UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Proyecto Final de Carrera

Archivador Digital de Expedientes

Código:

DSIC-144

Autor:

Pablo Guardiola Sánchez

Director del Proyecto:

Pedro José Valderas Aranda

Índice

Listado de Figuras

Listado de Ejemplos

1. Introducción

1.1. Ámbito

El trabajo realizado en este proyecto forma parte de **e-Dibam**, la primera plataforma de e-gobierno desarrollada por eyeOS para la Diputació de Barcelona. Esta plataforma se utilizará en más de 170 ayuntamientos de Cataluña y permitirá a los usuarios trabajar en la nube con sistemas de gestión de contenidos, gestión documental, procedimientos y servicios web integrados. Dentro de e-Dibam se implantarán distintas funcionalidades, entre ellas, **Arxivador Digital de Documents i Expedients**.

Arxivador Digital de Documents i Expedients, en adelante ADDE, es un repositorio de documentación de carácter definitivo que se genera en la fase de tramitación de peticiones realizadas por la ciudadanía al ayuntamiento. Esta herramienta desarrollada por ALTEN SPAIN para la Diputació de Barcelona, permite guardar la documentación en formato electrónico, garantizando su localización, consulta y autenticidad.

1.2. Motivación

En el actual entorno macroeconómico, las tecnologías de la información y la comunicación (TIC) constituyen un elemento clave en el desarrollo social y en la mejora de la competitividad. En la Diputació de Barcelona quieren alcanzar el reto de potenciar todas las posibilidades de uso de la tecnología disponible en favor de los municipios y de las personas que allí viven, quieren que las TIC se conviertan en una palanca de progreso, puesto que piensan que su uso estratégico e inteligente es la herramienta clave del futuro.

Bajo estas circunstancias surge e-Dibam con el lema “Un paso decidido hacia el gobierno electrónico”. e-Dibam es un conjunto de actuaciones de ámbito tecnológico, organizativo, documental y jurídico.

ALTEN es un grupo multinacional francés, líder en Europa en Consultoría, IT e Ingeniería desde el año 1988. Dedicada a ofrecer servicios globales de Tecnologías de la Información, mayoritariamente para el sector público. En España, tiene sedes en Madrid, Cataluña y Castilla y León. En este punto, y dado el alto número de proyectos

realizados en este ámbito, la Diputació de Barcelona, encarga a ALTEN SPAIN la funcionalidad de Gestión Documental de e-Dibam y, para ello, desarrolla ADDE, aplicación en la que se basa este proyecto.

1.3. Objetivo

El ámbito de la Societat del Coneixement de la Diputació de Barcelona tiene como objetivo principal conseguir que, mediante diferentes iniciativas de modernización en las tecnologías de la información y la comunicación y de impulso de la e-administración, los 311 ayuntamientos de la provincia avancen hacia el gobierno electrónico para que puedan prestar servicios públicos más próximos y efectivos a sus ciudadanos y ciudadanas.

Esta iniciativa contribuirá a que los ayuntamientos sean referentes en sistemas de información y de atención ciudadana ya que, mediante el acceso electrónico de la ciudadanía a los servicios públicos, se fomenta el uso y la mejora de las nuevas tecnologías.

e-Dibam incluye un conjunto de actuaciones dirigidas a desarrollar la administración electrónica más avanzada en los ayuntamientos de Barcelona. El proyecto tiene una primera fase donde se desarrolla una prueba piloto con 19 ayuntamientos, a los que ofrece una tecnología innovadora y avanzada de software libre llamada *private cloud computing**. Posteriormente y con la experiencia de este piloto, se elaborará el plan director de asistencia en e-gobierno dirigido a todos los municipios de nuestra provincia.

En cuanto a ADDE, tiene como objetivos principales, asegurar el acceso a los documentos a lo largo de su vida con garantía legal y de autenticidad y garantizar la preservación de los documentos y de las evidencias electrónicas con carácter histórico.

El proyecto propuesto tiene como objetivo conocer las distintas tecnologías que se utilizan en un proyecto real y plasmarlas en el desarrollo de una aplicación web para la gestión de expedientes, totalmente funcional y englobada en las tecnologías de la información y la comunicación.

1.4. Descripción de alto nivel

La aplicación ADDE no es un gestor de expedientes pero es más que un gestor documental genérico. A nivel funcional será un híbrido entre gestor documental y gestor de expedientes. Se encargará de almacenar toda la documentación que se genera alrededor de los trámites que forman parte del catálogo de trámites y asegurar su integridad, seguridad y completitud de cara a preparar la documentación para ponerla a disposición del archivo, teniendo la misma validez y eficacia que los originales en papel.

ADDE estará sobre una implantación de Alfresco diferente a la del registro de entrada y salida.

