



Proyecto cofinanciado por el Ministerio de Industria, Turismo y Comercio dentro del Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica 2004-2007 en el Programa Nacional de Tecnologías de Servicios de la Sociedad de la Información y Convocatoria Software de Código Abierto y Publicaciones en Internet.



Proyecto cofinanciado por el Fondo Europeo de Desarrollo Regional (FEDER)



Proyecto cofinanciado por el Gobierno del Principado de Asturias a través del IDEPA en su Programa de Innovación Empresarial IN+NOVA del año 2006.



PROGRAMA DE INNOVACIÓN EMPRESARIAL



Proyecto cofinanciado por GADD-Grupo Meana, S.A.



PROYECTO

Open CERTIAC

Software Abierto de Certificación Electrónica, Registro Telemático e Información y Atención Ciudadana

PLATAFORMA DE DESARROLLO DE SOFTWARE ABIERTO

Versión 1.2

01-06-2007

INDICE

1.-	SOFTWARE LIBRE DE FUENTES ABIERTAS	3
2.-	TIPOS DE LICENCIAS	3
3.-	PLATAFORMAS DE DESARROLLO DE SOFTWARE DE FUENTES ABIERTAS	5
4.-	PLATAFORMAS DE DESARROLLO JAVA.....	6
5.-	REQUISITOS DE LA PLATAFORMA	7
6.-	PLATAFORMA SELECCIONADA.....	8
6.1.-	COMPONENTES.....	8
6.1.1.-	<i>Struts</i>	8
6.1.2.-	<i>Framework del Principado (FW-PA)</i>	9
6.1.3.-	<i>Autenticación y Autorización</i>	11
6.1.4.-	<i>Acceso a Datos</i>	11
6.1.5.-	<i>Log</i>	12
6.1.6.-	<i>Taglibs</i>	12
6.1.7.-	<i>AJAX</i>	13
6.1.8.-	<i>Generación de Informes / Documentos</i>	13
6.2.-	DESARROLLO	14
6.2.1.-	<i>Gestor de RDBMS</i>	14
6.2.2.-	<i>Servidor de Aplicaciones</i>	14
6.2.3.-	<i>Entorno Integrado de Desarrollo</i>	15
6.2.4.-	<i>Control de Versiones</i>	16
6.2.5.-	<i>Constructor</i>	16
6.2.6.-	<i>Programación Orientada a Atributos (XDoclet)</i>	16
6.2.7.-	<i>Pruebas Unitarias y de Carga</i>	17
6.2.8.-	<i>Modelado UML</i>	17
6.2.9.-	<i>Herramientas XML</i>	17
6.2.10.-	<i>Gestor de Contenidos</i>	18
7.-	CRITERIOS DE DISEÑO	18
7.1.-	MULTICAPA.....	18
7.2.-	PATRONES	19
7.2.1.-	<i>MVC</i>	19
7.2.2.-	<i>Otros patrones a utilizar</i>	19
7.3.-	REUSABILIDAD.....	20
7.4.-	ACCESIBILIDAD.....	20
8.-	RESUMEN DE COMPONENTES Y HERRAMIENTAS	20

1.- SOFTWARE LIBRE DE FUENTES ABIERTAS

Por software libre o software de fuentes abiertas (OSS) se entiende aquel que se distribuye de forma gratuita y además viene acompañado de su código fuente, lo que permite modificarlo, reutilizarlo en otros programas y redistribuirlo libremente. La diferencia con el software propietario es tajante, ya que éste mantiene celosamente protegido el código fuente de los programas, a menudo objeto de patentes irracionales.

Cuando uno adquiere software sin sus fuentes, nunca llega a saber qué es lo que realmente hace el programa. Puede malgastar recursos de disco, memoria o ciclos de CPU perturbando la operación de otras aplicaciones, puede esconder pasatiempos o juegos, al estilo de lo que hacen los productos Microsoft, y lo que es mucho peor, puede incorporar módulos de espionaje de la actividad del usuario. En definitiva, el software propietario se comporta como una caja negra, cuyo funcionamiento interno desconocemos, ignorando qué clase de tretas puede llegar a jugarlos. Por el contrario, el software libre deja el código fuente disponible para el escrutinio público.

De esta manera, los programas crecen, nuevas funciones son añadidas, los agujeros de seguridad y fallos de funcionamiento se corrigen y todo ello en un clima de confianza, donde los cambios ocurren a la vista de todos. Cualquiera puede apuntar sugerencias, añadir código o mejorarlo. En consecuencia, los programas ya no pertenecen a una compañía, sino a una comunidad de programadores, de la que todos pueden, en principio, formar parte. Los usuarios ya no quedan a merced de las arbitrariedades de una única compañía ni cautivos de sus productos. De los monopolios de las empresas se pasará al de los productos. Los mejores productos dominarán los sectores del mercado.

2.- TIPOS DE LICENCIAS

- La licencia GPL (General Public License o Licencia Pública General) es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre. Software libre es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente. El software libre suele estar disponible gratuitamente en Internet, o a precio del coste de la distribución a través de otros medios; sin embargo no es obligatorio que sea así y, aunque conserve su carácter de libre, puede ser vendido comercialmente. La principal diferencia entre la GPL y otras licencias es que el software protegido por esta licencia sólo puede ser usado/modificado por proyectos de software libre, nunca podrán ser usados o modificados por un proyecto de software comercial.

