	0	0	33
Diseño del modelo de datos	4	15	0	0	19
Diseño de la interfaz	7	25	0	0	32
Diseño de la gestión de componentes JSF	28	45	0	0	73
Diseño de la lógica de negocio	20	46	0	0	66

Tareas	Jefe de proyecto	Analista	Programador 1	Programador 2	Horas totales
Desarrollo de la aplicación					**1.135**
Desarrollo de la interfaz	0	42	126	126	294
Desarrollo del dominio	0	15	80	80	175
Generación de componentes JSF	0	20	77	77	174
Gestión de librerías de componentes JSF	0	23	83	83	189
Desarrollo de la lógica de negocio	0	37	133	133	303
Pruebas					**96**
Pruebas unitarias	3	9	16	16	44
Pruebas de integración	4	8	5	5	22
Ejecución de pruebas	6	10	7	7	30
Despliegue de la aplicación					**20**
Despliegue del sistema	8	12	0	0	20
Mantenimiento					**65**
Mantenimiento del sistema	6	9	25	25	65
Documentación					**94**
Documentación del proyecto	11	83	0	0	94
Total horas por rol	**219**	**534**	**552**	**552**	**1.857**
Total coste por rol	**5.704,95 €**	**10.621,26 €**	**9.411,6 €**	**9.411,6€**	**35.149,41 €**

Tabla 6. Desglose de horas trabajadas por rol y por tarea.

Tal y como se indica al final de la Tabla 6, el coste total por empleados es de **35.149,41 euros**.

Equipamiento

Para que los anteriormente citados empleados puedan realizar su trabajo, es necesario que cuenten con un entorno de desarrollo apropiado. Una parte clave de él son los equipos informáticos que se ponen a su disposición. Dado que todo el software a utilizar en el desarrollo es completamente *open-source* (servidor Apache Tomcat y *frameworks* con licencia GPL), el coste de éste es nulo. También se incluye en esta categoría el material fungible utilizado (papel, cartuchos de tinta, bolígrafos, etc.). Este sería, pues, el resumen de gastos de equipamiento:

Descripción	Coste (€)	Cantidad	Amortización (años)	Coste total (€)
Ordenador de sobremesa de gama alta	1.000	4	2	500
Servidor propio	2.000	1	2	1.000
Software	0	4	-	0
Material fungible	40	4	-	160
Total				**1.660**

Tabla 7. Coste de equipamiento

Nótese que los equipos hardware se amortizan pasado un tiempo, de ahí que el coste imputable por ellos sea menor que el coste original.

Material

La infraestructura necesaria para el equipo de desarrollo designado consta de una oficina para realizar el mismo, conexión telefónica y a Internet, y servicio de limpieza y mantenimiento de las instalaciones:

Descripción	Coste (€/mes)	Cantidad (meses)	Coste total (€)
Alquiler oficina	1.500	3	4.500
Conexión telefónica y a Internet	40	3	120
Servicio de limpieza y mantenimiento	200	3	600
Total			**5.220**

Tabla 8. Coste de material.

Transporte

Otro aspecto a tener en cuenta en el cómputo de los gastos del proyecto son los relacionados con el transporte. Ya sea por reuniones con el cliente o por sesiones de implantación, los desplazamientos del equipo de desarrollo son inevitables:

Descripción	Coste total (€)
Transporte público y privado	600

Tabla 9. Coste de transporte.

Otros

De manera adicional, es necesario considera en todo proyecto una cantidad extra para cubrir posibles imprevistos durante el desarrollo:

Descripción	Coste total (€)
Costes adicionales	200

Tabla 10. Costes adicionales.

Presupuesto

Con todos los gastos listados en las tablas anteriores, este sería el coste total del proyecto:

Concepto	Coste (€)
Empleados	35.149,41
Equipamiento	1.660
Material	5.220
Transporte	600
Gastos adicionales	200
Total	**42.829,41**

Tabla 11. Resumen de gastos.

Ahora, con los porcentajes de riesgos, beneficio e IVA, se puede elaborar el presupuesto final de la realización del proyecto:

Concepto	Coste (€)
Gastos totales	42.829.41
Riesgos (20%)	8.565,88
Beneficio (20%)	8.565,88
Total (sin IVA)	59.961,17
IVA**[11]** (16%)	9.593,79
Total	**69.554,96**

Tabla 12. Presupuesto final.

Por tanto, tal como establece la Tabla 12, el coste total de desarrollo del proyecto para la empresa encargada (sin IVA) es de **59.961,17 € (Cincuenta y nueve mil novecientos sesenta y un euros y 17 céntimos de euro).**

La cantidad económica a desembolsar (con IVA) por la empresa contratante será de **69.554,96 € (Sesenta y nueve mil quinientos cincuenta y cuatro euros y noventa y seis céntimos de euro).**

3.2 Plan de trabajo

Una vez detallado el alcance, tiempo total de desarrollo y presupuesto del proyecto, se van a identificar las tareas en las que se descompone dicho desarrollo. Además, se va a incluir la planificación de las mismas en el tiempo indicado.

Identificación de tareas

Para conseguir afrontar el desarrollo de este proyecto de forma más ordenada y eficaz, se ha dividido el mismo en tareas más simples y manejables:

- Estudio preliminar:
  - Planteamiento del problema y esbozo de la solución: en primer lugar, es necesario especificar el problema que se pretende resolver con la realización de este proyecto y proporcionar una primera impresión de la solución que se quiere alcanzar.
  - Elaboración del Plan de proyecto: en esta fase se elaborará la gestión del proyecto, detallando la planificación, división de tareas y seguimiento del proyecto, así como el alcance del mismo y el coste de su desarrollo.
- Estado de la cuestión:
  - Estudio de las tecnologías disponibles: análisis de las tecnologías en actual uso en el ámbito del desarrollo de aplicaciones web.
  - Estudio de las librerías de componentes JSF: análisis detallado de las distintas librerías de componentes disponibles hoy en día para componer páginas JSF.
- Análisis del sistema:
  - Definición de requisitos: educción de requisitos impuestos por el cliente y clasificación de los mismos en diferentes categorías.
  - Especificación de requisitos: estudio más detallado de las características asociadas a cada requisito definido.

- Diseño: dado el análisis resultante de la fase anterior, se realiza el esbozo de la solución a desarrollar. Para ello, esta fase se divide de tal forma que se afronten los principales aspectos de toda aplicación, más aquellos propios del contexto concreto de este proyecto:
  - Diseño de la arquitectura: se diseñará la arquitectura general de la aplicación, estableciendo separaciones en grandes módulos.
  - Diseño del modelo de datos: se establecerá el modelo de datos que utilizará todo el sistema.
  - Diseño de la interfaz: se diseñará la interfaz de usuario que utilizará el cliente final.
  - Diseño de la gestión de componentes JSF: se generará el diseño del módulo de gestión de componentes JavaServer Faces.
  - Diseño de la lógica de negocio: se diseñará la lógica de negocio central de la aplicación.
- Desarrollo de la aplicación: esta fase se refiere al desarrollo propiamente dicho de los diferentes bloques que componen la aplicación:
  - Desarrollo de la interfaz: se construirá la interfaz de usuario con la que interactuará el usuario final.
  - Desarrollo del dominio: se desarrollará el código que representa la información del sistema.
  - Generación de componentes JSF: en esta fase se creará el código necesario para generar y organizar componentes JavaServer Faces de manera dinámica.
  - Gestión de librerías de componentes JSF: se generará el código encargado de gestionar todas las librerías y componentes disponibles en el sistema.
  - Desarrollo de la lógica de negocio: se realizarán las clases encargadas de gestionar las peticiones originadas en la interfaz de usuario.
- Pruebas:
  - Pruebas unitarias: se diseñarán las diferentes pruebas unitarias a realizar. Estas pruebas consisten en probar el correcto funcionamiento de un módulo de código dado.
  - Pruebas de integración: se planificarán las pruebas de integración del sistema diseñadas. Estas pruebas se centran en comprobar que los diferentes módulos

que componen un subsistema se integran correctamente, con el consiguiente correcto funcionamiento del subsistema.

- Ejecución de pruebas: se llevará a cabo la ejecución de las pruebas diseñadas y planificadas.

- Despliegue de la aplicación. Despliegue del sistema: el objetivo de esta fase es el de implantar correctamente la solución desarrollada. Entre los componentes a implantar, es necesario tener en cuenta los ficheros de configuración y de creación del sistema.

- Mantenimiento. Mantenimiento del sistema: aquí se corregirán los fallos y errores sin solución en el sistema desarrollado.

- Documentación. Documentación del proyecto: en esta fase se documentará todo el sistema, desde las tecnologías estudiadas para implementar hasta el manual de usuario de la aplicación, pasando por el análisis y diseño de las funcionalidades requeridas.

Planificación de tareas

De manera adicional al listado de tareas se va a mostrar la planificación realizada de la duración de estas. Esta planificación es estimada, por lo que puede sufrir modificaciones durante la realización del proyecto. La Figura 14 muestra esta planificación.

Se incluye también el Anexo II. Seguimiento de proyecto fin de carrera, el cual refleja esta planificación estimada comparada con la planificación real del desarrollo del mismo. Es necesario tener presente que la realización final del proyecto ha sido responsabilidad del alumno autor de este proyecto de fin de carrera, por lo que las condiciones de recursos, gastos y tiempos son completamente diferentes. En este anexo se puede comprobar el tiempo real empleado, frente la estimación inicial.

Task Name	Duration	Start	Finish
Proyecto	**78 days?**	**Thu 15/01/09**	**Mon 04/05/09**
Estudio preliminar	**7 days?**	**Thu 15/01/09**	**Fri 23/01/09**
Planteamiento del problema y esbozo de la	2 days?	Thu 15/01/09	Fri 16/01/09
Elaboración del Plan de proyecto	5 days?	Mon 19/01/09	Fri 23/01/09
Estado de la cuestión	**10 days?**	**Mon 26/01/09**	**Fri 06/02/09**
Estudio de las tecnologías disponibles	5 days?	Mon 26/01/09	Fri 30/01/09
Estudio de las librerías de componentes JS	5 days?	Mon 02/02/09	Fri 06/02/09
Análisis del sistema	**11 days?**	**Mon 02/02/09**	**Mon 16/02/09**
Definición de requisitos	5 days?	Mon 02/02/09	Fri 06/02/09
Especificación de requisitos	6 days?	Mon 09/02/09	Mon 16/02/09
Diseño	**9 days?**	**Tue 17/02/09**	**Fri 27/02/09**
Diseño de la arquitectura	4 days?	Tue 17/02/09	Fri 20/02/09
Diseño del modelo de datos	2 days?	Mon 23/02/09	Tue 24/02/09
Diseño de la interfaz	4 days?	Mon 23/02/09	Thu 26/02/09
Diseño de la gestión de componentes JSF	9 days?	Tue 17/02/09	Fri 27/02/09
Diseño de la lógica de negocio	8 days?	Wed 18/02/09	Fri 27/02/09
Desarrollo de la aplicación	**40 days?**	**Mon 23/02/09**	**Fri 17/04/09**
Desarrollo de la interfaz	30 days?	Mon 23/02/09	Fri 03/04/09
Desarrollo del dominio	20 days	Mon 23/02/09	Fri 20/03/09
Generación de componentes JSF	20 days?	Mon 02/03/09	Fri 27/03/09
Gestión de librerías de componentes JSF	23 days?	Mon 02/03/09	Wed 01/04/09
Desarrollo de la lógica de negocio	35 days?	Mon 02/03/09	Fri 17/04/09
Pruebas	**35 days?**	**Mon 02/03/09**	**Fri 17/04/09**
Pruebas unitarias	15 days?	Mon 09/03/09	Fri 27/03/09
Pruebas de integración	18 days?	Wed 11/03/09	Fri 03/04/09
Ejecución de pruebas	35 days?	Mon 02/03/09	Fri 17/04/09
Despliegue de la aplicación	**2 days?**	**Mon 20/04/09**	**Tue 21/04/09**
Despliegue del sistema	2 days?	Mon 20/04/09	Tue 21/04/09
Mantenimiento	**9 days?**	**Wed 22/04/09**	**Mon 04/05/09**
Mantenimiento del sistema	9 days?	Wed 22/04/09	Mon 04/05/09
Documentación	**65 days?**	**Thu 15/01/09**	**Wed 15/04/09**
Documentación del proyecto	65 days?	Thu 15/01/09	Wed 15/04/09

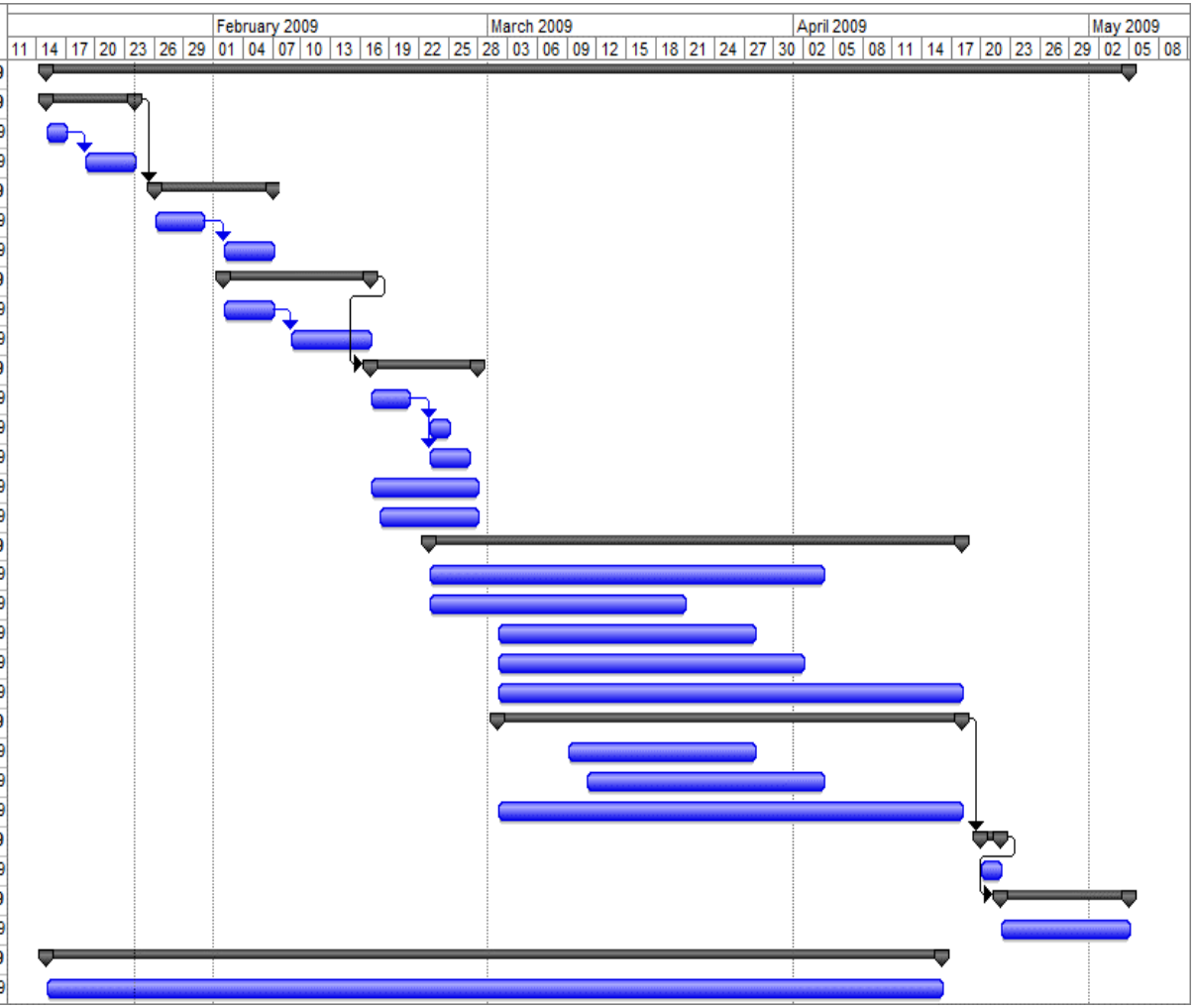

Figura 14. Planificación del proyecto.

3.3 Gestión de recursos

En esta sección se va hablar en profundidad de los recursos utilizados en la realización de este proyecto, concretamente los recursos humanos. Se comentará el perfil de cada tipo de empleado y la asignación de horas que tiene a cada tarea de las que componen el proyecto.

Especificación de recursos

Los recursos humanos que se van a utilizar en el proyecto son relativamente escasos dada la poca envergadura y complejidad del proyecto. El equipo de trabajo consta de los siguientes cuatro miembros:

- Jefe de proyecto: encargado de la planificación, gestión y seguimiento del proyecto.
- Analista: centrado en el análisis del sistema y diseño de la solución.
- 2 programadores: encargados fundamentalmente del desarrollo del código del proyecto, así como de la fase de pruebas y mantenimiento.

Los requisitos que estos miembros del equipo han de cumplir son los siguientes:

- Jefe de proyecto:
  - Experiencia mínima de 3 años gestionando proyectos de complejidad media y organización equipos de 5 a 10 personas.
  - Experiencia en la planificación de tareas y aprovechamiento de recursos humanos.
  - Experiencia en el trato con clientes.
  - Experiencia en resolver situaciones de crisis durante el desarrollo de proyectos.
- Analista:
  - Experiencia mínima de 2 años en la educción de requisitos dadas las peticiones del cliente.
  - Experiencia en el análisis, evaluación y diseño de aplicaciones web.
  - Extenso uso y conocimiento de UML**[12]**, patrones de diseño y patrones J2EE**[13]**.
- Programador:

- Experiencia mínima de 2 años desarrollando aplicaciones web J2EE.
- Extenso uso y conocimiento de UML, patrones de diseño y patrones J2EE.
- Extenso conocimiento de los principales marcos de trabajo J2EE actuales (JSF, Struts **[23]**, Spring **[34]**, Hibernate **[35]**, etc), así como en los principales servidores de aplicaciones (JBoss **[36]**, Tomcat **[37]**, iPlanet **[38]**, etc).

Asignación de recursos

La asignación de los diferentes recursos humanos a las fases del desarrollo, es decir, el desglose de las horas trabajadas por cada empleado en cada una de las distintas tareas, se puede encontrar en la Tabla 6. Desglose de horas trabajadas por rol y por tarea.

3.4 Gestión de riesgos

En todo proyecto, existe la posibilidad de que el desarrollo se vea retrasado o afectado de alguna manera por diversos factores. Dichos factores o imprevistos deben ser tomados en cuenta, identificados claramente desde un primer momento y ser concienzudamente analizados.

Identificación de riesgos

Los problemas que pueden llegar a suceder durante la realización del proyecto son de dos tipos: conocidos y desconocidos. Los desconocidos no se pueden prever, ni identificar, ni analizar, ni prevenir. Sin embargo, aquellos que son conocidos sí pueden ser considerados y tratados. De esta forma, el equipo de desarrollo puede tenerlos en cuenta y estar preparado. De manera general, estos son los principales riesgos conocidos que se consideran:

Identificador	Descripción	Causas	Consecuencias
R01	Falta de tiempo	▪ Mala planificación. ▪ Falta de experiencia.	▪ Retraso en los hitos planificados.
R02	Cliente no disponible	▪ Agenda del cliente demasiado apretada. ▪ Desvinculación del proyecto por parte del cliente.	▪ Mala toma de requisitos. ▪ Discrepancia entre petición del cliente y solución alcanzada.
R03	Baja de un miembro del equipo	▪ Enfermedad. ▪ Fallecimiento.	▪ Búsqueda de reemplazo. ▪ Repartición de tareas

Identificador	Descripción	Causas	Consecuencias
		▪ Despido.	entre el resto del equipo.
R04	Nuevos requisitos o funcionalidades	▪ Nueva solicitud por parte del cliente	▪ Replanificación de las tareas afectadas. ▪ Ajuste de tiempo y coste.
R05	Problemas de hardware o software	▪ Equipos defectuosos. ▪ Virus o ataque informático. ▪ Mala seguridad.	▪ Pérdida de información. ▪ Responsabilidades legales si el proyecto trabaja con datos personales. ▪ Retraso considerable.
R06	Problemas de infraestructura	▪ Terremotos, inundaciones, incendios. ▪ Interrupción del servicio de conexión a Internet. ▪ Problemas electrónicos.	▪ Pérdida de información. ▪ Retraso en el tiempo de desarrollo. ▪ Pérdida del equipo informático.
R07	Robo físico o informático	▪ Hurto de equipos informáticos. ▪ Hurto de información relacionada con el proyecto. ▪ Hurto de código perteneciente al proyecto.	▪ Responsabilidades legales. ▪ Retraso considerable. ▪ Análisis de viabilidad de continuación del proyecto.
R08	Pruebas insuficientes	▪ Planificación de pruebas mala calidad. ▪ No todos los aspectos de la aplicación se prueban.	▪ Fallos o errores no detectados. ▪ Producto de calidad inferior a la deseada. ▪ Incremento del coste

Identificador	Descripción	Causas	Consecuencias
			de mantenimiento.

