J2EE: Universal CMIS Client

Miguel Segura Anaya
ETIG / ETIS

Jose Juan Rodriguez

14 de Enero de 2013

Agradecimientos

Este proyecto, está dedicado a la luz de mi vida, Virginia. Sin su apoyo este proyecto ni siquiera se hubiera empezado.

Resumen

El estándar CMIS, siglas de Content Management Interoperability Services, es un estándar diseñado por los principales fabricantes de gestores de contenido empresarial para la su propio ámbito. CMIS permite el acceso de forma estándar y única a los sistemas de gestión de contenidos que lo implementan, entre ellos los más conocidos y extendidos como Microsoft SharePoint, IBM FileNet o Documentum.

El objetivo de este proyecto es desarrollar una aplicación J2EE que permita a un usuario tipo, realizar las operaciones más comunes en el uso de un gestor documental, estas operaciones incluirán el alta de documentos en el gestor, la creación de carpetas, el archivado de documentos en carpetas, la localización de los documentos por sus metadatos y el versionado y actualización de un documento.

Durante la ejecución de este proyecto hemos obtenido por un lado el conocimiento sobre el estándar CMIS, las estructuras de implementación y los servicios que toda implementación de CMIS debe cumplir. Por otro lado hemos estructurado un marco de desarrollo de aplicaciones J2EE, que nos ofrezca soluciones para todas las necesidades básicas de implementación dentro del patrón de diseño MVC.

El resultado del proyecto es una aplicación J2EE basada en los estándares actuales de desarrollo capaz de ejecutarse en la mayoría de los servidores de aplicaciones y con la capacidad de interactuar con la mayoría de gestores documentales que implementan el estándar CMIS.

Palabras clave

- CMIS
- Gestión documental
- J2EE
- MVC
- Spring
- iBatis
- Mootools
- JSON
- Apache Chemistry

Área del TFC

- J2EE

Índice de contenidos

Índice de figuras y tablas

1. Introducción

1.1. Justificación y contexto del TFC: punto de partida y aportación del TFC.

Millones y millones de papeles sin clasificar, almacenes llenos de facturas, documentos, informes, resúmenes de ventas que se abandonan esperando el día de ser destruidos. La gestión de la documentación ha sido durante muchos años un quebradero de cabeza para las organizaciones, que debían encontrar mecanismos lo más eficientes posibles para que toda esa información generada día a día pudiera ser reutilizada y aprovechada.

Con la llegada de la era digital, esta información es convertida de una manera más o menos automática en ficheros informáticas, que convierten el espacio físico en espacio en disco, y los millones de documentos en millones de ficheros ordenados con mayor o menor acierto en enormes árboles de carpetas.

Solucionado el problema del archivado surge la necesidad de solucionar el problema de la recuperación y clasificación de la información. Un fichero es más valioso si además de su contenido, podemos asociarle una serie de propiedades que nos permita localizarlo dentro de ese laberinto de carpetas. Surge entonces la gestión documental.

Estos gestores documentales permiten que toda esa información almacenada, clasificada y parametrizada sea fácilmente localizable por el usuario, utilizando diferentes criterios de localización dependiendo de las necesidades e independientemente de la localización de los mismos.

Surgen en paralelo productos de diferentes fabricantes, cada uno de ellos con su arquitectura, funcionalidades y características propias. En la actualidad, el mercado dispone de una amplia oferta de gestores documentales, cada uno de ellos utilizan tecnologías de implementación diferentes y la mayoría de ellos proporcionan una serie de APIs o mecanismos propietarios para la interconexión con otros sistemas. Las diferentes organizaciones optan por uno o varios de ellos y comienzan a utilizarlos con mayor o menor acierto.

Con posterioridad, estas organizaciones ven la necesidad de integrar estos gestores de contenidos en sus herramientas de trabajo diarias, ERPs, BPMs o CRMs deben poder recuperar información o alimentar estos gestores de manera que todas las herramientas en conjunto construyan una única base de conocimiento común y universal.

La diferente implementación de los gestores documentales hace prácticamente imposible que estas integraciones puedan llevarse a cabo de una manera estándar, ya que cada producto tiene sus propios mecanismos de integración y de acceso. Como resultado, las organizaciones se “casan” de por vida con los gestores documentales ya que un cambio de los mismos puede llegar a afectar a todos los demás sistemas.

El estándar CMIS, siglas de Content Management Interoperability Services, es un estándar diseñado por los principales fabricantes de gestores de contenido empresarial para la su propio ámbito. CMIS permite el acceso de forma estándar y única a los sistemas de gestión de contenidos que lo implementan, entre ellos los más conocidos y extendidos como Microsoft SharePoint, IBM FileNet o Documentum.

La utilización de CMIS nos permite la creación de aplicaciones que no tengan que dirigirse a un sistema de gestión de contenidos concreto, lo que asegura el futuro de las aplicaciones desarrolladas de esta manera al eliminar esta vinculación directa.

El estándar ofrece un catálogo completo de funcionalidades que nos permiten manejar los contenidos, metadatos de contenidos, control de versiones, contenidos de carpetas, asociaciones y transferencia de ficheros.

1.2. Objetivos del TFC.

El objetivo de este proyecto es desarrollar una aplicación J2EE que permita a un usuario tipo, realizar las operaciones más comunes en el uso de un gestor documental, estas operaciones incluirán el alta de documentos en el gestor, la creación de carpetas, el archivado de documentos en carpetas, la localización de los documentos por sus metadatos y el versionado y actualización de un documento.

La búsqueda de estas funcionalidades comunes a cualquier herramienta de gestión documental, nos permitirá un conocimiento bastante completo de la solución CMIS, ya que deberemos comprobar cómo este estándar da solución a todas estas necesidades

Las premisas de diseño del producto serán la simplicidad de la interface y el alto rendimiento del producto final. La aplicación J2EE se desarrollará siguiendo los estándares del modelo vista controlador, y se apoyará en frameworks actuales como Spring para la gestión de la navegación y de los beans y servicios y Mootols para dotar a la interface de la agilidad de la web 2.0.

Al igual que con el estudio del CMIS, la búsqueda de las soluciones inherentes en cualquier aplicación web de usuario, dará como resultado un framework de desarrollo que nos permita afrontar cualquier otra aplicación con las máximas garantías.

1.3. Enfoque y método seguido.

Para la realización de este proyecto, se ha seguido un método tradicional lineal, basado en la secuencia de etapas.

Se estableció inicialmente un plan de trabajo, donde se establecían de forma clara, los requerimientos funcionales de la aplicación, así como unos requerimientos técnicos.

A partir de estos requerimientos se realizó un análisis de los mismos, estableciéndose inicialmente una serie de funcionalidades básicas, descritas como casos de uso, que debían ser satisfechas por la aplicación de la manera más satisfactoria posible. Esto dio lugar a un documento de análisis.

Estas funcionalidades se agruparon por su similitud, a fin de optimizar el periodo de desarrollo, y a partir de estos grupos se establecieron unas necesidades técnicas. Esto dio lugar a un documento de diseño.

Por ejemplo, la aplicación requería de la entrada de datos por formulario, por lo que se buscó un mecanismo para la construcción y validación de los mismos, que fuera sólida y fácilmente implementable, lo mismo para los listados, la navegación, la gestión de errores, etc.

Una vez conocido con exactitud lo que se quería hacer y cómo se quería hacer, se inició el proceso de selección y construcción de la infraestructura de desarrollo y posteriormente del desarrollo en sí.

Como resultado final se obtuvo la aplicación que se describe con más detalle en los próximos capítulos de este documento.

1.4. Planificación del proyecto.

La planificación del proyecto se realizó siguiendo como referencia los hitos establecidos por los entregables del proyecto, por lo que se ha seguido con bastante exactitud. Como consecuencia de esta planificación y seguimiento se ha cumplido con el objetivo de satisfacer los requerimientos iniciales en el plazo deseado.

La programación de tareas con sus consiguientes hitos era la siguiente,

Cliente CMIS J2EE	**81 días**	**lun 24/09/12**	**lun 14/01/13**	
Plan de trabajo	8 días	lun 24/09/12	mié 03/10/12	
Entrega plan de trabajo	0 días	mié 03/10/12	mié 03/10/12	
Instalación entorno	**5 días**	**mié 03/10/12**	**mar 09/10/12**	**5**
Instalacion SO Suse	1 día	mié 03/10/12	mié 03/10/12	
Instalación BBDD DB2	1 día	jue 04/10/12	jue 04/10/12	7
Instalación Servidor Aplicaciones WA	1 día	vie 05/10/12	vie 05/10/12	8
Instalación Filenet	1 día	lun 08/10/12	lun 08/10/12	9
Instalación CMIS	1 día	mar 09/10/12	mar 09/10/12	10
Analisis funcional	**11 días**	**mié 03/10/12**	**mié 17/10/12**	**5**
Actores	1 día	mié 03/10/12	mié 03/10/12	5
Casos de uso	3 días	jue 04/10/12	lun 08/10/12	13
Procesos	7 días	mar 09/10/12	mié 17/10/12	14
Diseño	**16 días**	**jue 18/10/12**	**jue 08/11/12**	**12**
Diagrama de datos	1 día	jue 18/10/12	jue 18/10/12	12
Diagrama de clase	3 días	vie 19/10/12	mar 23/10/12	17
Diseño aplicación	2 días	mié 24/10/12	jue 25/10/12	18
Diseño formularios	5 días	vie 26/10/12	jue 01/11/12	19
Diseño listados	5 días	vie 02/11/12	jue 08/11/12	20
Entrega documento analisis y diseño	0 días	jue 08/11/12	jue 08/11/12	
Implementación	**32 días**	**jue 08/11/12**	**vie 21/12/12**	**22**
Implementación core	4 días	jue 08/11/12	mar 13/11/12	22
Alta documento	6 días	mié 14/11/12	mié 21/11/12	24
Exploración carpetas	3 días	jue 22/11/12	lun 26/11/12	25
Busqueda de documentos	8 días	mar 27/11/12	jue 06/12/12	26
Detalle documento	4 días	vie 07/12/12	mié 12/12/12	27
Gestión favoritos	7 días	jue 13/12/12	vie 21/12/12	28
Entrega maqueta	0 días	lun 17/12/12	lun 17/12/12	
Documentación	16 días	lun 24/12/12	lun 14/01/13	23
Pruebas y correciones	16 días	lun 24/12/12	lun 14/01/13	23
Entrega final	0 días	lun 14/01/13	lun 14/01/13	

Figura1: Plan de proyecto

1.5. Productos obtenidos.

El resultado de este TFC son los siguientes:

- La memoria, documento de resumen que sirve para definir y explicar los aspectos técnicos y funcionales del proyecto.

- La presentación, en formato, que a modo de resumen, remarca las fases del proyecto y el resultado final obtenido.

- La aplicación, basada en el estándar J2EE, entregada en formato de archivo EAR para su despliegue en un servidor WebSphere y en formato WAR para su despliegue en un servidor Tomcat.

- El servidor de pruebas proporcionado por Apache Chemistry que permite testear la aplicación si no se dispone de un servidor CMIS.

1.6. Breve descripción de los otros capítulos de la memoria.

En los próximos capítulos de la memoria, comentaremos los resultados obtenidos en las diferentes fases del proyecto, es decir, el resultado del análisis funcional, del diseño técnico y de la implementación, que incluirá las instrucciones de instalación y despliegue.

2. Análisis funcional

El análisis funcional es el primer producto obtenido en este proyecto. Contiene la especificación de las funcionalidades que tiene que cumplir el software a desarrollar.

En este apartado del documento, estudiaremos los diferentes tipos de actores que pueden interactuar con la aplicación, los casos de uso de estos actores y unos prototipos de la interface que al final tendrá la aplicación.

2.1. Actores

Usuario no autentificado

Este usuario se corresponde a cualquier usuario que accede a un url de la aplicación sin estar identificado en ella. Como todas las funcionalidades de la aplicación requieren que el usuario esté validado en ella, a este usuario se le dirigirá a la pantalla de identificación para que introduzca el usuario y la contraseña.

En ningún momento la aplicación permitirá ni el alta ni tendrá ningún proceso de recuperación de la contraseña, ya que usualmente los usuarios de un gestor documental son usuarios dados de alta en un directorio corporativo multifunción gestionado por otros mecanismos.

El usuario no autenticado podrá realizar las siguientes acciones:

- Identificarse en la aplicación

Usuario autentificado

Cuando el usuario no autentificado introduce su nombre y su contraseña y este es verificado por el sistema, pasa a ser un usuario autentificado. En la pantalla se le muestran el menú principal y sus opciones favoritas, a partir de este momento puede navegar por la aplicación, interactuando con los documentos, carpetas y tipos documentales del gestor documental a los que tiene permisos.

El usuario no autenticado podrá realizar las siguientes acciones:

- Cerrar sesión en la aplicación.
- Consultar sus accesos favoritos.
- Crear un documento.
- Archivar un documento en una carpeta.
- Buscar un documento.
- Navegar por las carpetas.
- Consultar un documento.
- Modificar los metadatos de un documento.
- Reservar un documento.
- Incorporar una nueva versión del documento.
- Eliminar un documento.
- Extraer un documento de una carpeta.

2.2. Casos de uso

2.2.1. Usuario no autenticado

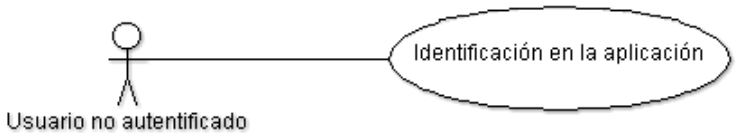


Figura 2: Casos de uso – Usuario no autenticado

Identificación en la aplicación
Funcionalidad: Permite a un usuario acceder a la aplicación
Actor: Usuario no autentificado
Precondición: El usuario está registrado en el directorio de la organización y conoce su identificador de usuario y contraseña.
Postcondición: El usuario accede a la aplicación con sus permisos correspondientes.
Flujo de eventos: El usuario introduce su identificador de usuario y contraseña en los campos correspondientes y pulsa el botón Aceptar. Si los datos son correctos accede a la aplicación.
Flujo alternativo: Los datos introducidos no son válidos, se le muestra al usuario un mensaje informativo a tal respecto,

2.2.2. Usuario autenticado – Gestión de documentos

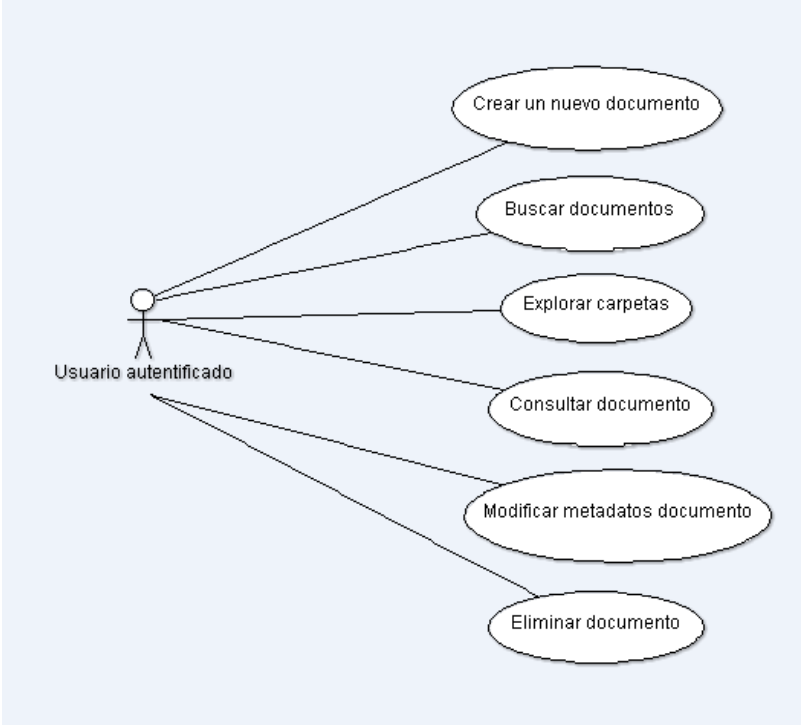


Figura 3: Casos de uso – Usuario autenticado. Gestión de documentos

Crear un nuevo documento
Funcionalidad: Permite a un usuario crear un nuevo documento
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación

Postcondición: Un nuevo documento se ha dado de alta en el gestor documental

Flujo de eventos: El usuario pulsa en uno de los botones de crear nuevo documento, se le muestra una pantalla para seleccionar el tipo documental, elige uno y se le muestra un formulario con los metadatos de ese tipo documental y un campo para subir un archivo. El usuario rellena el formulario pulsa el botón aceptar, se guarda el nuevo documento y se la muestra el formulario del nuevo documento en modo lectura.

Flujo alternativo: Los datos introducidos no son válidos, se le muestra al usuario un mensaje informativo a tal respecto.
El usuario quiere cancelar el proceso en cualquiera de los pasos, pulsa el botón de cancelar y vuelve a la pantalla en la que estaba. El usuario pulsa otra opción del menú para realizar otra acción diferente.

Buscar documentos

Funcionalidad: Permite a un usuario buscar un documento existente

Actor: Usuario autentificado

Precondición: El usuario está registrado en la aplicación

Postcondición: El usuario ha localizado los documentos que buscaba

Flujo de eventos: El usuario pulsa en uno de los botones de buscar documentos, se le muestra una pantalla para seleccionar el tipo documental, elige uno y se le muestra un formulario con los metadatos de ese tipo documental. El usuario rellena el formulario con los campos por los que desea buscar y pulsa el botón aceptar. Se muestran en un listado los documentos que cumplen con los requerimientos especificados en la búsqueda.

Flujo alternativo: Los datos introducidos no son coinciden con ningún documento del gestor documental. Se informa al usuario de que no se han encontrado resultados.
El usuario quiere cancelar el proceso en cualquiera de los pasos, pulsa el botón de cancelar y vuelve a la pantalla en la que estaba. El usuario pulsa otra opción del menú para realizar otra acción diferente.

Explorar carpetas

Funcionalidad: Permite a un usuario explorar el árbol de carpetas hasta alcanzar la carpeta que desea y poder consultar los documentos a ella asociados.