La aplicación dispondrá de cinco funcionalidades principales:

- Pantalla de bienvenida: incluye la pantalla de bienvenida.
- Pantalla resumen documentos pendientes: asentamientos y/o documentos escaneados pendientes se obtendrán a partir de la aplicación de registro ERES y del escáner.
- Pantalla resumen expedientes: se proporcionará una herramienta para la gestión de expedientes durante los principales estados del ciclo de vida hasta la disposición de los mismos.
- Pantalla de búsqueda general: se definirá una pantalla de búsqueda general con tal de localizar cualquier tipo de documento, trámite, expediente o documento pendiente.
- Pantalla de administración: se mostrará el detalle de las acciones de administración.

Dada la envergadura del proyecto, cada uno de los miembros del grupo de trabajo ha desarrollado módulos distintos. Por esta razón, el proyecto que se va a presentar es tan sólo una parte de ADDE, es decir, los módulos en los que he participado en el desarrollo. De manera que al ser una aplicación propietaria no haya problemas futuros de aspecto legal, puesto que no se corresponde con la aplicación final que se entregará al cliente.

Se trata de una aplicación J2EE, basada en Eclipse y tecnologías OpenSource. La aplicación utiliza Struts, el cual se basa en el Framework del Modelo-Vista-Controlador (MVC), lo que permite tener el código ordenado. Además Struts incluye un mecanismo llamado Tiles que permite reutilizar plantillas. También hace uso de Spring, Framework que inyecta las dependencias necesarias, permitiendo el desacoplamiento de objetos, es decir, que unos objetos no dependan de otros. A partir de aquí, Apache Maven es la herramienta que se utiliza para la gestión y la construcción del proyecto. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado. Como servidor de aplicaciones se usa Apache Tomcat. Para generar el contenido dinámico se hace uso de JavaServer Pages (JSP), en forma de documentos HTML. En ellas se utiliza la tecnología JSTL, que proporciona cuatro bibliotecas de etiquetas (Tag Libraries). Las JSP's permiten la utilización de código Java mediante scripts, de modo que también se usa JavaScript y jQuery, junto con AJAX y JSON para la resolución de los diálogos modales. Todas estas páginas definen su presentación mediante el lenguaje CSS. Por otra parte, como sistema de gestión de base de datos se emplea MySQL y como gestor documental, Alfresco, donde se guarda la estructura de carpetas y los ficheros físicos.

La solución aportada implementa las siguientes pantallas:

1. Pantalla de bienvenida: se accede tras pasar la autenticación, y se compone de las pestañas de expedientes y administración.

2. Pantalla de expedientes: se muestra el detalle de los expedientes para un trámite específico. Además permite crear expedientes nuevos, modificar expedientes y eliminar expedientes. Al hacer click sobre un expediente se muestra la pantalla de detalle de archivos, la cual, permite añadir y eliminar ficheros asociados a un expediente concreto.

3. Pantalla de administración: está formada por las pestañas gestión de usuarios y gestión de trámites.

1.5. Estructura del documento

En el siguiente apartado repasaremos el marco conceptual que rodea nuestro proyecto, examinando la actualidad en lo que concierne a los gestores de expedientes, gestores documentales, aplicaciones web (J2EE) y la e-Administración.

En la sección 3 analizaremos las diferentes tecnologías utilizadas en este proyecto, mencionando qué son y cómo las hemos utilizado, haciendo una breve descripción de su funcionamiento. También se introducirán las principales características de los distintos lenguajes de programación utilizados.

En el apartado 4 se expondrá el análisis del proyecto. Es aquí donde se describe el qué de forma detallada, esto es, paso a paso y sin hacer referencia concreta a la implementación ni a tecnologías específicas.

En el punto 5 nos centraremos en explicar en detalle cómo funciona la aplicación, es decir, se describirán los módulos, las clases principales y la metodología de desarrollo empleada.

En el apartado 6 observaremos un ejemplo de ejecución del proceso completo, desde los datos de entrada hasta los resultados obtenidos.

Finalmente encontramos tres secciones que contienen los usos futuros y posibles modificaciones, las conclusiones obtenidas y las referencias que han servido de gran apoyo para la elaboración de este proyecto.

2. Marco conceptual

2.1. Gestión de Expedientes

La gestión de expedientes es un acercamiento único a la gestión de las relaciones entre los documentos, los expedientes, las personas y los procesos. El valor real de la gestión colaborativa de expedientes reside en la simplificación, mejora y automatización de flujos de información complejos; en la disponibilidad de toda la información en una vista unificada.

Los principales problemas que encontramos actualmente en la gestión de expedientes son los relacionados con la existencia de archivos desconectados, inconsistentes o duplicados, con múltiples repositorios separados, sin control en los plazos de caducidad, con escasa confidencialidad y seguridad de los datos, con difícil acceso a los contenidos de los expedientes y con poco control y monitorización gráfica y estadística de la información por parte de la organización.