Tabla 13. Identificación de riesgos.

Análisis de riesgos

Una vez identificados los riesgos en la Tabla 13, es necesario considerar sus consecuencias sólo en términos relativos al producto final y sus condiciones de generación, es decir, el coste correspondiente a su desarrollo, la calidad alcanzada y el tiempo empleado. Para valorar el impacto en cada aspecto comentado de cada riesgo se utilizan tres valores cualitativos referidos a la intensidad de dicho impacto: alto, medio y bajo. Esta relación de riesgos e impacto puede verse en la Tabla 14.

Riesgo	Impacto en coste	Impacto en tiempo	Impacto en calidad
R01	Bajo	Alto	Medio
R02	Bajo	Medio	Alto
R03	Bajo	Alto	Medio
R04	Medio	Alto	Bajo
R05	Alto	Alto	Bajo
R06	Alto	Alto	Bajo
R07	Alto	Medio	Bajo
R08	Bajo	Medio	Alto

Tabla 14. Análisis de impacto de riesgos.

4 Solución

En este capítulo se explica el proceso de desarrollo llevado a cabo para elaborar la solución planteada, así como esta solución. En primer lugar, se procede a describir el proceso de desarrollo de la misma, para continuar con la presentación del producto obtenido.

4.1 El proceso de desarrollo

En esta descripción del proceso de desarrollo se empieza realizando el análisis del sistema a elaborar, comenzando con una educción de requisitos y terminando con la especificación de estos desde diferentes puntos de vista. Después, se pasa a explicar el diseño al que se ha llegado, tanto a nivel de sistema como a nivel detallado. Posteriormente, se describen las tecnologías y herramientas utilizadas para la realizar la implementación de la solución. Por último, se incluyen los resultados de las pruebas unitarias y de sistema llevadas a cabo en la aplicación.

El proceso de desarrollo software se define de la siguiente forma **[9]**:

"Se define como proceso (de desarrollo software) al conjunto ordenado de pasos a seguir para llegar a la solución de un problema u obtención de un producto, en este caso particular, para lograr la obtención de un producto software que resuelva un problema. (…) Cualquiera sea el proceso utilizado y aplicado en un desarrollo de software, y casi independientemente de él, siempre se debe aplicar un Modelo de Ciclo de Vida. (…) El Modelo de Proceso o Modelo de Ciclo de Vida utilizado para el desarrollo define el orden para las tareas o actividades involucradas, también definen la coordinación entre ellas, enlace y realimentación entre las mencionadas etapas. Entre los más conocidos se puede mencionar: Modelo en Cascada o secuencial, Modelo Espiral, Modelo Iterativo Incremental. De los antedichos hay a su vez algunas variantes o alternativas, más o menos atractivas según sea la aplicación requerida y sus requisitos.".

De acuerdo a esta definición, para completar exitosamente el proceso de desarrollo del producto software propuesto en este proyecto el primer paso es seleccionar un modelo de ciclo de vida que defina y orqueste las distintas fases del citado desarrollo. Para este proyecto se ha seleccionado el modelo de ciclo de vida en espiral. El modelo espiral de ciclo de vida consiste en la realización de iteraciones sobre un conjunto de actividades:

- Educción de requisitos: en esta fase se deducen y recolectan los requisitos que debe cumplir el producto desarrollado.
- Desarrollo: esta fase consiste en la implementación de la funcionalidad solicitada en la fase anterior.
- Evaluación: trata de evaluar la solución desarrollada en la fase anterior, proponiendo soluciones para problemas surgidos, mejoras, ampliaciones, correcciones, etc.

La idea principal de este ciclo de vida consiste en realizar un número finito de iteraciones sobre estas actividades, de tal forma que, salvo en la primera "ejecución" del mismo, la salida o resultado de la última actividad, la de evaluación, sirve como entrada para la primera de ellas, la de recolección de requisitos. De esta forma, se van generando prototipos incrementales, siendo cada uno mejor y más formado que el anterior. Esto hace también posible la disposición de prototipos actualizados en cualquier momento del proceso de desarrollo del producto software.

El flujo de este ciclo de vida se puede observar en la Figura 15:

Figura 15. Ciclo de vida en espiral.

Una vez dispuesto el ciclo de vida a seguir, se procede con el contenido de este apartado. Como ya se comentaba al iniciar este capítulo, este apartado consiste, en primer lugar, en el análisis de los requisitos del sistema. A continuación, se describe el diseño de la solución. Después, se explican los detalles de implementación de la misma. Finalmente, se analizan los resultados de las pruebas unitarias y de sistema.

Análisis

En esta fase se pretende obtener una especificación detallada de las necesidades y condiciones de la solución a desarrollar. Es de vital importancia porque sirve de base para todo el diseño e implementación posteriores.

En primer lugar, se va a proceder a educir los requisitos que ha de cumplir el sistema. Después, se procederá a especificar determinados aspectos del sistema a partir de esos requisitos.

Definición de requisitos

La principal función de la definición de requisitos es la de analizar las peticiones del cliente y obtener una serie de requisitos que debe cumplir el producto desarrollado. Este conjunto de requisitos debe representar lo más fielmente posible las necesidades del cliente. Para ello, debe estar formado por requisitos (de usuario) de diferentes tipos **[10]**:

- Funcionales: los requisitos funcionales describen:
  - Los servicios que proporciona el sistema.
  - La respuesta del sistema ante determinadas entradas.
  - El comportamiento del sistema en situaciones particulares.
- No funcionales: los requisitos no funcionales son restricciones de los servicios o funciones que ofrece el sistema, por ejemplo, cotas de tiempo, proceso de desarrollo, rendimiento, etc. En función del tipo de restricción se establecen subtipos de requisitos no funcionales:
  - Operacionales: recogen el entorno físico y técnico en el que el sistema va a operar.
  - Seguridad: definen los permisos de acceso a la aplicación.
  - Normativos: están relacionados con los factores políticos, legales y culturales que afectan al sistema.

De acuerdo a esta clasificación, se pasa a listar los requisitos educidos. Para facilitar su trazabilidad a lo largo del desarrollo, se les asignan identificadores de la siguiente forma:

- Funcionales: RF_<número de requisito>
- No funcionales: RNF_<tipo>_<número de requisito>

Donde el "número de requisito" es un número de dos cifras para identificar requisitos dentro del mismo tipo y el "tipo" se corresponde con el subtipo (O para operacional, S para seguridad y N para normativo) del requisito concreto.

En primer lugar, se lista el conjunto de requisitos funcionales recogidos:

Id requisito	Descripción
RF_01	Los roles de los que se compondrá el sistema serán el rol de administrador y el de usuario.

Id requisito	Descripción
RF_02	El sistema será gestionado por un usuario administrador, creado de antemano y sin posibilidad de ser dado de baja.
RF_03	Un usuario necesitará iniciar sesión en el mismo para hacer uso de él a través de su nombre de usuario y contraseña.
RF_04	El usuario se podrá desconectar en cualquier momento para finalizar su uso del sistema y así borrar su información temporal almacenada en el cliente.
RF_05	El usuario podrá modificar sus datos personales y de sistema.
RF_06	El usuario podrá crear un nuevo nodo web para realizar el proceso de maquetación.
RF_07	El usuario tendrá la posibilidad de guardar, cargar y borrar nodos web en proceso de maquetación.
RF_08	Durante el proceso de maquetación de un nodo web dado, el usuario podrá comenzar de nuevo el mismo o exportar el producto final de este proceso, que será una página JSF completamente válida y que incluye los componentes configurados. Este proceso de conversión de maqueta web a página JSF será realizado por el sistema de manera automática.
RF_09	Para crear un nodo web el usuario tendrá a su disposición un conjunto de librerías o maestros, cada uno con sus componentes, los cuales podrá arrastrar hacia el lienzo para colocarlos en la posición deseada.
RF_10	Una vez insertado un componente concreto en un nodo, el usuario podrá contraer o expandir el mismo.
RF_11	Una vez insertado un componente concreto en un nodo, el usuario podrá modificar sus propiedades, es decir, configurar todos sus atributos.
RF_12	Una vez insertado un componente concreto en un nodo, el usuario podrá reorganizarlo, es decir, subir o bajar su posición con respecto a los demás.
RF_13	Una vez insertado un componente concreto en un nodo, el usuario podrá borrarlo si lo desea.
RF_14	Un mismo componente puede ser insertado en el nodo un número de veces ilimitado.
RF_15	La cantidad de componentes que se pueden insertar en un nodo es ilimitada.
RF_16	El administrador será el encargado de borrar o modificar datos de los usuarios.
RF_17	El administrador podrá gestionar los maestros disponibles en el sistema.
RF_18	El administrador podrá agregar más maestros de componentes al conjunto inicial proporcionado por el sistema.
RF_19	El administrador tendrá la posibilidad de configurar componentes o maestros.
RF_20	El administrador podrá deshabilitar el uso por parte de los usuarios de un componente o maestro concretos.
RF_21	El administrador podrá modificar sus datos personales y su contraseña de acceso al sistema.
RF_22	El administrador podrá cerrar sesión en cualquier momento para finalizar su uso del sistema.

Tabla 15. Requisitos funcionales.

A continuación, el listado de requisitos no funcionales, separados por subtipo:

Requisitos no funcionales operacionales	
Id requisito	**Descripción**
RNF_O_01	La aplicación web desarrollada será portable, pudiendo ser desplegada en cualquier servidor de aplicaciones compatible en cualquier Sistema Operativo o plataforma (multiplataforma) y ser accesible desde los navegadores web más usados hoy en día, Mozilla Firefox, Ópera e Internet Explorer (cross-browser).
RNF_O_02	El idioma de la interfaz del sistema será el castellano.
RNF_O_03	De manera interna, los nodos web considerados en los requisitos funcionales serán transformados a páginas JSF, conteniendo estas componentes de las diferentes librerías o maestros disponibles.

Tabla 16. Requisitos no funcionales operacionales.

Requisitos no funcionales de seguridad	
Id requisito	**Descripción**
RNF_S_01	Es necesario superar satisfactoriamente un proceso de autenticación para poder acceder al sistema.
RNF_S_02	El sistema garantiza la privacidad de las contraseñas de los usuarios almacenando solamente el resultado de aplicar una función resumen a las mismas, es decir, no guardando en ningún caso las contraseñas en claro.

Tabla 17. Requisitos no funcionales de seguridad.

Requisitos no funcionales normativos	
Id requisito	**Descripción**
RNF_N_01	Los datos tratados y almacenados por el sistema se encuentran protegidos por la LOPD, por lo que sus derechos deben ser protegidos.

Tabla 18. Requisitos no funcionales normativos.

Especificación de requisitos

Una vez recogidos los requisitos que el sistema debe cumplir es necesario especificarlos. Esto consiste en detallarlos de tal forma que vayan modelando el aspecto final que debe tener el sistema. Para llevar este proceso a cabo, se va a separar en dos partes: el modelo de dominio y el modelo de sistema.

Modelo de contexto

El modelo de contexto trata de plasmar la representación de los conceptos y entidades que participan en el proyecto, incluyendo su estructura y organización. Para definir este modelo de contexto se van a utilizar un conjunto de diagramas UML.

- Funcional: esta parte del modelo de contexto especifica la funcionalidad principal que realiza el sistema. Para representarlo, se utiliza un diagrama de actividad que representa el comportamiento de inclusión de un componente JSF en una página determinada:

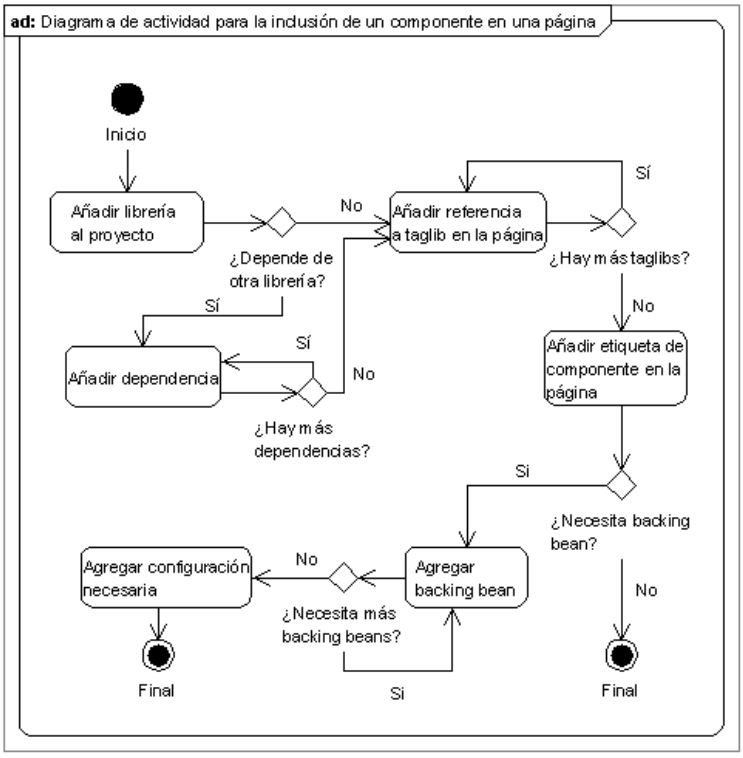

Figura 16. Diagrama de actividad para la inclusión de un componente en una página.

En el diagrama mostrado en la Figura 16 se puede observar cómo el proceso de incluir un nuevo componente en una página JSF se descompone en la agregación de diferentes elementos:

- Librería a la aplicación web.
- Dependencias.
- *Taglibs* o librerías de etiquetas a la página.
- *Backing beans*.
- Configuración necesaria.

- Estructural: se centra en la estructura del sistema, de los objetos que lo conforman. Siendo un sistema basado en objetos, se utiliza un diagrama de clases para mostrar las clases necesarias y las relaciones entre ellas. En la Figura 17 se expone la estructura de estos objetos, siendo los principales:
  - Usuario: usuario que crea/accede al sistema.
  - Librería: contiene una serie de componentes.
  - Componente: representa una componente JSF que puede ser utilizado en cualquier página.
  - Atributo: cada componente se configura a través de un conjunto de atributos.

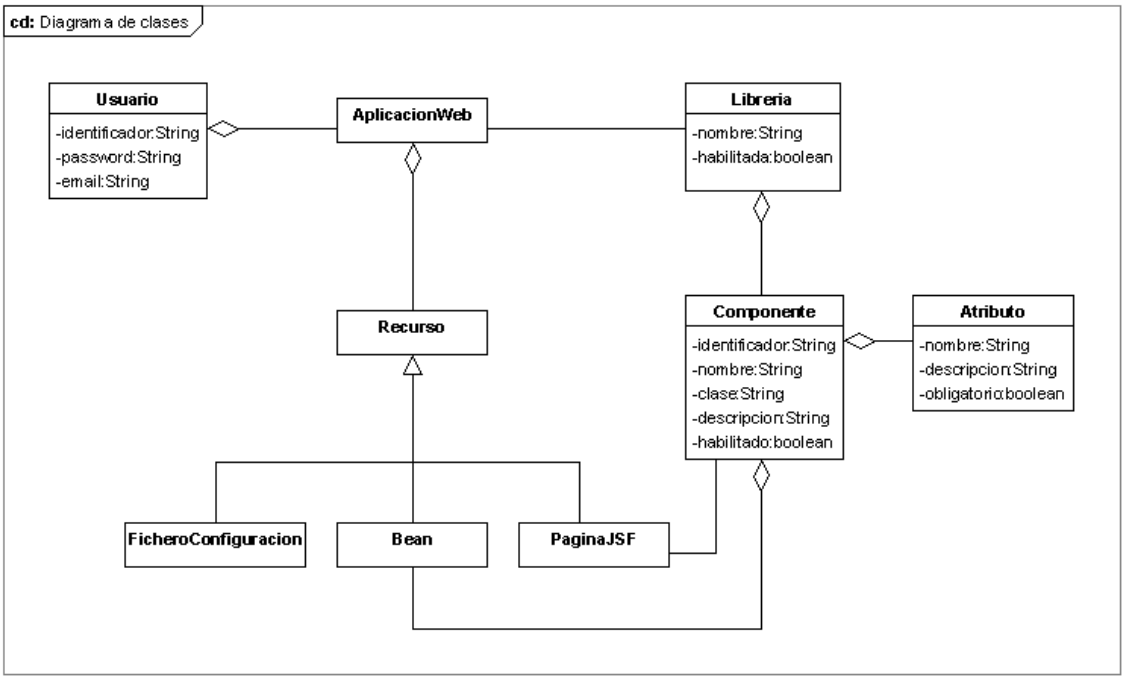

Figura 17. Diagrama de clases.

- Comportamiento: en esta parte se muestran los aspectos dinámicos del sistema, especificando el flujo de llamadas entre los objetos del sistema y los mensajes que intercambian. Para realizar este análisis se utilizan diagramas de secuencia, que permiten trazar estas llamadas y mensajes. La Figura 18 muestra el diagrama de secuencia que representa este flujo, siendo los objetos y mensajes los siguientes:
  - Usuario: en primer lugar, el usuario desarrollador crea una aplicación web.
  - AplicaciónWeb: después, la aplicación añade una librería al conjunto que ya posee.
  - Librería: la inclusión de la librería ocasiona, generalmente, la modificación de la uno o más ficheros de configuración.
  - AplicaciónWeb: a continuación, la aplicación crea una página JSF.
  - PáginaJSF: en la página creada se procede a incluir la etiqueta del componente deseado.
  - Componente: la inclusión del componente ocasiona la creación, si es necesario, de uno o más beans manejados.