Actor: Usuario autentificado

Precondición: El usuario está registrado en la aplicación

Postcondición: El usuario ha localizado la carpeta y los documentos que buscaba

Flujo de eventos: El usuario pulsa en uno de los botones de explorar carpetas, se le muestra una pantalla con las carpetas situadas en la carpeta raíz y los documentos existentes en esa carpeta. El usuario puede ir haciendo clic sucesivamente en una carpeta hasta llegar al punto del árbol de carpetas que desea. Una vez allí se muestran en un listado las carpetas que contiene esa carpeta y en otro los documentos almacenados en la carpeta actual.

Flujo alternativo: El usuario ha accedido a una carpeta que no quería y vuelve a la carpeta actual o a alguna anterior de su ruta de navegación. El usuario pulsa otra opción del menú para realizar otra acción diferente.

Consultar documento

Funcionalidad: Permite a un usuario consultar el contenido y los metadatos de un documento que ha localizado por una búsqueda o una exploración de carpetas.

Actor: Usuario autentificado

Precondición: El usuario está registrado en la aplicación y ha realizado una búsqueda o una exploración de carpetas

Postcondición: El usuario ha visualizado los datos y/o el contenido del documento

Flujo de eventos: Desde el listado de respuesta a la búsqueda o desde el listado de exploración de carpetas, el usuario hace clic sobre uno de los documentos. Se muestra un formulario en modo edición con los metadatos del documento así como un enlace para ver o descargar el contenido.

Flujo alternativo:

Modificar metadatos de un documento

Funcionalidad: Permite a un usuario modificar los metadatos de un documento que ha localizado por una búsqueda o una exploración de carpetas y del que estaba consultando los datos.
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación, ha realizado una búsqueda o una exploración de carpetas y ha accedido al detalle de un documento
Postcondición: El usuario ha modificado los metadatos del documento
Flujo de eventos: Desde el detalle del documento, el usuario pulsa en editar metadatos, aparece un formulario como el de crear documento pero con los datos actuales del documento. El usuario modifica los datos que desea y guarda los cambios. La pantalla muestra de nuevo los datos del documento en modo lectura.
Flujo alternativo: El usuario tras pulsar el botón editar metadatos cambia de opinión y pulsa el botón cancelar por lo que la aplicación vuelve a la visualización de metadatos sin hacer cambios

Eliminar documento
Funcionalidad: Permite a un usuario eliminar un documento que ha localizado por una búsqueda o una exploración de carpetas y del que estaba consultando los datos.
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación, ha realizado una búsqueda o una exploración de carpetas y ha accedido al detalle de un documento
Postcondición: El usuario ha eliminado el documento
Flujo de eventos: Desde el detalle del documento, el usuario pulsa en borrar documento. Se le muestra un mensaje pidiendo la confirmación de borrado, el usuario acepta y la aplicación vuelve a la pantalla de listado desde donde se había accedido a la aplicación.
Flujo alternativo: El usuario tras pulsar el botón borrar documento cambia de opinión y no acepta la confirmación. La pantalla vuelve al detalle de documento sin hacer cambios.

2.2.3. Usuario autenticado – Gestión de carpetas

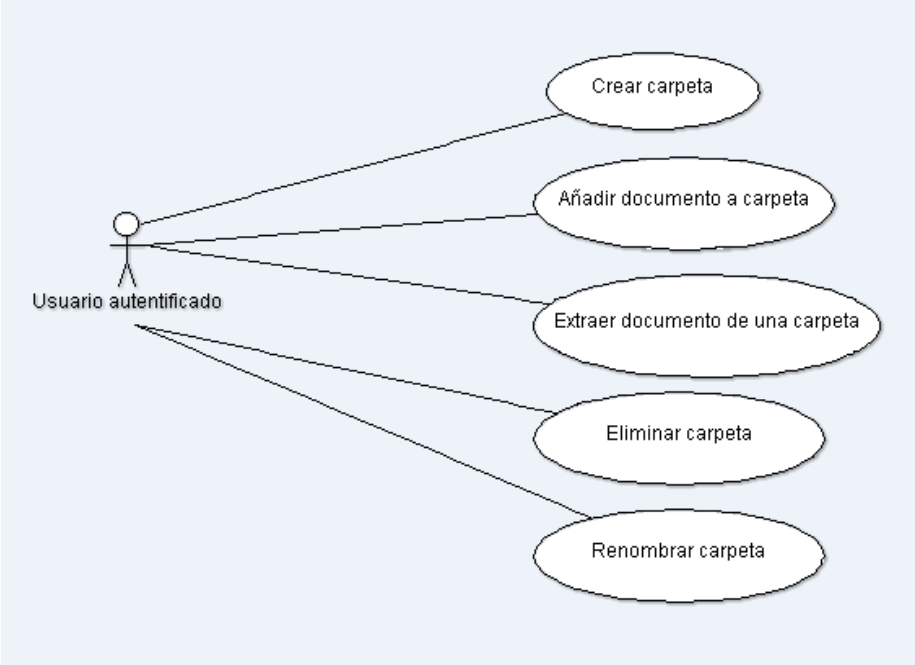


Figura 4: Casos de uso – Usuario autenticado. Gestión de carpetas

Crear carpeta
Funcionalidad: Permite a un usuario crear una nueva carpeta
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación y mediante la exploración de carpetas se sitúa en una carpeta concreta donde desea crear la subcarpeta.

Postcondición: Una nueva subcarpeta se ha creado dentro de la carpeta actual

Flujo de eventos: En la carpeta actual el usuario pulsa el botón Crear carpeta, aparece un formulario solicitando el nombre de la carpeta. Tras indicar el nombre pulsa Aceptar y la nueva carpeta se crea.

Flujo alternativo: En la carpeta actual el usuario pulsa el botón Crear carpeta, aparece un formulario solicitando el nombre de la carpeta. El usuario cambia de opinión y pulsa Cancelar, la aplicación vuelve a la carpeta donde esta sin hacer cambios.

Añadir documento a carpeta

Funcionalidad: Permite a un usuario añadir un documento a una carpeta

Actor: Usuario autentificado

Precondición: El usuario está registrado en la aplicación y mediante la exploración de carpetas o la búsqueda entra en el detalle de un documento.

Postcondición: El documento se ha añadido a la carpeta deseada

Flujo de eventos: Desde el detalle del documento, el usuario pulsa en Copiar en otra carpeta. Se muestra la carpeta raíz del árbol, el usuario navega hasta la carpeta deseada y pulsa el botón Pegar documento, el documento se asocia a esa carpeta.

Flujo alternativo: El usuario tras pulsar el botón Copiar en otra carpeta cambia de opinión y no lo pega. El documento estará disponible para pegar hasta que se seleccione otro o se cierre la sesión.

Extraer documento de carpeta

Funcionalidad: Permite a un usuario extraer un documento de una carpeta

Actor: Usuario autentificado

Precondición: El usuario está registrado en la aplicación y mediante la exploración de carpetas o la búsqueda entra en el detalle de un documento.

Postcondición: El documento se ha añadido eliminado de la carpeta seleccionada

Flujo de eventos: Desde el detalle del documento, el usuario pulsa en Eliminar de carpeta. El documento se desasocia de esa carpeta.

Flujo alternativo:

Eliminar carpeta

Funcionalidad: Permite a un usuario eliminar una nueva carpeta

Actor: Usuario autentificado

Precondición: El usuario está registrado en la aplicación y mediante la exploración de carpetas se sitúa en la carpeta concreta que desea eliminar.

Postcondición: La carpeta se ha eliminado

Flujo de eventos: En la carpeta actual el usuario pulsa el botón Borrar carpeta, aparece un mensaje pidiendo confirmación, si el usuario acepta, la carpeta se elimina.

Flujo alternativo: En la carpeta actual el usuario pulsa el botón Borrar carpeta, aparece un mensaje pidiendo confirmación, el usuario cambia de opinión y no lo acepta, el mensaje se cierra y no pasa nada.

Renombrar carpeta

Funcionalidad: Permite a un usuario renombrar una carpeta

Actor: Usuario autentificado

Precondición: El usuario está registrado en la aplicación y mediante la exploración de carpetas se sitúa en la carpeta concreta que desea renombrar.

Postcondición: La carpeta se ha renombrado

Flujo de eventos: En la carpeta actual el usuario pulsa el botón Renombrar carpeta, aparece un formulario con un campo para indicar el nuevo nombre, si el usuario acepta, la carpeta se renombra.

Flujo alternativo: En la carpeta actual el usuario pulsa el botón Renombrar carpeta, aparece un formulario con un campo para indicar el nuevo nombre. El usuario cambia de opinión y pulsa Cancelar, la aplicación vuelve a la carpeta donde esta sin hacer cambios.

2.2.4. Usuario autenticado – Gestión de versiones

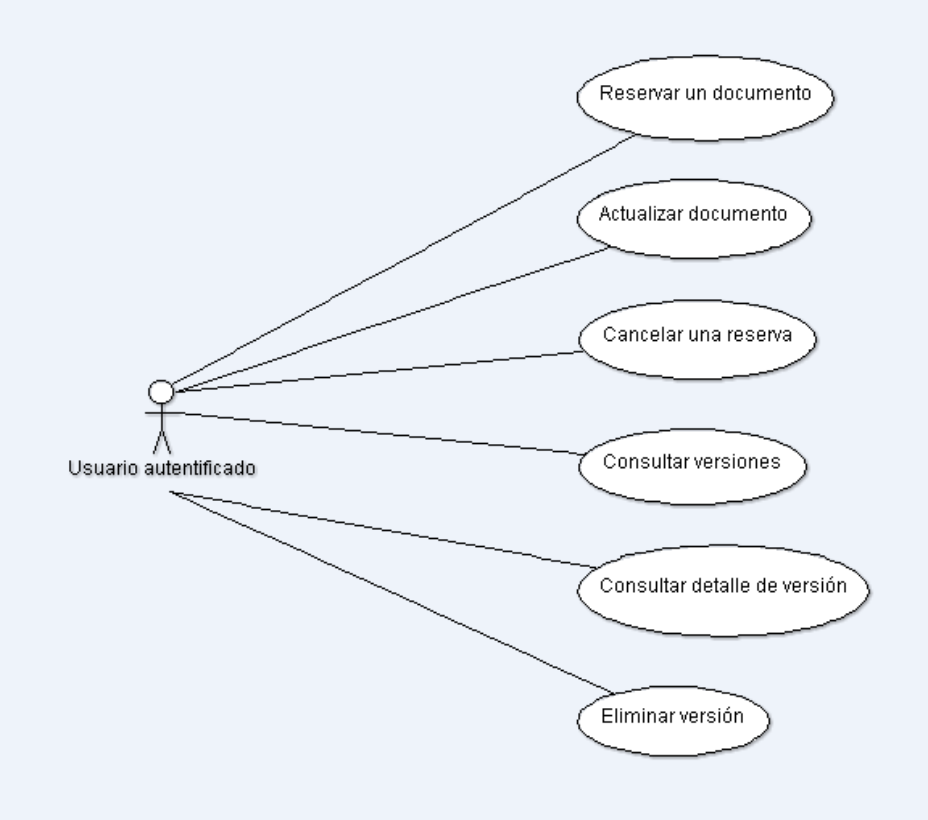


Figura 5: Casos de uso – Usuario autenticado. Gestión de versiones

Reservar documento
Funcionalidad: Permite a un usuario hacer la reserva de un documento que ha localizado por una búsqueda o una exploración de carpetas y del que estaba consultando los datos con el objetivo de modificar el contenido y crear una nueva versión.
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación, ha realizado una búsqueda o una exploración de carpetas y ha accedido al detalle de un documento
Postcondición: El usuario ha reservado el documento
Flujo de eventos: Desde el detalle del documento, el usuario pulsa en reservar documento. Se le muestra un mensaje pidiendo la confirmación de reserva, el usuario acepta y la aplicación vuelve a la pantalla de detalle donde se indica que el documento está reservado.
Flujo alternativo: El usuario tras pulsar el botón reservar documento cambia de opinión y no acepta la confirmación. La pantalla vuelve al detalle de documento sin hacer cambios.

Actualizar documento
Funcionalidad: Permite a un usuario actualizar el contenido y metadatos de un documento creando una nueva versión del mismo.
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación, ha realizado una búsqueda o una exploración de carpetas y ha accedido al detalle de un documento que había reservado anteriormente.
Postcondición: El usuario ha actualizado el contenido y los metadatos de un documento
Flujo de eventos: Tras localizar el documento anteriormente reservado y acceder a su detalle, el usuario pulsa en actualizar documento. Se mostrará un formulario con los metadatos existentes y un campo de archivo para subir la nueva versión. Al guardar el documento se crea la nueva versión y se muestra en modo visualización.
Flujo alternativo: El usuario tras pulsar el botón actualizar el documento cambia de opinión y

pulsa cancelar. La pantalla vuelve al detalle de documento sin hacer cambios.

Cancelar una reserva
Funcionalidad: Permite a un usuario cancelar la reserva de un documento que había reservado anteriormente.
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación, ha realizado una búsqueda o una exploración de carpetas y ha accedido al detalle de un documento que había reservado anteriormente.
Postcondición: El usuario ha eliminado la reserva del documento
Flujo de eventos: Tras localizar el documento anteriormente reservado y acceder a su detalle, el usuario pulsa en cancelar la reserva. Se le muestra un mensaje pidiendo la confirmación de la cancelación, el usuario acepta y la aplicación vuelve a la pantalla de detalle del documento.
Flujo alternativo: El usuario tras pulsar el botón cancelar la reserva cambia de opinión y no acepta la confirmación. La pantalla vuelve al detalle de documento sin hacer cambios.

Consultar versiones
Funcionalidad: Permite a un usuario consultar las versiones de un documento que ha localizado por una búsqueda o una exploración de carpetas y del que estaba consultando los datos.
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación, ha realizado una búsqueda o una exploración de carpetas y ha accedido al detalle de un documento
Postcondición: El usuario ha consultado las versiones de un documento
Flujo de eventos: Desde el detalle del documento, el usuario pulsa en consultar versiones. Se le muestra un listado donde se muestran de forma ordenada todas las versiones del documento.
Flujo alternativo:

Consultar detalle de versión
Funcionalidad: Permite a un usuario seleccionar una de las versiones de un documento mostradas en la consulta de versiones para ver su detalle.
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación, ha realizado una búsqueda o una exploración de carpetas, ha accedido al detalle de un documento y ha pulsado en consultar versiones.
Postcondición: El usuario ha consultado el detalle de una versión de un documento
Flujo de eventos: Desde el detalle del documento, el usuario pulsa en consultar versiones. Se le muestra un listado donde se muestran de forma ordenada todas las versiones del documento. El usuario hace clic en una de las versiones y visualiza el detalle de la misma.
Flujo alternativo: Una vez en el listado de versiones de documento, el usuario pulsa cancelar y vuelve al detalle del documento en vigor.

Eliminar versión
Funcionalidad: Permite a un usuario seleccionar una de las versiones de un documento mostradas en la consulta de versiones para ver su detalle y desde el eliminarla.
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación, ha realizado una búsqueda o una exploración de carpetas, ha accedido al detalle de un documento, ha pulsado en consultar versiones y ha seleccionado una del as versiones.
Postcondición: El usuario ha eliminado una versión de un documento
Flujo de eventos: Desde el detalle del documento, el usuario pulsa en consultar versiones. Se le muestra un listado donde se muestran de forma ordenada todas las versiones del documento. El usuario hace clic en una de las versiones y visualiza el detalle de la misma. El usuario pulsa eliminar versión, se muestra un mensaje de confirmación, si el usuario lo acepta, se elimina la versión y se vuelve al listado de versiones.
Flujo alternativo: Una vez en el detalle de la versión el usuario pulsa eliminar, se muestra el mensaje de confirmación pero el usuario no lo acepta. Se vuelve a la pantalla de detalle sin ningún cambio.

2.2.5. Usuario autenticado – Gestión de favoritos

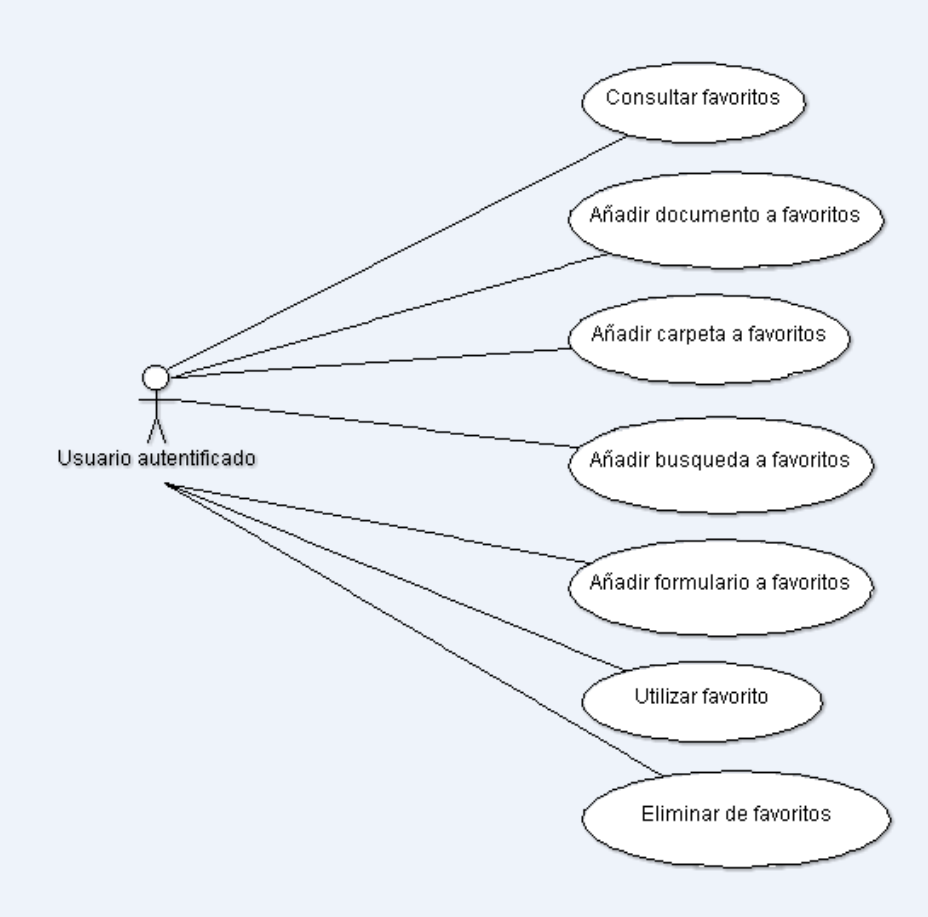


Figura 6: Casos de uso – Usuario autenticado. Gestión de favoritos

Consultar favoritos
Funcionalidad: Permite a un usuario consultar sus favoritos.
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación.
Postcondición: El usuario ha consultado sus favoritos.
Flujo de eventos: Desde cualquier parte de la aplicación, el usuario pulsa en la opción del menú principal, favoritos. La aplicación muestra en su parte central cuatro bloques de favoritos, uno de documentos otro de carpetas, otro de formularios y otro de búsquedas.
Flujo alternativo:

Añadir documento a favoritos
Funcionalidad: Permite a un usuario añadir a sus favoritos un documento que ha localizado por una búsqueda o una exploración de carpetas y del que estaba consultando los datos.
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación, ha realizado una búsqueda o una exploración de carpetas y ha accedido al detalle de un documento
Postcondición: El usuario ha añadido el documento actual a los favoritos.
Flujo de eventos: Desde el detalle del documento, el usuario pulsa en añadir a favoritos. Se le muestra un mensaje pidiendo la confirmación, el usuario acepta y la aplicación vuelve al documento donde se muestra el botón de eliminar de favoritos y una marca de documento favorito.