- La licencia LGPL(Lesser General Public License o Licencia Pública General Menor) hace menos que la Licencia Pública General ordinaria para proteger las libertades del usuario. También proporciona a los desarrolladores de programas libres menos ventajas sobre los programas no libres competidores. Se utiliza en determinados casos, por ejemplo cuando se desea fomentar lo más ampliamente posible el uso de una determinada librería, de forma que ésta se convierta en un estándar. Para conseguir esto, se debe permitir a los programas no libres el uso de estas librerías. Un caso más frecuente es aquel en el que una librería libre hace el mismo trabajo que el que realizan las librerías no libres más ampliamente usadas. En este caso, hay poco que ganar limitando la librería únicamente al software libre, de manera que se usa la Licencia Pública General Menor LGPL. Cualquier software licenciado bajo LGPL puede ser incorporado a otro software cualquiera que sea su licencia, pero en el caso de realizar modificaciones sobre el mismo, la licencia deberá ser necesariamente LGPL o GPL.
- La licencia BSD es la licencia otorgada principalmente para los sistemas BSD (Berkeley Software Distribution) y pertenece al grupo de licencias de software Libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL, estando muy cercana al software de dominio público (aquel por el que no es necesario solicitar ninguna licencia y cuyos derechos de explotación son para toda la humanidad, porque pertenece a todos por igual y cualquiera puede hacer uso de él, siempre con fines legales y consignando su autoría original). La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre, se puede desarrollar software propietario sobre software libre BSD, si bien el software original siempre sigue siendo libre. Uno de los problemas de esta licencia se encuentra en que, llegado el caso, puede permitir a una empresa aprovecharse del trabajo de una comunidad entera y, sobre el desarrollo, construir una aplicación propietaria sin devolver nada a la comunidad.
- La licencia de Apache Software es similar a la licencia LGPL, ya que permite el uso, modificación y distribución del código en formato fuente u objeto, pero debe mantener o reproducir, respectivamente, el copyright del código junto con la licencia suministrada por Apache. Además, debe aparecer una referencia a Apache en la documentación suministrada y no deben ser utilizados los nombres de Apache para promocionar el producto ni para nombrarlo. No hay obligación de hacer públicos los cambios realizados, pero en caso de redistribución los cambios deben licenciarse bajo la misma la licencia.

3.- PLATAFORMAS DE DESARROLLO DE SOFTWARE DE FUENTES ABIERTAS

Con el auge Web, los clientes demandan ambientes de plataforma abierta. Las soluciones de código abierto, basadas en estándares, se han convertido en elementos cada vez más importantes en muchas organizaciones.

La característica principal del software abierto es el hecho de que se distribuye con su código fuente. Esto permite adaptar los programas a las necesidades existentes, evaluar el código fuente y con ello su calidad, y corregir fácil y rápidamente los errores. El desarrollo de estos programas se realiza a través de una comunidad de programadores que están en contacto mediante Internet. Estos desarrollos cooperativos han demostrado ser sumamente eficientes. Se aprovechan al máximo otros desarrollos previos, utilizando librerías de funciones o módulos desarrollados por terceros, los programadores se esfuerzan en escribir código reutilizable. Esta característica le otorga al software abierto una gran facilidad para interactuar con otros programas e integrarse en sistemas más complejos.

Otra característica es la licencia de uso que acompaña a estos programas. La más difundida es la licencia GPL (Licencia Pública General), que permite la modificación del código fuente con la condición de mantener el software obtenido bajo la misma licencia.

Hay varias alternativas para el desarrollo de aplicaciones en plataformas de fuentes abiertas. Una de ellas es Java, plataforma de software desarrollada por Sun Microsystems, de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos. Hoy en día Java junto a .Net de Microsoft son las dos plataformas más utilizadas para desarrollar aplicaciones Web, ya que están diseñadas específicamente para realizar esta tarea y tienen detrás una comunidad de desarrolladores que se encarga de generar e incorporar nuevas funcionalidades.

Otra plataforma de desarrollo puede ser Mono, la cual es una implementación de varias tecnologías, algunas de ellas propietarias de Microsoft:

- Un compilador para el lenguaje C# y Visual Basic.Net
- Un entorno de ejecución virtual: Un compilador JIT (Just-In-Time = justo-a-tiempo, esto es, que compila el código justo antes de ser ejecutado), un compilador AOT (Ahead-Of-Time = antes-de-tiempo, esto es, que compila a código nativo un archivo y de esta forma no necesita la compilación JIT cada vez que se ejecute el programa), gestión automática de memoria, un interprete (mint), motor multiproceso.

- Una máquina virtual para los bytecodes del Lenguaje Intermedio Común (CLI).
- Una implementación de la librería de clases de .NET: manipulación XML, Remoting, Reflection.Emit, XSLT, etc.
- Librería de clases multiplataforma para el acceso a bases de datos: Postgress, MySQL, DB2, TDS, Sybase, Oracle, ODBC y Gnome-GDA.
- Librería de clases UNIX: Mono.Posix.
- Librería de clases GNOME: la familia Gtk#.

Mono pretende traer al mundo del software libre lo mejor del software propietario para crear una plataforma de desarrollo también multiplataforma y con una alta productividad.