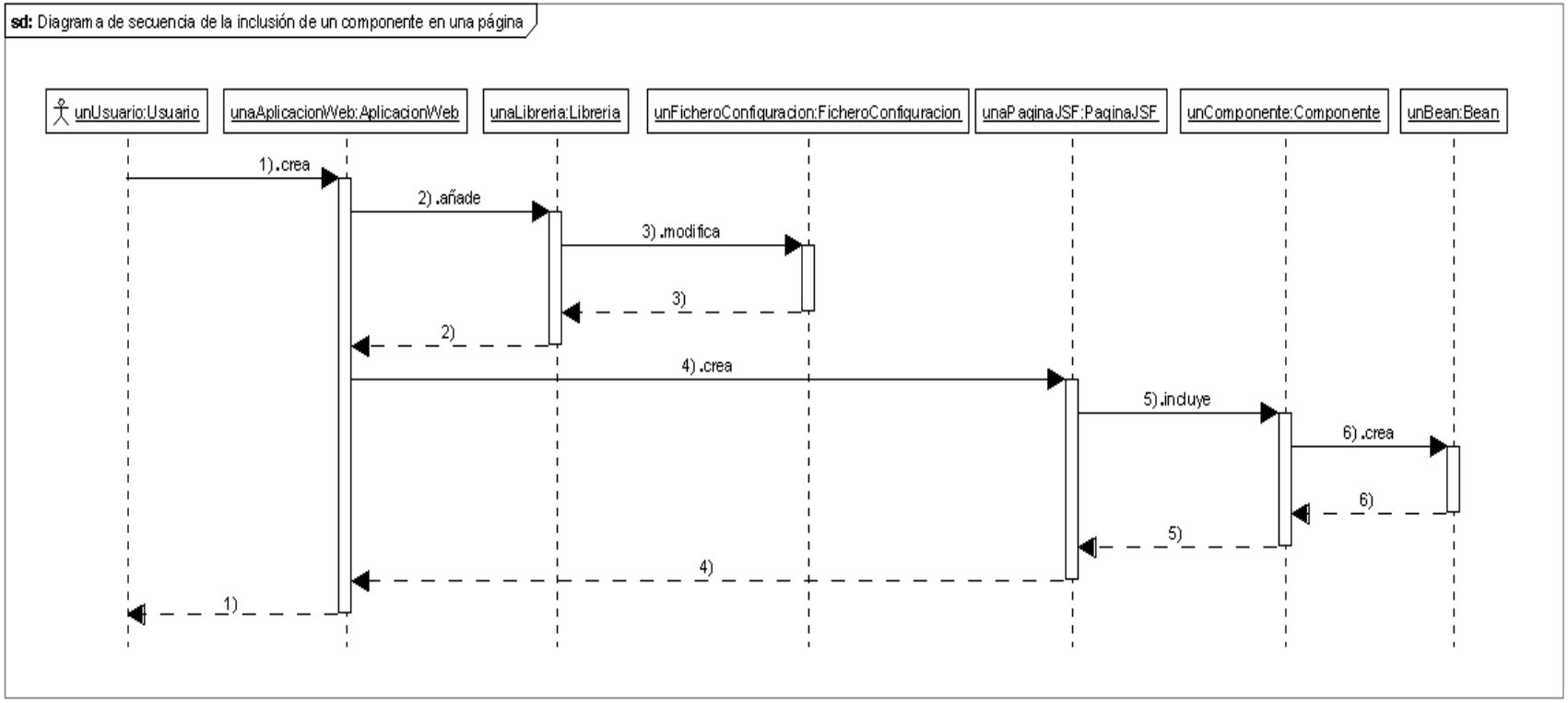

Figura 18. Diagrama de secuencia para la inclusión de un componente en una página.

Modelo de sistema

El objetivo del modelo de sistema es representar las funcionalidades que necesita tener el futuro sistema, incluyendo también los objetos que formarán parte del mismo. Siendo este modelo realizado durante la fase de análisis, carece de gran nivel de detalle, por lo que será ampliado en la fase de diseño. Las partes del modelo de sistema son dos: funcional y estructural, diferenciadas por la perspectiva del sistema en el que se centra cada una.

- Funcional: esta parte se centra en la funcionalidad que debe proporcionar el sistema a desarrollar. Para ello, se utiliza diagramas de casos de uso, en los que se puede observar la relación entre actores y acciones a realizar, así como la relación entre los actores mismos. Esto se puede observar en las figuras Figura 19, Figura 20, Figura 21 y Figura 22. A su vez, esta sería la descripción de cada figura:
  - Figura 19. Casos de uso: actores. Representa la relación entre los diferentes actores del sistema. El actor base sería el "Usuario no autenticado", el cual extiende el "Usuario". De la misma manera, el actor "Administrador" extiende a este último.
  - Figura 20. Casos de uso: administración. En esta figura se muestran las tareas de administración que puede realizar el usuario administrador en el sistema. Estas se descomponen en dos grupos:
    - Gestión de usuarios: administra los usuarios que componen el sistema. Permite listar usuarios, agregar nuevos, y modificar o borrar existentes.
    - Gestión de librerías: administra las librerías contenidas en el sistema. Posibilita su listado, agregación de nuevas, listado de componentes y habilitado/deshabilitado de componentes/librerías.
  - Figura 21. Casos de uso: acceso al sistema. Muestra las acciones disponibles para el usuario no autenticado (registrarse e iniciar sesión) y el usuario autenticado (modificar datos y cerrar sesión).
  - Figura 22. Casos de uso: usuarios. Esta figura muestra el uso del sistema que puede hacer un usuario autenticado. Este uso se divide en:
    - Gestión de páginas: listado, creación, edición y borrado de páginas.
    - Creación de páginas: listad de componentes agregados, agregar, mover, borrar, configurar, contraer y expandir componentes, borrar todos y exportar página.

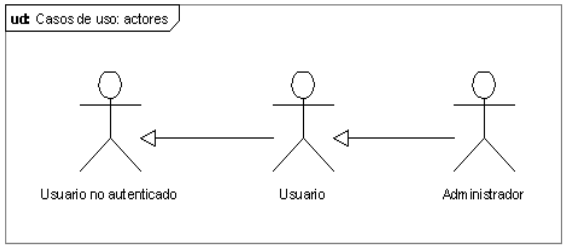

Figura 19. Casos de uso: actores.

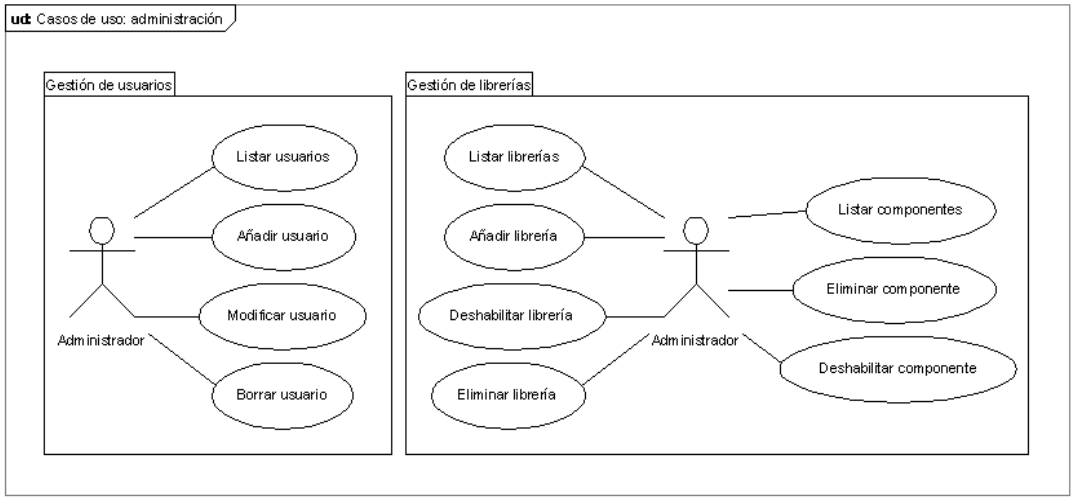

Figura 20. Casos de uso: administración.

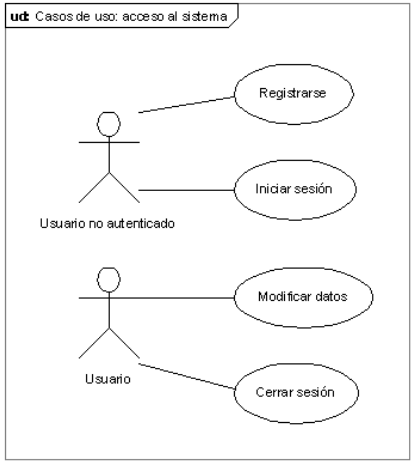

Figura 21. Casos de uso: acceso al sistema.

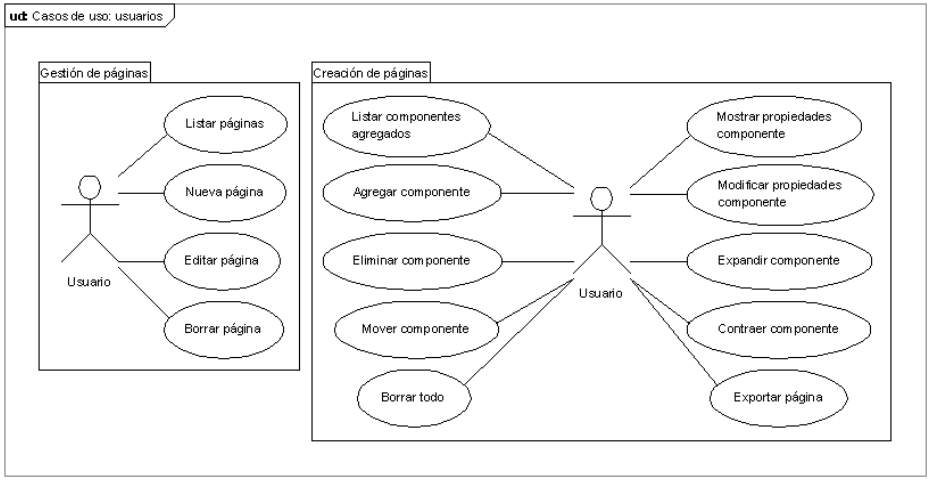

Figura 22. Casos de uso: usuarios.

- Estructural: para definir la estructura de los objetos del dominio se utiliza el diagrama Entidad/Relación, el cual recoge los conceptos y reglas de relación existentes en el sistema:

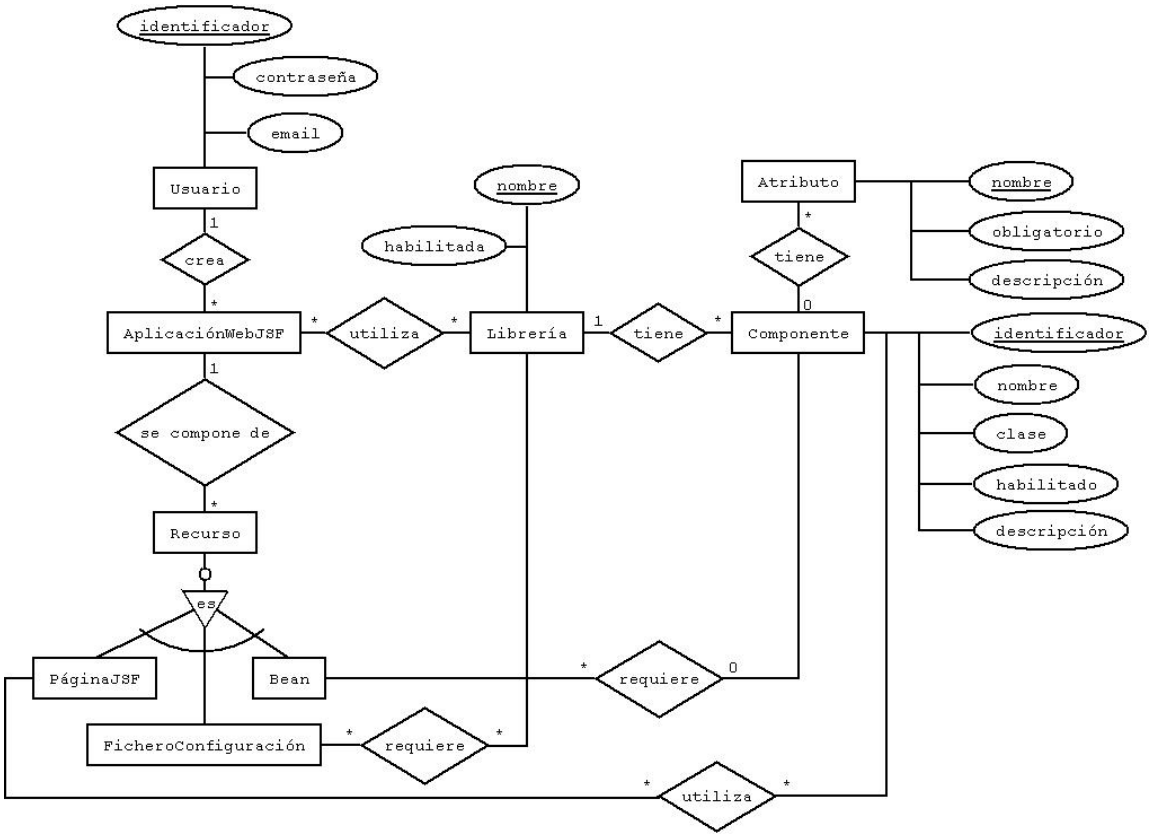

Figura 23. Diagrama de Entidad/Relación.

En la Figura 23 se puede observar el diagrama E/R citado. En él se ve que las entidades principales (aquellas con atributos) son:

- Usuario: representa al usuario que crea la aplicación web.
- Librería: hace referencia a una librería de componentes.
- Componente: representa un componente JSF perteneciente a una librería.
- Atributo: se refiere a los atributos que permiten modificar la apariencia de un componente dado.

Diseño

La fase que sigue al análisis del sistema a desarrollar es la fase de diseño. En ella, se pretende construir un sistema que se ajuste a las necesidades requeridas (requisitos funcionales) y a las limitaciones impuestas (requisitos no funcionales). En primer lugar, se va a ofrecer un diseño del sistema visto "desde arriba", es decir, centrado en la estructura general y exterior que tiene el producto a desarrollar. Después, se proporciona un diseño más detallado, en el que se puede observar como esa estructura general se amplía y concreta para definir con mayor precisión el diseño. Es en este diseño detallado final donde los casos de uso se verán cumplidos, aún cuando no se especifique claramente que módulo o subsistema se encarga de cada uno.

Diseño de sistema

En este apartado se expone la arquitectura del sistema y las tecnologías utilizadas para su desarrollo. La arquitectura sirve de estructura general del sistema, mientras que las tecnologías ayudan a su funcionamiento.

Arquitectura

La arquitectura del sistema está basada en el ya introducido patrón Modelo-Vista-Controlador (MVC, apartado "Tendencia actual"). La principal característica de este patrón es la clara separación de responsabilidades:

- Modelo: se encarga de gestionar los datos recuperados de la capa de persistencia. Proporciona acceso y operaciones sobre estos datos.
- Controlador: su función es gestionar las peticiones realizadas por la Vista, enlazándolas con el Modelo.
- Vista: proporciona la presentación final de los datos y resultados de operaciones devueltos por el Modelo.

En este proyecto, la capa de persistencia es de especial relevancia, no por su tamaño o extensión, sino por su complejidad. La estructura de los datos a almacenar y la interpretación de los mismos requieren una clara separación entre la parte del Modelo del patrón MVC y esta capa de persistencia, la cual se refleja en la Figura 24.

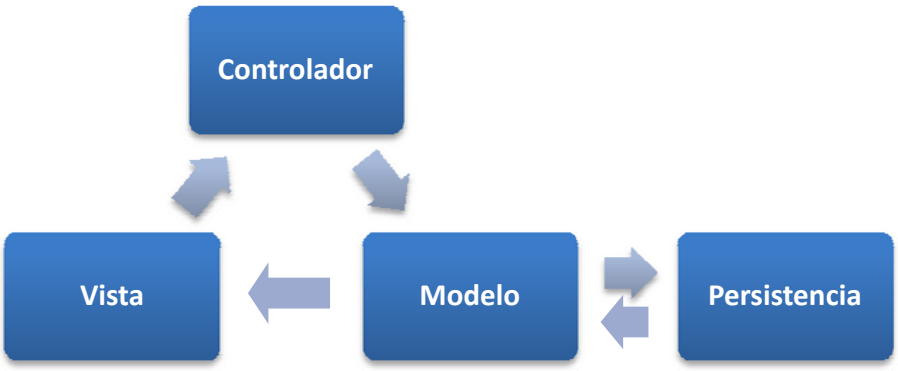

Figura 24. Arquitectura de la aplicación.

Se va a concretar ahora cada una de las partes que componen esta arquitectura, aplicadas al presente proyecto. Además, se citarán tecnologías que posibilitan su funcionamiento, que serán explicadas en el próxima apartado (Tecnologías), y componentes que serán descritos en profundidad en la siguiente sección (Diseño detallado). La Figura 25 ilustra la citada arquitectura concretada:

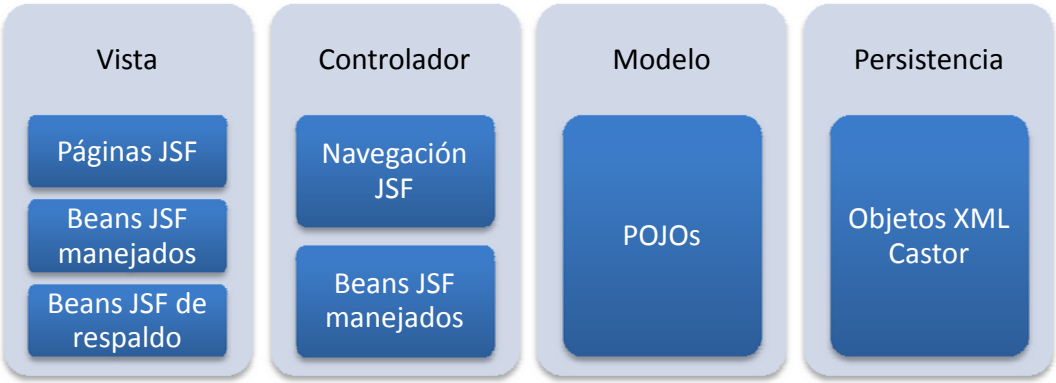

Figura 25. Arquitectura de la aplicación concretada.

Ahora se va a describir cada componente de la arquitectura mostrada:

- Vista:
  - Páginas JSF: representan la presentación final de la información. Es lo ve el usuario final de la aplicación, aquello con lo que interactúa.
  - Beans JSF manejados: son beans normales (POJOs, *Plain Old Java Objects*) que se utilizan para guardar temporalmente información relacionada con la presentación de los datos y la naturaleza de los mismos, como pueden ser el usuario actual de la aplicación, el valor de determinados campos de un formulario, etc.

- Beans JSF de respaldo: también son POJOs pero con una función especialidad: los datos que almacenan no son regulares, sino que contienen componentes JSF instanciados.

- Controlador:
  - Navegación JSF: para realizar la función principal de la parte del Controlador, toda implementación del estándar de JSF utiliza las reglas de navegación definidas en los ficheros de navegación para redirigir el control de una página a otra, en función de la página inicial y del valor de redirección recibido.
  - Beans JSF manejados: para determinar el valor de redirección que se transmite a las reglas de navegación, se suelen utilizar beans manejados (POJOs) que, además de almacenar información temporal, contienen métodos que devuelven valores de redirección en función de sus condiciones y estados internos.

- Modelo:
  - POJOS: la parte del Modelo se encuentra completamente formada por simples clases Java, que según la petición recibida del Controlador y los datos recuperados de la capa de Persistencia realizan una serie de operaciones y generan unos resultados.

- Persistencia:
  - Objetos XML Castor: para el almacenamiento persistente de datos se ha decidido utilizar ficheros XML. Estos serán gestionados, tratados y asociados a objetos, gracias a la librería Castor, la cual será explicada en el apartado Tecnologías.

Tecnologías

En este apartado se van a describir las diferentes tecnologías utilizadas en este proyecto para posibilitar la arquitectura y funcionalidades requeridas. Se va a explicar cada una de estas tecnologías tanto de manera general (qué hace y cómo lo hace) como de manera concreta aplicada a la aplicación (qué tarea concreta desempeña y dónde).

JavaServer Faces

La tecnología JSF fue descrita en las secciones 2.2 y 2.3. Básicamente, establece un estándar en el desarrollo de la capa de presentación en las aplicaciones web J2EE, que consiste en el uso de componentes para formar esta presentación. Estos componentes son completamente independientes y pueden proceder de diferentes librerías de terceros. Además, proporciona un motor de navegación de reglas así como la gestión de beans manejados y de respaldo. A

continuación se va a introducir la implementación del estándar JSF que se utiliza, así como las librerías que lo complementan.