Toda organización que disponga de información de clientes, productos, sistemas de facturación, gestión de incidencias, departamento de atención al cliente, que deba implantar procesos de gestión de calidad, que ofrezca servicios bancarios (apertura de cuentas, créditos, hipotecas) o servicios relacionados con los seguros, necesita gestionar sus expedientes de manera eficiente, segura y eficaz.

En el caso concreto de la Administración Pública, nos encontramos con la nueva Ley de Administración Electrónica que requiere de soluciones que coordinen una gestión integrada de trámites, expedientes y procesos administrativos, junto con contenidos y documentos, todo ello a través de portales, en un entorno seguro con certificación digital y que se integre con los sistemas actuales.

Tiene que permitir la creación, tramitación, seguimiento, control y monitorización de cualquier tipo de expediente además de búsquedas avanzadas de expedientes y datos, control de plazos y alarmas, elaboración de listados.

Esto incluye una bandeja de entrada de expedientes (con posibilidad de asignar expedientes a otros usuarios), con registro histórico de acciones, lista de tareas y apertura, modificación o cierre de casos: expedientes, proyectos o protocolos.

La "carpeta compartida" o "Electronic case folder" actúa muchas veces como un repositorio para todos los procesos, tareas, datos y documentos del expediente que además es compartido por múltiples usuarios.

El valor añadido y la flexibilidad de un buen sistema de gestión de expedientes, radica en que los usuarios puedan completar tareas, añadir nuevas, generar nuevos procesos o incluir nuevos datos en cualquier momento del proceso.

En cuanto a gestión documental, debe permitir la creación de formularios online, integración con firma electrónica, generación automática y manual de documentos (diversos formatos) y plantillas, desde un repositorio único integrado y debe permitir además vistas en tiempo real de los datos asociados al expediente, a los archivos y los documentos.

2.2. Gestión documental

Se entiende por Gestión Documental el conjunto de normas técnicas y prácticas usadas para administrar el flujo de documentos de todo tipo en una organización, permitir la recuperación de información desde ellos, determinar el tiempo que los documentos deben guardarse, eliminar los que ya no sirven y asegurar la conservación indefinida de los documentos más valiosos, aplicando principios de racionalización y economía.

- Software de gestión documental

Sistema de gestión documental (en inglés, Document Management System) son todos aquellos programas de ordenador creados para la gestión de grandes cantidades de documentos, suele rastrear y almacenar documentos electrónicos o imágenes de documentos en papel. Estos documentos no tienen una organización clara de sus contenidos, al contrario de lo que suele suceder con la información almacenada en una base de datos. La combinación de este tipo de bibliotecas de documentos con índices almacenados en una base de datos permite el acceso rápido mediante diversos métodos a la información contenida en los documentos. Estos generalmente se encuentran comprimidos y además de texto pueden contener cualquier otro tipo de documentos multimedia como imágenes o vídeos.

Los sistemas de gestión de documentos comúnmente proporcionan medios de almacenamiento, seguridad, así como capacidades de recuperación e indexación. El término tiene algún traslapo con los conceptos de Content Management Systems y a menudo es visto como un componente de Sistemas de Gestión de Contenido de Empresa y relacionado con la Gestión de Activo Digital.

2.3. Aplicación web

En la ingeniería de software se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.

Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, a la independencia del sistema operativo, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales. Existen aplicaciones como los webmails, wikis, weblogs, tiendas en línea y la propia Wikipedia que son ejemplos bien conocidos de aplicaciones web.

Es importante mencionar que una página Web puede contener elementos que permiten una comunicación activa entre el usuario y la información. Esto permite que el usuario acceda a los datos de modo interactivo, gracias a que la página responderá a cada una de sus acciones, como por ejemplo rellenar y enviar formularios, participar en juegos diversos y acceder a gestores de base de datos de todo tipo.

Aunque existen muchas variaciones posibles, una aplicación web está normalmente estructurada como una aplicación de tres-capas. En su forma más común, el navegador web ofrece la primera capa y un motor capaz de usar alguna tecnología web dinámica (ejemplo: PHP, Java Servlets o ASP, ASP.NET, CGI, ColdFusion, embPerl, Python (programming language) o Ruby on Rails) constituye la capa de enmedio. Por último, una base de datos constituye la tercera y última capa.

El navegador web manda peticiones a la capa de enmedio que ofrece servicios valiéndose de consultas y actualizaciones a la base de datos y a su vez proporciona una interfaz de usuario.