Flujo alternativo: El usuario tras pulsar el botón de añadir a favoritos cambia de opinión y no acepta la confirmación. La pantalla vuelve al detalle de documento sin hacer cambios.

Añadir carpeta a favoritos

Funcionalidad: Permite a un usuario añadir a sus favoritos una carpeta que ha localizado por una exploración de carpetas.

Actor: Usuario autentificado

Precondición: El usuario está registrado en la aplicación, ha realizado una exploración de carpetas.

Postcondición: El usuario ha añadido el documento actual a los favoritos.

Flujo de eventos: Desde la carpeta actual, el usuario pulsa en añadir a favoritos. Se le muestra un mensaje pidiendo la confirmación, el usuario acepta y la aplicación vuelve a la carpeta donde se muestra el botón de eliminar de favoritos y una marca de carpeta favorita.

Flujo alternativo: El usuario tras pulsar el botón de añadir a favoritos cambia de opinión y no acepta la confirmación. La pantalla vuelve a la carpeta sin hacer cambios.

Añadir búsqueda a favoritos

Funcionalidad: Permite a un usuario añadir a sus favoritos unos criterios de búsqueda de documentos para su posterior reutilización.

Actor: Usuario autentificado

Precondición: El usuario está registrado en la aplicación, ha realizado una búsqueda.

Postcondición: El usuario ha añadido la búsqueda actual a los favoritos.

Flujo de eventos: Desde el detalle formulario de búsqueda el usuario pulsa en añadir a favoritos. Se le muestra un mensaje pidiendo la confirmación, el usuario acepta y la aplicación solicita un nombre para guardar la búsqueda. El usuario indica el nombre, pulsa aceptar y un mensaje indica que la búsqueda se ha añadido a los favoritos. Al aplicación vuelve al formulario de búsqueda tal y como estaba.

Flujo alternativo: El usuario tras pulsar el botón de añadir a favoritos cambia de opinión y no acepta la confirmación. La pantalla vuelve al detalle de búsqueda sin hacer cambios.

Añadir formulario a favoritos

Funcionalidad: Permite a un usuario añadir a sus favoritos un formulario de creación de documentos con algunos datos precargados para su posterior reutilización.

Actor: Usuario autentificado

Precondición: El usuario está registrado en la aplicación, ha comenzado la creación de un nuevo documento.

Postcondición: El usuario ha añadido el formulario de creación de documento con los datos actuales a los favoritos.

Flujo de eventos: Desde el detalle formulario de creación de documento, el usuario pulsa en añadir a favoritos. Se le muestra un mensaje pidiendo la confirmación, el usuario acepta y la aplicación solicita un nombre para guardar el formulario. El usuario indica el nombre, pulsa aceptar y un mensaje indica que el formulario se ha añadido a los favoritos. La aplicación vuelve al formulario tal y como estaba.

Flujo alternativo: El usuario tras pulsar el botón de añadir a favoritos cambia de opinión y no acepta la confirmación. La pantalla vuelve al detalle de formulario sin hacer cambios.

Utilizar favorito

Funcionalidad: Permite a un usuario utilizar uno de sus favoritos para ir a un documento, una carpeta, crear un documento o iniciar una búsqueda.

Actor: Usuario autentificado

Precondición: El usuario está registrado en la aplicación y ha consultado sus favoritos.

Postcondición: El usuario ha ido al detalle de un documento, a una carpeta, ha iniciado la creación de un documento o la búsqueda de unos documentos desde uno de sus favoritos.

Flujo de eventos: Desde cualquier parte de la aplicación, el usuario pulsa en la opción del menú principal, favoritos. La aplicación muestra en su parte central cuatro bloques de favoritos, uno de documentos otro de carpetas, otro de formularios y otro de búsquedas- El usuario puede seleccionar cualquiera de ellos simplemente haciendo clic en el descriptor. Dependiendo el tipo de favorito se lleva a cabo una u otra opción.

Flujo alternativo: El usuario accede al listado de favoritos pero no hace clic en ninguno. No se

produce ninguna acción hasta que el usuario pulse en otra opción del menú de la aplicación.

Eliminar favorito
Funcionalidad: Permite a un usuario uno de sus
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación y ha consultado sus favoritos o está en uno de sus documentos favoritos o en una carpeta favorita.
Postcondición: El usuario ha eliminado uno de sus favoritos.
Flujo de eventos: Desde cualquier parte de la aplicación, el usuario pulsa en la opción del menú principal, favoritos. La aplicación muestra en su parte central cuatro bloques de favoritos, uno de documentos otro de carpetas, otro de formularios y otro de búsquedas. Cada uno de los favoritos tiene un icono de borrado a la derecha, al pulsarlo se le muestra un mensaje de confirmación de borrado, si se acepta, el favorito se borra. Se actualiza el listado de favoritos. Al acceder a un documento o carpeta favorita se muestra un botón de eliminar de favoritos. Si el usuario lo pulsa, se muestra un mensaje de aceptación que si el usuario lo acepta elimina el documento de favoritos. Se actualiza el documento o la carpeta eliminando la información de favorito.
Flujo alternativo: El usuario hace clic en una de las dos opciones para eliminar de favoritos pero no acepta el mensaje de confirmación. La pantalla se queda como estaba.

2.3. Prototipado de pantallas

Las pantallas de la aplicación seguirán un diseño compacto y coherente. Podremos distinguir entre cuatro tipos de pantallas:

Pantalla de logan:

La pantalla de logan solo se utilizara para permitir al usuario introducir la sus credenciales de acceso a la aplicación, por tanto no contendrá más que dos campos de entrada y un botón de aceptar. El resto de elementos de esta pantalla serán meros elementos estéticos o promocionales.

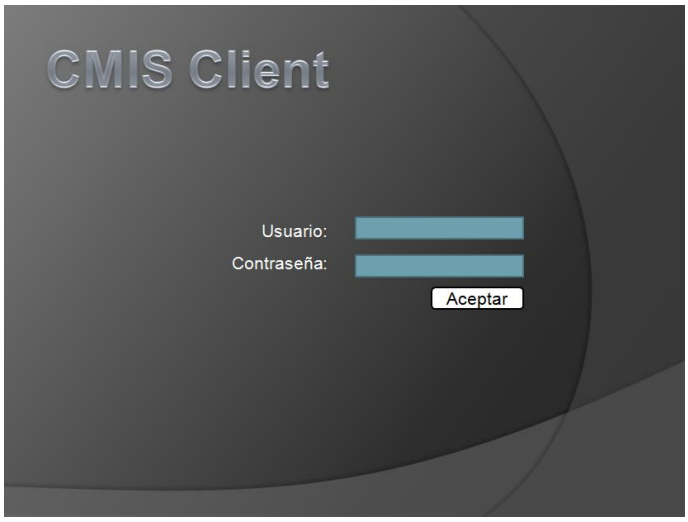


Figura 7: Prototipado de pantallas – Usuario Pantalla de logan

Elementos comunes pantallas interiores:

Todas las pantallas internas, tendrán el mismo aspecto general y estará compuesto por cuatro zonas principales:

- La cabecera, zona superior que contendrá el logo del producto, la información sobre el usuario conectado y el enlace para cerrar la sesión.
- El menú lateral izquierdo, que contendrá los enlaces a las diferentes secciones del programa, el usuario podrá hacer clic en ellas lo que modificará el contenido mostrado en la zona de contenidos.
- El pie de la aplicación, incluirá el copyright de la aplicación, así como información y enlaces sobre la misma.
- La zona de contenidos, la zona de contenidos modificará la información mostrada dependiendo de la navegación del usuario por el menú pero básicamente tendrá pantallas de tres tipos que se describen en los siguientes sub-apartados.

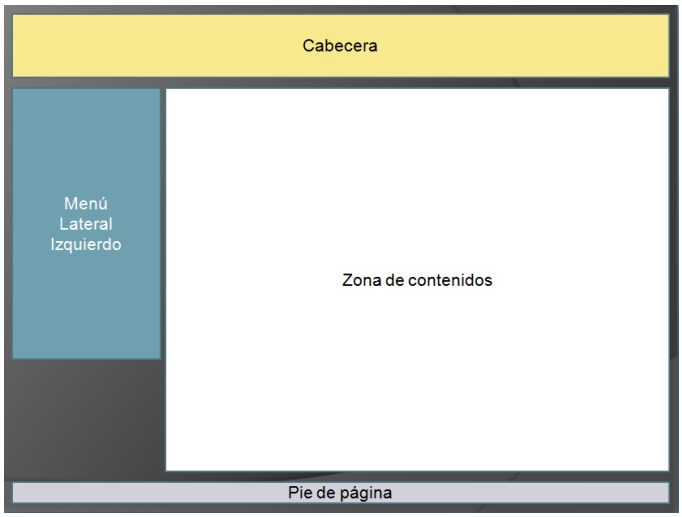


Figura 8: Prototipado de pantallas – Elementos comunes pantallas interiores

Pantalla de listado:

Las pantallas de listado serán aquellas donde se muestren varios elementos de contenido de un mismo tipo, de manera que puedan ser identificados y seleccionados para acceder a su detalle de metadatos o de contenido.

Los listados podrán disponer de unos campos para filtrado que se situarán en la parte superior de la zona de contenidos.

Las columnas permitirán seleccionar la columna y la dirección de ordenado de los resultados.
Los resultado estarán paginados pudiéndose seleccionar la página anterior y siguiente, la primera y la última o situarse en una página concreta.

Figura 9: Prototipado de pantallas – Pantalla de listado

Pantalla de detalle de información:

La pantalla de detalle d información mostrará los metadatos de un documento o objeto que hayamos seleccionado en un listado. Este tipo de pantalla mostrará los metadatos mediante una lista de parejas clave valor que estarán en modo lectura y por lo tanto no podrán modificarse.

En la parte superior de esta información el usuario dispondrá de una serie de botones de acción principales, que hacen referencia a procesos básicos de la aplicación, como pueden ser volver a la página anterior, editar, mover, borrar o copiar.

En la parte derecha parecerán acciones más relacionadas a la gestión documental, como pueden ser, hacer un checo-in, un check-out, ascender versión, consultar versiones anteriores, etc.

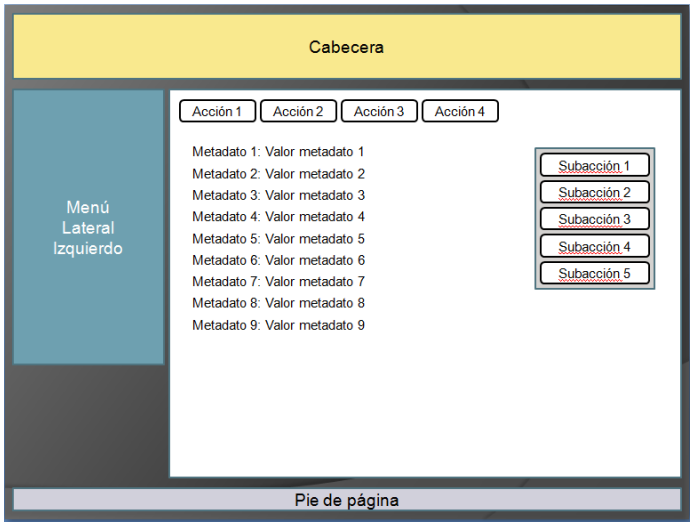


Figura 10: Prototipado de pantallas – Pantalla de detalle de información

Pantalla de formulario de entrada:

La pantalla de entrada de datos permitirá al usuario tanto dar de alta un formulario nuevo, como modificar los datos de un documento ya existente.

El formato del formulario es similar al de detalle y muestra la lista de metadatos mediante una serie de pares clave valor, pero en este caso el valor se muestra en campos editables.

Figura 11: Prototipado de pantallas – Pantalla de formulario de entrada

3. Diseño

Tras definir que tiene que hacer la aplicación, pasamos a definir cómo va a hacerlo. En este apartado especificamos los diferentes aspectos técnicos en los que se apoyará la aplicación.

3.1. Base arquitectónica – J2EE

La aplicación se desarrollará utilizando la plataforma de programación J2EE. J2EE es una plataforma Java, creada para la construcción de aplicaciones orientadas al cliente web y que cubre prácticamente todas las necesidades que este tipo de aplicaciones requiere.

J2EE da solución a necesidades del desarrollador como son la seguridad, la persistencia, la intercomunicación entre sistemas, la transaccionalidad estableciendo estándares como el de portlet, servlet, EJB o web services.

Evidentemente en este proyecto no utilizaremos todas estas herramientas pero si muchas de ellas.

3.2. Patrón de diseño – MVC

El modelo MVC hace referencia a un patrón de diseño de aplicaciones que especifica cómo debe ser estructurada una aplicación, las capas que la componen y la funcionalidad de cada una. Este modelo concretamente separa las aplicaciones Web en tres componentes distintos; los datos, la lógica de la aplicación y la interfaz de usuario con el objetivo de independizar lo máximo posible unos de otros a fin que las modificaciones estén lo más aisladas posibles.

Esta separación permite que se pueda plantear el diseño de cada una de las partes de manera independiente y eligiendo para cada una de ellas las herramientas que más se ajusten a nuestras necesidades.

Por este motivo podemos a continuación describir como se solucionarán cada uno de los componentes del patrón.

3.2.1. Modelo

En la arquitectura MVC la lógica de negocio de la aplicación, incluyendo el acceso a los datos y su manipulación, está encapsulada dentro del modelo. El Modelo lo forman una serie de componentes de negocio independientes del Controlador y la Vista, permitiendo así su reutilización y el desacoplamiento entre las capas.

Dentro del modelo de datos distinguiremos dos partes, la dedicada a comunicarse con el gestor documental, y la que almacenará las preferencias y configuraciones del usuario y que estará en base de datos.

La comunicación con el gestor documental, se llevará a cabo utilizando las librerías OpenCMIS de Apache Chemistry (http://chemistry.apache.org/project/cmis.html) ya que la comunicación directa con el protocolo CMIS se hace a un nivel muy bajo y estas librerías ya implementan todos los elementos necesarios y están debidamente probadas y optimizadas.

Para la comunicación con el gestor de base de datos, se utilizará el framework MyBatis (http://www.mybatis.org/core/es/index.html). MyBatis es un framework de persistencia que ofrece gran control sobre el SQL generado para las consultas a la vez que un mapeo muy robusto entre entidades y tablas.

3.2.2. Vista

Las respuestas HTML que se enviarán al navegador de usuario, serán generadas por la vista. Como normalmente estas respuestas muestran datos proporcionados por el Controlador, por lo que el código HTML de la página no será fijo si no que deberá ser generado de forma dinámica, por lo que su implementación correrá a cargo de una página JSP o un servlet.

Para la elaboración de la interface de usuario, utilizaremos como base de construcción de las páginas los JSPs, ampliando la funcionalidad básica con etiquetas JSTL (http://jstl.java.net/). Los JSP nos van a permitir tener total control sobre el código que generamos, mientras que las etiquetas JSTL nos permitirán reducir a la mínima expresión el uso de “scriptlets” obteniendo como resultado un código más limpio y claro.

Los jsps se organizarán en subcarpetas dentro de la carpeta "/WEB-INF/jsp” de esta manera los jsp no serán accesible de forma directa por los usuarios y solo serán utilizables como vista utilizada por el controlador.

Las partes comunes a todas las páginas, como son el menú, la cabecera y el pié de página se centralizarán en fragmentos de JSP únicos que se incluirán en el resto de páginas.

Se intentará que el trabajo en el browser sea el máximo posible, para ello se realizarán todas las validaciones y las ayudas de los formularios utilizando javascript, más concretamente utilizando Mootools (http://mootools.net/).

Mootools nos ofrece compatibilidad con la mayoría de los navegadores, modularidad y orientación a objetos, juntamente con un amplio catálogo de librerías y funcionalidades que aportan dinamismo y calidad a la interface de usuario.

3.2.3. Controlador

El controlador es el elemento de la aplicación al cual se dirigen todas las peticiones a la capa intermedia que se realicen desde el cliente, su misión es determinar las acciones a realizar

para cada una de estas peticiones e invocar al resto de los componentes de la aplicación (Modelo y Vista) para que realicen las acciones requeridas en cada caso, encargándose también de la coordinación de todo el proceso.

El controlador que utilizamos para construir la aplicación será el de Spring, concretamente en su modalidad de anotaciones. Con este modelo de construcción de controladores es posible enlazar las urls de las páginas directamente a los métodos de las clases controladoras, inyectar los servicios que nos den acceso al modelo de la aplicación y generar los contenedores de información que enviaremos a los JSP de presentación.