- Sun JSF RI: la implementación del estándar que se utiliza en este proyecto es la desarrollada por Sun Microsystems. Contiene un número limitado y básico de componentes, suficientes para elaborar páginas relativamente sencillas.

- JBoss RichFaces: para ampliar este conjunto de componentes y dotar de dinamismo al sistema desarrollado se utiliza la librería RichFaces. Esta librería es fruto de la unión del trabajo de Red Hat y Exadel. Es completamente independiente de la implementación de JSF que se utilice así que se puede reutilizar de un proyecto a otro aunque se cambie la implementación del estándar. Entre sus características cabe destacar:

  - Soporte AJAX a nivel de página, en lugar de a nivel de componente.
  - Posibilidad de personalizar el aspecto de los componentes con “pieles”.
  - Validación de datos de entrada.
  - Buen soporte y gestión de eventos.

  Además, cuenta con una amplia gama de componentes, de todos los tipos:

  - Soporte AJAX: formulario AJAX, AJAX listener, región AJAX.
  - Iteración de datos: rejilla de datos, tabla de datos, lista de datos, ordenado, filtrado y navegación en tablas.
  - Menús: contextuales, desplegables, barra de menú.
  - Árboles: simples, recursivos.
  - Salida enriquecida: panel, panel modal, diálogo, separadores, pestañas, barras de herramientas.
  - Entrada enriquecida: calendario, subida de ficheros, caja de sugerencias.
  - Varios: efectos, Google Maps, Virtual Earth.

Esta tecnología representa el pilar central de toda la aplicación: convierte el proyecto desarrollado en una meta-aplicación, es decir, la herramienta utiliza JSF para generar JSF. Su uso está generalizado en toda la aplicación, sirviendo de base para las páginas de la capa de Vista o presentación, los beans de respaldo, los beans manejados, las reglas de navegación, etc.

Debido a la potencia de la implementación de JavaServer Faces unida a la librería de componentes RichFaces, no es necesario incluir ningún marco de trabajo más para componer las partes de Vista y Controlador de la aplicación.

AJAX

Como ya se explicó en el apartado Tendencia actual, AJAX significa *Asynchronous Javascript and XML*, Javascript y XML asíncrono en castellano, y consiste en la utilización de tecnologías que existían previamente para dotar de dinamismo a las páginas web. La librería descrita en el punto anterior, JBoss RichFaces, proporciona soporte AJAX para todos sus componentes, siendo posible enviar y recuperar datos del servidor para refrescar sólo aquellas secciones de la página que realmente se vean afectadas.

Se hace un gran uso de esta tecnología en el proceso de maquetación de una página JSF con la aplicación desarrollada. Toda la tarea de agregar componentes al lienzo, modificar sus propiedades, aspecto y posicionamiento se realiza de manera completamente asíncrona, siendo sus cambios reflejados tanto en el cliente como en el servidor sin que el usuario final perciba retraso alguno en su experiencia de uso.

Facelets

Facelets es un marco de de trabajo que se basa en el uso de plantillas para organizar componentes en páginas JSP (JavaServer Pages). Permite ser utilizado sobre una gran cantidad de marcos de trabajo que se superponen sobre estas páginas JSP. En concreto, en este proyecto se utiliza sobre los componentes JSF. Estas son las propiedades más interesantes de Facelets:

- Trabajo basado en plantillas.
- Fácil composición de componentes.
- Creación de etiquetas lógicas a medida.
- Funciones para expresiones.
- Desarrollo amigable para el diseñador gráfico.
- Creación de librerías de componentes.

Castor XML

La librería Castor posibilita el acceso y gestión de ficheros XML de manera que sea lo más simple posible: orientada a objetos. A partir del XSD (*XML Schema* Definition, Definición de Esquema XML) de un fichero XML, Castor XML es capaz de generar toda una estructura de clases para representar la información y su estructura contenida en ese fichero. De esta forma, una vez realizado este proceso, este conjunto de clases puede tratar con cualquier fichero XML que siga el esquema especificado inicialmente, obteniendo instancias de clases para acceder y recorrer los datos almacenados.

En este proyecto, Castor XML se utiliza para gestionar la capa de persistencia de la arquitectura diseñada. Permite asociar entidades dentro de un fichero XML a objetos Java, con lo que juega un papel esencial en el almacenamiento persistente de los descriptores de librerías de componentes incluidas, de los datos de los usuarios y de las páginas maquetadas.

Diseño detallado

En este apartado se va a detallar cada uno de los componentes de la arquitectura introducida previamente en el apartado anterior (Figura 25. Arquitectura de la aplicación concretada.). En primer lugar, se va a comenzar por la capa de persistencia, seguida de las partes de Modelo, Controlador y Vista, en ese orden.

Persistencia

La capa de persistencia constituye los cimientos de la aplicación: es la responsable del almacenamiento de toda la información persistente del sistema. Esta información se guarda en ficheros XML, los cuales se construyen de acuerdo a los esquemas XSD que se van a comentar más adelante, y el formato de estos depende del tipo de la misma. La elección de este tipo de formato viene dada, fundamentalmente, por la independencia que otorga respecto a soluciones que involucran bases de datos, teniendo que asegurar una configuración externa a la aplicación. La información que se va a almacenar viene dada por los requisitos iniciales y puede verse en las figuras Figura 23 y Figura 17, de forma que se obtienen los esquemas mostrados más adelante. Esta información a almacenar es:

- Datos de usuario: se almacenan los siguientes datos:
  - Identificador: indica el identificador global del usuario.
  - Contraseña: representa la contraseña de acceso al sistema. Sólo se almacena el resumen MD5 de la contraseña original, por motivos de seguridad.
  - Email: dirección de correo electrónico de contacto.
- Datos de librería de componentes:
  - Nombre: nombre de la librería.
  - Habilitada: indica si la librería se encuentra habilitada o deshabilitada.
  - Componentes: para cada componente se almacena lo siguiente:
    - Identificador: identificador global del componente.
    - Nombre: nombre del componente.
    - Clase: clase Java que se utiliza para instanciar el componente.
    - Descripción: descripción que la librería guarda de cada componente.
    - Habilitado: indica si el componente se encuentra habilitado o deshabilitado.

- Campos: un componente tiene una serie de campos. Para cada uno de ellos se almacena lo siguiente:
  - Nombre: nombre del campo.
  - Descripción: descripción que guarda el componente de este campo.
  - Obligatorio: indica si es obligatorio proporcionar un valor para este campo durante la instanciación del componente.

- Datos de página maquetada:
  - Identificador: identificador global de la página.
  - Nombre: nombre de la página.
  - Descripción: contiene la descripción de esta página.
  - Propietario: identificador del usuario creador.
  - Fecha: fecha de creación.
  - Hora: hora de creación.
  - Componentes: se almacena el conjunto de componentes que la forman, de manera que el orden en el fichero indica el orden en la página:
    - Identificador: identificador del componente usado.
    - Campos: valores de los campos usados:
      - Nombre: nombre del campo.
      - Valor: valor del campo.

Por tanto, los esquemas XSD encargados de representar estas estructuras de información son los siguientes:

- Fichero esquema XSD de usuarios:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://dei.inf.uc3m.es"
xmlns:tns="http://dei.inf.uc3m.es"
elementFormDefault="qualified">

      <complexType name="usuarioType">
            <sequence>
                  <element name="identificador" type="string"/>
```

```
                  <element name="password" type="string"/>
                  <element name="email" type="string"/>
            </sequence>
      </complexType>

      <element name="usuarios">
            <complexType>
                  <sequence>
                        <element name="usuario" type="tns:usuarioType"
maxOccurs="unbounded"/>
                  </sequence>
            </complexType>
      </element>
</schema>
```

- Fichero esquema XSD de librerías:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://dei.inf.uc3m.es"
xmlns:tns="http://dei.inf.uc3m.es"
elementFormDefault="qualified">

      <complexType name="campoType">
            <sequence>
                  <element name="nombre" type="string"/>
                  <element name="obligatorio" type="boolean"/>
                  <element name="descripcion" type="string"/>
            </sequence>
      </complexType>

      <complexType name="camposType">
            <sequence>
                  <element name="campo" type="tns:campoType"
maxOccurs="unbounded"/>
            </sequence>
      </complexType>

      <complexType name="componenteType">
            <sequence>
                  <element name="identificador" type="string"/>
                  <element name="nombre" type="string"/>
                  <element name="clase" type="string"/>
                  <element name="habilitado" type="boolean"/>
                  <element name="descripcion" type="string"/>
                  <element name="campos" type="tns:camposType"/>
            </sequence>
      </complexType>

      <complexType name="componentesType">
            <sequence>
                  <element name="componente" type="tns:componenteType"
maxOccurs="unbounded"/>
            </sequence>
      </complexType>

    <complexType name="libreriaType">
            <sequence>
                  <element name="nombre" type="string"/>
                  <element name="habilitada" type="boolean"/>
```

```
                  <element name="componentes" type="tns:componentesType"/>
            </sequence>
      </complexType>

      <element name="librerias">
            <complexType>
                  <sequence>
                        <element name="libreria" type="tns:libreriaType"
maxOccurs="unbounded"/>
                  </sequence>
            </complexType>
      </element>
</schema>
```

- Fichero esquema XSD de páginas:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://dei.inf.uc3m.es"
xmlns:tns="http://dei.inf.uc3m.es"
elementFormDefault="qualified">

      <complexType name="campoType">
            <sequence>
                  <element name="nombre" type="string"/>
                  <element name="valor" type="string"/>
            </sequence>
      </complexType>

      <complexType name="camposType">
            <sequence>
                  <element name="campo" type="tns:campoType"
maxOccurs="unbounded"/>
            </sequence>
      </complexType>

      <complexType name="componenteType">
            <sequence>
                  <element name="identificador" type="string"/>
                  <element name="campos" type="tns:camposType"/>
            </sequence>
      </complexType>

      <complexType name="componentesType">
            <sequence>
                  <element name="componente" type="tns:componenteType"
maxOccurs="unbounded"/>
            </sequence>
      </complexType>

    <complexType name="paginaType">
            <sequence>
                  <element name="identificador" type="string"/>
                  <element name="nombre" type="string"/>
                  <element name="descripcion" type="string"/>
                  <element name="propietario" type="string"/>
                  <element name="fecha" type="string"/>
                  <element name="hora" type="string"/>
                  <element name="componentes" type="tns:componentesType"/>
            </sequence>
```

```
	</complexType>

	<element name="paginas">
		<complexType>
			<sequence>
				<element name="pagina" type="tns:paginaType"
maxOccurs="unbounded"/>
			</sequence>
		</complexType>
	</element>
</schema>
```

De esta forma, un conjunto de ficheros XML de ejemplo de la aplicación podrían ser los siguientes:

- Fichero de usuarios:

```
<?xml version="1.0" encoding="UTF-8"?>
<usuarios>
	<usuario>
		<identificador>Usuario1</identificador>
		<password>d17b01bcbb7080b7c49ed41253926115<password>
		<email>usuario1@dominio.com</email>
	</usuario>
	<usuario>
		<identificador>Usuario2</identificador>
		<password>26f3aaafaaabd6456eb1d368756d2497<password>
		<email>usuario2@dominio.com</email>
	</usuario>
</usuarios>
```

- Fichero de librerías:

```
<?xml version="1.0" encoding="UTF-8"?>
<librerias>
	<libreria>
		<nombre>Librería 1</nombre>
		<habilitada>true</habilitada>
		<componentes>
			<componente>
				<identificador>1</identificador>
				<nombre>Componente 1</nombre>
				<clase>paquete.subpaquete.Clase1</clase>
				<habilitado>true</habilitado>
				<descripcion>Descripción</descripcion>
				<campos>
					<campo>
						<nombre>Campo1</nombre>
						<obligatorio>true</obligatorio>
						<descripcion>Descripción</descripcion>
					</campo>
					<campo>
						<nombre>Campo2</nombre>
						<obligatorio>true</obligatorio>
						<descripcion>Descripción</descripcion>
					</campo>
```

```
                        </campos>
                  </componente>
            </componentes>
      </libreria>
<librerias>
```

- Fichero de páginas:

```
<?xml version="1.0" encoding="UTF-8"?>
<paginas>
      <pagina>
            <identificador>1</identificador>
            <nombre>Página 1</nombre>
            <descripcion>Descripción</descripcion>
            <propietario>Usuario1</propietario>
            <fecha>23/02/1985</fecha>
            <hora>14:23</hora>
            <componentes>
                  <componente>
                        <identificador>1</identificador>
                        <campos>
                              <campo>
                                    <nombre>Campo1</nombre>
                                    <valor>Valor</valor>
                              </campo>
                        </campos>
                  </componente>
                  <componente>
                        <identificador>1</identificador>
                        <campos>
                              <campo>
                                    <nombre>Campo2</nombre>
                                    <valor>Valor</valor>
                              </campo>
                        </campos>
                  </componente>
            </componentes>
      </pagina>
</paginas>
```

A continuación se va a mostrar el diagrama de clases encargadas del proceso de serialización o *marshalling* (pasar de objetos Java, POJOs, a ficheros XML) y deserialización o *unmarshalling* (pasar de ficheros XML a POJOs), así como de la encapsulación del acceso a estos datos, todo ello separado de acuerdo al tipo de fichero que tratan:

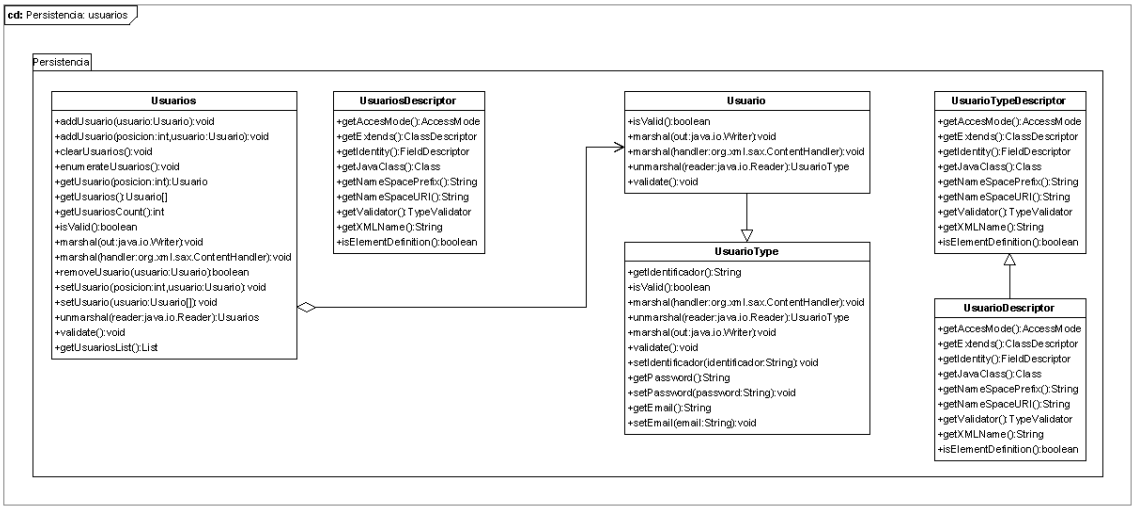

Figura 26. Diagrama de clases de la capa de persistencia: usuarios.

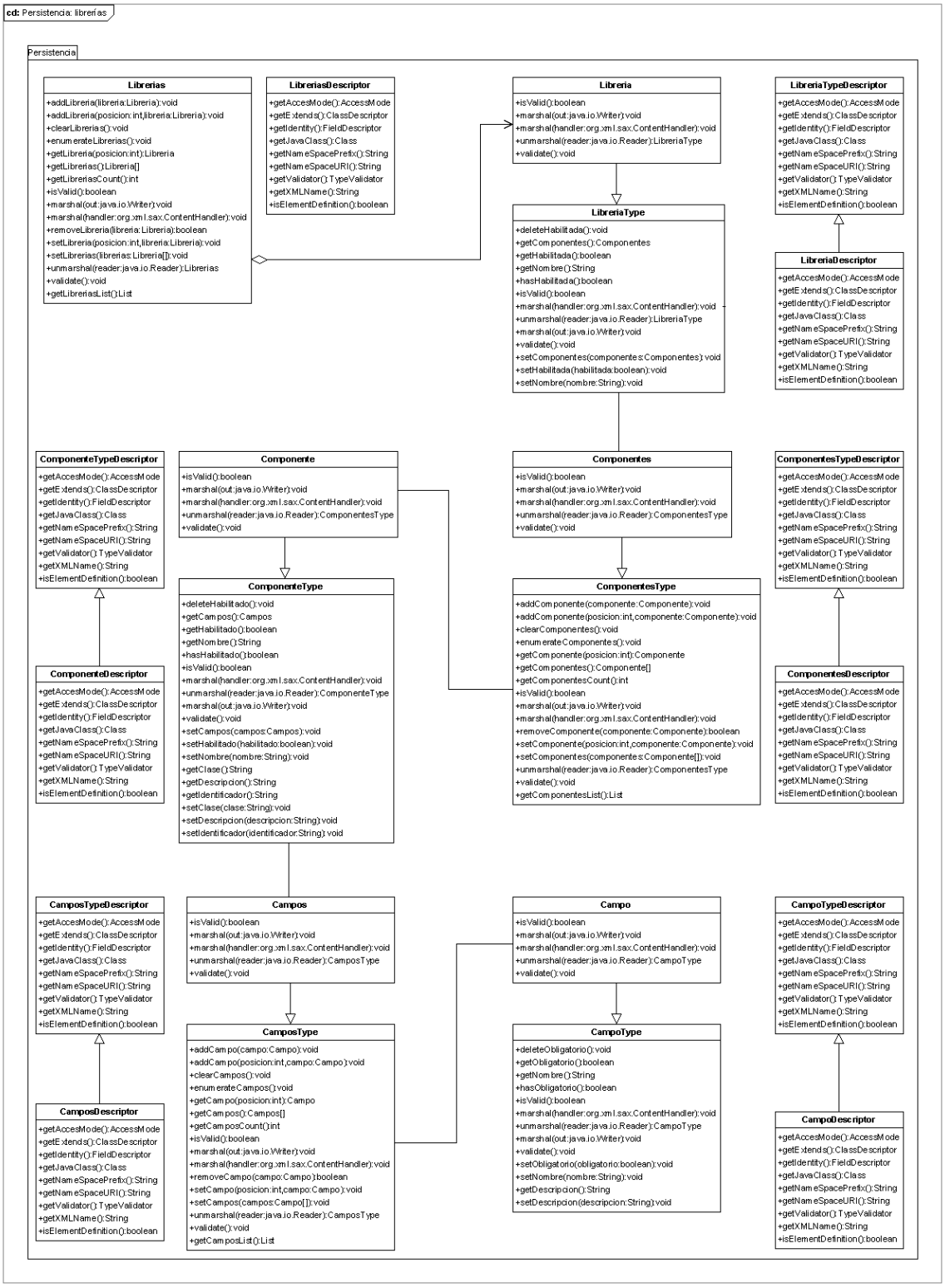

Figura 27. Diagrama de clases de la capa de persistencia: librerías.

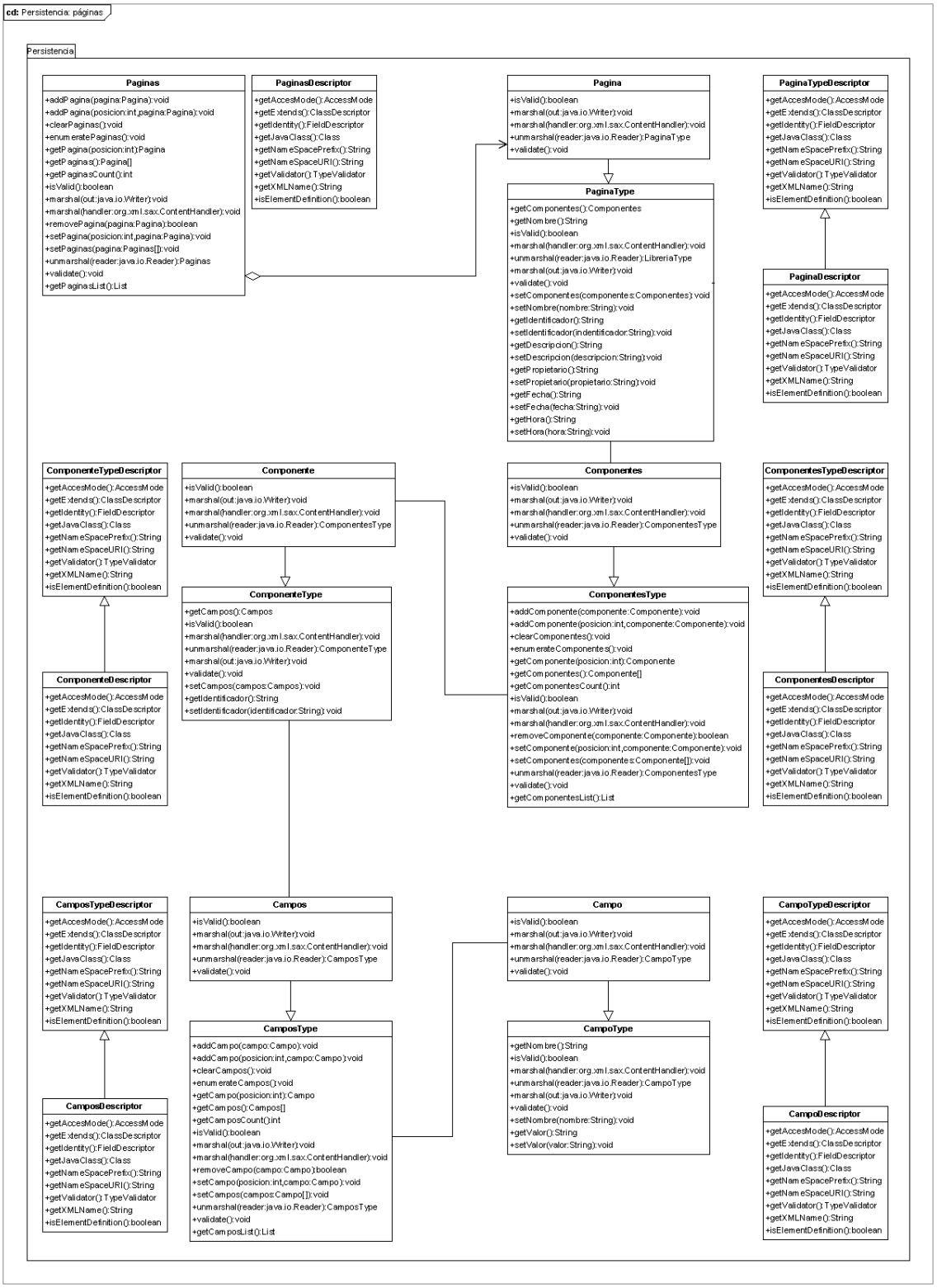

Figura 28. Diagrama de clases de la capa de persistencia: páginas.

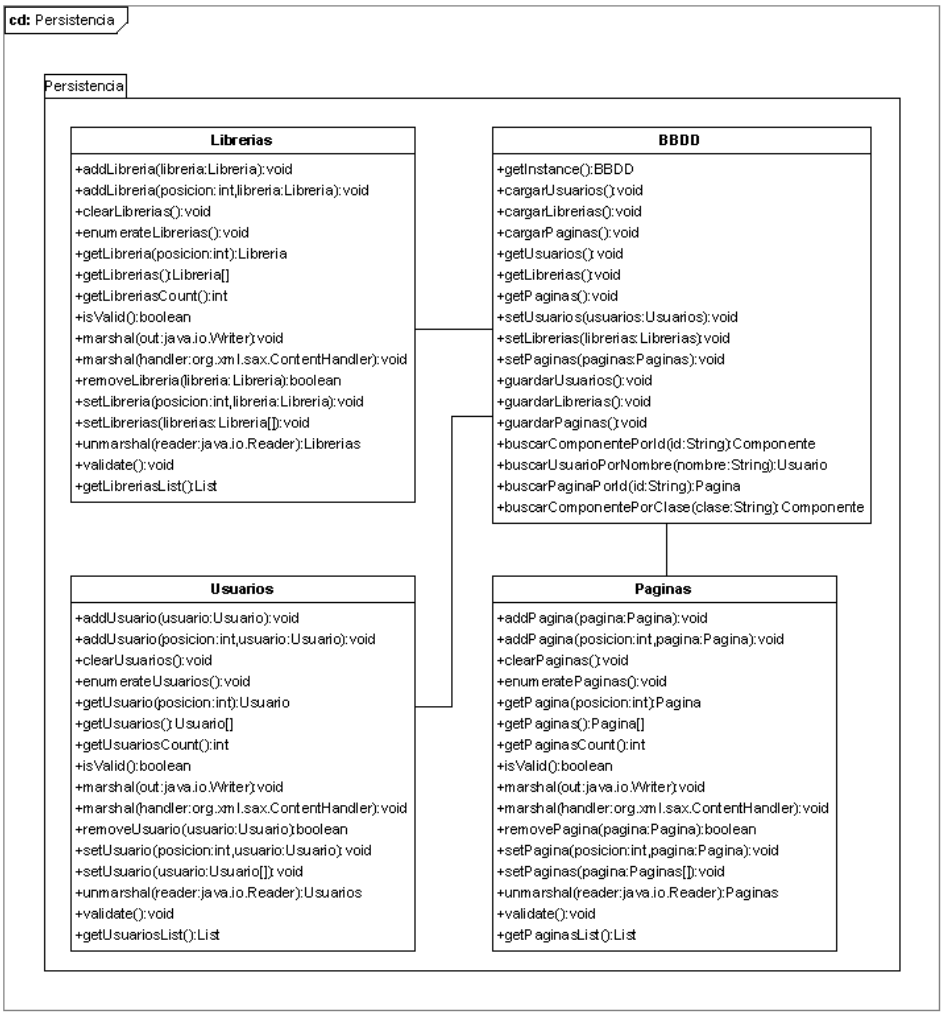

Figura 29. Diagrama de clases para la capa de persistencia.

En este último diagrama se puede observar lo más relevante de toda la capa de persistencia: la clase BBDD, que implementa el patrón *Singleton* (en el que sólo va a existir una instancia de la clase para toda la aplicación), encapsula todo el acceso a los datos almacenados en la "base de datos", siendo en este caso ficheros XML. Siendo común para toda la aplicación, sólo se recuperarán los datos del almacenamiento persistente la primera vez que sea accedido, estando desde entonces disponibles para las siguientes llamadas. Así mismo, se consigue que todos los objetos que intenten recuperar los usuarios, librerías o páginas obtengan siempre la misma referencia, de tal forma que se ahorre en reserva de memoria.

Modelo

La parte del Modelo es la que realmente lleva a cabo las funcionalidades solicitadas inicialmente. Se comunica con la capa de persistencia para recuperar los datos necesarios y, usando estos, responder a las peticiones realizadas por la parte de la Vista, redirigidas adecuadamente por el Controlador. Por tanto, estas serían las funcionalidades principales que debe cumplir el Modelo:

- Generar el componente JSF correspondiente al componente representado en la capa de persistencia.
- Generar el panel de configuración de los campos o atributos de un componente dado.
- Generar el gestor de componente para un componente dado, el cual está formado por el panel de configuración correspondiente y la representación final del componente JSF.
- Generar la paleta de todas las librerías y sus componentes disponibles en el sistema.
- Elaborar el lienzo con los componentes seleccionados que representa una página JSF.
- Comprobar los datos de acceso al sistema para un usuario determinado.
- Gestionar usuarios, componentes, librerías y páginas.
- Permitir la carga de una nueva librería de forma dinámica.

Para poder realizar todas estas funciones, se elabora el siguiente conjunto de clases:

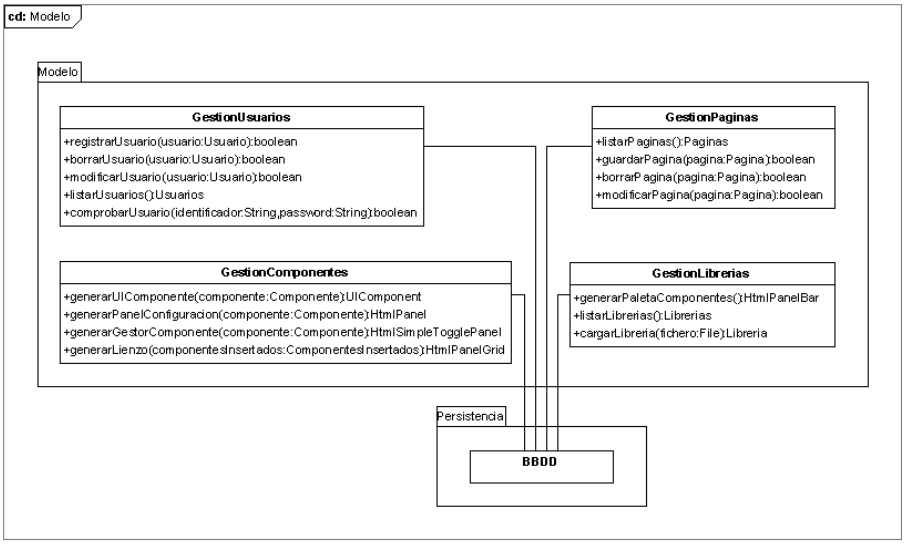

Figura 30. Diagrama de clases para la parte del Modelo.

Aunque no se encuentran propiamente en el paquete del Modelo, son utilizadas también las siguientes clases:

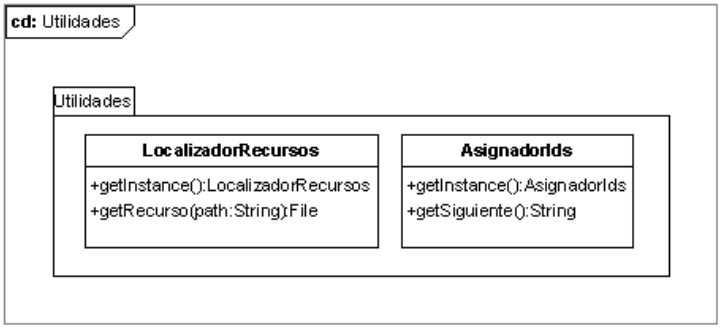

Figura 31. Diagrama de clases de utilidades.

- LocalizadorRecursos: se encarga de encapsular la recuperación de recursos internos de la aplicación web.
- AsignadorIds: genera identificadores diferentes para cada llamada realizada.

Vista y Controlador

Debido a la relación tan cercana que tienen estas dos partes, el diseño detallado de una de ellas tiene mucho que ver con la otra, por lo que se van a explicar de manera conjunta. La estructura de este apartado es la siguiente:

1. Descripción de páginas de la Vista.
2. Descripción de beans manejados utilizados por Vista y Controlador.
3. Reglas de navegación entre elementos de la Vista usadas por el Controlador.

En primera lugar, se van a introducir las páginas JSF que conforman la Vista. Tienen la extensión “xhtml” porque son gestionadas por el marco de trabajo Facelets:

Nombre de la página	Descripción
inicio.xhtml	Sirve de punto de entrada a la aplicación. Muestra un formulario donde introducir identificador de usuario y contraseña y permite la posibilidad de iniciar sesión o registrarse en el sistema.
registro.xhtml	Posibilita el registro de un usuario cualquiera en el sistema.
indexAdmin.xhtml	Representa el menú de opciones disponibles para el usuario administrador de la aplicación. Estas opciones son: modificar sus datos de usuario, administrar las librerías disponibles en el sistema, administrar los usuarios registrados y cerrar sesión.
modificarDatos.xhtml	Sirve para modificar los datos del usuario actual del sistema.
modificarUsuario.xhtml	Permite la modificación de los datos de un usuario cualquiera.

Nombre de la página	Descripción
adminLibrerias.xhtml	Permite realizar las siguientes operaciones con las librerías: cargar una nueva librearía, listar las existentes y deshabilitar y/o borrar componentes y/o librerías enteras.
adminUsuarios.xhtml	Gestiona el conjunto de usuarios registros, pudiendo listar, agregar, borrar o modificar los mismos.
indexUsuario.xhtml	Muestra las opciones disponibles para un usuario normal: administrar páginas, modificar datos y cerrar sesión.
adminPaginas.xhtml	Permite realizar las siguientes operaciones con las páginas: listarlas, crear una nueva, editar una existente o borrar una dada.
editarPagina.xhtml	Proporciona la funcionalidad de maquetación de una página JSF. Además, proporciona las opciones de guardar el estado actual, eliminar todos los componentes agregados a la misma, exportar el resultado o terminar de editarla.

Tabla 19. Páginas de la Vista.

A continuación, se van a describir los beans manejados utilizados por la aplicación. Aunque inicialmente se diseñó la arquitectura de forma que cada parte, Vista y Controlador, tuviese sus propios beans manejados, se llegó a la conclusión de que era más sencillo y eficiente concentrar la funcionalidad de ambos tipos de beans en uno sólo de ellos. Por tanto, los mismos beans que almacenan los datos temporales de la Vista durante el acceso de un usuario dejan disponibles al mismo tiempo las funcionalidades ofrecidas por el Modelo. Además, los beans de respaldo no existen como tales, es decir, no se definen inicialmente con la aplicación, sino que son definidos en tiempo de ejecución. Todos los beans se encuentran en el mismo subpaquete “beans”. Con esto explicado, este sería el conjunto de beans manejados usados:

Nombre del bean	Descripción
DatosRegistro	Almacena los datos relativos al registro de usuarios (identificador, contraseña, email) y permite llevarlo a cabo.
DatosInicio	Guarda los datos relativos al inicio de sesión (identificador, contraseña) y realiza las comprobaciones oportunas.
DatosUsuario	Almacena datos temporales sobre un usuario determinado (identificador), dejando disponibles también las opciones relativas a él (modificar, borrar).
DatosLibreria	Contiene la información relativa a una librería dada (nombre, componentes), así como las funciones correspondientes (modificar, borrar).
DatosPagina	Sirve de contenedor para los datos de una página (nombre, componentes insertados), dejando disponibles las funcionalidades relacionadas (guardar, borrar, exportar, etc.).
ComponentesInsertados	Contiene el listado de componentes insertados en la página actual.
ComponenteInsertado	Guarda los datos de un componente insertado en una página (identificador, valores de campos).

Tabla 20. Beans manejados utilizados por Vista y Controlador.

Por último, se muestra un diagrama de estados que ilustra las reglas de navegación entre páginas, usando etiquetas o identificadores de acción, que utiliza el Controlador. Los estados representan páginas JSF (o XHTML) y las entradas representan identificadores de acción.

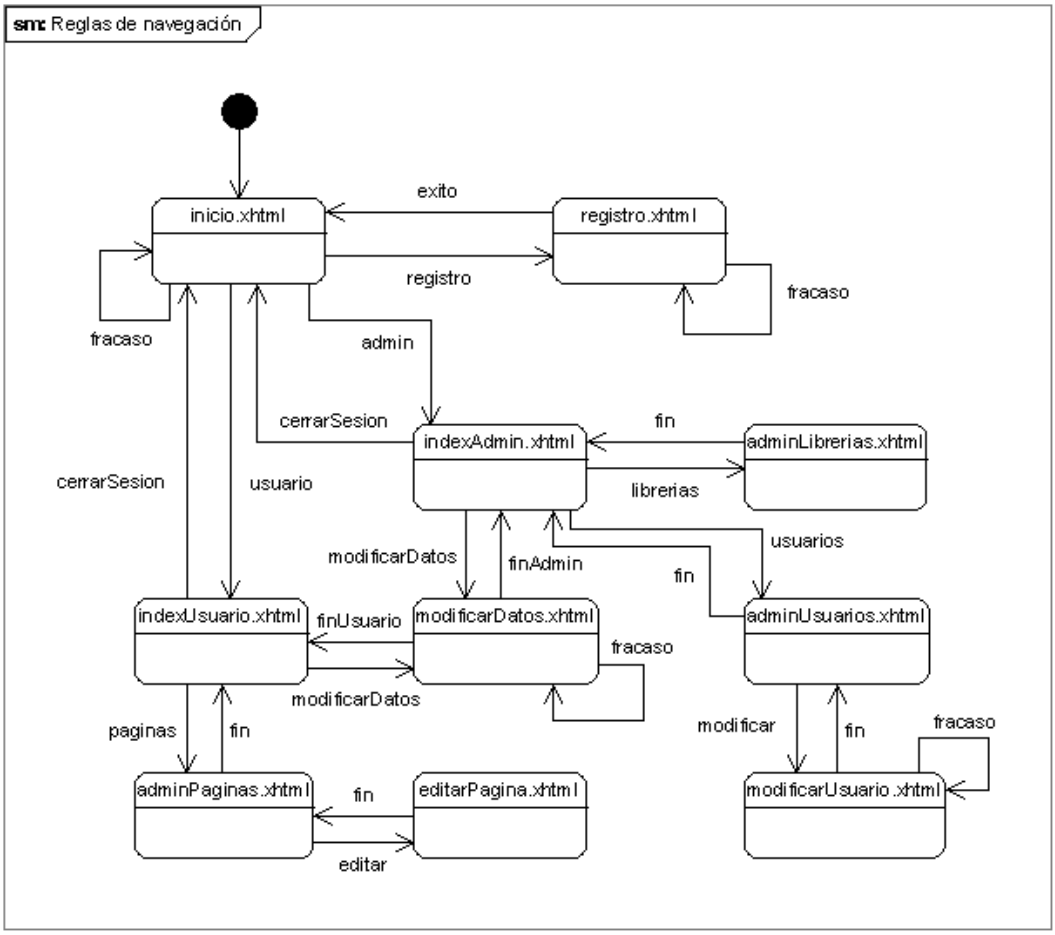

Figura 32. Reglas de navegación del Controlador.

Implementación

Durante la elaboración del sistema planteado y diseñado se han seguido unos detalles de implementación determinados. En esta sección se van a discutir algunos de ellos, como pueden ser las herramientas usadas durante el desarrollo y la organización interna de la aplicación.

Herramientas

Para poder llevar a cabo la implementación del sistema desarrollado se ha necesitado utilizar determinas herramientas. Estas herramientas o utilidades van desde entornos de desarrollo hasta aplicaciones externas, navegadores o editores:

- Eclipse 3.4.1: se ha utilizado Eclipse como IDE (*Integrated Development Environment*, Entorno de Desarrollo Integrado) por las ventajas que ofrece en comparación con sus competidores (Netbeans, JCreator, IntelliJ IDEA, Stylus Studio, etc). Entre estas, se pueden citar las siguientes: menor cantidad de recursos necesarios, disponibilidad de plugins o conectores para todas las librerías utilizadas en el proyecto y soporte para la mayor parte de servidores de aplicaciones J2EE existentes. Evidentemente, el código desarrollado puede ser llevado a cualquiera de los otros IDEs, por lo que tampoco limita la elección del IDE en gran medida.

- Java SE 1.6/6.0: la versión de Java utilizada tanto para compilar como para ejecutar la aplicación (levantar el servidor de aplicaciones) es la 6.0. Aún pudiendo conservar compatibilidad utilizando las versiones 5.0 o 1.4.2, se ha decidido optar por esta debido a la potencia del lenguaje desde la 5.0 y la necesidad del servidor de aplicaciones, como se verá más adelante, de utilizar la versión 6.0. Además, esta versión contiene todas las mejoradas incorporadas hasta la fecha, por lo que permite que la ejecución del sistema se beneficie de ello.

- Apache Tomcat 6.0.18: como servidor de aplicaciones/contenedor J2EE se ha optado por Apache Tomcat, en su versión 6.0.18. Se eligió Tomcat en primera instancia por ser uno de los servidores de aplicaciones *open-source* más utilizado hoy en día, fácil de configurar y con mayor documentación online. La versión utilizada sigue el mismo razonamiento que la versión de Java: utilizar la última versión ofrecida siempre significa beneficiarse de todas las mejoras que se hayan incorporado. La única limitación de esta versión, como ya se comentó antes, es la necesaria utilización del JDK 6.0.

- Vim: como editor de texto para los ficheros de esquemas XSD o ficheros XML se ha usado la herramienta Vim, en su versión para Windows. La elección de este editor de texto no puede ser argumentada en términos de funcionalidades o capacidades, ya que es completamente personal.

- Navegadores: para las pruebas de funcionamiento durante el proceso de implementación del sistema se utilizaron los navegadores Internet Explorer 6.0 y Mozilla Firefox 3.0, dado que representan a la gran mayoría de usuarios.

Organización

La organización interna del código del sistema depende en gran medida de la parte de la arquitectura diseñada en la que se enmarque cada elemento, así como de la funcionalidad que desempeña. La aplicación se encuentra estructurada en un único proyecto web de Eclipse, cuya jerarquía de directorios puede observarse en la Figura 33. El nombre en clave para el sistema ha sido "JSFComposer". Este nombre representa fielmente la principal funcionalidad requerida: componer páginas JSF.

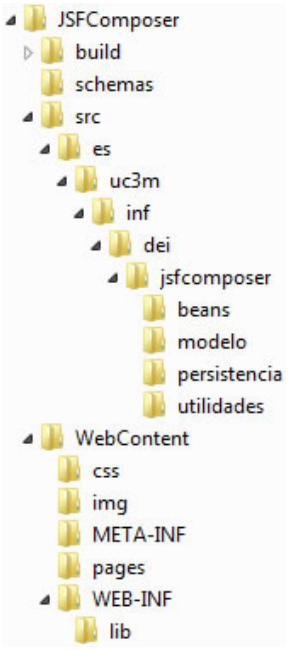

Figura 33. Organización proyecto Eclipse.

A partir de este proyecto web de Eclipse, la estructura de la aplicación web resultante es la siguiente:

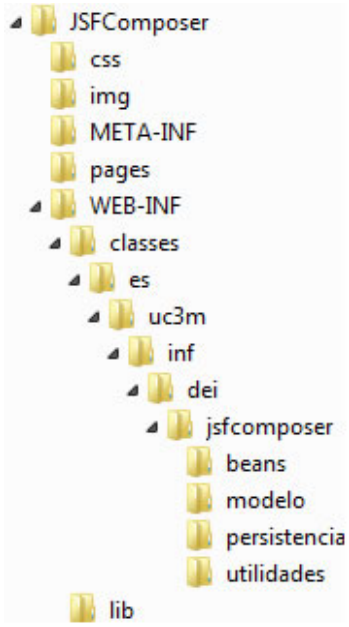

Figura 34. Organización aplicación web.

La descripción de cada directorio de la aplicación web, así como de los diferentes paquetes, puede verse en las tablas Tabla 21 y Tabla 22:

Directorio	Descripción
css	Contiene todas las hojas de estilo utilizadas por las páginas de la aplicación.
img	Almacena todas aquellas imágenes que utilice el sistema.
META-INF	Guarda el manifiesto descriptivo de la aplicación web.
pages	Contiene todas las páginas JSF/XHTML explicadas en la Tabla 19.
WEB-INF	Almacena todos aquellos recursos accesibles sólo por el servidor de aplicaciones y el propio código de la aplicación. Entre otros ficheros importantes, guarda aquellos XMLs que contienen las librerías, usuarios y páginas, así como la configuración de JSF respecto a las reglas de navegación.
WEB-INF\lib	Guarda todas las librerías necesarias para el correcto despliegue y funcionamiento del sistema.
WEB-INF\classes	Contiene el resultado de compilar todo el código fuente de la aplicación.

Tabla 21. Descripción de directorios de la aplicación web.

Paquete	Descripción
es.uc3m.inf.dei.jsfcomposer.beans	Este paquete contiene todos los beans manejados descritos en la Tabla 20.
es.uc3m.inf.dei.jsfcomposer.modelo	En este paquete se almacenan las clases descritas en la Figura 30.
es.uc3m.inf.dei.jsfcomposer.persistencia	Aquí se guardar todas las clases mostradas en las figuras Figura 26, Figura 27, Figura 28 y Figura 29.
es.uc3m.inf.dei.jsfcomposer.utilidades	Este paquete contiene las clases expuestas en la Figura 31.

Tabla 22. Descripción de paquetes de código fuente.

4.2 El producto del desarrollo

El sistema desarrollado sirve de herramienta para la composición de páginas JSF. En esta sección se va a describir brevemente este producto desarrollado, incluyendo capturas de prototipos de alto nivel. Además, se van a proporcionar los detalles necesarios para implantar el sistema implementado para dejarlo funcionando y disponible.

Descripción breve

El tipo de herramienta desarrollada es una aplicación web. Por tanto, es necesario utilizar un navegador web para poder acceder a ella. El punto inicial de acceso al sistema es la página inicio.xhtml, tal como se indica en la Figura 32. A través de ella se puede navegar por todo el sistema y realizar las funcionalidades requeridas inicialmente. Seguidamente, se muestra la estructura de menús disponibles en la aplicación, en función del rol de cada usuario:

- Inicio
  - Registro de usuario
  - Iniciar sesión
    - Menú de de administrador
      - Modificar datos
      - Cerrar sesión
      - Administrar usuarios
        - Crear usuario

- Listar usuarios
- Editar usuario
- Borrar usuario

- Administración librerías
  - Añadir librería
  - Listar librerías
  - Listar componentes
  - Deshabilitar/habilitar librería
  - Deshabilitar/habilitar componente
  - Eliminar librería
  - Eliminar componente

- Menú de usuario
  - Modificar datos
  - Cerrar sesión
  - Administrar páginas
    - Listar páginas
    - Editar página
    - Borrar página
    - Crear página
      - Agregar componente
      - Eliminar componente
      - Contraer/expandir componente
      - Reorganizar componente
      - Guardar página

- Borrar contenido página
- Exportar página

Es necesario aclarar que el producto final de la implementación del sistema es un fichero WAR que contiene la estructura descrita en la Figura 34 y que puede ser automáticamente desplegado en un contenedor de aplicaciones J2EE compatible. Debido a que el nombre en clave de este proyecto para los detalles de implementación ha sido JSFComposer, el fichero WAR resultante de todo el proceso de elaboración se llamará JSFComposer.war.

A continuación, se va a proceder a explicar las diferentes pantallas que se le muestran al usuario, proporcionando de esta forma una guía de referencia sobre los distintos escenarios que se puede encontrar.

Acceso a la aplicación