2.4. e-Administración

La e-Administración o Administración electrónica hace referencia a la incorporación de la tecnologías de la información y las comunicaciones en dos vertientes: desde un punto de vista intraorganizativo transformar las oficinas tradicionales, convirtiendo los procesos en papel, en procesos electrónicos, con el fin de crear una oficina sin papeles y desde una perspectiva de la relaciones externas habilitar la vía electrónica como un nuevo medio para la relación con el ciudadano y empresas. Es una herramienta con un elevado potencial de mejora de la productividad y simplificación de los diferentes procesos del día a día que se dan en las diferentes organizaciones.

La definición de la Comisión Europea de la Unión Europea es la siguiente: “La Administración electrónica es el uso de las TIC en las AAPP, combinado con cambios organizativos y nuevas aptitudes, con el fin de mejorar los servicios públicos y los procesos democráticos y reforzar el apoyo a las políticas públicas”

La e-Administración alcanza a las comunicaciones internas de una oficina como las comunicaciones entre oficinas de diferentes organizaciones.

Uno de los objetivos es la introducción de transparencia y responsabilidad para alcanzar un mejor e-Gobierno dentro de las organizaciones.

Para la puesta en marcha de la e-Administración, es necesario cambiar la mentalidad tradicional de que la organización es el centro de atención, pasando a ser el cliente el centro de todas las actividades de la organización. Se debe introducir sistemas transparentes de trabajo, eliminando la dependencia específica de personas para realizar las diferentes tareas.

La e-Administración se ha visto impulsada por la aparición de las tecnologías de la información y comunicación (teléfono, Internet,...), facilitando a los clientes la interacción con las organizaciones y a los trabajadores flexibilizando las condiciones de trabajo (flexibilidad horaria, teletrabajo, movilidad...) y mejorando dichas condiciones.

Las ventajas que tiene la e-Administración para los clientes de las organizaciones son:

1) Disponibilidad: Se puede interactuar con las organizaciones las 24 horas del día (por teléfono con servicios de atención telefónica o por Internet a través de oficinas virtuales). No es necesario cernirse a un horario de oficinas. Ya no existen los días festivos.

2) Facilidad de acceso: Ya no es necesario acudir a la oficina presencial de la organización para realizar las gestiones, se puede hacer desde cualquier parte del mundo a través del teléfono o Internet. Las oficinas están disponibles para los usuarios en cualquier lugar.

3) Ahorro de tiempo: Para realizar una gestión se puede realizar desde casa o cualquier lugar que deseemos, sin la necesidad de tener que desplazarse a la oficina presencial, esperar una cola para ser atendido, la atención (explicación de lo que se desea realizar y realización de dicha actividad), y el regreso a casa.

En España la Agencia Tributaria fue pionera en la puesta en marcha de la e-Administración. Actualmente diferentes administraciones públicas están desarrollando la e-Administración dentro de programas de mejora y prácticamente todos los organismos disponen de oficinas virtuales en las que ofrecen información y trámites por vía electrónica.

- Sede electrónica

Las sedes electrónicas, que sustituyen a las actuales oficinas virtuales, son un punto de acceso electrónico a aquellos servicios que requieran la autenticación de los ciudadanos o de la administración, dotado de especiales condiciones de identificación, seguridad y responsabilidad que garantizan una información veraz, actualizada y completa. Mediante el dominio específico reservado a las mismas (.gob.es) y el certificado de sede queda asegurada su identificación, de manera que el ciudadano tiene la certeza de que se encuentra en un sitio de la administración y de que nadie ha realizado una suplantación del mismo, así como que las conexiones que se establezcan en las sedes electrónicas son seguras para salvaguardar la necesaria confidencialidad en los intercambios de datos que se realicen.

3. Contexto tecnológico

Este proyecto integra diferentes tecnologías, cada una de ellas utilizada para una labor específica. En este apartado las analizaremos una a una y expondremos algún breve ejemplo. Además se enunciarán los distintos lenguajes de programación empleados y sus principales características.

3.1. Eclipse

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido". Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE).

La base para Eclipse es la Plataforma de cliente enriquecido (del Inglés Rich Client Platform RCP). Los siguientes componentes constituyen la plataforma de cliente enriquecido:

- Plataforma principal - inicio de Eclipse, ejecución de plugins
- OSGi - una plataforma para bundling estándar.
- El Standard Widget Toolkit (SWT) - Un widget toolkit portable.
- JFace - manejo de archivos, manejo de texto, editores de texto
- El Workbench de Eclipse - vistas, editores, perspectivas, asistentes

El entorno de desarrollo integrado (IDE) de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido.

En cuanto a las aplicaciones clientes, Eclipse provee al programador con frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, entre otros.

Eclipse ha sido el IDE utilizado para el desarrollo de la aplicación, por su alta compatibilidad con el lenguaje de programación Java y por su facilidad de integración de frameworks.