La plataforma seleccionada finalmente ha sido Java, ya que es una plataforma utilizada ampliamente en el mundo del desarrollo Web de fuentes abiertas, tiene mucha documentación a disposición del desarrollador, es una plataforma madura (lleva mucho tiempo siendo utilizada y ha demostrado su valía en el desarrollo de aplicaciones Web) y permite extender su funcionalidad mediante librerías y componentes.

4.- PLATAFORMAS DE DESARROLLO JAVA

La plataforma Java 2 Enterprise Edition (J2EE) es fruto de la colaboración de SUN con los líderes del sector del software empresarial (IBM, Apple, Bea Systems, Oracle, Inprise, Hewlett-Packard, Novell, etc.) para definir una plataforma robusta y flexible orientada a cubrir las necesidades empresariales en e-business y business-to-business.

J2EE utiliza la plataforma Java 2 Standard Edition (J2SE), para tender una completa, estable, segura, y rápida plataforma Java en el ámbito de la empresa. Permite ahorrar a la compañía, porque habilita una plataforma que reduce de manera significativa los costos y la complejidad de desarrollo de soluciones multicapa, resultando en servicios que pueden ser desarrollados rápidamente y ampliados fácilmente.

J2EE aprovecha muchas de las características de la plataforma Java, como la portabilidad, el acceso a bases de datos, la tecnología para la interacción con los recursos existentes de la empresa y un modelo de seguridad que protege los datos incluso en las aplicaciones para Internet. Sobre esta base, J2EE añade el soporte completo para componentes EJBs, servlets y la tecnología JavaServer Pages. El estándar J2EE incluye todas las especificaciones y pruebas de conformidad que permiten la portabilidad de las aplicaciones a través de la amplia gama de sistemas empresariales compatibles con J2EE.

También existe la plataforma Java 2 Micro Edition (J2ME), que es una familia de especificaciones que definen varias versiones minimizadas de la plataforma Java 2. Estas versiones minimizadas pueden ser usadas para programar en dispositivos electrónicos, desde teléfonos móviles, PDAs, hasta en tarjetas inteligentes, etc. Estos dispositivos presentan en común que no disponen de abundante memoria ni mucha potencia en el procesamiento, ni tampoco necesitan de todo el soporte que brinda el J2SE (plataforma estándar de Java usada en sistemas de escritorio y servidor). J2ME es la versión de Java orientada a los dispositivos móviles. Debido a que los dispositivos móviles tienen una potencia de cálculo baja e interfaces de usuario pobres, es necesaria una versión específica de Java destinada a estos dispositivos, ya que el resto de versiones de Java, J2SE o J2EE, no encajan dentro de este esquema. J2ME es por tanto, una versión “reducida” de J2SE.

Para el desarrollador, existen múltiples entornos gráficos de desarrollo para el lenguaje Java (Visual J++ de Microsoft, JBuilder de Borland, Java Studio de Sun, etc.). Entre todos estos entornos, cabe destacar la herramienta Eclipse por ser de libre distribución.

5.- REQUISITOS DE LA PLATAFORMA

La plataforma de desarrollo tiene que incorporar los siguientes elementos que permitan satisfacer las necesidades del puesto de desarrollo así como su posterior despliegue:

- Herramienta de documentación y modelado: en la fase de análisis el modelado de sistemas software es una técnica que permite tratar la complejidad inherente a estos sistemas. El uso de modelos ayuda al ingeniero de software a visualizar el sistema a construir. Además, los modelos de un nivel de abstracción mayor pueden utilizarse para la comunicación con el cliente. Las herramientas de modelado pueden ayudar a verificar la corrección del modelo.
- Sistema administrador de bases de datos relacionales (RDBMS): herramienta para gestionar la base de datos, el conjunto de datos que va a manejar el software desarrollado.
- Entorno integrado de desarrollo (IDE): programa compuesto por un conjunto de herramientas de alta productividad para construir y mantener aplicaciones.
- Repositorio de código fuente: sistema cliente-servidor que permite a los desarrolladores realizar el seguimiento de las diferentes versiones del código fuente de un proyecto. Se trata de una herramienta muy potente que permite registrar todos los cambios efectuados sobre los archivos, recuperar versiones anteriores del código, gestionar los conflictos que pueden producirse en entornos en los que los desarrolladores se encuentran distribuidos geográficamente, conocer qué cambios se han efectuado sobre un archivo determinado, quién los ha realizado y cuándo.
- Constructor: herramienta para la automatización de tareas complejas en el desarrollo del software, como la configuración, compilación, instalación o distribución.

- Monitor: herramienta para la monitorización y seguimiento del código fuente que facilite del proceso de depuración a los desarrolladores.
- Servidor de aplicaciones: entorno optimizado de ejecución para componentes de aplicaciones de servidor. Proporciona un entorno robusto de ejecución, de elevado rendimiento y escalabilidad específicamente diseñado para soportar sistemas de aplicaciones preparados para Internet.
- Pruebas unitarias y de carga: herramienta para la automatización de las pruebas de caja negra, respuesta y carga sobre aplicaciones Web.
- Generador de informes: herramienta para la presentación de la información manejada por el software en diversos formatos entendibles para otras aplicaciones.
- Gestor de contenidos: herramienta que permite de forma sencilla actualizar el contenido y manejar los documentos publicados en el sitio Web.