La configuración del controlador de Spring será el siguiente:

Activaremos el uso del los controladores anotados añadiendo las siguientes líneas en el fichero de configuración de Spring:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

   <bean
class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotationHandlerMapping"/>

   <bean
class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter"/>

   // ... (definiciones de beans controladores) ...

</beans>
```

En la parte (definiciones de beans controladores), añadiremos la definición de nuestros controladores. Inicialmente utilizaremos varios beans para no tener uno solo muy grande e inmanejable pudiendo organizar de esta manera funcionalidades asociadas.

De esta manera crearemos por ejemplo un controlador para los favoritos, con métodos para crear, listar y eliminar. Uno para carpetas, otro para documentos, etc.

Los controladores tendrán el siguiente aspecto:

```
@Controller
public class FavoritosController {

   @RequestMapping("/crearFavorito.html")
   public ModelAndView crearFavoritoHandler() {
            return …;
   }

   @RequestMapping("/eliminarFavorito.html")
   public ModelAndView eliminarFavoritoHandler (@RequestParam("favoritoId") int favoritoId) {
      return …;
   }

   @RequestMapping("/listarFavoritos.html")
   public ModelAndView listarFavoritosHandler() {
      return …;
   }
}
```

Las anotaciones determinarán el método a ejecutar, y en el caso de necesitar parámetros en la llamada, se enlazarán mediante la anotación @RequestParam.

En este modelo de controlador cobra especial importancia el objeto de retorno ModelAndView, ya que este objeto contendrá por un lado el jsp de la Vista que se invocará y por el otro lado, los datos del modelo que se enviará a ese jsp.

Por ejemplo:

```
ModelAndView mav = new ModelAndView("/WEB-INF/jsp/favoritos/list.jsp");
mav.addObject("favoritos", favoritosService.list(username));
```

En este caso estamos enviando la lista de favoritos que hemos recuperado mediante el servicio de favoritos al jsp list.jsp. Este jsp podrá utilizar la información enviada para representarla por pantalla.

3.3. Esquema de ejecución de las páginas

La ejecución de las páginas seguirá el siguiente diagrama:

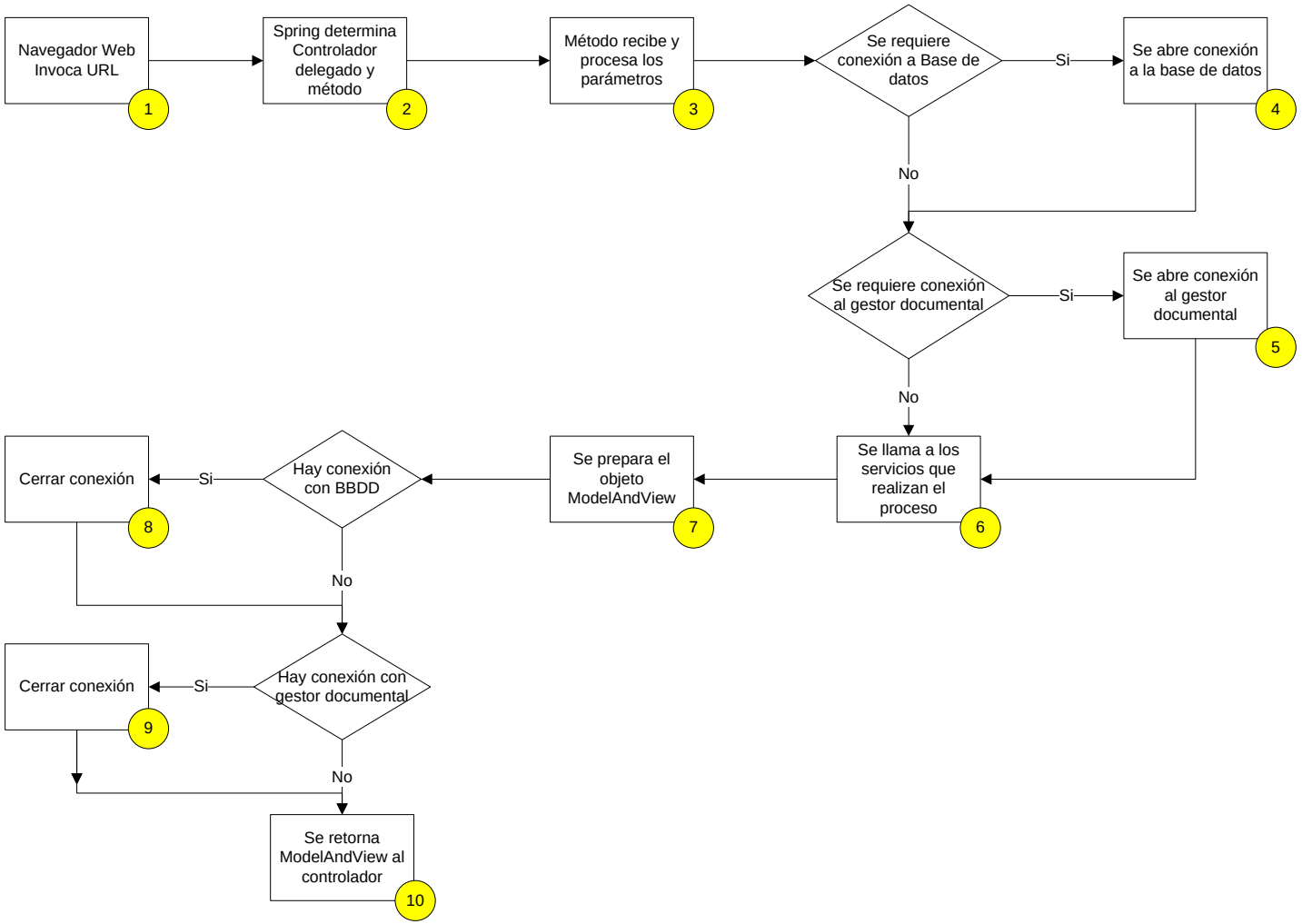


Figura 12: Diseño – Esquema de ejecución de páginas

1 La aplicación genera una llamada (request) a través de una url en el navegador, esta url es del tipo /raíz_de_contexto/*.html. Al detectar la raíz de contexto, el servidor de aplicaciones envía la petición a nuestra aplicación.

2 El controlador de Spring, busca entre los controladores que tiene registrados, cual da servicio a la url solicitada y llama al método correspondiente pasando los parámetros de la url a los parámetros anotados. Comienza la ejecución del método.

3 El método java invocado hace un preprocesado de los parámetros recibidos, si esto es necesario.

4 Si el proceso requiere interacción con la base de datos, se crea una conexión con ella. El objetivo de crear la conexión en este punto es por un lado el de poder reutilizarla por los diferentes servicios, de manera que una sola conexión sirva para todos. Por otro lado se evitan problemas de conexiones no cerradas o múltiples aperturas en bucles, problemas que pueden aparecer cuando la conexión se abre dentro de los servicios.

5 Si el proceso requiere interacción con el gestor documental, se crea una conexión con él. El objetivo de crear la conexión en este punto es el mismo que el del punto anterior.

6 En este punto se llaman a los diferentes servicios que realizan el procesado de datos. A estos datos se les pasará (cuando lo necesiten) las conexiones, los parámetros y otros posibles datos almacenados en sesión, como puede ser el nombre de usuario. Los servicios podrán devolver datos como respuesta al procesado.

7 Ya disponemos de todos los datos que necesitamos por tanto somos capaces de construir el objeto ModelAndView, fijar la jsp de retorno y preparar los datos que necesitaremos en el jsp.

8 Si hemos abierto una conexión a la base de datos, la cerraremos en un bloque finally, para que se cierre aunque se produzca un error.

9 Si hemos abierto una conexión al gestor documental, la cerraremos en un bloque finally, para que se cierre aunque se produzca un error.

10 Para finalizar se hace un return del ModelAndView. El controlador de Spring en este punto invoca el jsp indicado y pone a su disposición la información contenida en el objeto.

3.4. Gestión de errores

Para gestionar los posibles errores que se puedan producir se seguirá una estrategia de envío y no procesado de las excepciones hasta que lleguen al controlador. Es decir, todos los métodos llamados desde el método controlador, harán un throws Exception de manera que envíen la excepción al controlador.

El controlador tendrá una estructura de código como la siguiente:

```
try {
        …
        Proceso principal
        …
} cath (Exception e){
        …
```

```
            Gestión de errores
            …
} finally {
            …
            Cierre de conexiones
            …
}
```

En el caso de no producirse ningún error el proceso realizará todas las tareas del try, que generará el ModelAndView correspondiente y lo enviará. Se procesará el bloque finally que cerrará las conexiones.

Si se produce un error, la excepción llevará la ejecución al catch, que montarán un ModelAndView contra un error.jsp. Este error jsp informará sobre el error que se ha producido. Se procesará el bloque finally que cerrará las conexiones.

3.5. Trazabilidad

La trazabilidad de la aplicación se realizará utilizando la librería log4j. Esta librería permite la configuración de los niveles y destinos de los logs en ficheros de configuración permitiendo adecuar el detalle de los mismos en cada momento sin necesidad de modificar el código.

Para su implementación se añadirá al principio de cada clase una línea como esta:

```
static Logger log = Logger.getLogger(Clase.class);
```

En los sitios claves del código se añadirán mensajes de log de diferentes niveles dependiendo de las necesidades.

```
if(log.isDebugEnabled())log.debug("Mensaje de nivel DEBUG");
if(log.isInfoEnabled())log.info("Mensaje de nivel  INFO");
log.warn("Mensaje de nivel  WARN");
  log.error("Mensaje de nivel  ERROR");
  log.fatal("Mensaje de nivel  FATAL");
```

3.6. Servicios

Como hemos comentado en el esquema de ejecución, el código de la aplicación estará orientado a servicios, es decir todo código que realice alguna lógica de negocio compleja se organizará mediante métodos estáticos de clases servicio.

En nuestra aplicación dispondremos de los siguientes servicios:

- LoginService: Implementará los mecanismos de login.
- DocumentService: Englobará todas las funcionalidades relacionadas con los documentos.
- FolderService: Englobará todas las funcionalidades relacionadas con las carpetas.
- FavoriteService: Englobará todas las funcionalidades relacionadas con los favoritos.
- VersionService: Englobará las funcionalidades relacionadas con el control de versiones de los documentos.

Como ya hemos comentado, los servicios implementarán una serie de métodos estáticos que se llamarán desde la aplicación, de manera que las clases no serán instanciadas.

La plantilla de un servicio será la siguiente:

```
public class <Nombre>Service {

        public static <tipo retorno> metodo1(<Parametros>) throws Exception {
                return …;
        }

        public static <tipo retorno> metodo2(<Parametros>) throws Exception {
                return …;
        }

        …

}
```

3.7. Organización de las clases

A continuación se incluye una primera aproximación al diagrama de clases de la aplicación, estas clases se organizarán en seis paquetes principales que describimos a continuación:

- es.pfc.cmis.client.bean: contendrá las clases de objeto, la representación de los objetos reales, carpeta, documento, etc.
- es.pfc.cmis.client.controller: contendrá los controladores de la aplicación, completados con las anotaciones de Spring.
- es.pfc.cmis.client.dao: contendrá las clases de acceso a datos.
- es.pfc.cmis.client.service: contendrá las clases con los servicios.
- es.pfc.cmis.client.servlet: contendrá los servlets que pueda necesitar la aplicación.
- es.pfc.cmis.client.util: contendrá clases con utilidades varias.

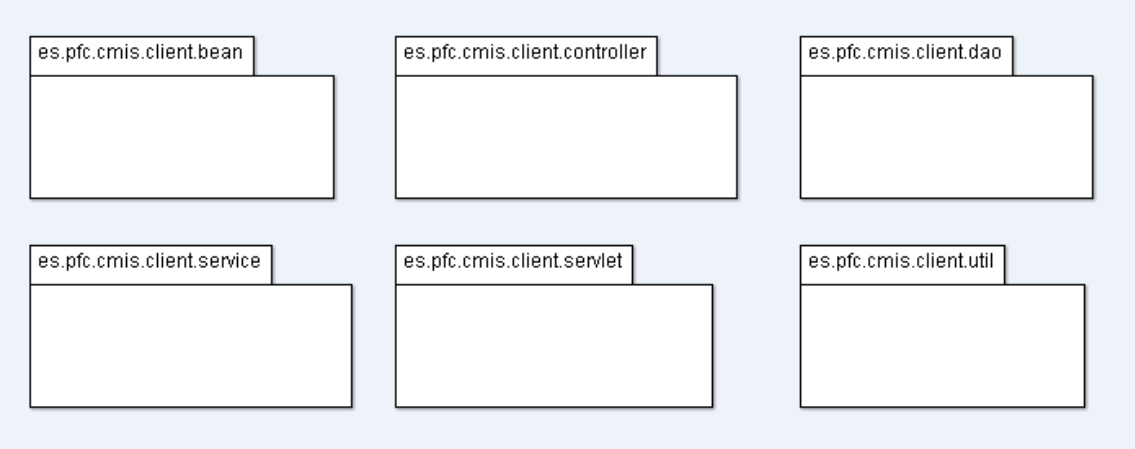


Figura 13: Diseño – Organización de las clases

3.8. Seguridad de la aplicación

Nuestra aplicación distinguirá únicamente entre usuarios autentificados y no autentificados, por tanto la seguridad consistirá en comprobar si esa validación se ha realizado. El mecanismo

utilizado será muy sencillo. Cualquier petición a la aplicación pasará por un filtro que comprobará si la información de usuario se encuentra en la sesión, si no es así se redirigirá al formulario de login.

El usuario indicará su usuario y contraseña que se validará mediante una conexión al gestor documental, si esta información es correcta, se añadirá la información de usuario a la sesión y se redirigirá a la home de la aplicación. Al existir ya la información de usuario en la sesión, el filtro no redirigirá y el usuario podrá navegar por la aplicación.

Para cerrar la sesión, bastará con que el usuario cierre el navegador o pulse el botón de cerrar sesión que simplemente eliminará la información del usuario de la sesión con lo que volverá a saltar el filtro a la pantalla de login.

El siguiente diagrama muestra los pasos del proceso.

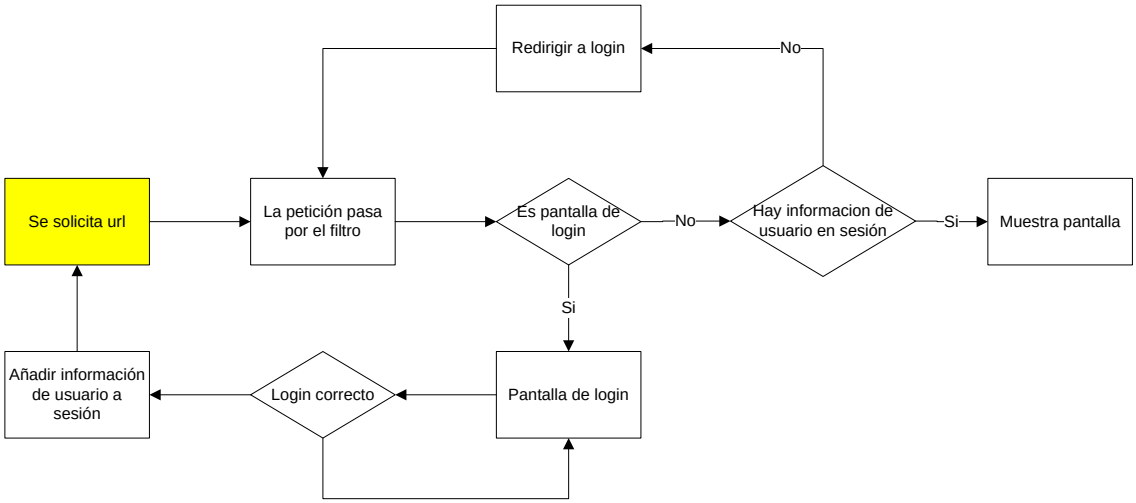


Figura 14: Diseño – Diagrama de proceso de control de seguridad

3.9. Base de datos

La importancia de la base de datos en este proyecto es muy pequeña ya que la mayoría de la información, se almacena en el gestor documental. De esta manera no tenemos que preocuparnos de guardar información sobre los documentos, carpetas o usuarios.

Sin embargo la funcionalidad relativa a los favoritos, o bien no está soportada por los gestores documentales o bien una implementación en los propios gestores representaría un peor rendimiento. La experiencia en gestores documentales indica que no son buenas herramientas para el almacenaje de información relacional.

En nuestro caso vamos a utilizar una base de datos convencional para almacenar esta información.

Utilizaremos cuatro tablas, una para cada tipo de favorito ya que la información que estos guarden será diferente, y en muy pocos casos, por no decir ninguno, necesitaremos una combinación de favoritos de distintos tipos.

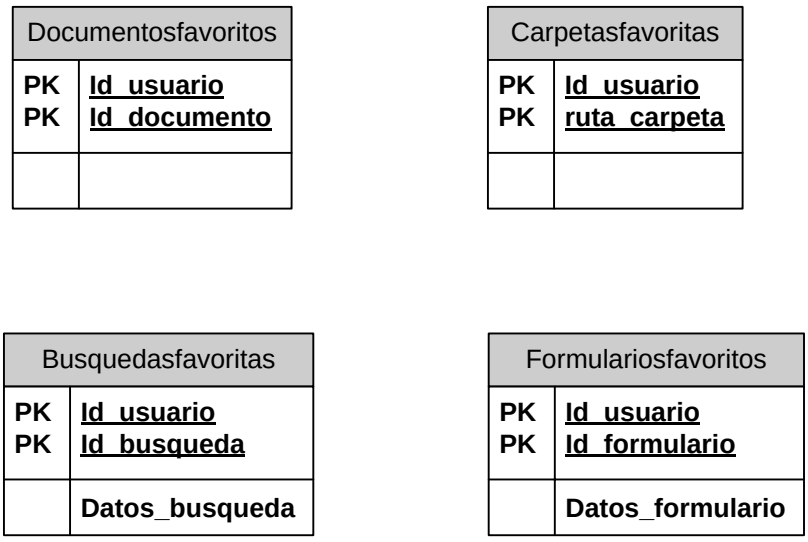


Figura 15: Diseño – Diagrama de tablas de datos

- La tabla de documentos favoritos estará formada por una columna que almacenará el nombre único del usuario y otra el id único del documento. Estas dos columnas formarán la clave primaria.

- La tabla de carpetas favoritas estará formada por una columna que almacenará el nombre único del usuario y otra la ruta completa de la carpeta. Estas dos columnas formarán la clave primaria.

- La tabla de búsquedas favoritas estará formada por una columna que almacenará el nombre único del usuario y otra el nombre identificador que le ha asignado el usuario. Estas dos columnas formarán la clave primaria. Una tercera columna guardara la información serializada de la búsqueda.

- La tabla de formularios favoritos estará formada por una columna que almacenará el nombre único del usuario y otra el nombre identificador que le ha asignado el usuario. Estas dos columnas formarán la clave primaria. Una tercera columna guardara la información serializada del formulario.

Como vemos las tablas no tendrán relaciones ni claves foráneas y los índices generados implícitamente por las claves únicas serán suficientes para un rendimiento correcto en las búsquedas.

4. Implementación

4.1. Decisiones de diseño e implementación

El diseño y la implementación de la aplicación están siguiendo el plan establecido en el documento de diseño, solo se han tenido que realizar una serie de correcciones que describimos a continuación.

- **Corrección en el diseño de tablas de favoritos**: Se ha detectado un error en el diseño de las tablas de favoritos ya que en el diseño inicial, estas no almacenaban el nombre del documento favorito, por lo que este debía ser recuperado del gestor documental con el correspondiente problema de rendimiento. En el diseño final el nombre del documento y de la carpeta se guardan en la tabla de favoritos, de esta manera no hay que conectarse al gestor documental.

  Por otra parte las carpetas disponen de id al igual de los documentos por lo que en la tabla de favoritos se guarda el id y no la ruta como se había diseñado inicialmente.

- **Gestión de conexiones al gestor documental**: Inicialmente en el documento de diseño, se establecía la necesidad de cerrar la conexión con el gestor documental al final de cada ejecución. Dado el sistema de conexión que se utiliza en Apache Chemistry, basadas en WebServices y Atom, no existe el concepto de conexión como tal, de manera que estas no tienen que ser creadas ni cerradas.

  En lugar de la conexión la aplicación establece una sesión que no tiene que ser cerrada.

- **Gestión de conexiones a la base de datos**: Similar a lo que sucede con las conexiones con el gestor documental, Ibatis no trabaja con el concepto de conexión, sino que utiliza un objeto de sesión. Sin embargo en este caso si se tiene que cerrar la sesión por lo que en este caso si utilizamos la técnica del "try – catch – finally" tal y como estaba previsto en el documento de diseño.

- **Nueva funcionalidad Consultar documentos reservados**: Se ha detectado que una funcionalidad no prevista en el diseño es necesaria para la correcta funcionalidad de la aplicación. La descripción del caso de uso es la siguiente:

Consultar documentos reservados
Funcionalidad: Permite a un usuario visualizar los documentos a los que ha realizado una reserva o check-out y está trabajando con ellos.
Actor: Usuario autentificado
Precondición: El usuario está registrado en la aplicación y ha pulsado en la opción Documentos reservados del menú lateral izquierdo.
Postcondición: El usuario ha consultado la lista de sus documentos reservados.
Flujo de eventos: Desde cualquier parte de la aplicación, el usuario pulsa en la opción del menú principal, Documentos reservados. La aplicación muestra en su parte central una lista con los documentos reservados por el usuario en ese momento.
Flujo alternativo:

- **Búsqueda directa por consulta**: Se ha detectado que una funcionalidad no prevista en el diseño es necesaria para potenciar la aplicación. La descripción del caso de uso es la siguiente:

Buscar documentos por consulta
Funcionalidad: Permite a un usuario buscar un documento existente por consulta
Actor: Usuario autentificado

Precondición: El usuario está registrado en la aplicación
Postcondición: El usuario ha localizado los documentos que buscaba
Flujo de eventos: El usuario pulsa en uno de los botones de buscar documentos, se le muestra una pantalla para indica la consulta que desea realizar. El usuario rellena el formulario con los campos por los que desea buscar y pulsa el botón aceptar. Se muestran en un listado los documentos que cumplen con los requerimientos especificados en la búsqueda.
Flujo alternativo: Los datos introducidos no son coinciden con ningún documento del gestor documental. Se informa al usuario de que no se han encontrado resultados. El usuario quiere cancelar el proceso en cualquiera de los pasos, pulsa el botón de cancelar y vuelve a la pantalla en la que estaba. El usuario pulsa otra opción del menú para realizar otra acción diferente.

- **Configuración de los parámetros de conexión al gestión documental**: Se han situado los parámetros de conexión al gestor documental en un fichero de propiedades situado en la ruta de clases, esto permitirá la configuración de los mismos si n que sea necesario recompilar el código.

- **Generación automática de objetos MyBatis**: Para la creación de los beans y de los ficheros xml de configuración de Mybatis se ha utilizado el sistema de generación automática que proporciona MyBatis Generator. El fichero /CMISClient/src/es/pfc/cmis/client/dao/xml/generatorConfig.xml contiene la configuración establecida para este generador.

- **Uso de librerías adicionales de Mootools:** Para la realización de la interfaz de usuario, se han utilizado una serie de librerías de javascript creadas para su uso con el framework Mootools. Estas librerías han dado solución a una serie de funcionalidades básicas y comunes a varias páginas mejorando considerablemente el resultado final. Las librerías utilizadas han sido:

    - **Arian Mootools Datepicker:** Nos proporciona un control para introducir las fechas en los formularios con el formato correcto.
    - **mBox:** Nos ha proporcionado vistosas ventanas modales y diálogos ce confirmación de decisiones.
    - **Omnigrid:** Permite la creación de las tablas de datos paginadas que hemos utilizado para nuestro listados.

- **Carga de datos en las páginas con tablas:** Hay muchas pantallas que muestran tablas de datos. Para implementarlas se ha utilizado un componente de Mootools llamado omnigrid que permite la carga de datos mediante JSON sin tener que recargar la pantalla entera. El funcionamiento de este tipo de pantallas es el siguiente.

    - Se invoca a una página enviando parámetros si estos son necesarios.
    - Esta pantalla contiene un div con un identificador, por ejemplo:

        ```
        <div id="favoriteDocumentListgrid" ></div>
        ```

    - Mediante javascript y tras terminarse de cargar la página, cosa que controlamos mediante "domready, creamos un grid en el div anterior. Por ejemplo:

```
window.addEvent("domready", function(){
```

```
datagridDocumentList = new omniGrid('favoriteDocumentListgrid', {
        columnModel: cmu,
    url:"favoriteDocumentListAjax.html",
    perPageOptions: [10,20,50,100,200],
    perPage:100,
    page:1,
    pagination:false,
    serverSort:false,
    alternaterows: true,
    showHeader:true,
    sortHeader:true,
    resizeColumns:false,
    multipleSelection:false,

    width:405,
    height: 200
  });

datagridDocumentList.addEvent('click', onGridSelectFD);
```

Podemos observar como en la creación del objeto especificamos entre otros muchos parámetros la url de la cual obtendremos el JDOM que nos traerá los datos.

En la página también dispondremos de la función que responderá al clic en una de las líneas del grid. Por ejemplo en este caso abriremos un detalle de documento.

```
function onGridSelectFD(evt) {
        location.href='browseDocument.html?idDocument='+evt.target.getDataByRow(evt.r
        ow).id_DOCUMENTO;
  }
```

Por último tendremos una sección donde indicaremos que columnas de los objetos JDOM queremos que se muestren en la tabla.

```
var cmu = [
        {
  header: "Nombre",
  dataIndex: 'nombre',
  dataType:'string',
  width:400
}
];
```

- **Construcción de los JDOM:** Los paquete de datos JDOM los generaremos de la misma forma que una página normal de la aplicación, e decir desde el controlador de Spring, solo que cambiaremos algunas cosas.

  Los métodos del controlador añadirán el parámetro Response, ya que será directamente a través del response como enviaremos los datos. No se utilizará un jsp.

  Fijaremos el código de caracteres para no tener problemas con caracteres extraños.

  ```
  response.setCharacterEncoding("UTF-8");
  ```

Crearemos un objeto suministrado por la biblioteca Jackson que nos transformará un objeto java en un JDOM.

ObjectMapper mapper = new ObjectMapper();

Utilizaremos este objeto para generar el JDOM resultante de la transformación del objeto directamente contra la salida del response.

mapper.writeValue(response.getWriter(), omniGrid);

En este caso hemos enviado un objeto omniGrid que detallamos a continuación.

- **Objeto OmniGrid:** Para mecanizar el uso de los grids de datos hemos creado un objeto al que hemos llamado OmniGrid, este objeto transporta los datos que el "grid" de Mootools necesita. Estos son:

  private Integer page; Número de página actual
  private Integer total; Numero de elementos totales
  private Collection<?> data; Datos a pintar en esa página

  Cabe destacar que el hecho de cortar los datos y enviar solamente los que necesitamos optimiza considerablemente el rendimiento y el tráfico generado.

- **Búsqueda por consulta:** Un caso especial es la pantalla de búsqueda por consulta. En toda la aplicación hemos intentado dejar la mínima información posible en sesión. Este caso ha sido especial ya que para detectar los errores de sintaxis, necesitábamos enviar la consulta. Si esta era incorrecta nos devolvía el error y lo informábamos pero si era correcta y redirigíamos a la pantalla del grid teníamos que volver a hacer la consulta.

  Para evitar esto, dejamos los resultados de la consulta en sesión y al llamar a la pantalla con la tabla, el método que monta el JDOM los recoge de ahí sin volver a realizar la consulta.

- **Ventanas modales:** Por motivos estéticos, se han sustituido las ventanas modales estándar de javascript, por unas más atractivas distribuidas en la librería para Mootools mBox. Se pueden ver en las confirmaciones de borrado y al guardar búsquedas y plantillas de documentos.

4.2. Requerimientos de software

La aplicación necesita los siguientes elementos para su correcto funcionamiento:

- Servidor de aplicaciones J2EE. En nuestro caso estamos utilizando un servidor de aplicaciones WebSphere Application Server, pero la aplicación debería funcionar en cualquier plataforma con algunos ajustes. Hemos adaptado la misma para que funcione igualmente en Tomcat, aunque por motivos de bibliotecas de Apache Chemistry, se tienen que hacer algunos ajustes. Incluimos los dos instalables, el EAR es para WebSphere y el WAR para Tomcat.

  A su vez el Tomcat necesita unas librerías adicionales que WAS ya trae incluidas y que se indican en el apartado "Instalación en Tomcat 7.0.34".

  Es posible descargarse una versión de desarrollo de websphere en:

- http://www.ibm.com/developerworks/ssa/websphere/was/

Y la versión de Tomcat puede descargarse en:

- http://tomcat.apache.org/download-70.cgi

- Servidor de base de datos. En nuestro caso estamos utilizando un servidor de base de datos DB2, pero es posible utilizar cualquier otro compatible con los estándares JDBC, como un Derby o un MySQL.

  Es posible descargarse una versión libre del db2 en la siguiente url:

  - http://www-01.ibm.com/software/data/db2/express-c/download.html

Para poder probar la aplicación es necesaria la conexión con un servidor de gestión documental que soporte el estándar CMIS. En nuestro caso el servidor al que nos conectamos es un FileNet Content Engine V5.1 con el modulo de CMIS instalado.

Algunos gestores documentales requieren unos ajustes especiales como puede leerse en el siguiente enlace:

http://chemistry.apache.org/java/developing/dev-repository-specific-notes.html

La aplicación ha sido testeada y diseñada para la utilización con navegador Firefox 17.0.1, el uso de cualquier otro navegador puede presentar incompatibilidades.

Puede descargarse el navegador Firefox desde este enlace

http://www.mozilla.org/es-ES/firefox/fx/

4.3. Consideraciones para la instalación y configuración de la aplicación

4.3.1. Creación de tablas de la base de datos

La aplicación necesita de la creación de cuatro tablas en una base de datos relacional, en nuestro caso hemos utilizado una base de datos DB2 en Linux. Los siguientes DDL crean las tablas en el esquema db2inst1 y el tablespace USERSPACE1.

Si se utiliza otro tipo de base de datos u otro esquema o tablespace, se tendrán que corregir los DDL.

```
------------------------------------------------
-- DDL Statements for table "DB2INST1"."CARPETASFAVORITAS"
------------------------------------------------
CREATE TABLE "DB2INST1"."CARPETASFAVORITAS"  (
                  "ID_USUARIO" VARCHAR(50) NOT NULL ,
                  "ID_CARPETA" VARCHAR(50) NOT NULL ,
                  "NOMBRE" VARCHAR(200) )
                  IN "USERSPACE1" ;

-- DDL Statements for primary key on Table "DB2INST1"."CARPETASFAVORITAS"
ALTER TABLE "DB2INST1"."CARPETASFAVORITAS"
        ADD CONSTRAINT "CC1353965880067" PRIMARY KEY
```

```
                    ("ID_USUARIO",
                     "ID_CARPETA");

------------------------------------------------
-- DDL Statements for table "DB2INST1"." DOCUMENTOSFAVORITOS"
------------------------------------------------
CREATE TABLE "DB2INST1"."DOCUMENTOSFAVORITOS"  (
                  "ID_USUARIO" VARCHAR(50) NOT NULL ,
                  "ID_DOCUMENTO" VARCHAR(50) NOT NULL ,
                  "NOMBRE" VARCHAR(200) )
                 IN "USERSPACE1" ;

-- DDL Statements for primary key on Table
 "DB2INST1"."DOCUMENTOSFAVORITOS"
ALTER TABLE "DB2INST1"."DOCUMENTOSFAVORITOS"
       ADD CONSTRAINT "CC1353965880068" PRIMARY KEY
             ("ID_USUARIO",
              "ID_DOCUMENTO");

------------------------------------------------
-- DDL Statements for table "DB2INST1"."FORMULARIOSFAVORITOS"
------------------------------------------------
CREATE TABLE "DB2INST1"."FORMULARIOSFAVORITOS"  (
                     "ID_USUARIO" VARCHAR(50) NOT NULL ,
                     "ID_FORMULARIO" VARCHAR(100) NOT NULL ,
                     "DATOS_FORMULARIO" CLOB(1048576) LOGGED NOT COMPACT NOT NULL
)
                    IN "USERSPACE1" ;

-- DDL Statements for primary key on Table
"DB2INST1"."FORMULARIOSFAVORITOS"

ALTER TABLE "DB2INST1"."FORMULARIOSFAVORITOS"
          ADD CONSTRAINT "CC1353966056630" PRIMARY KEY
                    ("ID_USUARIO",
                     "ID_FORMULARIO");

------------------------------------------------
-- DDL Statements for table "DB2INST1"."BUSQUEDASFAVORITAS"
------------------------------------------------
CREATE TABLE "DB2INST1"."BUSQUEDASFAVORITAS"  (
                     "ID_USUARIO" VARCHAR(50) NOT NULL ,
                     "ID_BUSQUEDA" VARCHAR(100) NOT NULL ,
                     "DATOS_BUSQUEDA" CLOB(1048576) LOGGED NOT COMPACT NOT NULL )
                    IN "USERSPACE1" ;

-- DDL Statements for primary key on Table
"DB2INST1"."BUSQUEDASFAVORITAS"