3.2. J2EE

Java Platform, Enterprise Edition o Java EE (anteriormente conocido como Java 2 Platform, Enterprise Edition o J2EE hasta la versión 1.4), es una plataforma de programación—parte de la Plataforma Java—para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java con arquitectura de N capas distribuidas y que se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process, Java EE es también considerada informalmente como un estándar debido a que los proveedores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por The Java Community Process / JCP.

Java EE incluye varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, servlets, portlets (siguiendo la especificación de Portlets Java), JavaServer Pages y varias tecnologías de servicios web. Ello permite al desarrollador crear una Aplicación de Empresa portable entre plataformas y escalable, a la vez que integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel.

3.3. Struts

Struts es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma Java EE (Java Enterprise Edition). Struts permite reducir el tiempo de desarrollo. Su carácter de "software libre" y su compatibilidad con

todas las plataformas en las que Java Entreprise esté disponible lo convierten en una herramienta altamente disponible.

Struts se basa en el patrón de arquitectura de software Modelo-Vista-Controlador (MVC) el cual se utiliza ampliamente y es considerado de gran solidez. De acuerdo con este Framework, el procesamiento se separa en tres secciones diferenciadas llamadas el modelo, las vistas y el controlador.

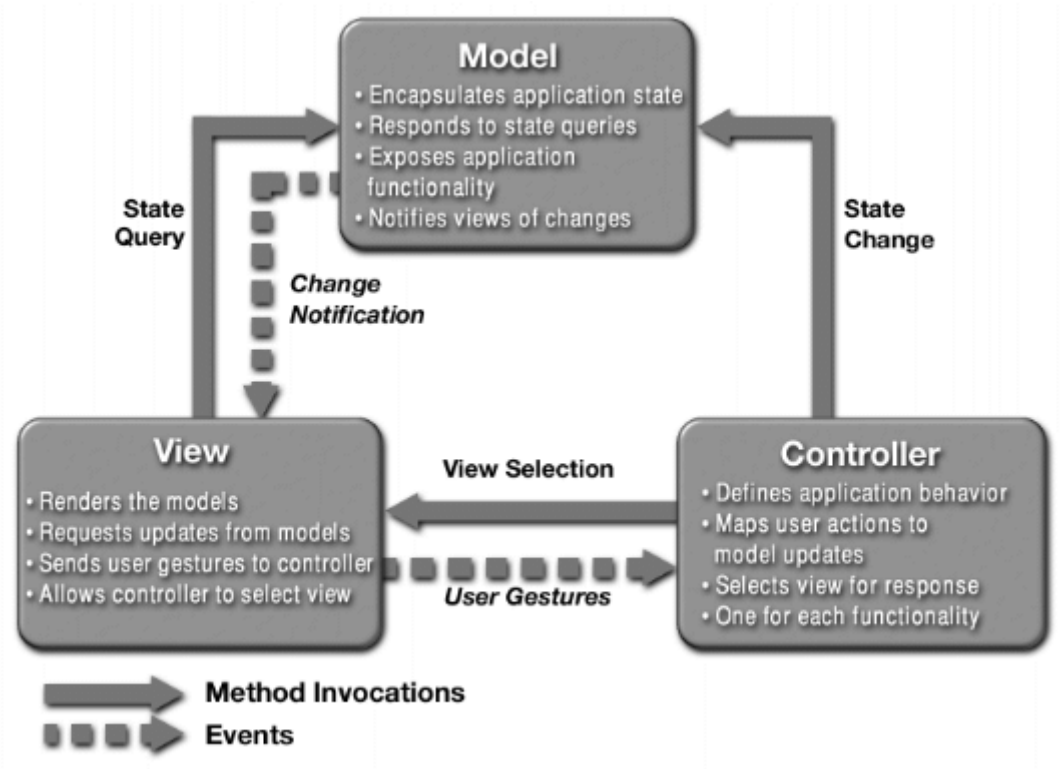


Figura 1. Modelo Vista Controlador

En nuestro trabajo, se ha utilizado porque simplifica notablemente la implementación de una arquitectura según el patrón MVC. Struts separa muy bien lo que es la gestión del workflow de la aplicación, del modelo de objetos de negocio y de la generación de interfaz.

3.4. Spring

Spring Framework es una plataforma que nos proporciona una infraestuctura que actúa de soporte para desarrollar aplicaciones Java. Spring maneja toda la

infraestructura y así te puedes centrar en tu aplicación. Coloquialmente, Spring es el “pegamento” que une todos los componentes de la aplicación, maneja su ciclo de vida y la interacción entre ellos.

Spring Framework es un contenedor ligero (“lightweight container”) en contraposición a un servidor de aplicaciones J2EE. En el caso de una aplicación web, te basta con un contenedor de servlets como Tomcat o Jetty.