6.- PLATAFORMA SELECCIONADA

6.1.- Componentes

6.1.1.- Struts

El Modelo Vista Controlador MVC es un patrón arquitectural muy difundido hoy en día en aplicaciones de entorno Web. La evolución de lo que se conoce como “Modelo 2” de aplicaciones Web (separación de responsabilidades de presentación, negocio y navegación) avanza un poco más en el reparto de tareas en la aplicación. Pese a que hay distintos puntos de vista acerca de la forma de aplicar e implementar este patrón, en esencia las ideas principales sobre su estructura y funcionalidad son las mismas. El MVC tiene tres piezas claves que se reparten la responsabilidad de la aplicación:

- El modelo (Model): responsable de toda la lógica y estado del dominio de negocio.
- La vista (View): responsable de la presentación del dominio de negocio.
- El controlador (Controller): responsable del flujo de control, la navegabilidad y el estado de la entrada del usuario.

El proyecto Struts proporciona un marco de trabajo MVC estándar a la comunidad Java. Este framework del proyecto Jakarta es la implementación java orientada a aplicaciones Web más difundida del patrón MVC. Provee su propio controlador (ActionServlet), y se integra con otras tecnologías para proveer el modelo y la vista. La navegación se

configura en ficheros XML externos y se organiza la lógica y responsabilidades siguiendo la distribución del MVC.

Por lo tanto, para desarrollar basándose en una arquitectura “Modelo 2”, variación del paradigma Modelo-Vista-Controlador (MVC), se ha seleccionado el Framework Struts. Struts proporciona un Servlet que se encarga de manejar la lógica de negocio, mientras que toda la lógica de presentación recae en las páginas JavaServer Pages (JSPs). También proporciona herramientas para desarrollar con MVC así como subcomponentes para el desarrollo de páginas Web, como pueden ser Struts-Menú, Struts-Tiles o las librerías de etiquetas JSP html, logic, bean, etc.

6.1.2.- Framework del Principado (FW-PA)

Para la implementación de la aplicación se utilizará toda la infraestructura proporcionada por el Framework de Desarrollo J2EE del Principado de Asturias (jerarquía de actions, módulos de autenticación y autorización, logging, configuración, seguridad, etc.), lo cual permitirá acortar los tiempos de desarrollo al no tener que configurar y probar una plataforma de desarrollo en Java partiendo desde cero. Simplifica el desarrollo de las aplicaciones, proporcionando un conjunto de herramientas y librerías que implementan algunos de los componentes más habituales en aplicaciones Web, definiendo estándares de desarrollo y calidad en las aplicaciones J2EE. Esta plataforma proporciona una base tecnológica común a todos los nuevos desarrollos, empleando estándares abiertos y herramientas de código libre. Todo esto redundará en un menor tiempo de desarrollo, una homogeneización de los sistemas a mantener y un incremento de la calidad de los mismos.

Se actualizarán a versiones más actuales algunos componentes que se incluyen en el Framework del Principado de Asturias, como por ejemplo el componente DisplayTag, para acceder a nuevas funcionalidades que proporcionan estas nuevas versiones y que son importantes para la aplicación a desarrollar.

Las funcionalidades soportadas por el Framework PA son las siguientes:

- Extensión del Framework Struts con una colección propia de clases Action.
- Acceso a datos a través de objetos DAO.
- Automatización de la carga de consultas SQL desde ficheros de propiedades.
- Plantillas (Tiles) para la creación rápida de páginas JSP.
- Facilidades para la generación de informes en formato PDF.
- Etiquetas JSP para la inclusión de listas, barras de navegación, fechas y calendarios en las páginas Web.

- Integración de formularios (ActionForm) con entidades (ValueObject) de la aplicación.
- Utilidades para la gestión de tablas de datos en formato {atributo, valor}.
- Facilidades para la obtención de listas paginadas como resultado de consultas.
- Herramienta para la generación automática de menús.
- Infraestructura para pruebas unitarias.
- Infraestructura para pruebas unitarias en contenedor.
- Integración de una consola de monitorización y gestión basada en el estándar JMX.
- Jerarquía propia de excepciones.
- Monitorización y control integrado de errores
- Sistema de configuración centralizado.
- Sistema de inicialización y arranque configurable.
- Componentes para el acceso a pools de conexiones.
- Sistema de logging con varios niveles.
- Gestión de logging desde la consola de administración.
- Monitor de rendimiento.
- Sistema de monitorización para las clases Action.
- Generación de estadísticas de acceso a las aplicaciones.
- Generación de estadísticas de excepciones no controladas en las aplicaciones.
- Componente para monitorizar el estado del sistema sobre el que corre la aplicación.
- Infraestructura para filtros gestionados “en caliente”.
- Inicialización de componentes configurable.

El Framework de Desarrollo del Principado de Asturias promueve el uso intensivo de Patrones de Diseño. A fin de lograr una homogeneidad efectiva en las aplicaciones realizadas, se propone una Arquitectura de Referencia que describe la arquitectura de las aplicaciones desarrolladas con el FW-PA. Una arquitectura de referencia es una descripción de los elementos de los que se compone una aplicación, y de las relaciones entre estos elementos. Manejar arquitecturas de referencia es tremendamente beneficioso, ya que permite:

- Homogeneizar las aplicaciones: al usar la arquitectura de referencia, las aplicaciones son estructuralmente iguales, cambiando sólo los elementos en concreto, pero no la forma que tienen de relacionarse. Esto tiene un impacto directo en el esfuerzo en desarrollo y mantenimiento.