ALTER TABLE "DB2INST1"."BUSQUEDASFAVORITAS"
          ADD CONSTRAINT "CC1355437032043" PRIMARY KEY
                    ("ID_USUARIO",
                     "ID_BUSQUEDA");
```

4.3.2. Configuración del servidor

El desarrollo de la aplicación y las pruebas se han realizado con un servidor de aplicaciones WebSphere Applicaciones Server 7.0.5. Dado la dificultad de conseguir una versión de este servidor, aunque exista una descargable para desarrolladores, especificaremos como hacer la instalación en un servidor WAS y en un Tomcat.

La aplicación está desarrollada para trabajar con una codificación utf-8, si el servidor está configurado para recibir otro tipo de codificación, puede producirse errores en los formularios si se usan caracteres no comunes.

Para solventar este problema, basta con añadir el parámetro:

-Dclient.encoding.override=UTF-8

Como parámetro de configuración de la máquina virtual del servidor de aplicaciones.

4.3.2.1. Instalación en WAS 7.0.0.25 (WebSphere Application Server)

Una vez tenemos el servidor de aplicaciones instalado, tenemos que configurar el origen de datos. Para ello utilizamos la consola administrativa del servidor de aplicaciones.

Figura 16: Instalación en WAS 7.0.0.25 – Orígenes de datos 1

En el menú lateral derecho seleccionamos Recursos > JDBC > Proveedores de JDBC.

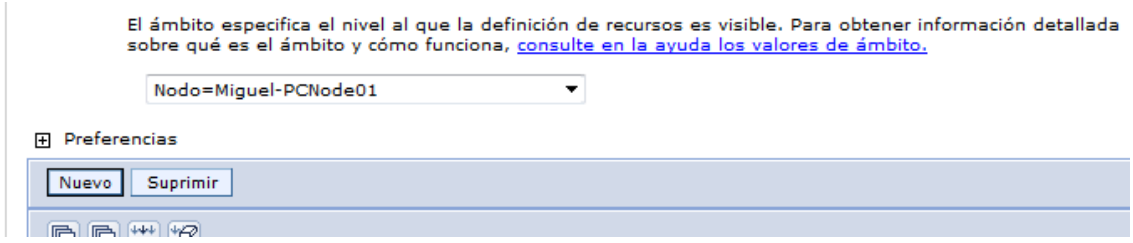


Figura 17: Instalación en WAS 7.0.0.25 – Orígenes de datos 2

En la pantalla principal seleccionamos el nodo y pulsamos el botón nuevo. No aparece un asistente, en la primera pantalla indicamos los datos del tipo de base de datos y de proveedor.

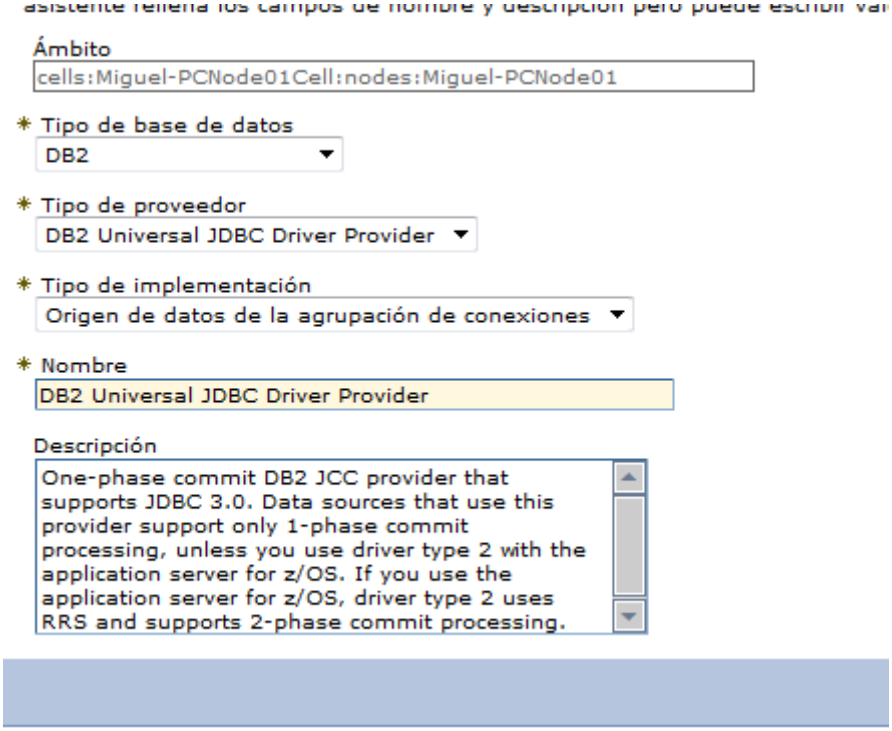
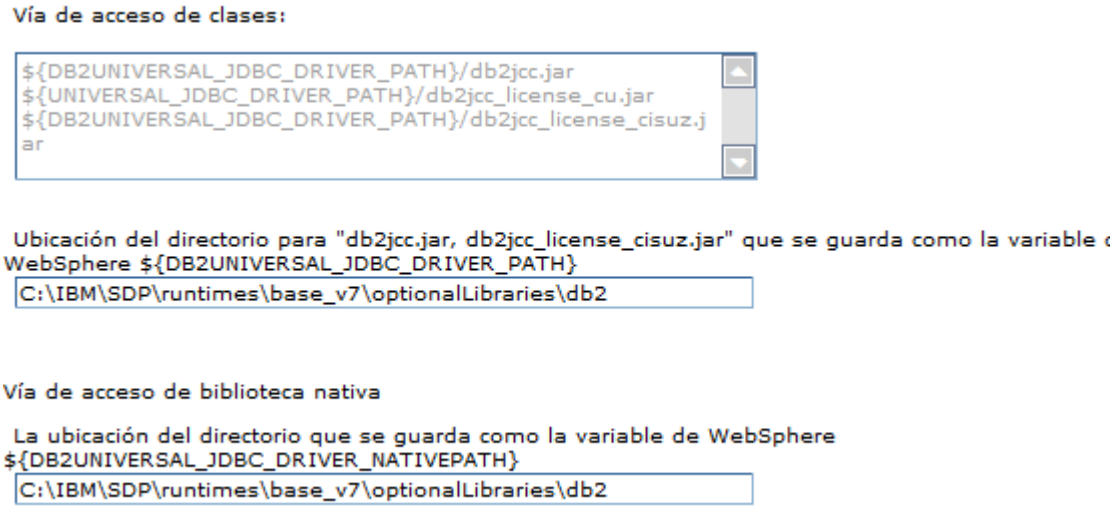
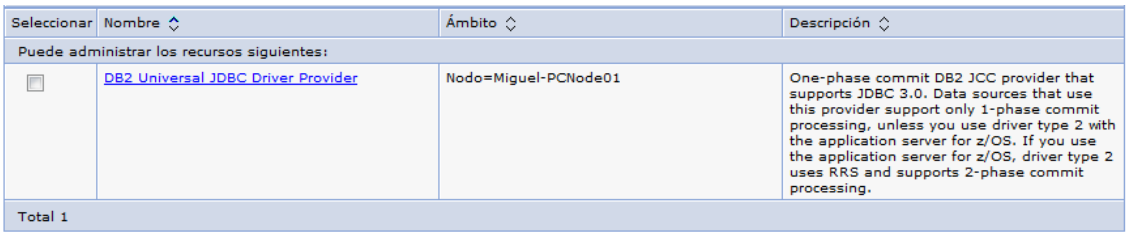


Figura 18: Instalación en WAS 7.0.0.25 – Orígenes de datos 3

En la segunda indicamos la ruta a las bibliotecas donde están los drivers JDBC para conectarnos a la base de datos.



En la tercera pantalla nos aparece un resumen de la selección realizada, y solo tenemos que pulsar finalizar y grabar la configuración. En el listado de proveedores aparecerá el que acabamos de crear.



Seleccionar	Nombre	Ámbito	Descripción
Puede administrar los recursos siguientes:			
☐	DB2 Universal JDBC Driver Provider	Nodo=Miguel-PCNode01	One-phase commit DB2 JCC provider that supports JDBC 3.0. Data sources that use this provider support only 1-phase commit processing, unless you use driver type 2 with the application server for z/OS. If you use the application server for z/OS, driver type 2 uses RRS and supports 2-phase commit processing.
Total 1			

Figura 19: Instalación en WAS 7.0.0.25 – Orígenes de datos 4

Necesitamos también crear un alias de autenticación. Para ello dentro de Seguridad global > JAAS hacemos clic en Datos de autenticación de J2C.

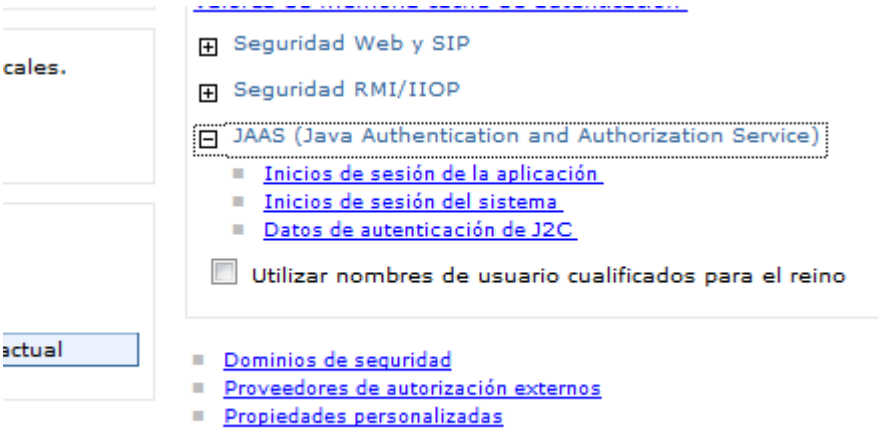


Figura 20: Instalación en WAS 7.0.0.25 – Orígenes de datos 5

En la pantalla principal hacemos clic en Nuevo y nos aparecerá un formulario donde podremos indicar un nombre identificativo y un nombre de usuario y password de la base de datos. Indicamos los datos de nuestra base de datos, pulsamos Aceptar y guardamos los cambios.

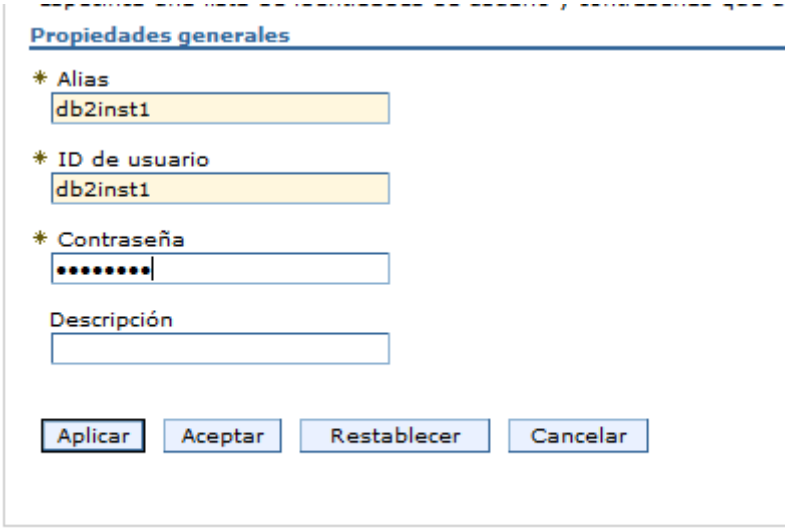


Figura 21: Instalación en WAS 7.0.0.25 – Orígenes de datos 6

Tras crear el proveedor y el alias de autenticación, pasamos a crear el origen de datos. Para ello en el menú de la derecha seleccionamos la opción Recursos > JDBC > Orígenes de datos.
En la pantalla principal el nodo y pulsamos el botón Nuevo.
Nos aparecerá otro asistente, en la primera pantalla indicamos el nombre del origen de datos y su nombre JNDI. Este nombre lo utilizaremos posteriormente para hacer el enlace.

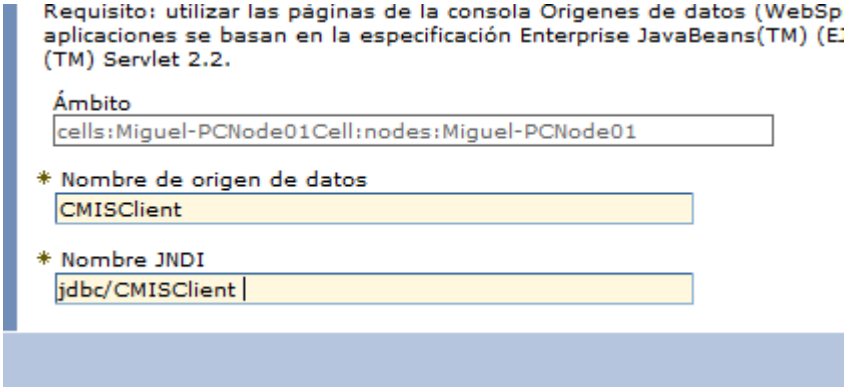


Figura 22: Instalación en WAS 7.0.0.25 – Orígenes de datos 7

Si se indica otro nombre diferente, evidentemente habrá que hacer los cambios pertinentes en las otras configuraciones relacionadas.