Las características más destacables de Spring son la inversión de control y la inyeccción de dependencias. Abreviado del inglés IoC y DI respectivamente. Cuando diseñas una aplicación en Java dispones de muchos objetos que se relacionan entre sí mediante composición. Para enlazar dos objetos tendrías que inyectarle a uno de ellos una instancia del otro. Esto lo realiza Spring, se llama Inversión de control, porque es Spring quien se encarga de estas dependencias, instancia los objectos y los inyecta por reflexión. A grandes rasgos, declaras en un XML los componentes de tu aplicación y sus dependencias. Spring lee este XML, llamado Application Context, crea los componentes y sus relaciones entre ellos.

Spring es bastante grande, por ello el proyecto esta dividido en módulos. No siempre se utiliza en un proyecto todo lo que tiene Spring. Por poner un ejemplo, podrías utilizar Struts para la parte web, módulo utilizado en nuestro proyecto.

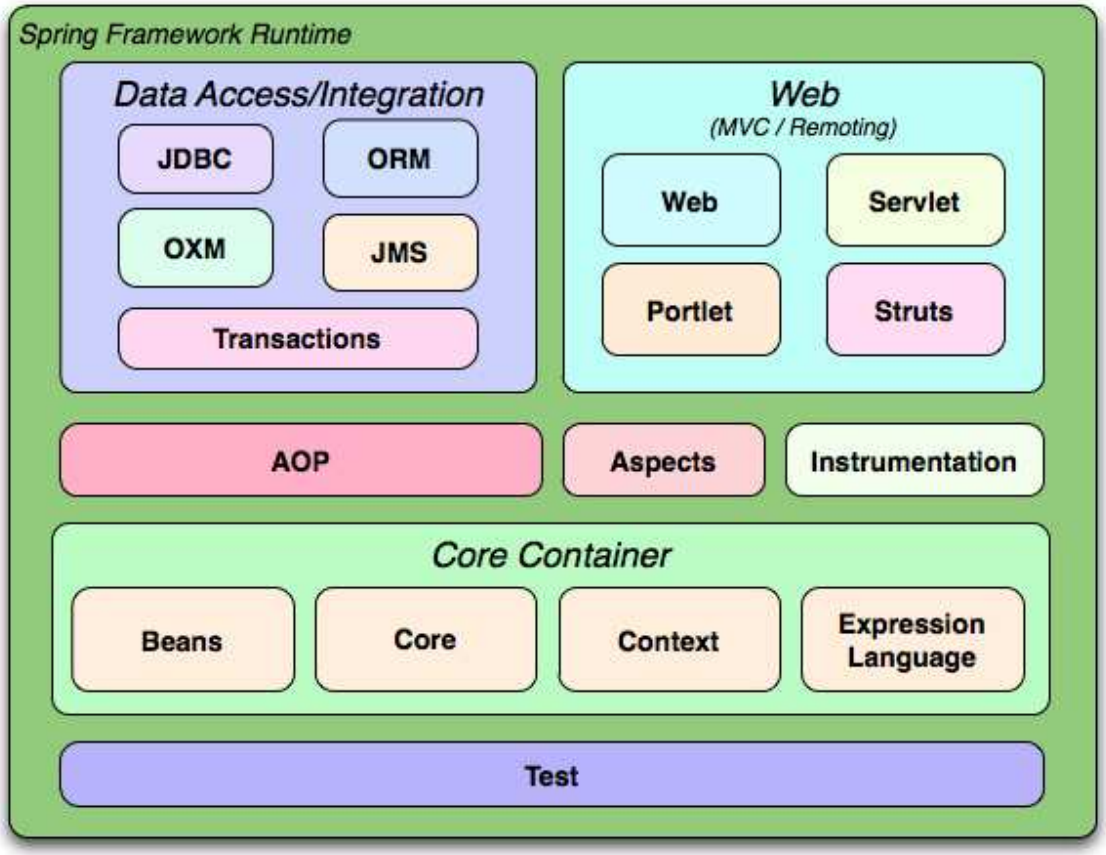


Figura 2. Módulos de Spring

3.5. Maven

Maven es una herramienta de software para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002. Maven tiene un modelo de configuración de construcción muy simple, basado en un formato XML.

Maven utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.

Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos Open Source en Java, de Apache y otras organizaciones y desarrolladores.

En este proyecto, Maven se ha empleado para generar el empaquetado en un archivo WAR, lo que facilita la instalación al servidor web.

- WAR

Este componente es un archivo que contiene un archivo JAR que posee uno o más módulos web. Pudiendo ser desde un simple sistema JSP a un servicio web.