- Extender las mejores prácticas y tecnologías: la arquitectura de referencia ha de mantenerse, de manera que se vayan introduciendo cambios basados en cambios tecnológicos o en el establecimiento de mejores prácticas.

6.1.3.- Autenticación y Autorización

Tanto para la autenticación como la autorización se ha seleccionado el estándar Java JAAS (Java Authentication and Authorization Service). JAAS puede ser usado de dos maneras:

- Para autenticación de los usuarios: determinando quien está ejecutando código Java, es decir, el usuario que se ha autenticado en la aplicación. Lo que permite definir diferentes formas de acceso a las aplicaciones
- Para autorización de los usuarios: determinando si el usuario que está ejecutando código Java tiene los permisos necesarios para realizar determinadas operaciones. Lo que permite asignar roles a los usuarios.

6.1.4.- Acceso a Datos

La capa de acceso a los datos que maneja la aplicación requiere la persistencia de datos y el uso de transacciones. Tanto los EJBs como Hibernate proporcionan estas funcionalidades, entre otras.

Los EJBs (Enterprise JavaBeans) son componentes Java que permiten definir la lógica de negocio junto con los datos que se manejan. Son creados, gestionados y destruidos por el contenedor de EJBs en el que se alojan, el cual proporciona control para la seguridad, las transacciones y el ciclo de vida. Por ello, el servidor de aplicaciones que se utiliza con los EJBs tiene que proporcionar un servidor de EJBs con su respectivo contenedor de EJBs.

Hibernate permite definir la persistencia de un modelo relacional a partir de clases de objetos Java, los cuales son mapeados utilizando simples ficheros XML de configuración. También permite definir consultas SQL mediante un lenguaje totalmente portable (HQL) válido para cualquier base de datos.

Comparando estas dos alternativas, finalmente se ha decidido utilizar Hibernate. El desarrollo de los objetos Java presenta una menor complejidad. La aplicación es totalmente independiente del servidor de aplicaciones (como se ha mencionado anteriormente, se utiliza Tomcat para el desarrollo). La escalabilidad no es necesaria, ya que el número de usuarios concurrentes no va a representar una gran carga para la aplicación.

6.1.5.- Log

Para monitorizar el estado de la aplicación en tiempo de ejecución los desarrolladores normalmente añaden sentencias `println()` a su código para generar información con la que poder revisar la salida de error estándar mientras se está ejecutando la aplicación. Algunas veces las sentencias `println()` bien situadas también pueden ayudar durante el proceso de depuración. Aunque esta aproximación es sencilla de codificar, no es apropiada para aplicaciones desplegadas en entornos de producción, ya que la salida de la consola es temporal.

Escribir esa información a un fichero de log es una mejor opción, ya que así se podrá recuperar el fichero más tarde para su análisis. En general, se pueden utilizar los logs para obtener un mejor entendimiento del estado de la aplicación. Por ejemplo, se puede registrar la entrada y salida de los métodos dentro del código y luego buscar en los logs más tarde, para entender mejor lo que está sucediendo en la aplicación.

Dentro del mundo Java existen componentes que proporcionan a los desarrolladores mecanismos sencillos y configurables para escribir información de log a ficheros. Utilizando estos marcos de trabajo se puede cambiar fácilmente el nivel de log en los ficheros de propiedades asociados, para que durante los procesos de desarrollo y depuración, el nivel de log se seleccione para una captura minuciosa, pero cuando se despliegue la aplicación, el nivel se seleccione como mínimo, para que sólo se registren los problemas reales.

Para la inserción de sentencias de log se ha seleccionado el componente Log4J, que permite insertar sentencias de log sin influir demasiado en el rendimiento de la aplicación, puesto que está bastante optimizado para estas tareas. Log4J puede ser controlado mediante un fichero de configuración y su salida puede ser enviada a varios tipos de dispositivos, como por ejemplo ficheros, servidores remotos, daemons, etc.

Log4J es el más extendido y popular entre la comunidad de desarrolladores, por lo que no se han evaluado otras alternativas.

6.1.6.- Taglibs

Aparte de las ya mencionadas Struts taglibs, se utilizan las librerías de tags JavaServer Pages Standard Tag Library (JSTL) que encapsulan las funcionalidades básicas que se suelen necesitar en toda aplicación Web. La JSTL proporciona tags de iteración, condicionales, tags para manejar documentos XML, de internacionalización, etc.

6.1.7.- AJAX

En las aplicaciones Web tradicionales los usuarios interactúan mediante formularios, que al enviarse, realizan una petición al servidor Web. El servidor se comporta según lo enviado en el formulario y contesta enviando una nueva página Web. Se desperdicia mucho ancho de banda, ya que gran parte del HTML enviado en la segunda página Web ya estaba presente en la primera. Además, de esta manera no es posible crear aplicaciones con un grado de interacción similar al de las aplicaciones habituales.

En aplicaciones AJAX (Asynchronous JavaScript And XML) se pueden enviar peticiones al servidor Web para obtener únicamente la información necesaria, empleando SOAP o algún otro lenguaje para servicios Web basado en XML, y usando JavaScript en el cliente para procesar la respuesta del servidor Web. Esto redundante en una mayor interacción gracias a la reducción de información intercambiada entre servidor y cliente, ya que parte del proceso de la información lo hace el propio cliente, liberando al servidor de ese trabajo. La contrapartida es que la descarga inicial de la página es más lenta al tenerse que bajar todo el código JavaScript.