La página inicial a la que accede el usuario no autenticado es inicio.xhtml (Figura 35). En ella puede observarse la descripción general del sistema acompañada de un área específica para iniciar sesión. Además, puede hacer clic en el botón de "Registro" para comenzar el proceso de registro en el sistema.

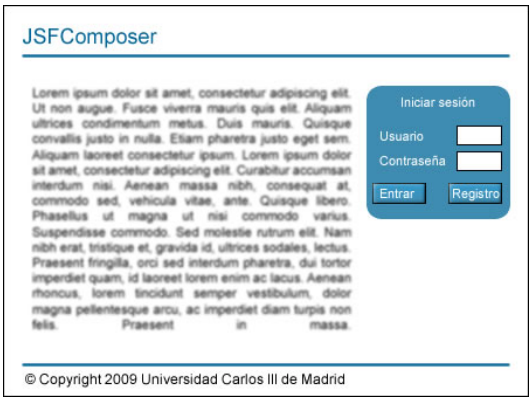

Figura 35. Captura de inicio.xhtml.

Registro de nuevo usuario

El proceso de alta de un nuevo usuario se realiza en la página registro.xhtml (Figura 36). En esta página se le pregunta al usuario qué nombre, contraseña y dirección de email desea introducir como información personal. Se le pide que contraseña especifique la contraseña dos veces para asegurar que no se producen errores de tecleo al introducirla. Una vez termine de rellenar este formulario, pulsando el botón de "Aceptar" se procederá a finalizar su registro, mostrando un mensaje si ocurriese algún error o redirigiendo a la página inicio.xhtml si todo saliese bien. También puede hacer clic en el botón "Cancelar" para volver a la página inicial sin finalizar el registro.

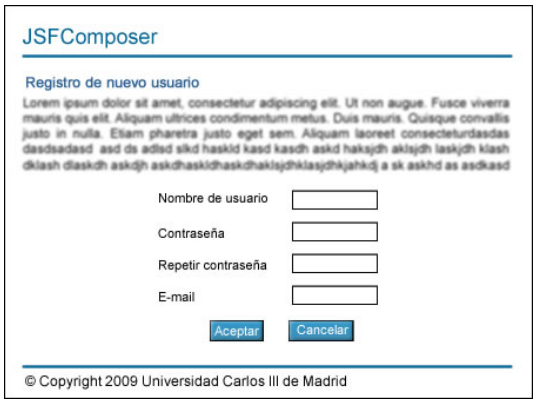

Figura 36. Captura de registro.xhtml.

Menú principal de Administrador

Una vez ha iniciado sesión el usuario, si es el administrador se encontrará con la página indexAdmin.xhtml (Figura 37). Se le presentan las siguientes opciones:

- Cerrar sesión: finaliza la sesión de este usuario y vuelve a la página inicial.
- Modificar datos: permite cambiar los datos personales del usuario.
- Administrar librerías: permite gestionar las librerías que contiene el sistema.
- Administrar usuarios: posibilita la gestión de los usuarios registrados.

Figura 37. Captura de indexAdmin.xhtml.

Modificación de datos propios

La opción de "Modificar datos" de los menús de administrador y usuario llevan a la página modificarDatos.xhtml (Figura 38 y Figura 39). El aspecto de la misma cambia en función del rol, permitiendo modificar usuario, contraseña y email en el caso de un usuario normal y dejando cambiar sólo contraseña y email en el caso del administrador.

Figura 38. Captura de modificarDatos.xhtml del Administrador.

Figura 39. Captura de modificarDatos.xhtml del usuario.

Administración de usuarios

La opción de administración de usuarios del menú del administrador lleva a la página adminUsuarios.xhtml (Figura 40). En esta es posible agregar un nuevo usuario, volver al menú, editar un usuario existente o borrar uno dado.

Figura 40. Captura de adminUsuarios.xhtml.

Modificación de datos ajenos

La edición de un usuario desde la página de administración de usuarios lleva a modificarUsuario.xhtml (Figura 41). Aquí se permite cambiar los datos de la misma forma que cuando se hace sobre el usuario actual, solo que ahora se hace sobre uno dado.

Figura 41. Captura de modificarUsuario.xhtml.

Administración de librerías

La última funcionalidad que tiene el administrador es la de gestionar las librerías incluidas en el sistema. Ésta se proporciona en la página adminLibrerias.xhtml (Figura 42). En ella es posible cargar una nueva librería, deshabilitar una dada o alguno de sus componentes y borrar una librería o alguno de sus componentes.

Figura 42. Captura de adminLibrerias.xhtml.

Menú principal del usuario

El menú principal del usuario se presenta en la página indexUsuario.xhtml (Figura 43). Las opciones que se le muestran son:

- Cerrar sesión: finaliza su sesión.
- Modificar datos: permite cambiar sus datos de usuario.
- Administrar páginas: gestión de las páginas creadas por el usuario.

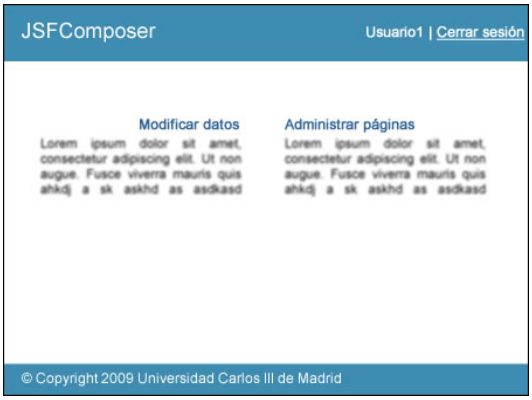

Figura 43. Captura de indexUsuario.xhtml.

Administración de páginas

La página adminPaginas.xhtml (Figura 44) permite gestionar las páginas creadas por el usuario. Se le permite crear una nueva (lo que le lleva a la página editarPagina.xhtml), editar una dada (que también lleva a la misma página pero con los componentes utilizados presentes) y borrar una página.

Figura 44. Captura de adminPaginas.xhtml.

Edición de página

La última funcionalidad que tiene el usuario es al mismo tiempo la más importante: la edición de una página. Esta se proporciona en la página editarPagina.xhtml (Figura 45). En ella es posible agregar nuevos componentes a la página en construcción, reorganizar los ya existentes, alterar sus propiedades, contraerlos/expandirlos e incluso eliminiarlos. Además, se puede limpiar el lienzo actual por completo, guardar el progreso alcanzado o exportar la página compuesta a su formato final: una página JSF válida que contiene las librerías y componentes configurados correctamente.

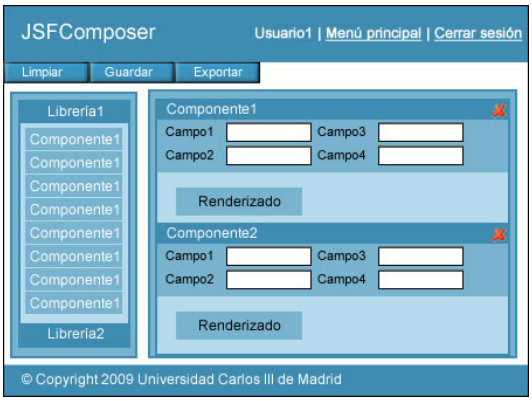

Figura 45. Captura de editarPagina.xhtml.

Vista previa

Desde la ventana editarPagina.xhtml se permite la generación de una vista preliminar de cómo quedaría la página JSF que se está maquetando. Esta visión previa representa fielmente el estado final de la página. Puede verse un ejemplo de esta visualización en la Figura 46:

Figura 46. Captura de una vista previa.

Exportación

El resultado de la exportación final de la página desarrollada es un fichero comprimido ZIP. Este fichero contiene por un lado la página JSF maquetada. Por otro lado, contiene también el conjunto de librerías (ficheros JAR) que incluyen los componentes utilizados en la página generada.

Detalles de implantación

El proceso de implantación consiste en la instalación definitiva del sistema desarrollado en el entorno final, sea este del tipo que sea. Esta instalación se traduce en la copia de todos los ficheros necesarios de la aplicación, así como de la copia y adaptación de los ficheros de configuración requeridos y la instalación de herramientas externas, para el correcto funcionamiento de la aplicación.

En el caso del sistema desarrollado, gracias a la utilización de ficheros XML como almacenamiento persistente, se consigue un alto grado de independencia del entorno de implantación. Por tanto, sólo es necesario desplegar el fichero WAR mencionado en el apartado anterior en un servidor de aplicaciones/contenedor J2EE compatible con el utilizado durante el desarrollo, Apache Tomcat 6.0.18, así como poseer la versión 6.0 del kit de desarrollo Java de Sun instalada en el entorno final. Con todo esto, la aplicación elaborada puede ser desplegada automáticamente, quedando disponible en la siguiente dirección URL:

http://<dirección>:<puerto>/JSFComposer

Donde “dirección” indica la dirección IP o nombre del servidor que contiene la aplicación y “puerto” es el puerto asociado al servidor de aplicaciones, contenido en el citado servidor, en el que se despliega la aplicación desarrollada. Además, los datos iniciales de acceso al sistema en modo administrador son:

- Usuario: admin
- Contraseña: admin*pass

5 Evaluación

Este capítulo tiene como función principal demostrar la validez de la solución elaborada. Para ello, en primer lugar, se va a introducir el proceso de evaluación seguido, compuesto por la forma de evaluación y los casos de prueba. Después, se procederá a realizar el análisis de los resultados obtenidos durante este proceso de evaluación. De estos resultados dependerá la calidad y validez del sistema implementado para resolver el problema planteado inicialmente.

5.1 Proceso de evaluación

El proceso de evaluación seguido para analizar la validez de la solución desarrollada sigue una metodología determinada. Esta metodología o forma de evaluación se va a explicar a continuación, para después proceder a describir los casos de prueba establecidos.

Forma de evaluación

Es importante seleccionar la forma de evaluación adecuada a cada tipo de proyecto. Hay algunos en los que el flujo de ejecución está muy definido y los resultados son completamente deterministas. En este tipo de proyectos pueden utilizarse una evaluación basada en el uso de pruebas unitarias, las cuales realizan operaciones con el sistema de las que se conoce el resultado correcto con completa seguridad. Estas pruebas unitarias evaluarían el nivel de “acierto” o validez de los resultados reales que arrojaría el sistema comparándolos con los esperados.

Otro tipo de proyectos no siguen una línea o flujo de ejecución tan definido, sino que por la interfaz externa que proporcionan o por la naturaleza del sistema en sí, disponen de otra manera de ser utilizados. Esto provoca que la utilización del tipo de evaluación introducido antes, el basado en pruebas unitarias, genere una cantidad inmensa de casos de prueba y, lo que es más preocupante, que muchos de ellos sean redundantes o innecesarios. Además, no garantiza la completitud de las pruebas. Un tipo de aplicación o sistema que se ajustaría a esta descripción es el de las aplicaciones web. La naturaleza de este sistema posibilita evaluar determinadas capas o módulos usando pruebas unitarias, pero no todo el sistema. Es por esto que se necesita una forma más eficaz y adecuada para evaluar este tipo de sistemas.

La forma de evaluación de las aplicaciones de esta clase puede ser realizada basándose en las funcionalidades que el sistema debería proporcionar (y evidentemente controlando su resultado) y en los objetivos marcados durante su concepción.

De esta forma, la evaluación del sistema elaborado se va a centrar en el aspecto funcional del mismo, es decir, se va a analizar si la aplicación desarrollada satisface los objetivos marcados inicialmente y si permite llevar a cabo satisfactoriamente los casos de uso definidos durante la fase de análisis.

Casos de prueba

Tal como se ha explicado en el apartado anterior, se va a evaluar el producto elaborado desde un punto de vista funcional. Es por esto que los casos de prueba van a consistir en el correcto cumplimiento de los objetivos marcados (sección 1.2, Objetivos) y en la satisfactoria realización de los casos de uso definidos en el análisis (Modelo de sistema). De esta forma, distinguiríamos dos tipos de pruebas:

- Cumplimiento de objetivo.
- Realización de caso de uso.

Siguiendo esta separación, en las tablas que siguen se muestran los casos de prueba seleccionados para el sistema desarrollado. El identificador de caso de prueba sigue el formato "CP_<tipo>_XX", donde "tipo" indica si es de cumplimiento de objetivo (O) o de realización de caso de uso (U) y XX es un número de dos cifras autoincrementativo. Al mismo tiempo, se va a describir la manera de implementar cada caso de prueba explicado: el/los recurso/s físico/s en el/los que se realiza, la entrada esperada, el procesamiento intermedio realizado y la salida deseada:

Caso de prueba	**CP_O_01**
Descripción	Facilitar la elaboración de una página JSF a partir de componentes de librerías desconocidas.
Recursos involucrados	▪ adminPaginas.xhtml ▪ editarPagina.xhtml ▪ Beans manejados relacionados. ▪ Paquete del Modelo.
Entrada recibida	▪ Componentes desconocidos seleccionados.
Procesamiento a realizar	1. El usuario inicia sesión. 2. Accede a la administración de sus páginas 3. Crea o edita una página ya existente. 4. Agrega una serie de componentes a la página con los que nunca antes había trabajado. 5. Exporta la página satisfactoriamente sin preocuparse de cómo se han insertado dichos componentes.

Salida esperada	▪ Página JSF válida con los componentes seleccionados correctamente agregados y configurados, así como organizados en el orden utilizado al maquetar la página.

Tabla 23. Explicación de caso de prueba CP_O_01.

Caso de prueba	**CP_O_02**
Descripción	Agilizar el proceso de selección, inclusión y configuración de componentes JSF.
Recursos involucrados	▪ adminPaginas.xhtml ▪ editarPagina.xhtml ▪ Beans manejados relacionados. ▪ Paquete del Modelo.
Entrada recibida	▪ Componentes desconocidos seleccionados. ▪ Configuración determinada para dichos componentes.
Procesamiento a realizar	1. El usuario inicia sesión. 2. Accede a la administración de sus páginas 3. Crea o edita una página ya existente. 4. Agrega una serie de componentes a la página con los que nunca antes había trabajado. 5. Personaliza las propiedades de estos componentes a según la configuración deseada.
Salida esperada	▪ Página JSF guardada con los componentes seleccionados por el usuario, correctamente configurados.

Tabla 24. Explicación de caso de prueba CP_O_02.

Caso de prueba	**CP_O_03**
Descripción	Permitir la incorporación de librerías genéricas de componentes.
Recursos involucrados	▪ adminLibrerias.xhtml ▪ Beans manejados relacionados ▪ Paquete del Modelo.

Entrada recibida	▪ Ficheros JAR de librerías de componentes no probadas anteriormente con la aplicación.
Procesamiento a realizar	1. El administrador inicia sesión en el sistema. 2. Accede a la página de administración de librerías. 3. Selecciona la librería que desea incorporar y la carga en el sistema utilizando el formulario correspondiente.
Salida esperada	▪ La librería aparece mostrada correctamente en la página de administración de librerías. ▪ Los componentes de la librería se encuentran listados junto a ella.

Tabla 25. Explicación de caso de prueba CP_O_03.

Caso de prueba	**CP_O_04**
Descripción	Ser accesible vía web, *Cross-browser* (Internet Explorer y Mozilla Firefox únicamente), portable y configurable.
Recursos involucrados	▪ Implementación del sistema como aplicación web J2EE desplegable en un servidor de aplicaciones compatible con Tomcat ▪ Compatibilidad de las páginas desarrolladas con los navegadores más utilizados.
Entrada recibida	▪ Fichero WAR de la aplicación. ▪ Diferentes máquinas (Ubuntu Linux y Microsoft Windows XP) con servidores de aplicaciones (Apache Tomcat) para implantar el sistema. ▪ Diferentes navegadores (Internet Explorer 6.0 y Mozilla Firefox 2.0) en diferentes sistemas operativos (Ubuntu Linux y Microsoft Windows XP) para realizar pruebas de acceso al sistema. ▪ Varias configuraciones de librerías.
Procesamiento a realizar	1. Desplegar el fichero WAR de la aplicación en las diferentes máquinas. 2. Realizar la totalidad de los casos de prueba listados en esta sección en cada unos de los clientes accediendo a cada una de las máquinas, probando también con las diferentes configuraciones de librerías.
Salida esperada	▪ El mismo resultado visual y de funcionalidad en cada uno de los clientes y para cada una de las máquinas.

Tabla 26. Explicación de caso de prueba CP_O_04.

Caso de prueba	CP_U_01
Descripción	Registro de usuario no autenticado.
Recursos involucrados	▪ inicio.xhtml ▪ registro.xhtml ▪ Beans asociados.
Entrada recibida	▪ Nombre de usuario. ▪ Contraseña (por duplicado). ▪ Dirección de correo electrónico.
Procesamiento a realizar	1. Un usuario no autenticado accede al sistema. 2. Selecciona la opción de Registro. 3. Introduce sus datos de usuario y presiona el botón de aceptar.
Salida esperada	▪ Fichero XML de usuarios modificado de manera que siga el esquema especificado en la sección Persistencia y que contenga los datos introducidos.

Tabla 27. Explicación de caso de prueba CP_U_01.

Caso de prueba	CP_U_02
Descripción	Inicio de sesión de usuario no autenticado.
Recursos involucrados	▪ inicio.xhtml ▪ Bean DatosInicio.
Entrada recibida	▪ Nombre de usuario. ▪ Contraseña de acceso.
Procesamiento a realizar	1. El usuario no autenticado accede al sistema. 2. Introduce sus datos en las casillas correspondientes y presiona el botón de inicio de sesión.
Salida esperada	▪ Acceso a la página que contiene el menú del usuario normal (indexUsuario.xhtml).

Tabla 28. Explicación de caso de prueba CP_U_02.

Caso de prueba	CP_U_03
Descripción	Modificación de datos de usuario autenticado.
Recursos involucrados	▪ modificarDatos.xhtml ▪ Bean DatosInicio.
Entrada recibida	▪ Datos de acceso del usuario. ▪ Nuevos valores para los datos a modificar.
Procesamiento a realizar	1. El usuario no autenticado accede al sistema. 2. Introduce sus datos de acceso y presiona el botón de inicio de sesión. 3. Selecciona la opción de Modificar datos. 4. Introduce los nuevos valores en las casillas correspondientes y presiona el botón de aceptar.
Salida esperada	▪ El fichero XML de usuarios modificado de manera acorde.

Tabla 29. Explicación de caso de prueba CP_U_03.

Caso de prueba	CP_U_04
Descripción	Cierre de sesión de usuario autenticado.
Recursos involucrados	▪ indexAdmin.xhtml ▪ indexUsuario.xhtml
Entrada recibida	▪ Datos de acceso de usuario.
Procesamiento a realizar	1. El usuario inicia sesión en el sistema. 2. En cualquier momento, desde su menú principal utiliza la opción de cerrar sesión.
Salida esperada	▪ Cualquier acceso futuro a recursos privilegiados (páginas para las cuales se necesita estar autenticado) redirigirán automáticamente a la página de acceso inicial al sistema.