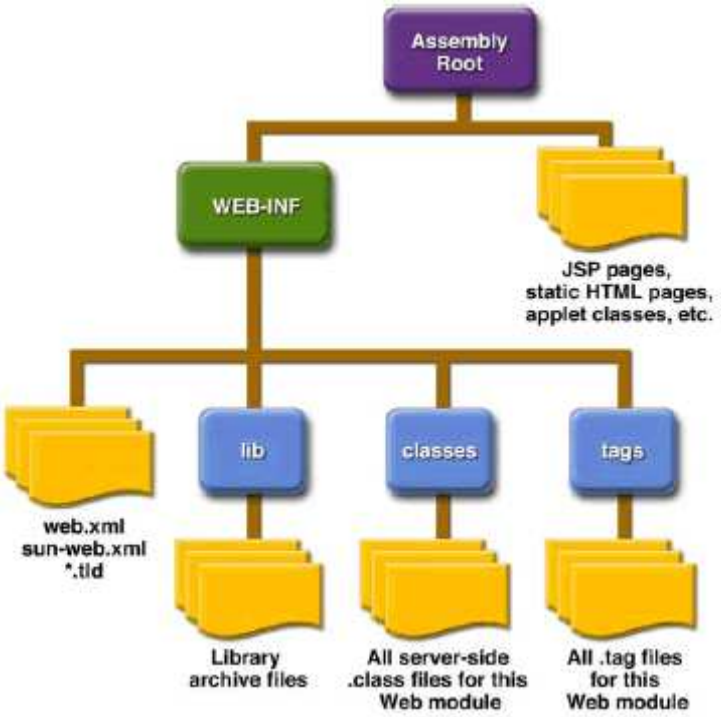


Figura 3. Estructura archivo WAR

3.6. Tomcat

Tomcat (también llamado Jakarta Tomcat o Apache Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

Tomcat es un servidor web con soporte de servlets y JSP's. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila

JSP’s convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

En el proyecto se ha utilizado Tomcat como servidor web, es una de las piezas fundamentales puesto que se encarga de contestar a las peticiones, realizadas por el usuario, de forma adecuada, entregando como resultado una página web o información de todo tipo de acuerdo a los comandos solicitados.

3.7. MySQL

MySQL es un sistema de administración de bases de datos (Database Management System, DBMS) para bases de datos relacionales, multihilo y multiusuario. Utiliza el lenguaje SQL (Structured Query Language) que es el estándar de consulta a bases de datos a nivel mundial.

También es muy destacable, la condición de Open Source de MySQL, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente.

3.8. Alfresco

Alfresco es un sistema de administración de contenidos libre, basado en estándares abiertos y de escala empresarial para sistemas operativos tipo Unix y Otros. Se distribuye en dos variantes diferentes:

- Alfresco Community Edition: Es software libre, con licencia LGPL de código abierto y estándares abiertos.
- Alfresco Enterprise Edition: Se distribuye bajo licencia de código abierto y estándares abiertos con soporte comercial y propietario a escala empresarial.

Está diseñado para usuarios que requieren un alto grado de modularidad y rendimiento escalable. Alfresco incluye un repositorio de contenidos, un framework de portal web para administrar y usar contenido estándar en portales, una interfaz CIFS que provee compatibilidad de sistemas de archivos en Windows y sistemas operativos tipo Unix, un sistema de administración de contenido web, capacidad de virtualizar

aplicaciones web y sitios estáticos vía Apache Tomcat, búsquedas vía el motor Lucene y flujo de trabajo en jBPM. Alfresco está desarrollado en Java.

En el proyecto desarrollado Alfresco es utilizado como software de gestión documental para documentos, concretamente, se guarda la estructura de carpetas y los ficheros físicos.

A continuación se detallan los lenguajes de programación utilizados en el proyecto, junto con una breve descripción de cada uno de ellos.

3.9. Java

Java es el lenguaje de programación escogido para el desarrollo de toda la aplicación. Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores.

El lenguaje Java se creó con cinco objetivos principales:

- Debería usar la metodología de la programación orientada a objetos.
- Debería permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Debería incluir por defecto soporte para trabajo en red.
- Debería diseñarse para ejecutar código en sistemas remotos de forma segura.
- Debería ser fácil de usar y tomar lo mejor de otros lenguajes orientados a objetos, como C++.

3.10. 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.

Las JSP's permiten la utilización de código Java mediante scripts. Además, es posible utilizar algunas acciones JSP predefinidas mediante etiquetas. Estas etiquetas pueden

ser enriquecidas mediante la utilización de Bibliotecas de Etiquetas (TagLibs o Tag Libraries) externas e incluso personalizadas.

El funcionamiento general de la tecnología JSP es que el Servidor de Aplicaciones interpreta el código contenido en la página JSP para construir el código Java del servlet a generar. Este servlet será el que genere el documento (típicamente HTML) que se presentará en la pantalla del Navegador del usuario.