AJAX constituye una técnica de desarrollo Web para crear aplicaciones interactivas mediante la combinación de tres tecnologías ya existentes:

- HTML (o XHTML) y hojas de estilos en cascada (CSS) para presentar la información.
- Document Object Model (DOM) y JavaScript, para interactuar dinámicamente con los datos.
- XML y XSLT, para intercambiar y manipular datos de manera desincronizada con un servidor Web (aunque las aplicaciones AJAX pueden usar otro tipo de tecnologías, incluyendo texto llano, para realizar esta labor).

DWR es la librería JAVA que facilita a los desarrolladores el uso de AJAX en las aplicaciones Web. Básicamente, se utiliza JavaScript para comunicarse con el servidor y actualizar dinámicamente las páginas Web, y en el servidor existe un Java Servlet que procesa las peticiones y envía las respuestas al cliente.

6.1.8.- Generación de Informes / Documentos

Para la generación de informes se ha seleccionado la herramienta JasperReports, que permite la generación de informes ricos en variedad de formatos, como puede ser PDF, HTML, XLS, CSV y XML. Como editor de informes se utilizará iReport, perfectamente integrado con JasperReports.

También se utilizarán plantillas de transformación XSLT para generar documentos HTML a partir de documentos XML generados por el controlador.

Como alternativa en la generación de informes, cabe destacar JFreeReport, que es una herramienta menos utilizada y con una documentación más escasa, y que como desventaja principal, no posee un diseñador visual de informes como el iReport.

6.2.- Desarrollo

6.2.1.- Gestor de RDBMS

Como gestor de base de datos se ha seleccionado MySQL versión 5.0, cuya licencia de código libre se ajusta perfectamente a los propósitos de éste proyecto. MySQL es la base de datos más usada en el mundo del software libre, es rápida, flexible, no consume muchos recursos, es fácil de administrar y proporciona todas las funcionalidades que se necesitaban para este proyecto. Además proporciona utilidades para administrar, migrar y manejar las bases de datos.

Como alternativas a MySQL se plantea la otra base de datos que se suele usar en los desarrollos de software libre, PostgreSQL. Las dos soportan transacciones, rollbacks, subselects, vistas, integridad referencial, triggers, etc, pero MySQL tiene un mayor rendimiento, tanto a la hora de conectarse al servidor como a la hora de servir consultas. También consume muchos menos recursos, lo que contribuye a mejorar su estabilidad en su servidor en comparación con PostgreSQL.

6.2.2.- Servidor de Aplicaciones

En principio, para la fase de desarrollo se ha seleccionado el contenedor de servlets Tomcat, por su facilidad de mantenimiento y agilidad en el proceso de desarrollo, ya que permite al desarrollador trazar fácilmente el código fuente y a la hora de arrancar tarda mucho menos que otros contenedores.

Para versiones de producción se ha seleccionado el servidor de aplicaciones JBoss, estándar en aplicaciones J2EE. JBoss es un servidor de aplicaciones J2EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo que lo soporte. JBoss es el primer servidor de aplicaciones de código abierto, preparado para la producción y certificado J2EE 1.4, disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones de e-business. Combinando una arquitectura orientada a servicios con una licencia de código abierto, JBoss puede ser descargado, utilizado, incrustado, y distribuido sin restricciones por la

licencia. Por este motivo es la plataforma más popular de middleware para desarrolladores, vendedores independientes de software y, también, para grandes empresas.

Las características destacadas de JBoss incluyen:

- Producto de licencia de código abierto sin coste adicional.
- Cumple los estándares.
- Confiable a nivel de empresa.
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente.
- Servicios middleware para cualquier objeto de Java.
- Soporte completo para JMX.
- Implementa todo el paquete de servicios de J2EE.
- Integra EJB 3.0, Hibernate, Tomcat, etc.

La selección de JBoss frente a otros servidores de aplicaciones como pueden ser Jetty o Geronimo se ha realizado teniendo en cuenta todas estas características y la compatibilidad con el contenedor de desarrollo (Tomcat), ya que JBoss tiene integrado Tomcat, con lo cual cualquier aplicación desarrollada en Tomcat se despliega sin ningún problema ni cambios en JBoss.

6.2.3.- Entorno Integrado de Desarrollo

Como entorno de desarrollo se ha seleccionado la última versión disponible de Eclipse, una plataforma de desarrollo libre e integrada con Java, a la cual se le pueden añadir diversos plugins que permiten extender la funcionalidad del entorno y que son de gran ayuda para el desarrollador. En principio, se instala con el plugin Web Tools Platform (WTP) Project, que incluye herramientas que facilitan el desarrollo de aplicaciones Web en J2EE.

Otro entorno de desarrollo es el NetBeans para Java, es menos flexible que Eclipse y presenta una usabilidad más compleja. Su gran desventaja respecto a Eclipse radica en el amplio número de plugins que éste último tiene, los cuales facilitan y automatizan diversas tareas para el desarrollador.

6.2.4.- Control de Versiones

Para el control de versiones se ha seleccionado CVS (Concurrent Versions System). CVS permite a varios desarrolladores trabajar al mismo tiempo sobre el mismo código, fusionar los cambios realizados, restaurar versiones anteriores, generar nuevas ramas y versiones del código, etc.