Tabla 30. Explicación de caso de prueba CP_U_04.

Caso de prueba	CP_U_05
Descripción	Listado de usuarios de usuario administrador.
Recursos involucrados	▪ adminUsuarios.xhtml
Entrada recibida	▪ Datos de acceso de administrador.
Procesamiento a realizar	1. El usuario administrador se autentica en el sistema e inicia sesión. 2. Desde su menú principal, accede a la opción de administración de usuarios.
Salida esperada	▪ Listado de todos los usuarios contenidos en el fichero XML de usuarios, salvo el administrador.

Tabla 31. Explicación de caso de prueba CP_U_05.

Caso de prueba	CP_U_06
Descripción	Agregación de usuario de usuario administrador.
Recursos involucrados	▪ adminUsuarios.xhtml
Entrada recibida	▪ Datos de acceso de administrador. ▪ Datos de usuario a agregar.
Procesamiento a realizar	1. El usuario administrador se autentica en el sistema e inicia sesión. 2. Desde su menú principal, accede a la opción de administración de usuarios. 3. En esta página introduce los datos del nuevo usuario y presiona el botón de agregar.
Salida esperada	▪ Fichero XML de usuarios correctamente modificado.

Tabla 32. Explicación de caso de prueba CP_U_06.

Caso de prueba	CP_U_07
Descripción	Modificación de usuario de usuario administrador.
Recursos involucrados	▪ adminUsuarios.xhtml
Entrada recibida	▪ Datos de acceso de administrador. ▪ Nombre o identificador de usuario a modificar.

	▪ Nuevos valores para los datos del usuario a modificar.
Procesamiento a realizar	1. El usuario administrador accede al sistema. 2. Desde su menú principal, accede a la opción de administración de usuarios. 3. Busca el usuario con el nombre del que quiere modificar y pulsa en el botón de edición. 4. En la página de modificación de datos que aparece, cambia sus valores actuales por los nuevos.
Salida esperada	▪ Fichero XML correctamente actualizado.

Tabla 33. Explicación de caso de prueba CP_U_07.

Caso de prueba	**CP_U_08**
Descripción	Borrado de usuario de usuario administrador.
Recursos involucrados	▪ adminUsuarios.xhtml
Entrada recibida	▪ Datos de acceso de administrador. ▪ Nombre o identificador de usuario a modificar.
Procesamiento a realizar	1. El usuario administrador accede al sistema. 2. Desde su menú principal, accede a la opción de administración de usuarios. 3. Busca el usuario con el nombre del que quiere modificar y pulsa en el botón de borrar. 4. Confirma el diálogo de confirmación que aparece.
Salida esperada	▪ Fichero XML correctamente actualizado.

Tabla 34. Explicación de caso de prueba CP_U_08.

Caso de prueba	**CP_U_09**
Descripción	Listado de librerías de usuario administrador.
Recursos involucrados	▪ adminLibrerias.xhtml
Entrada recibida	▪ Datos de acceso de administrador.

Procesamiento a realizar	1. El usuario administrador accede al sistema. 2. Desde su menú principal, accede a la opción de administración de librerías.
Salida esperada	▪ Listado de todas las librerías y sus componentes contenidos en el fichero XML de librerías.

Tabla 35. Explicación de caso de prueba CP_U_09.

Caso de prueba	**CP_U_10**
Descripción	Agregación de librería de usuario administrador.
Recursos involucrados	▪ adminLibrerias.xhtml
Entrada recibida	▪ Fichero JAR de librería de componentes.
Procesamiento a realizar	1. El administrador inicia sesión en el sistema. 2. Accede a la página de administración de librerías. 3. Selecciona la librería que desea incorporar y la carga en el sistema utilizando el formulario correspondiente.
Salida esperada	▪ Fichero XML de librerías actualizado correctamente.

Tabla 36. Explicación de caso de prueba CP_U_10.

Caso de prueba	**CP_U_11**
Descripción	Deshabilitado de librería de usuario administrador.
Recursos involucrados	▪ adminLibrerias.xhtml
Entrada recibida	▪ Nombre de librería a deshabilitar.
Procesamiento a realizar	1. El administrador inicia sesión en el sistema. 2. Accede a la página de administración de librerías. 3. Busca la librería deseada y pulsa en el botón de deshabilitar.
Salida esperada	▪ Fichero XML de librerías correctamente actualizado.

Tabla 37. Explicación de caso de prueba CP_U_11.

Caso de prueba	**CP_U_12**
Descripción	Borrado de librería de usuario administrador.

Recursos involucrados	▪ adminLibrerias.xhtml
Entrada recibida	▪ Nombre de librería a borrar.
Procesamiento a realizar	1. El administrador inicia sesión en el sistema. 2. Accede a la página de administración de librerías. 3. Busca la librería deseada y pulsa en el botón de borrado. 4. Confirma el diálogo de confirmación que aparece.
Salida esperada	▪ Fichero XML de librerías correctamente actualizado.

Tabla 38. Explicación de caso de prueba CP_U_12.

Caso de prueba	**CP_U_13**
Descripción	Listado de componentes de usuario administrador.
Recursos involucrados	▪ adminLibrerias.xhtml
Entrada recibida	▪ Datos de acceso de administrador.
Procesamiento a realizar	1. El administrador inicia sesión en el sistema. 2. Accede a la página de administración de librerías.
Salida esperada	▪ Listado de todos los componentes incluidos en el fichero XML de librerías.

Tabla 39. Explicación de caso de prueba CP_U_13.

Caso de prueba	**CP_U_14**
Descripción	Deshabilitado de componente de usuario administrador.
Recursos involucrados	▪ adminLibrerias.xhtml
Entrada recibida	▪ Nombre o identificador de componente a deshabilitar.
Procesamiento a realizar	1. El administrador inicia sesión en el sistema. 2. Accede a la página de administración de librerías. 3. Busca el componente deseado y pulsa en la opción de deshabilitar.
Salida esperada	▪ Fichero XML de librerías actualizado correctamente.

Tabla 40. Explicación de caso de prueba CP_U_14.

Caso de prueba	CP_U_15
Descripción	Borrado de componente de usuario administrador.
Recursos involucrados	▪ adminLibrerias.xhtml
Entrada recibida	▪ Nombre o identificador de componente a borrar.
Procesamiento a realizar	1. El administrador inicia sesión en el sistema. 2. Accede a la página de administración de librerías. 3. Busca el componente deseado y pulsa en la opción de borrado.
Salida esperada	▪ Fichero XML de librerías actualizado correctamente.

Tabla 41. Explicación de caso de prueba CP_U_15.

Caso de prueba	CP_U_16
Descripción	Listado de páginas de usuario.
Recursos involucrados	▪ adminPaginas.xhtml
Entrada recibida	▪ Datos de acceso de usuario.
Procesamiento a realizar	1. El usuario inicia sesión en el sistema utilizando sus datos. 2. Accede a la página de administración de páginas.
Salida esperada	▪ Listado de las páginas creadas por este usuario, contenidas en el fichero XML de páginas.

Tabla 42. Explicación de caso de prueba CP_U_16.

Caso de prueba	CP_U_17
Descripción	Creación de página de usuario.
Recursos involucrados	▪ adminPaginas.xhtml
Entrada recibida	▪ Datos de acceso de usuario. ▪ Nombre de la página. ▪ Descripción de la página.

Procesamiento a realizar	1. El usuario inicia sesión en el sistema utilizando sus datos. 2. Accede a la página de administración de páginas. 3. Pulsa en el botón de nueva página. 4. Introduce el nombre y la descripción de la página en el diálogo que aparece. 5. Comienza a agregar componentes a la misma.
Salida esperada	▪ Fichero XML de páginas correctamente actualizado.

Tabla 43. Explicación de caso de prueba CP_U_17.

Caso de prueba	**CP_U_18**
Descripción	Edición de página de usuario.
Recursos involucrados	▪ editarPagina.xhtml
Entrada recibida	▪ Datos de acceso de usuario. ▪ Nombre o identificador de la página.
Procesamiento a realizar	1. El usuario accede al sistema utilizando sus datos de acceso. 2. Accede a la página de administración de páginas. 3. Busca la página deseada y pulsa en la opción de edición.
Salida esperada	▪ Página mostrada en el estado en el que se guardó.

Tabla 44. Explicación de caso de prueba CP_U_18.

Caso de prueba	**CP_U_19**
Descripción	Borrado de página de usuario.
Recursos involucrados	▪ adminPaginas.xhtml
Entrada recibida	▪ Datos de acceso de usuario. ▪ Nombre o identificador de la página.
Procesamiento a realizar	1. El usuario accede al sistema utilizando sus datos de acceso. 2. Accede a la página de administración de páginas. 3. Busca la página deseada y pulsa en la opción de borrado.

	4. Confirma el diálogo de confirmación que aparece.
Salida esperada	▪ Fichero XML de páginas correctamente actualizado.

Tabla 45. Explicación de caso de prueba CP_U_19.

Caso de prueba	**CP_U_20**
Descripción	Listado de componentes agregados de usuario.
Recursos involucrados	▪ editarPagina.xhtml
Entrada recibida	▪ Datos de acceso de usuario. ▪ Nombre o identificador de página.
Procesamiento a realizar	1. El usuario inicia sesión en el sistema. 2. Accede a la página de administración de páginas. 3. Busca la página deseada y pulsa en el botón de edición.
Salida esperada	▪ Listado de componentes agregados a la página dada en el momento en el que se guardó.

Tabla 46. Explicación de caso de prueba CP_U_20.

Caso de prueba	**CP_U_21**
Descripción	Agregación de componente de usuario.
Recursos involucrados	▪ editarPagina.xhtml
Entrada recibida	▪ Nombre de librería y componente.
Procesamiento a realizar	1. El usuario accede al sistema. 2. Accede a la página de administración de páginas. 3. Edita una página existente o crea una nueva. 4. Busca el componente deseado y lo agrega a la página en edición.
Salida esperada	▪ Fichero XML de páginas actualizado correctamente.

Tabla 47. Explicación de caso de prueba CP_U_21.

Caso de prueba	**CP_U_22**
Descripción	Borrado de componente de usuario.

Recursos involucrados	▪ editarPagina.xhtml
Entrada recibida	▪ Nombre de componente.
Procesamiento a realizar	1. El usuario inicia sesión en el sistema. 2. Accede a la página de administración de páginas. 3. Edita una página existente. 4. Busca el componente que desea eliminar de la página y pulsa en el botón de borrar.
Salida esperada	▪ Fichero XML de páginas actualizado correctamente.

Tabla 48. Explicación de caso de prueba CP_U_22.

Caso de prueba	**CP_U_23**
Descripción	Movimiento de componente de usuario.
Recursos involucrados	▪ editarPagina.xhtml
Entrada recibida	▪ Nombre de componente insertado.
Procesamiento a realizar	1. El usuario inicia sesión en el sistema. 2. Accede a la página de administración de páginas. 3. Edita la página deseada. 4. Busca el componente insertado deseado y lo reorganiza.
Salida esperada	▪ Fichero XML de páginas actualizado correctamente.

Tabla 49. Explicación de caso de prueba CP_U_23.

Caso de prueba	**CP_U_24**
Descripción	Borrado de componentes de usuario.
Recursos involucrados	▪ editarPagina.xhtml
Entrada recibida	▪ Nombre de página.
Procesamiento a realizar	1. El usuario inicia sesión en el sistema. 2. Accede a la página de administración de páginas.

	3. Edita la página deseada. 4. Pulsa la opción de borrar todos los componentes insertados. 5. Confirma el diálogo de confirmación que aparece.
Salida esperada	▪ Fichero XML de páginas correctamente actualizado.

Tabla 50. Explicación de caso de prueba CP_U_24.

Caso de prueba	**CP_U_25**
Descripción	Muestra de propiedades de componente de usuario.
Recursos involucrados	▪ editarPagina.xhtml
Entrada recibida	▪ Nombre de página. ▪ Nombre de componente.
Procesamiento a realizar	1. El usuario inicia sesión en el sistema. 2. Accede a la página de administración de páginas. 3. Edita la página deseada. 4. Selecciona el componente deseado y muestra sus propiedades.
Salida esperada	▪ Valores de configuración del componente dado almacenados en el fichero XML de páginas.

Tabla 51. Explicación de caso de prueba CP_U_25.

Caso de prueba	**CP_U_26**
Descripción	Modificación de propiedades de componente de usuario.
Recursos involucrados	▪ editarPagina.xhtml
Entrada recibida	▪ Nombre de página. ▪ Nombre de componente. ▪ Nuevos valores de configuración para el componente.
Procesamiento a realizar	1. El usuario inicia sesión en el sistema. 2. Accede a la página de administración de páginas. 3. Edita la página deseada.

	4. Selecciona el componente deseado y modifica sus propiedades.
Salida esperada	▪ Fichero XML de páginas actualizado correctamente.

Tabla 52. Explicación de caso de prueba CP_U_26.

Caso de prueba	**CP_U_27**
Descripción	Expansión de componente de usuario.
Recursos involucrados	▪ editarPagina.xhtml
Entrada recibida	▪ Nombre página. ▪ Nombre de componente.
Procesamiento a realizar	1. El usuario inicia sesión en el sistema. 2. Accede a la página de administración de páginas. 3. Edita la página deseada. 4. Selecciona el componente deseado (contraído inicialmente) y lo expande utilizando el botón correspondiente.
Salida esperada	▪ Representación visual del contenedor del componente actualizada.

Tabla 53. Explicación de caso de prueba CP_U_27.

Caso de prueba	**CP_U_28**
Descripción	Contracción de componente de usuario.
Recursos involucrados	▪ editarPagina.xhtml
Entrada recibida	▪ Nombre de página. ▪ Nombre de componente.
Procesamiento a realizar	1. El usuario inicia sesión en el sistema. 2. Accede a la página de administración de páginas. 3. Edita la página deseada. 4. Selecciona el componente deseado (expandido inicialmente) y lo contrae utilizando el botón correspondiente.
Salida esperada	▪ Representación visual del contenedor del componente actualizada.

Tabla 54. Explicación de caso de prueba CP_U_28.

Caso de prueba	CP_U_29
Descripción	Exportación de página de usuario.
Recursos involucrados	▪ editarPagina.xhtml
Entrada recibida	▪ Nombre de página.
Procesamiento a realizar	1. El usuario inicia sesión en el sistema. 2. Accede a la página de administración de páginas. 3. Edita la página deseada. 4. Pulsa la opción de exportar y guarda el fichero resultante.
Salida esperada	▪ Fichero JSF que contiene la página maquetada, con todos los componentes insertados debidamente configurados y ordenados.

Tabla 55. Explicación de caso de prueba CP_U_29.

Caso de prueba	CP_U_30
Descripción	Guardado de página de usuario.
Recursos involucrados	▪ editarPagina.xhtml
Entrada recibida	▪ Nombre de página.
Procesamiento a realizar	1. El usuario inicia sesión en el sistema. 2. Accede a la página de administración de páginas. 3. Edita la página deseada. 4. Pulsa la opción de guardar.
Salida esperada	▪ Fichero XML de páginas correctamente actualizado.

Tabla 56. Explicación de caso de prueba CP_U_30.

5.2 Análisis de resultados

Tras la ejecución de los casos de prueba descritos en la sección anterior, se han generado una serie de resultados. En la Tabla 57 se muestra el desglose de los mismos. Después, se ofrecerá una justificación del fracaso de determinados casos.

Caso de prueba	Resultado
CP_O_01	Exitoso
CP_O_02	Exitoso
CP_O_03	Exitoso
CP_O_04	Exitoso
CP_U_01	Exitoso
CP_U_02	Exitoso
CP_U_03	Exitoso
CP_U_04	Exitoso
CP_U_05	Exitoso
CP_U_06	Exitoso
CP_U_07	Exitoso
CP_U_08	Exitoso
CP_U_09	Exitoso
CP_U_10	Exitoso
CP_U_11	Exitoso
CP_U_12	Exitoso
CP_U_13	Exitoso
CP_U_14	Exitoso
CP_U_15	Exitoso
CP_U_16	Exitoso
CP_U_17	Exitoso
CP_U_18	Exitoso
CP_U_19	Exitoso
CP_U_20	Exitoso
CP_U_21	Fallido
CP_U_22	Exitoso
CP_U_23	Exitoso
CP_U_24	Exitoso
CP_U_25	Exitoso
CP_U_26	Fallido
CP_U_27	Exitoso
CP_U_28	Exitoso
CP_U_29	Exitoso

Tabla 57. Resultado de evaluación de casos de prueba.

Como puede observarse en la Tabla 57, la ejecución de los siguientes casos de prueba ha resultado en fallo:

- CP_U_21: determinados componentes JSF requieren como configuración inicial uno o más beans manejados presentes para poder funcionar en condiciones. Debido a que esto no está contemplado por la aplicación, el uso de este tipo de componentes provoca errores en tiempo de ejecución.

- CP_U_26: en determinados componentes, tales como los afectados en el caso de prueba CP_U_21, la modificación de sus propiedades de manera errónea (no conteniendo un valor del formato necesario, no existiendo un bean manejado de respaldo, etc.) lleva a un estado inestable de la aplicación.

Para solucionar los casos de prueba fallidos recién mencionados se toman dos medidas:

- Limitación de los componentes disponibles en la versión implantable: se restringe el uso de los componentes que causan problemas en los citados casos de prueba CP_U_21 y CP_U_26. Esto se realiza desde la propia aplicación, deshabilitando los componentes conflictivos.

- Propuesta de trabajo futuro: como posible ampliación futura se comentará la remodelación o extensión del sistema desarrollado de forma que gestione correctamente y dé soporte a este tipo de componentes.