En la segunda pantalla seleccionamos el proveedor que habíamos creado con anterioridad:

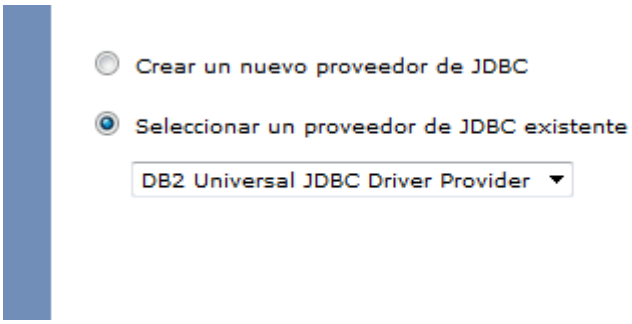


Figura 23: Instalación en WAS 7.0.0.25 – Orígenes de datos 8

En la tercera pantalla especificamos los datos de la base de datos a la que nos queremos conectar.

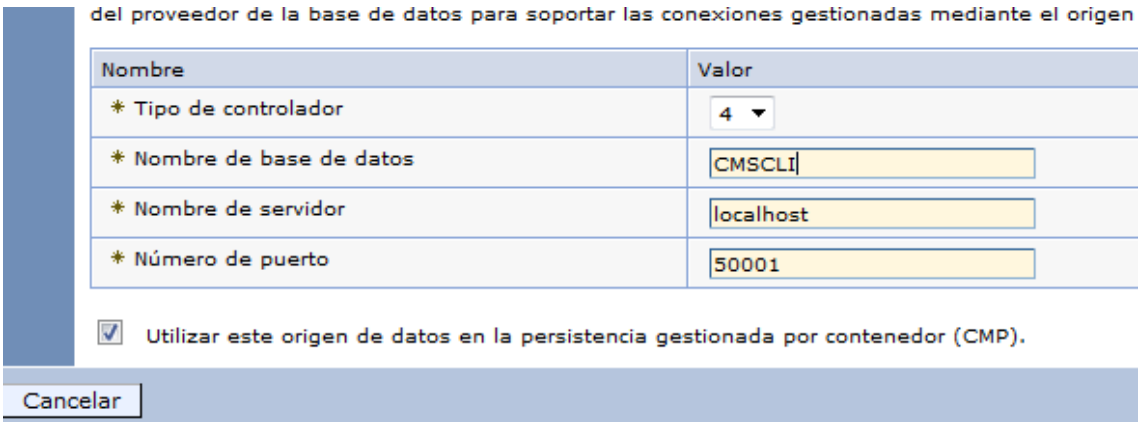


Figura 24: Instalación en WAS 7.0.0.25 – Orígenes de datos 9

En el paso cuarto seleccionamos el alias de autenticación que habíamos creado antes.

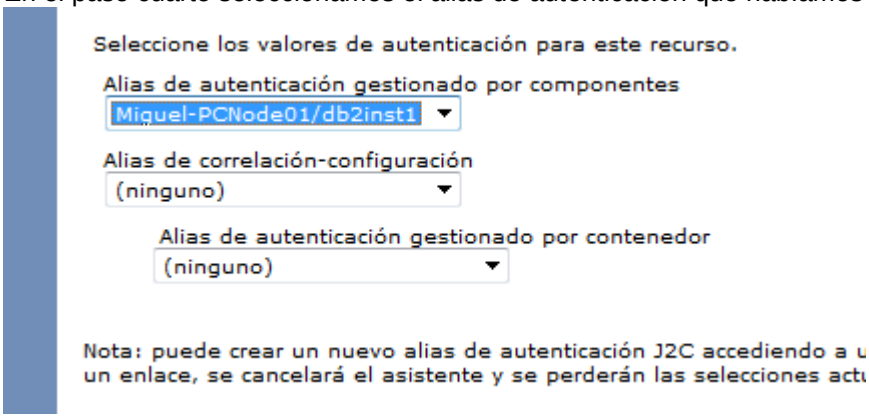


Figura 25: Instalación en WAS 7.0.0.25 – Orígenes de datos 10

En la quinta pantalla nos aparece un resumen de la selección realizada, y solo tenemos que pulsar finalizar y grabar la configuración. En el listado de orígenes de datos aparecerá el que acabamos de crear.

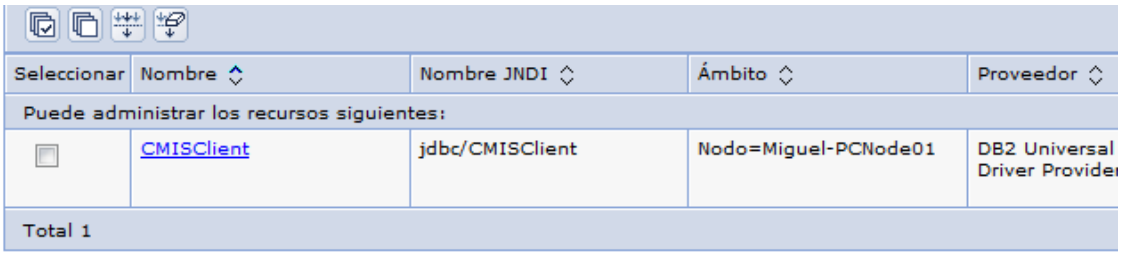


Figura 26: Instalación en WAS 7.0.0.25 – Orígenes de datos 11

Podemos hacer una conexión de prueba para comprobar que la configuración es correcta.

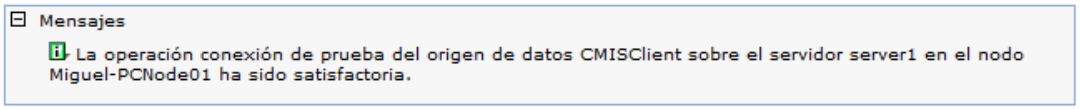


Figura 27: Instalación en WAS 7.0.0.25 – Orígenes de datos 12

Nos queda desplegar la aplicación. Para ello vamos a la opción de menú Aplicaciones > Tipos de aplicación > Aplicaciones de empresa de WebSphere y en la pantalla principal seleccionamos la opción Instalar:

Figura 28: Instalación en WAS 7.0.0.25 –Despliegue 1

Nos aparecerá un asistente, en la primera pantalla seleccionamos la ruta del archivo CMISClientEAR.ear y pulsamos siguiente.

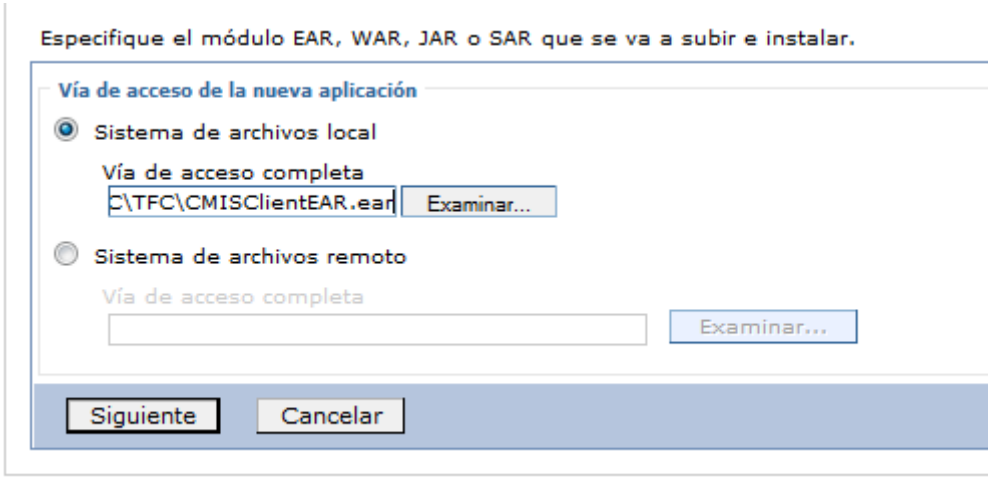


Figura 29: Instalación en WAS 7.0.0.25 –Despliegue 2

En la segunda y la tercera pantalla no cambiamos nada y pulsamos siguiente

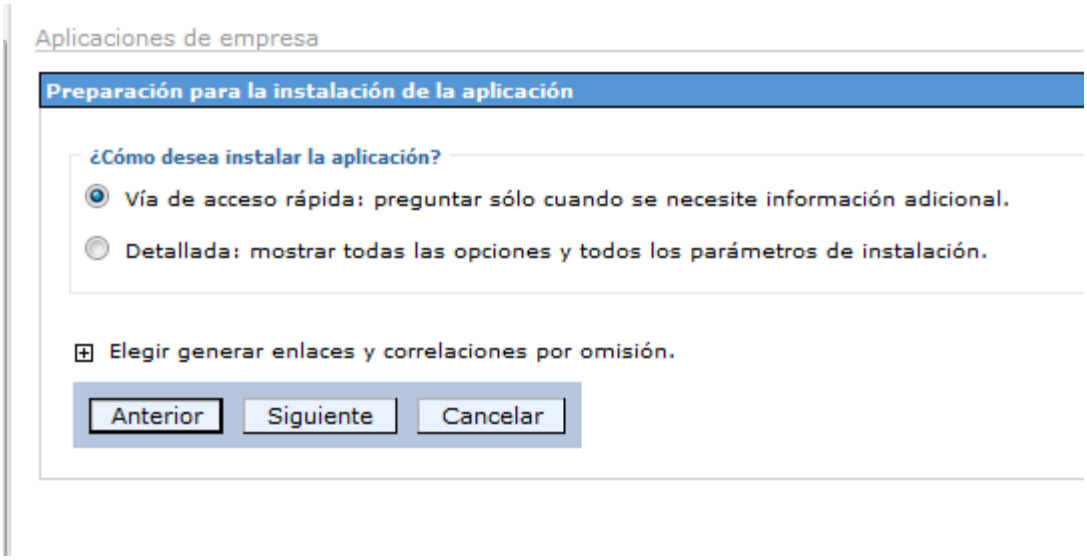


Figura 30: Instalación en WAS 7.0.0.25 –Despliegue 3

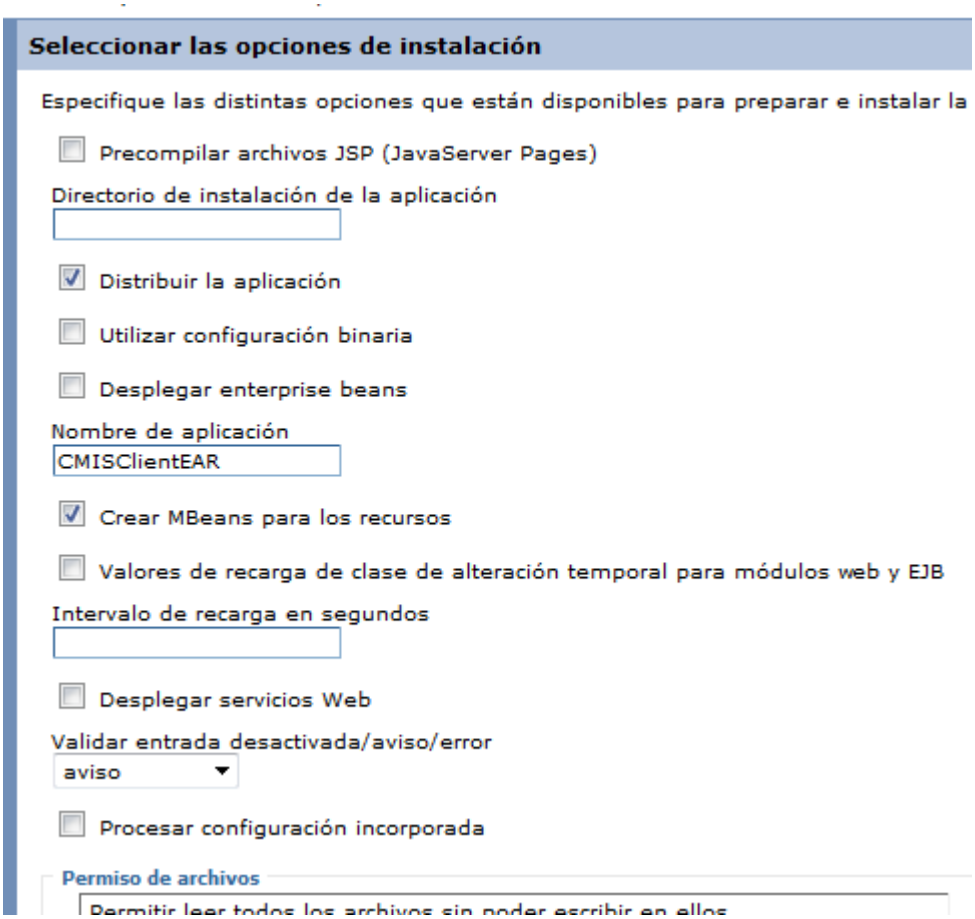


Figura 31: Instalación en WAS 7.0.0.25 –Despliegue 4

En la cuarta pantalla (paso 2) comprobamos que el modulo web tenga asignado correctamente el servidor y pulsamos siguiente.

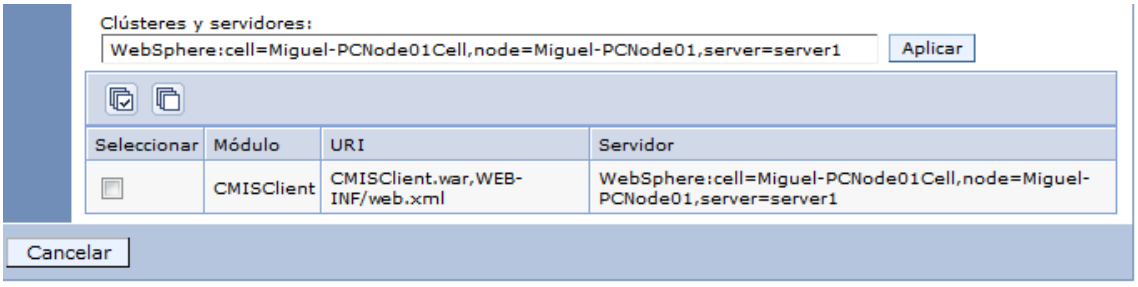


Figura 32: Instalación en WAS 7.0.0.25 –Despliegue 5

En la quinta pantalla tenemos que asociar el origen de datos y el alias de autenticación que habíamos creado a la referencia de la aplicación

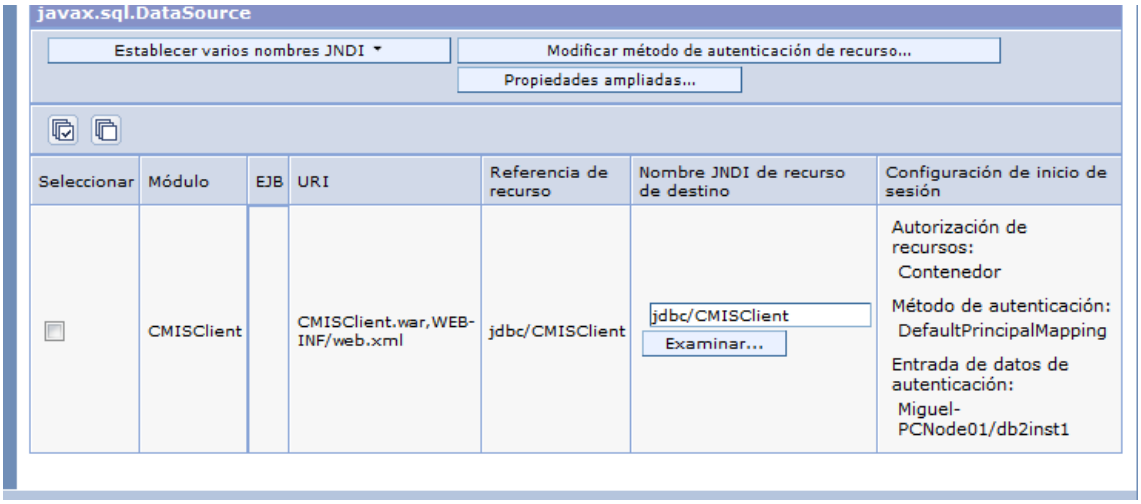


Figura 33: Instalación en WAS 7.0.0.25 –Despliegue 6

La sexta pantalla nos enseña un resumen, tenemos que pulsar finalizar y en la siguiente pantalla esperar a que se realice la instalación y pulsar Guardar.

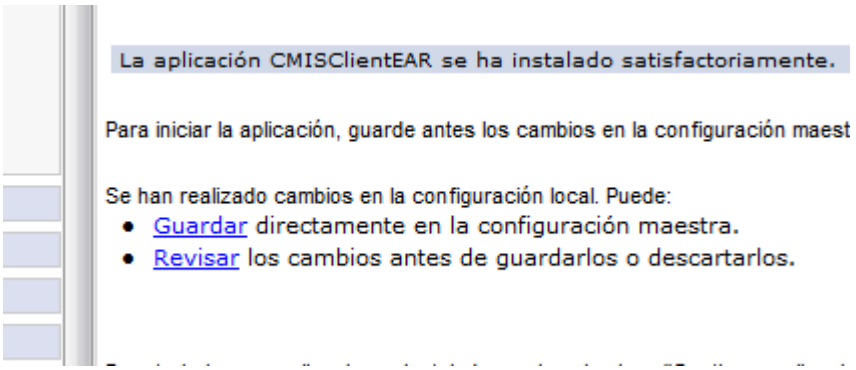


Figura 34: Instalación en WAS 7.0.0.25 –Despliegue 7

La aplicación aparecerá en el listado de aplicaciones como parada. Solo tenemos que seleccionarla y pulsar el botón Iniciar. La aplicación arrancará y mostrará el estado de iniciada.

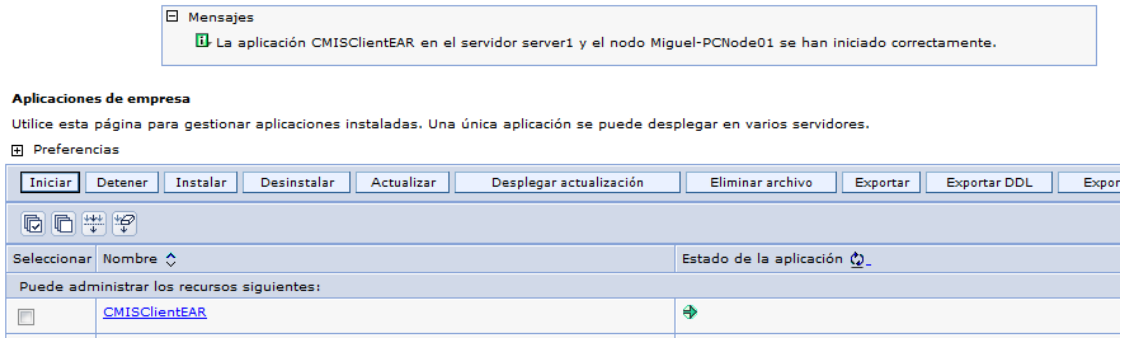


Figura 35: Instalación en WAS 7.0.0.25 –Despliegue 8

Ya podemos acceder a la aplicación mediante la url :

http://<nombre servidor>:<puerto WAS>/CMISClient

por defecto es:

http://localhost:9080/CMISClient

4.3.2.2. Instalación en Tomcat 7.0.34

Tras realizar una instalación estándar, el primer paso que tenemos que hacer es incluir unas librerías que la aplicación necesita y que el servidor Tomcat no incorpora. Estas librerías son las siguientes:

- db2jcc.jar
- db2jcc_license_cu.jar
- javax.servlet.jsp.jstl-1.2.1.jar
- javax.servlet.jsp.jstl-api-1.2.1.jar
- Las librerías las tenemos que copiar en la carpeta lib del servidor tomcat.

Tenemos que tener en cuenta que las librerías db2jcc.jar y db2jcc_license_cu.jar son las necesarias para conectar con el servidor de base de datos DB2, si se utiliza otro gestor de base de datos, tendremos que incluir las propias de este gestor y no las del DB2.

Tras añadir las librerías configuramos el origen de datos, para ellos hacemos las siguientes modificaciones en los siguientes archivos:

Añadimos el siguiente texto:

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

Dentro de la sección <Context></Context> del archivo conf/context.xml

Añadimos el siguiente texto:

```
<Resource name="jdbc/CMISClient" auth="Container"
                type="javax.sql.DataSource"
                driverClassName="com.ibm.db2.jcc.DB2Driver"
                password="******"
                username="db2inst1"
                url="jdbc:db2://localhost:50001/CMISCLI"/>
```

Dentro de la sección <GlobalNamingResources></GlobalNamingResources> del archivo conf/server.xml. Estos datos deben ajustarse la configuración del servidor de base de datos que vayamos a utilizar.

Una vez configurado el origen de datos, es posible que debamos aumentar el Heap Size del servidor Tomcat.

Queda finalmente desplegar la aplicación, para ello tenemos que copiar el fichero de la aplicación CMISClient.war en el directorio webapps. Recordar que el fichero war a diferencia del EAR está

preparado con las librerías del Apache Chemistry con su configuración estándar y no con la configuración para WebSphere.

Tras arrancar el servidor podremos acceder a la aplicación con la url:

http://<nombre servidor>:<puerto tomcat>/CMISClient

por defecto es:

http://localhost:8080/CMISClient

4.3.3. Configuración de la conexión con el gestor documental

La configuración del gestor documental se realiza a través del fichero de propiedades CMISConnection.properties que se encuentra en la raíz del directorio de clases del contenedor war , CMISClient.war\WEB-INF\classes. Este fichero permite configurar el tipo de conexión, a seleccionar entre:

- WEBSERVICES
- ATOM
- LOCAL

Cada uno de los tipos de conexiones tiene sus urls de conexión a especificar. También se configura en este fichero el ID del repositorio a conectarnos, mediante el parámetro REPOSITORY_ID.

Mas detalles sobre los parámetros de conexión pueden encontrarse en la url:

http://chemistry.apache.org/java/examples/example-create-session.html

4.3.4. Configuración de los logs

La configuración del log de la aplicación se realiza a través del fichero de configuración xml log4j.xml que se encuentra en la raíz del directorio de clases del contenedor war, CMISClient.war\WEB-INF\classes.

La configuración actual es la salida de debug por consola.

4.4. Instalación del servidor CMIS de pruebas

Si no disponemos de un gestor que proporcione servicios de CMIS podemos utilizar un servidor de desarrollo y pruebas suministrado por el propio proyecto Apache Chemistry. Este servidor se denomina OpenCMIS InMemory Repository. Este depósito no está destinado para su uso en producción.

El OpenCMIS InMemory Repository es una implementación de un repositorio CMIS que tiene el contenido y los metadatos en memoria. Por lo tanto, todos los datos almacenados en el depósito se pierden después de cada reinicio. Esta implementación tiene dos propósitos principales:
Proporcionar un motor rápido para realizar pruebas sin ninguna dependencia o configuración adicional.

Proporcionar un ejemplo de implementación de la interfaz de servidor de OpenCMIS.

Puede encontrarse más información sobre el OpenCMIS InMemory Repository en la siguiente url:

http://chemistry.apache.org/java/developing/repositories/dev-repositories-inmemory.html

La configuración de este servidor es extremadamente sencilla, basta con desplegar una aplicación war en un servidor de aplicaciones j2EE como podría ser Tomcat o WebSphere Applicationa server.

Para desplegar la aplicación en un servidor Tomcat, tenemos que copiar el fichero de la aplicación chemistry-opencmis-server-inmemory-0.8.0.war en el directorio webapps. Tras arrancar el servidor podremos acceder a la aplicación con la url:

http://<nombre servidor>:<puerto tomcat>/chemistry-opencmis-server-inmemory-0.8.0

por defecto es:

http://localhost:8080/chemistry-opencmis-server-inmemory-0.8.0/

En esta página podremos observar información acerca del servidor, como por ejemplo las urls de conexión y el nombre del repositorio. En nuestro caso hemos utilizado la conexión ATOM, que según indica esta página es:

http://localhost:8080/chemistry-opencmis-server-inmemory-0.8.0/atom

y el nombre del repositorio es A1

Por tanto modificaremos el fichero de configuración del la aplicación CMISClient para indicar los siguientes valores:

- BINDING_TYPE=ATOM
- ATOMPUB_URL=http://localhost:8080/chemistry-opencmis-server-inmemory-0.8.0/atom
- REPOSITORY_ID=A1

El servidor OpenCMIS InMemory Repository también puede desplegarse en el servidor de aplicaciones WebSphere Application Server, para ello se tiene que desplegar como una aplicación normal pero se tiene que hacer una modificación en el fichero de despliegue WAR. Las instrucciones se encuentran en el fichero chemistry-opencmis-server-inmemory-0.8.0.war\WEB-INF\websphere\how-to-setup.txt que indica lo siguiente:

Para poder utilizar la infraestructura del servidor OpenCMIS en WebSphere 7.0.0.5 o posterior, copie los archivos web.xml y webservices.xml desde / WEB-INF/websphere a / WEB-INF.

5. Valoración económica

Para la realización de una valoración económica de este proyecto podemos diferenciar dos partidas o bloque de valoración independiente.

Por un lado el coste en software o hardware de la puesta en marcha del proyecto. Si tenemos en cuenta que este proyecto está desarrollado completamente basado en estándares de desarrollo y no se ha utilizado ninguna funcionalidad propia de un servidor de aplicaciones concreto, puede ser desplegado en cualquier servidor del que disponga la empresa. En el caso de no disponer de ninguno, siempre podría utilizar un Open Source.

Lo mismo sucede con la base de datos, por lo que el coste de implantación se vería limitado a la posible inversión en hardware, si esta es necesaria.

En cuanto al coste de desarrollo y posterior de mantenimiento y evolutivos, la utilización de frameworks actuales y la arquitectura diseñada sencilla y directa, se prevé razonable.

6. Conclusiones

6.1. Conclusiones funcionales

Estoy muy satisfecho en cómo se ha desarrollado el proyecto, en lo referente a la relación entre el alcance inicial previsto y las funcionalidades finales desarrolladas, ya que la desviación ha sido muy pequeña. Sin duda esto ha beneficiado al modelo de gestión lineal de proyecto, que es más efectivo cuanto más claros son los requerimientos iniciales.

Definir todos los casos de uso y utilizarlos como guía de desarrollo me ha permitido tener claro en todo momento el grado de avance del desarrollo y lo que me quedaba por desarrollar pudiendo distribuir el tiempo de la forma lo más optima posible.

Como sucede en la mayoría de los proyectos, a lo largo del desarrollo de la aplicación y mientras iba satisfaciendo las funcionalidades definidas en la fase de análisis, han surgido nuevas necesidades, o posibilidades de mejoras.

Hemos podido hacer una selección entre aquellas que era necesario realizar para mantener la consistencia de la aplicación, aquellas que podemos realizar porque la inversión de tiempo y los resultados obtenidos lo justificaban y aquellas que son posibles mejoras y que se podrán plantear en futuras mejoras de la aplicación.

En el primer grupo de ampliaciones necesarias, situamos la pantalla de consulta de documentos reservados. Inicialmente no estaba contemplada y se preveía que el usuario buscara los documentos reservados mediante la exploración de carpetas o la herramienta de búsqueda. Sin embargo, esta nueva funcionalidad permitirá al usuario, ver de un solo vistazo, todos los documentos en los que está trabajando en ese momento, haciendo su trabajo más ágil y preciso.

En el segundo grupo de mejoras rápidas y efectivas pero no necesarias, podemos situar las mejoras visuales generadas por las tablas de datos y las ventanas modales de Mootools. Las primeras han permitido establecer un modelo de generación de listados con paginación de datos rápido y efectivo, mientras que los segundos han mejorado el aspecto visual de la aplicación.

En el último grupo situamos todas aquellas evoluciones de futuro que puede tener el producto, y que son muchas. Se merecen una sección propia y las hemos situado en el capítulo 7, al que hemos llamado evolución futura.

6.2. Conclusiones técnicas

De la misma manera que sucedía con las conclusiones funcionales, técnicamente estoy extremadamente satisfecho de cómo ha evolucionado el proyecto, ya que el diseño planteado en la fase dos se ha demostrado muy efectivo y me ha permitido un desarrollo, rápido, modular y con muy bajo nivel de errores.

Empezaré comentando el tema del CMIS. Si nos paramos a pensar en los diferentes que pueden llegar a ser los distintos gestores documentales y en todas las funcionalidades que se han descrito en el estándar CMIS, llegamos a la conclusión que el trabajo desarrollado en su elaboración ha tenido que ser enorme y el resultado obtenido ha sido excelente. Es difícil encontrar una fisura en su definición y no hay funcionalidad que no haya sido descrita y detallada.

Otra característica de esta definición sería su enfoque práctico, ya que basa su estructura en la definición de los objetos involucrados en la gestión documental y su relación entre ellos. Deja un

margen de libertad a los fabricantes con la inclusión de las extensiones, pero estas solo son necesarias para funcionalidades muy específicas que en mi caso no he tenido que utilizar en ningún momento.

Alineado a este estándar, está la implementación de Apache Chemistry de la que destacaría su claridad y simplicidad que me ha permitido su utilización sin apenas tener que dirigirme a la documentación salvo en casos muy específicos, como podría ser los formatos en las búsquedas o la posición de determinadas propiedades de los documentos o las clases documentales. Ha sido muy positivo también el descubrimiento y uso del servidor de pruebas en memoria, que ha simplificado las pruebas y ha permitido compactar más este proyecto. Tiene algunos fallos pero permite realizar casi todas las tareas básicas si problemas.

Si hablamos de la plataforma J2EE, primero querría destacar como el patrón de desarrollo MVC, ha resultado extraordinariamente efectivo en este proyecto.

Fue muy efectivo en la construcción del modelo, ya que pude inicialmente establecerlo de forma rápida y apenas tuve que corregirlo a lo largo del proyecto. Esto fue gracias principalmente a la utilización de la librería de Apache Chemistry, de la cual solo tuve que hacer alguna simplificación de objetos para tener disponible toda la comunicación con el gestor documental. También destacaría el magnífico funcionamiento del generador de código de Ibatis que me suministró no solo los beans y Daos de forma inmediata, sino que también me suministró de forma automática toda la estructura para realiza las búsquedas.

De esta manera, el acceso de base de datos solo tuve que corregirlo al modificar las tablas, pero de nuevo el generador se encargó de hacerlo. Y a los objetos generados para simplificar los retornados por Apache Chemistry solo tuve que añadirles alguna propiedad más como resultado de las necesidades del bloque de controlador o de vista.

El controlador de Spring ha sido sin duda el mayor descubrimiento de este proyecto. Si es cierto que cuesta un poco configurarlo al principio, la posterior generación inmediata de las urls conforme se van añadiendo métodos a las clases controladoras hace que la construcción de la aplicación sea extremadamente sencilla y rápida.

También tienen la ventaja de que podemos realizar las salidas a pantalla de diferente manera, de esta forma hemos podido hacer redirecciones a otras url, salidas a jsp con el ModelAndView o salidas en formato de JDOM utilizando el mismo sistema y estructura.

Me ha resultado también increíblemente práctico el tratamiento de los uploads de ficheros, que siempre había sido una complicación y que Spring trata como si fuera un parámetro más. Por último la posibilidad de controlar los parámetros de la url de forma automática y convertirlos directamente en parámetros de llamada al método evita realizar comprobaciones y protege las url de accesos incorrectos.

En lo que a la parte de la vista se refiere, me he centrado en la maquetación de divs y en la explotación del jstl y del Mootools. El tema de los divs me ha ayudado mucho para el posicionamiento de los objetos y me ha permitido poder desarrollar gran parte del proyecto sin tener que definir el diseño final. Al estar este ligado con una única css, me ha permitido poder desarrollar esa css y aplicarla cuando ya tenía todo el proyecto desarrollado, demostrando cómo es posible independizar casi completamente, diseño y código html.

El tema de Mootools era algo que me interesaba especialmente. Quería comprobar si lo que me habían comentado era cierto y la modularidad y aplicación de esta librería era tan sencilla como parecía. La verdad es que si y si bien ha sido la parte que más me ha costado aplicar, es la que visual y funcionalmente mas enriquece la interface. La existencia también de tantas librerías desarrolladas por otros programadores ayuda también a potenciar este framework.

Mención especial merece el Omnigrid. Es una librería que llevaba tiempo buscando y que hace exactamente lo que una aplicación web necesita. Combina a la perfección las funcionalidades básicas de ordenado, paginado, etc. con la flexibilidad suficiente en la carga de datos para que sea eficiente a nivel de tráfico y rendimiento. Es una herramienta a tener muy en cuenta para futuros desarrollos.

6.3. Conclusiones personales

Personalmente he intentado que este proyecto aportara nuevos conocimientos a los que ya poseía, y me ha servido para aprender mucho sobre javascript, JSON, Spring controller, anotaciones y evidentemente Apache Chemistry y CMIS. Estoy muy satisfecho con el resultado, tanto por la experiencia que he adquirido en temas de gestión documental, como por la construcción de este entorno de desarrollo de aplicaciones propio.

También he aprendido mucho con toda la teoría aplicada en el análisis y diseño de la aplicación, sobre todo la estructuración de las funcionalidades en casos de uso, y la dependencia de la arquitectura en la mejor implementación de estas necesidades.

Sin duda todo esto me será de gran ayuda en un futuro próximo, si no lo está haciendo ya.

7. Evolución futura

La herramienta que hemos desarrollado en este proyecto, ha cubierto de forma más o menos eficaz las funcionalidades básicas que un usuario pueda necesitar. Sin duda quedan pendientes muchas otras funcionalidades o mejoras de las actuales que podrían formar parte la aplicación en un futuro. Algunas de ellas serían:

- Compatibilidad con otros navegadores. Las incompatibilidades de los navegadores están por solventar. La aplicación funciona correctamente en Firefox pero presenta problemas con las ventas modales y las hojas de estilos en otros navegadores.

- Gestión del control de acceso a los contenidos. Este es un apartado que no hemos abordado en este proyecto, debido principalmente a su complejidad. En la aplicación actual, la seguridad se asigna a los documentos mediante la herencia recibida de la definición del tipo documental.

  En determinadas ocasiones, los usuarios pueden desear reajustar esos permisos para limitar los accesos a las carpetas o a los documentos, solo a las personas o grupos de ellos desearan.

  La evolución de la aplicación permitiría que todo objeto de contenido del tipo carpeta o documento, tuviera una configuración de seguridad modificable por el dueño del documento o por personas o grupos a los que se le hubiera dado estos privilegios.

  La aplicación necesitaría acceso directo a los directorios, para recuperar los usuarios y grupos directamente de ellos, ya que este servicio los gestores documentales no lo dan.

- Utilización de listas de valores en formularios. Hay gestores documentales que permiten asociar a un campo concreto de un tipo documental, una lista de valores permitidos. La aplicación debería ser capaz de recuperar esta información y mostrar al usuario una lista para que seleccionara la opción deseada sin tener que escribirla.

- Similar a la funcionalidad anterior, un posible valor añadido que me parece muy interesante sería la de poder asociar los valores posibles de un formulario, a fuentes de datos externas. Por ejemplo, imaginemos que en un tipo documental tenemos que añadir el código de un proveedor. Sería ideal que este valor, en lugar de introducirlo idealmente pudiéramos localizarlo buscando el proveedor por su nombre contra el ERP corporativo para añadirlo. Esto aportaría más robustez ante errores tipográficos y daría consistencia a la información corporativa.

- Otra funcionalidad interesante sería la de poder incorporar documentos al gestor documental, directamente desde un escáner. El funcionamiento sería similar a la creación actual de un documento seleccionando el fichero de disco, pero en este caso se llamaría a un componente que arrancaría el escaneado de un documento. Un paso más sería la integración de componentes de reconocimiento de texto que se encargarían de rellenar automáticamente los campos de metadatos.

- También me parece interesante la posibilidad de incorporar documentos por lotes. El funcionamiento sería tan simple como el que requerir al usuario la importación de un

documento de texto tabulado o un fichero xml donde se indicara la ruta de los archivos y los metadatos de los mismos. El programa necesitaría otro componente de acceso a disco para leer estos documentos y enviarlos al servidor mediante algún sistema de comunicación abierto como un web service.

- Una última propuesta de mejora, es la creación de un sistema de gestión de conexiones a los gestores documentales. Esto permitiría la configuración de varios repositorios diferentes de manera que el usuario pudiera trabajar de forma indistinta en varios de ellos, llegando incluso a poder transferir información de uno a otro con un simple copiar y pegar.

Como vemos las posibilidades y opciones de mejora son infinitas. Solo hay que tener en cuenta las necesidades de los usuarios y de las empresas conjuntamente con las posibilidades que la gestión documental y los gestores documentales nos ofrecen.

-

Glosario

Actores, En el Lenguaje Unificado de Modelado (UML), un actor "especifica un rol jugado por un usuario o cualquier otro sistema que interactúa con el sujeto."

Anotaciones (java), En programación, una Anotación Java es una forma de añadir metadatos al código fuente Java que están disponibles para la aplicación en tiempo de ejecución.

Apache Chemistry, es un grupo de proyectos de la Apache Software Foundation que provee implementaciones de código abierto de la especificación CMIS que regula la interoperabilidad de los sistemas de gestión de contenidos.

APIs, es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

ATOM, concepto que hace referencia a dos estándares relacionados:

- El Formato de Redifusión Atom es un fichero en formato XML usado para Redifusión web.
- Mientras que el Protocolo de Publicación Atom (resumido en Inglés AtomPub o APP) es un protocolo simple basado en HTTP para crear o actualizar recursos en Web.

Base de datos, conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso

Beans (java), Un Bean es un componente software que tiene la particularidad de ser reutilizable y así evitar la tediosa tarea de programar los distintos componentes uno a uno.

Check-in, un documento al que se le quita la reserva, puede ser reservado por otro usuario.

Check-out, un documento reservado (checkout), sólo podrán accederse el resto de en modo sólo lectura, evitando así, que múltiples usuarios modifiquen el documento sobrescribiendo los cambios realizados por sus propios compañeros.

CMIS, siglas de Content Management Interoperability Services, es un estándar diseñado por los principales fabricantes de gestores de contenido empresarial para la su propio ámbito.

EAR, Es un formato para empaquetar en un sólo archivo varios módulos. Permite desplegar varios de esos módulos en un servidor de aplicaciones. Contiene archivos XML llamados descriptores de despliegue, que describen cómo realizar dicha operación.

EJB, Los EJB proporcionan un modelo de componentes distribuido estándar del lado del servidor**.**

Excepciones (java), Java lanza una excepción en respuesta a una situación poco usual. El programador también puede lanzar sus propias excepciones. Las excepciones en Java son objetos de clases derivadas de la clase base Exception.

FileNet, software desarrollado actualmente por IBM para ayudar a las empresas a realizar procesos de negocio y de manejo de contenido

Filtro (J2EE), tipo especial de servlet que nos ayuda a analizar o transformar las peticiones

hechas a las páginas Web, ya sean jsps o servlets.

Framework, estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado.

Gestión documental, conjunto de normas técnicas y prácticas usadas para administrar el flujo de documentos de todo tipo en una organización.

HTML, siglas de HyperText Markup Language («lenguaje de marcado de hipertexto»), hace referencia al lenguaje de marcado predominante para la elaboración de páginas web que se utiliza para describir y traducir la estructura y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes.

J2EE, Java Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4; traducido informalmente como Java Empresarial), es una plataforma de programación—parte de la Plataforma Java—para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java

JDBC, Java Database Connectivity, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

JNDI, Interfaz de Nombrado y Directorio Java (Java Naming and Directory Interface) es una Interfaz de Programación de Aplicaciones (API) de Java para servicios de directorio. Permite a los clientes descubrir y buscar objetos y datos a través de un nombre.

JSON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

JSP, JavaServer Pages (JSP) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

JSTL, la tecnología JavaServer Pages Standard Tag Library (JSTL) es un componente de Java EE. Extiende las ya conocidas JavaServer Pages (JSP) proporcionando cuatro bibliotecas de etiquetas (Tag Libraries) con utilidades ampliamente utilizadas en el desarrollo de páginas web dinámicas.

Log4j, biblioteca open source desarrollada en Java por la Apache Software Foundation que permite a los desarrolladores de software elegir la salida y el nivel de granularidad de los mensajes o “logs” (data logging) a tiempo de ejecución y no a tiempo de compilación.

Mootols, MooTools (My object oriented tools) es un Framework web orientado a objetos para JavaScript, de código abierto, compacto y modular.

MVC, es un patrón o modelo de abstracción de desarrollo de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos.

MyBatis, es una herramienta de persistencia disponible para Java y .NET que se encarga de mapear sentencias SQL y procedimientos almacenados con objetos a partir de ficheros XML o anotaciones.

Origenes de datos, representa todo lo relativo a una fuente de datos configurada por el usuario

para conectarse a una Base de datos.

Portlet, componentes modulares de las interfaces de usuario gestionadas y visualizadas en un portal web.

Scriptlet, un scriplet en JSP es un fragmento de código Java escrito entre los símbolos <% y %>.

Servlet, componentes de la parte del servidor de Java EE, encargados de generar respuestas a las peticiones recibidas de los clientes.

Spring, el Spring Framework (también conocido simplemente como Spring) es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java

SQL, el lenguaje de consulta estructurado o SQL (por sus siglas en inglés structured query language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas

Tomcat, es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS.

Versión, una versión, revisión o edición de un documento, es el estado en el que se encuentra dicho producto en un momento dado de su desarrollo o modificación.

WAR, es un archivo JAR (con la extensión WAR) usado para **distribuir** una **colección** de **archivos** JSP, servlets, clases Java, archivos XML y contenido web estático (HTML). En conjunto constituyen una aplicación Web.

Web 2.0, comprende aquellos sitios web que facilitan el compartir información, la interoperabilidad, el diseño centrado en el usuario1 y la colaboración en la World Wide Web.

Web services, tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

WebSphere Application Server, servidor de aplicaciones de software, es el producto estrella dentro de la familia WebSphere de IBM. WAS está construido usando estándares abiertos tales como J2EE, XML, y Servicios Web.

Bibliografia

- Content Management Interoperability Services (CMIS) Version 1.0
  http://docs.oasis-open.org/cmis/CMIS/v1.0/os/cmis-spec-v1.0.html

- OpenCMIS Client API Developer's Guide
  http://chemistry.apache.org/java/developing/guide.html

- Apache Chemistry OpenCMIS 0.8.0 Javadoc
  http://chemistry.apache.org/java/0.8.0/maven/apidocs/

- OpenCMIS InMemory Repository
  http://chemistry.apache.org/java/developing/repositories/dev-repositories-inmemory.html

- MyBatis
  http://www.mybatis.org/core/es/index.html

- Introduction to MyBatis Generator
  http://www.mybatis.org/generator/index.html

- JacksonInFiveMinutes
  http://wiki.fasterxml.com/JacksonInFiveMinutes

- Spring - Web MVC framework
  http://static.springsource.org/spring/docs/3.2.x/spring-framework-reference/html/mvc.html

- JSP Standard Tag Library (JSTL) Tutorial
  http://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm

- Aprende Posicionamiento con CSS en 10 Pasos
  http://www.trisfera.com/ejemplosRetos/position_float/

- Regular Expression Tutorial
  http://www.regular-expressions.info/tutorial.html

- MooTools a JavaScript framework
  http://mootools.net/

- OmniGrid - Advanced DataGrid for Mootools
  http://www.omnisdata.com/omnigrid/

- Arian Mootools Datepicker
  https://github.com/arian/mootools-datepicker/blob/master/README.md

- mBox 0.2
  http://htmltweaks.com/mBox

- Short introduction to log4j
  http://logging.apache.org/log4j/1.2/manual.html

- Wikipedia
  http://es.wikipedia.org/wiki/Wikipedia:Portada