No se descarta en un futuro utilizar SVN (Subversion), una nueva implementación destinada a reemplazar a CVS y eliminar las deficiencias de éste.

6.2.5.- Constructor

Para la construcción y despliegue en el servidor de aplicaciones se ha seleccionado la herramienta Ant, que viene integrada con el entorno de desarrollo Eclipse y permite realizar fácilmente dichas tareas. Es una herramienta de construcción basada en XML que permite definir tareas que se ejecutan para realizar determinadas acciones.

Ant es el más extendido y popular entre la comunidad de desarrolladores, por lo que no se han evaluado otras alternativas.

6.2.6.- Programación Orientada a Atributos (XDoclet)

Este paradigma se integra dentro de la programación orientada a objetos, y se basa en la utilización de atributos en el código fuente para especificar comportamientos de las clases que normalmente se describen en ficheros de configuración. El objetivo es evitar la dispersión de la información en varios puntos como código fuente y distintos ficheros de configuración.

El origen de este paradigma está en la especificación EJB, que requiere un gran número de ficheros de configuración, clases e interfaces en muchas ocasiones redundantes. Con el fin de facilitar el desarrollo de sistemas EJB se creó XDoclet, que interpretaba una serie de etiquetas Javadoc especiales y generaba los ficheros redundantes a partir de ellas. Hoy en día XDoclet permite realizar programación orientada a atributos no sólo para EJB, sino para muchas otras tareas, por ejemplo generación de ficheros de configuración de Hibernate, generación de servicios Web, etc.

XDoclet se usa para añadir metadatos a los atributos de ciertas clases (clases Value Object) mediante tags especiales Javadoc. Gracias a estos metadatos, se pueden generar los mapeos de Hibernate mediante una tarea Ant, necesarios para acceder a los registros de las tablas de la base de datos.

6.2.7.- Pruebas Unitarias y de Carga

Para las pruebas unitarias se ha seleccionado JUnit, estándar de facto para este tipo de pruebas. JUnit permite automatizar todo el proceso de realización de pruebas unitarias para software orientado a objetos, en particular Java. En este caso se utilizará para realizar pruebas unitarias de lógica de negocio. No se utilizará ninguna extensión particular de JUnit para ésta aplicación.

En cuanto a pruebas de rendimiento se utilizará OpenSTA, que permite realizar pruebas de carga variando el número de usuarios concurrentes a la aplicación.

6.2.8.- Modelado UML

Como herramienta de software libre para el modelado UML se ha probado ArgoUML, que permite diseñar los diferentes diagramas UML (casos de uso, clases, estados, actividad, secuencia, colaboración, etc.). Presenta las carencias de que no permite generar el modelo de datos para exportarlo a una base de datos, y la documentación que genera es escasa, los diagramas generados se graban como imágenes.

Para la parte de desarrollo de software de modelado UML y documentación se ha seleccionado la herramienta Enterprise Architect de Sparx System, que permite la integración con Eclipse. Además de generar todos los diagramas UML, permite diseñar el modelo de datos junto con sus respectivas sentencias SQL de creación, y viceversa, generar el modelo de datos a partir de una base de datos ya creada. La documentación que genera es extensa y permite formatearla mediante el uso de plantillas.

6.2.9.- Herramientas XML

Para la edición de documentos XML en el Eclipse se ha seleccionado el plugin XMLBuddy, que facilita las tareas del desarrollador para la generación, formateado y validación de estos documentos.

Como herramienta para la generación de las hojas de estilo se ha seleccionado el XMLSpy Home Edition de Altova, que es una versión libre del entorno de desarrollo XMLSpy para el modelado, edición, depuración y transformación de las diferentes tecnologías XML, que permite la integración con Eclipse. Mediante esta herramienta se puede trabajar con archivos XML, generar schemas, DTDS y plantillas de transformación XSLT. Es una de las herramientas más potentes para el trabajo con ficheros XML y facilita enormemente el trabajo con estos ficheros.

6.2.10.- Gestor de Contenidos

En la parte del proyecto dedicada al registro telemático, se necesita generar componentes fácilmente integrables en un gestor de contenidos. El gestor de contenidos seleccionado es OpenCMS, el cual es un Gestor de Contenidos Open Source que ayuda a crear y mantener sitios Web completos de forma sencilla y sin necesidad de conocimientos de Html. Está basado en tecnología Java y XML, e integra un editor WYSIWYG con una interfaz de usuario similar a las habituales aplicaciones de ofimática que consigue crear todo tipo de plantillas para la Web corporativa.

Como alternativa a OpenCMS podría estar Apache Lenya, también basado en Java, pero todavía en fase de desarrollo no apta para producción. La elección final de OpenCMS como gestor de contenido se basa en el hecho de soportar nativamente Java y en la posibilidad de crear módulos que extiendan la funcionalidad de OpenCMS.

7.- CRITERIOS DE DISEÑO

7.1.- Multicapa

Una arquitectura multicapa particiona todo el sistema en distintas unidades funcionales: presentación, lógica de negocio y acceso a datos. Esto asegura una división clara de responsabilidades y hace que el sistema sea más mantenible y extensible. Los sistemas con tres o más capas se han probado como más escalables y flexibles que un sistema cliente-servidor, en el que no existe la capa central de lógica de negocio.