Una vez tomadas estas medidas, todos los casos de prueba se ejecutarían con resultado satisfactorio, por lo que el producto desarrollado cumpliría todos los objetivos y limitaciones marcadas y resolvería correctamente el problema planteado.

6 Conclusión

La descripción del trabajo finaliza con una recapitulación de los hitos alcanzados durante la realización del mismo. La recapitulación se inicia con una visión sobre diferentes aspectos del trabajo realizado, como pueden ser aportaciones realizadas o trabajos futuros planteables. Su cometido es el de permitir al autor realizar una auto-evaluación del trabajo desarrollado, así como plantearse posibles mejoras o extensiones, todo ello desde un punto de vista personal. Además, al final del capítulo se ofrece la opinión de dicho autor, formada a lo largo del desarrollo del proyecto.

6.1 Aportaciones realizadas

A lo largo del desarrollo de este proyecto se ha experimentado con las tecnologías utilizadas (explicadas en el apartado Tecnologías) de manera intensiva, incluso llevando en ocasiones hasta el límite de sus funcionalidades conocidas a muchas de ellas. Como ejemplo, pueden citarse varios casos:

- Inclusión de un componente en una página: normalmente, un componente se agrega a una página en el propio código de la página. En el caso de este proyecto, la inclusión del componente no sólo tenía que ser en código Java (programáticamente) sino que además tenía que posibilitarse este proceso para un componente genérico, del cual sólo se disponía de la clase que lo implementaba.

- Estructura interna de librerías JSF: para el proceso de generación de la "base de datos" inicial de librerías, fue necesario estudiar la estructura interna de las librerías de componentes JSF, consistentes en un fichero comprimido JAR que contiene el conjunto de clases correspondientes, así como ficheros descriptores.

- AJAX: durante la realización de esta aplicación se ha experimentado profusamente con AJAX, haciendo posible la inclusión y modificación de componentes incluidos en una página de manera completamente asíncrona y proporcionando una experiencia de usuario todo lo rica posible.

Todos estos casos vienen a ejemplificar la cantidad y variedad de tecnologías y formas de uso aprendidas durante la implementación de este proyecto.

6.2 Trabajos futuros

En casi todo proyecto, a pesar de elaborar una planificación acorde a los objetivos marcados, generalmente se quedan cosas en el tintero. Posibles mejoras, funcionalidades casi terminadas pero eliminadas a última hora por motivos de estabilidad, ideas abandonadas en etapas previas,

etc. Este proyecto no iba a ser diferente. A continuación, se van a introducir algunas de estas ideas:

- Administración de beans: cuando el usuario personaliza un componente sólo puede introducir texto plano como valor para sus atributos o campos. Muchos componentes JSF se benefician, si no lo requieren, del uso de beans manejados como valores para estos campos. De esta forma, se le permite al usuario probar realmente como lucirían los componentes que está personalizando con datos dinámicos, incluso con estructuras enteras recuperadas de la base de datos. La introducción de la funcionalidad de gestión y creación de beans manejados en la aplicación proporcionaría una herramienta muchísimo más completa, subsanando muchas de las complicaciones encontradas por el camino a la hora de *renderizar* componentes "avanzados", componentes que requieren el uso de beans como medio de configuración.

- Mejora de la organización de componentes: en el producto desarrollado, los componentes insertados en el lienzo se pueden ordenar verticalmente. Además, aquellos componentes que son de tipo de contenedor, es decir, permiten albergar otros como hijos, sólo permiten ser configurados de manera genérica, no como padres de otros. Por tanto, sería interesante la implementación de una ordenación basada en componentes compuestos, tanto verticales como horizontales, considerando también aquellos de tipo rejilla y demás. Esto conseguiría que el usuario pudiese realmente maquetar la página en su totalidad sin la necesidad de retocar la disposición de los componentes una vez exportada la página creada.

- Exportación: actualmente, cuando se exporta el resultado de la creación de una página, sólo se obtiene un fichero con extensión XHTML que representa la página creada o fichero comprimido que proporcionan las librerías utilizadas, pero no los ficheros de configuración necesarios para incorporar completamente el resultado a un proyecto dado. Una mejora evidente sería proporcionar un fichero comprimido como producto final que contuviese todos estos elementos.

- Resolución de los problemas encontrados en la evaluación de los casos de prueba diseñados: el problema de agregación de componentes y modificación de sus propiedades debido a la falta de beans manejados se verá resuelto con el primer punto de los trabajos futuros propuestos.

6.3 Problemas encontrados

Durante la realización de este proyecto se ha encontrado una serie de dificultades. Todas ellas han sido superadas para poder llegar a la correcta finalización del mismo. A continuación, se van a explicar estos problemas encontrados:

- Viabilidad del proyecto: uno de los objetivos fundamentales y de obligado cumplimiento era que el sistema desarrollado permitiese incorporar nuevas librerías en cualquier momento de su vida. Esta inclusión no debía obligar al administrador a poseer conocimiento alguno sobre la misma o los componentes que la formaban. Esta meta se traduce en que, en todo momento, la aplicación está trabajando con componentes genéricos, no sabe si está gestionando una etiqueta o una caja de texto, un panel o un componente de Google Maps. Esta generalización acarrea gran cantidad de complicaciones, dado que los componentes se generan "programáticamente", es decir, en código Java. Esto significa que en tiempo de ejecución (es más, durante una llamada AJAX) se le pide al sistema *renderizar* un componente del cual sólo tiene almacenada la clase que implementa su "tag" o etiqueta de componente. Este proceso no es nada trivial y en un punto avanzado del desarrollo incluso se dudó de la viabilidad de llevar a cabo semejante tarea.

- Estándar JSF: aunque de mejor problemática que el anterior punto, la especificación del estándar de JavaServer Faces y la forma de implementarlo de cada librería ha sido una dificultad considerable, ya que ha sido necesario extraer las características comunes, por pocas que fuesen, que permitiesen desarrollar la funcionalidad principal marcada, al tiempo que aparecían numerosas discrepancias.

6.4 Opiniones personales

Personalmente, la realización de este proyecto de fin de carrera le ha proporcionado al proyectando conocimientos avanzados sobre JSF, su ciclo de vida, la estructuración y generación de componentes y demás características avanzadas del *framework*. Esto, sumado al intensivo uso de AJAX, supone una gran formación para cualquier desarrollador/arquitecto J2EE.

Además, la realización desde los bocetos iniciales hasta el prototipo de alto nivel del producto desarrollado y la elaboración de la presente memoria le otorgan gran desenvolvimiento en el proceso habitual de desarrollo de productos software, facilitándole en gran medida la inserción en proyectos reales de grandes y medianas empresas.

7 Bibliografía

1. Pressman, Roger S. (2005). *Software Engineering: A Practitioner's Approach*. McGraw-Hill.

2. Jacobi, Jonas y R. Fallows, John (2006). *Pro JSF and Ajax: Building Rich Internet Components*. Apress.

3. Dudney, Bill; Lehr, Jonathan; Willis, Bill y Mattingly LeRoy (2004). *Mastering JavaServer Faces*. Wiley.

4. JavaServer Faces, http://java.sun.com/javaee/javaserverfaces, último acceso el 2 de Marzo de 2009.

5. JavaServer Faces application lifecycle, http://www.ibm.com/developerworks/library/j-jsf2/index.html?ca=drs-j1105, último acceso el 3 de Marzo de 2009.

6. JavaServer Faces component development, http://www.ibm.com/developerworks/library/j-jsf4, último acceso el 3 de Marzo de 2009.

7. Comparativa librerías JSF AJAX, http://www.jsfmatrix.net, último acceso el 23 de Diciembre de 2008.

8. Wikipedia: Gestión de proyectos, http://es.wikipedia.org/wiki/Gesti%C3%B3n_de_proyectos, último acceso el 20 de Enero de 2009.

9. IEEE Std (1993). *IEEE Software Engineering Standard: Glossary of Software Engineering Terminology*. IEEE Computer Society Press.

10.Wikipedia: Clasificación e identificación de requerimientos, http://es.wikipedia.org/wiki/Computer_software#Clasificaci.C3.B3n_e_identificaci.C3.B3n_de_requerimientos, último acceso el 20 de Enero de 2009.

11. IVA en España, http://es.wikipedia.org/wiki/IVA, último acceso el 21 de Enero de 2009.

12. UML, http://www.uml.org, último acceso el 20 de Noviembre de 2008.

13. J2EE, http://java.sun.com/javaee, último acceso el 19 de Octubre de 2008.

14. Modelo-Vista-Controlador, http://www.proactiva-calidad.com/java/patrones/mvc.html, último acceso el 18 de Octubre de 2008.

15. Sun Java applets, http://java.sun.com/applets, último acceso el 25 de Julio de 2008.

16. Macromedia/Adobe Flash, http://www.adobe.com/products/flashplayer, último acceso el 25 de Julio de 2008.

17. CGI, http://hoohoo.ncsa.uiuc.edu/cgi, último acceso el 25 de Julio de 2008.

18. Patrones de diseño, http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php, último acceso el 26 de Julio de 2008.

19. AJAX, http://www.w3schools.com/Ajax/Default.Asp, último acceso el 26 de Julio de 2008.

20. Velocity, http://velocity.apache.org, último acceso el 2 de Enero de 2009.

21. JavaCC, https://javacc.dev.java.net, último acceso el 2 de Enero de 2009.

22. Tapestry, http://tapestry.apache.org, último acceso el 2 de Enero de 2009.

23. Struts, http://struts.apache.org/1.x/struts-tiles, último acceso el 2 de Enero de 2009.

24. Struts Tiles, http://struts.apache.org, último acceso el 2 de Enero de 2009.

25. Sun Microsystems, http://es.sun.com, último acceso el 23 de Febrero de 2009.

26. Apache MyFaces, http://myfaces.apache.org, último acceso el 4 de Diciembre de 2008.

27. Sun JSF RI, https://javaserverfaces.dev.java.net, último acceso el 5 de Diciembre de 2008.

28. JBoss RichFaces, http://www.jboss.org/jbossrichfaces, último acceso el 10 de Febrero de 2009.

29. JBoss RichFaces showcase, http://livedemo.exadel.com/richfaces-demo/index.jsp último acceso el 11 de Febrero de 2009.

30. IceSoft Technologies IceFaces, http://www.icefaces.org/main/home, último acceso el 9 de Enero de 2009.

31. Vedana Rich Client Faces, http://www.rcfaces.org/project-summary.html, último acceso el 9 de Enero de 2009.

32. Oracle ADF Faces, http://www.oracle.com/technology/products/adf/adffaces/index.html, último acceso el 9 de Enero de 2009.

33. Apache MyFaces Tomahawk, http://myfaces.apache.org/tomahawk/index.html, último acceso el 9 de Enero de 2009.

34. Spring, http://www.springsource.org, último acceso el 10 de Enero de 2009.

35. Hibernate, http://www.hibernate.org, último acceso el 10 de Enero de 2009.

36. JBoss, http://www.jboss.org, último acceso el 21 de Febrero de 2009.

37. Apache Tomcat, http://tomcat.apache.org, último acceso el 21 de Febrero de 2009.

38. Sun iPlanet, http://docs.sun.com/app/docs/doc/816-2670?l=es, último acceso el 21 de Febrero de 2009.

39. JBoss Seam, http://www.jboss.com/products/seam, último acceso el 10 de Enero de 2009.

40. Stripes, http://www.stripesframework.org/display/stripes/Home, último acceso el 10 de Enero de 2009.

41. Oracle TopLink, http://www.oracle.com/technology/products/ias/toplink/index.html, último acceso el 10 de Enero de 2009.

Anexo I. Control de versiones

En este primer anexo se va a proporcionar un registro detallado de las versiones de este documento que han sido generadas a lo largo del desarrollo del mismo. Esta información es importante porque ayuda a entender la progresión que ha seguido el trabajo realizado, la frecuencia de versiones, etc. Para cada versión se indicará el identificador correspondiente, la fecha de entrega de la misma y una breve descripción de su contenido. El formato de los identificadores será el siguiente:

DOC_<número>

Donde "número" es un entero de dos dígitos que se incrementa de versión en versión. El listado de versiones puede verse en la Tabla 58.

Identificador	Fecha de entrega	Contenido
DOC_01	28/04/2008	Primeros prototipos del sistema.
DOC_02	05/08/2008	Primera versión del documento como tal. Contiene los capítulos 1 y 2 completados.
DOC_03	26/01/2009	Reformulación drástica de los dos primeros capítulos y primera versión del tercero.
DOC_04	03/03/2009	Capítulos 4 y 5 terminados y realizadas correcciones en el tercero.
DOC_05	05/03/2009	Capítulo 6, glosario de términos y anexos terminados.
DOC_06	06/03/2009	Revisión general de toda la memoria.
DOC_07	03/04/2009	Correcciones realizadas sobre toda la memoria y capítulo 7.
DOC_08	15/04/2009	Correcciones realizadas sobre toda la memoria.
DOC_09	06/05/2009	Revisión final de toda la memoria.

Tabla 58. Control de versiones.

Anexo II. Seguimiento de proyecto fin de carrera

Este anexo se encarga de exponer y detallar el seguimiento que se ha realizado de este proyecto fin de carrera, es decir, la monitorización del estado del mismo durante todo el tiempo que ha durado su desarrollo. En primer lugar se va a explicar el procedimiento seguido para monitorizar el proyecto. Después, se va a proporcionar la planificación inicial que se elaboró para el mismo. Por último, se presentará la planificación final de este, es decir, la programación real que se ha llevado a cabo.

Forma de seguimiento

La manera de monitorizar el estado de este proyecto ha consistido en reuniones virtuales entre tutor y proyectando en las que se evaluaban los avances realizados en la elaboración del mismo, resultando de estas reuniones una serie de añadidos y/o modificaciones que aplicar a la siguiente versión del documento. Asimismo, en la siguiente reunión se evaluaban también los cambios o añadidos solicitados, de manera que toda alteración del documento fuese revisado en todo momento por el tutor del proyecto.

De forma adicional, se han ido recogiendo horas reales de desarrollo y fechas de comienzo y fin de cada capítulo de esta memoria, de manera que se pudiese efectuar una clara comparación entre la planificación inicial elaborada y la planificación resultante tras terminar todo el proceso de desarrollo.

Planificación inicial

Inicialmente, este proyecto de fin de carrera fue planificado para ser realizado entre los años 2006 y 2008, años en los que el proyectando aún contaba con una gran cantidad de créditos por superar en sus estudios. Este hecho ocasionó que la mayor parte de este tiempo planificado fuese dedicado a estudiar la viabilidad del sistema planteado y a realizar prototipos de bajo y medio nivel (de ahí la entrega de primeros prototipos en Abril de 2008).

Posteriormente, el proyecto fue replanificado para llevarse a cabo una vez el proyectando hubiese concluido los citados créditos. Esta última planificación es la que realmente se toma como inicial, ya que tiene un planteamiento mucho más real y asequible. Puede observarse en la Tabla 59.

Tarea	Fecha de inicio	Fecha de fin
Introducción	20/07/2008	22/07/2008
Estado de la cuestión	20/07/2008	31/07/2008
Planteamiento del problema	01/08/2008	05/08/2008
Gestión del proyecto	06/08/2008	15/08/2008
Solución	15/08/2008	15/09/2008
Evaluación	16/09/2008	26/09/2008
Conclusiones	27/09/2008	30/09/2008
Anexos	27/09/2008	30/09/2008

Tabla 59. Planificación inicial.

Planificación final

Siguiendo la planificación inicial, se intentó ajustar a las fechas programadas para finalizar en el tiempo previsto. Sin embargo, circunstancias laborales y personales del proyectando ocasionaron un ligero retraso en el inicio del desarrollo y una mayor cantidad de tiempo asignado a cada tarea, resultando la planificación final real que puede observarse en la Tabla 60. Asimismo, se reorganizaron y redefinieron algunas tareas, con el fin de mejorar la claridad de los contenidos y su estructura.

Tarea	Fecha de inicio	Fecha de fin	Horas dedicadas
Introducción	27/10/2008	06/11/2008	15
Estudio del problema	08/11/2008	08/12/2008	38
Gestión del proyecto software	08/12/2008	25/01/2008	37
Solución	26/01/2008	03/03/2009	58
Evaluación	01/03/2009	03/03/2009	6
Conclusión	01/03/2009	04/03/2009	4
Anexos	01/03/2009	04/03/2009	7
Total			**165**

Tabla 60. Planificación final.