La capa de presentación expone los servicios de la capa de lógica de negocio a los usuarios. Sabe cómo procesar una petición de cliente, cómo interactuar con la capa de lógica de negocio, y cómo seleccionar la siguiente vista a mostrar.

La capa de la lógica de negocio contiene los objetos y servicios de negocio de la aplicación. Recibe peticiones de la capa de presentación, procesa la lógica de negocio basada en las peticiones, y media en los accesos a los recursos de la capa de datos. Los componentes de la capa de lógica de negocio se benefician de la mayoría de los servicios a nivel de sistema, como el control de seguridad, de transacciones y de recursos.

La capa de acceso a datos maneja la persistencia de los datos y las transacciones. Contiene la base de datos relacional.

7.2.- Patrones

7.2.1.- MVC

El Modelo Vista Controlador MVC es el patrón de diseño arquitectural recomendado para aplicaciones interactivas Java. Separa los conceptos de diseño, y por lo tanto decrementa la duplicación de código, el centralizamiento del control y hace que la aplicación sea más extensible. MVC también ayuda a los desarrolladores con diferentes habilidades a enfocarse en sus habilidades principales y a colaborar a través de interfaces claramente definidos. MVC es el patrón de diseño arquitectural para la capa de presentación y el estándar en aplicaciones Web.

7.2.2.- Otros patrones a utilizar

Los más destacados son:

- Singleton: creación de una única instancia de una clase.
- Factory: centralización del sitio donde se crean los objetos de una misma familia.
- Decorator: añadir responsabilidades a una objeto concreto de forma dinámica.
- Data Transfer Object (VO – Value Object): objeto serializable para la comunicación entre las diferentes capas de la aplicación y para la transferencia de datos sobre la red.
- Data Access Object (DAO): objeto de acceso a datos para abstraer y encapsular todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos.
- Service Locator: objeto que retorna los recursos listos para utilizar independientemente de cómo se hayan obtenido, reduciendo la complejidad de código y proporcionando un punto de control.
- Business Delegate: objeto que llama a métodos remotos de la capa de la lógica de negocio para simplificar su utilización en la capa de presentación.
- Intercepting Filter: objeto que procesa las peticiones y respuesta entre el cliente y los componentes Web.
- View Helper: objeto que se encarga de aglutinar código común (JSP custom tags).

7.3.- Reusabilidad

Para la construcción de la aplicación se empleará una metodología de desarrollo ágil basada en componentes. La aplicación estará construida en base a pequeños componentes que se reutilizarán en diversas partes de la aplicación. Para cada uno de estos componentes se generarán diagramas de clases, diagramas del modelo de datos y documentación de casos de uso. Aparte de esto se generará un manual de uso de la aplicación.

7.4.- Accesibilidad

La aplicación a desarrollar consta de varias partes diferenciadas. Una de ellas es el módulo de registro, que necesita una gran agilidad, tanto por parte del programa como del usuario que lo utiliza. Es por esto por lo que para el módulo de registro se ha tomado la decisión de incluir Javascript donde fuese necesario para facilitar el trabajo diario al funcionario, de forma que aunque se pierda accesibilidad y movilidad, se gana facilidad de uso y agilidad de cara al usuario. De todas formas se procurará que la aplicación sea accesible en la medida de lo posible.

Todo el Javascript que se incluirá en las páginas será al menos compatible con los navegadores Mozilla Firefox e Internet Explorer.

Para otros módulos de la aplicación, como puede ser el registro telemático, se tendrá muy en cuenta la accesibilidad y la movilidad, diseñando las páginas de acuerdo a los estándares de accesibilidad e incluyendo el menor Javascript posible.

8.- RESUMEN DE COMPONENTES Y HERRAMIENTAS

Componente	Versión	Enlace	Licencia
Java J2SE SDK	1.4.2_11	http://java.sun.com/j2se/1.4.2/	Sun License
Java J2EE	Librerías proporcionadas por el contenedor de aplicaciones seleccionado		
Struts	1.1	http://struts.apache.org/	Apache Software
Framework PA	1.4	http://www.fundacionctic.org/	
Hibernate	3.1.2	http://www.hibernate.org/	LGPL
Log4J	1.0.4	http://jakarta.apache.org/commons/logging/	Apache

			Software
DisplayTag	1.1	http://displaytag.sourceforge.net/11/	LGPL
JSTL	1.0.6	http://jakarta.apache.org/taglibs/	Apache Software
DWR AJAX	1.1	http://getahead.ltd.uk/dwr/	LGPL
JasperReports	1.2.0	http://jasperreports.sourceforge.net/	LGPL

Herramienta	Versión	Enlace	Licencia
MySQL	5.0.19	http://www.mysql.com/	GPL
Apache Tomcat	5.0.28	http://tomcat.apache.org/	Apache Software
JBoss	4.0.3 SP1	http://www.jboss.com/	LGPL
Eclipse	3.1.2	http://www.eclipse.org/	Eclipse Project License
CVS	2.5.03	http://www.march-hare.com/	GPL
Ant	1.6.5	http://ant.apache.org/	Apache Software
XDoclet	1.2	http://xdoclet.sourceforge.net/xdoclet/	GPL
JUnit	3.8.1	http://www.junit.org/	LGPL
OpenCMS	6.0.4	http://www.opencms.org/	LGPL