Es posible enriquecer el lenguaje de etiquetas utilizado por JSP. Para ello debemos extender la capa de alto nivel JSP mediante la implementación de Bibliotecas de Etiquetas (Tags Libraries). Un ejemplo de estas bibliotecas son las propocionadas por Sun bajo la denominación de JSTL.

Los JSPs son en realidad servlets: un JSP se compila a un programa en Java la primera vez que se invoca, y del programa en Java se crea una clase que se empieza a ejecutar en el servidor como un servlet. La principal diferencia entre los servlets y los JSPs es el enfoque de la programación: un JSP es una página Web con etiquetas especiales y código Java incrustado, mientras que un servlet es un programa Java puro que recibe peticiones y genera a partir de ellas una página web.

3.11. 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.

Estas bibliotecas de etiquetas extienden de la especificación de JSP (la cual a su vez extiende de la especificación de Servlet). Su API nos permite además desarrollar nuestras propias bibliotecas de etiquetas.

3.12. JavaScript

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

JavaScript se interpreta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

3.13. jQuery

jQuery es una biblioteca o framework de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres y privativos. jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. Para ello utiliza las funciones $() o jQuery().

3.14. AJAX

Ajax, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las

páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

3.15. JSON

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.

3.16. CSS

El nombre hojas de estilo en cascada viene del inglés Cascading Style Sheets, del que toma sus siglas. CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.

La información de estilo puede ser adjuntada como un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo "style".

4. Análisis

Este apartado está dedicado a la descripción de lo que se realiza en nuestro proyecto, centrándose en qué consiste, es decir, las principales funcionalidades requeridas por la aplicación.

La organización de esta sección es la siguiente:

- Descripción de la aplicación: Breve descripción del objetivo de la aplicación.
- Perfiles: Contiene la descripción de los roles o perfiles de usuarios que pueden acceder a la aplicación, así como, el detalle de los estados para cada uno.
- Flujos principales: Descripción de los flujos principales de la aplicación, analizando detalles funcionales.
- Modelo de dominio: Mostrará el conjunto de clases conceptuales del problema y las relaciones entre sí.

4.1. Descripción de la aplicación

La aplicación permitirá almacenar toda la documentación que se genera alrededor de los trámites que forman parte del catálogo de trámites y asegurar su integridad, seguridad y completitud de cara a preparar la documentación para ponerla a disposición del archivo.

De esta forma, el usuario recibirá una serie de documentos por parte de un tercero, el cual desea realizar un trámite en el órgano donde se encuentra implantada la aplicación. A partir de aquí, el usuario podrá crear un expediente asociado al trámite concreto que quiere realizar dicha persona. Una vez creado el expediente, el usuario tendrá la posibilidad de gestionar todos estos documentos.

Por otra parte, cuando la institución necesite modificar su catálogo de trámites, el usuario tendrá la opción de gestión del mismo. Esto es, podrá dar de alta nuevos trámites, modificarlos y eliminarlos.

Por último, y no menos importante, la aplicación también permitirá la gestión de los usuarios.

4.2. Perfiles de la aplicación

La aplicación está compuesta de dos perfiles de usuario descritos a continuación como “Administrador” y “Administrador Alfresco”.

No se dispondrá del perfil “Administrador Alfresco” a nivel de la aplicación. La gestión de este perfil se realizará directamente sobre Alfresco.

A continuación, se detalla el comportamiento de los perfiles mencionados.

4.2.1. Rol Administrador

Los usuarios de la aplicación, enmarcados en este perfil, son los responsables de la tramitación de expedientes a partir de los documentos de entrada. Por otro lado, también se encargan de gestionar las principales entidades del sistema.

Debe poder realizar las siguientes funcionalidades:

- Expedientes: Crear, modificar metadatos y eliminar.
- Documentos: Incorporar y borrar.
- Gestión de usuarios: Mantenimiento de estos, mediante operaciones de alta, baja y modificación.
- Gestión de trámites: Mantenimiento a través de funciones de alta, baja y modificación.

Tras pasar la autenticación, el usuario accederá a la pantalla principal “Pantalla Inicial”, la cual se compone de 3 pestañas con el contenido siguiente:

1. Pantalla de bienvenida: Se mostrarán las principales opciones de la aplicación.
2. Pantalla resumen expedientes: Dispone las funcionalidades relacionadas con expedientes.
3. Pantalla administración: Se compone de las pantallas “Gestión Usuarios” y “Gestión Trámites”, implementan las funciones asociadas a usuarios y trámites, respectivamente.

En el siguiente diagrama se muestra el detalle de todos los estados posibles para un usuario de la aplicación:

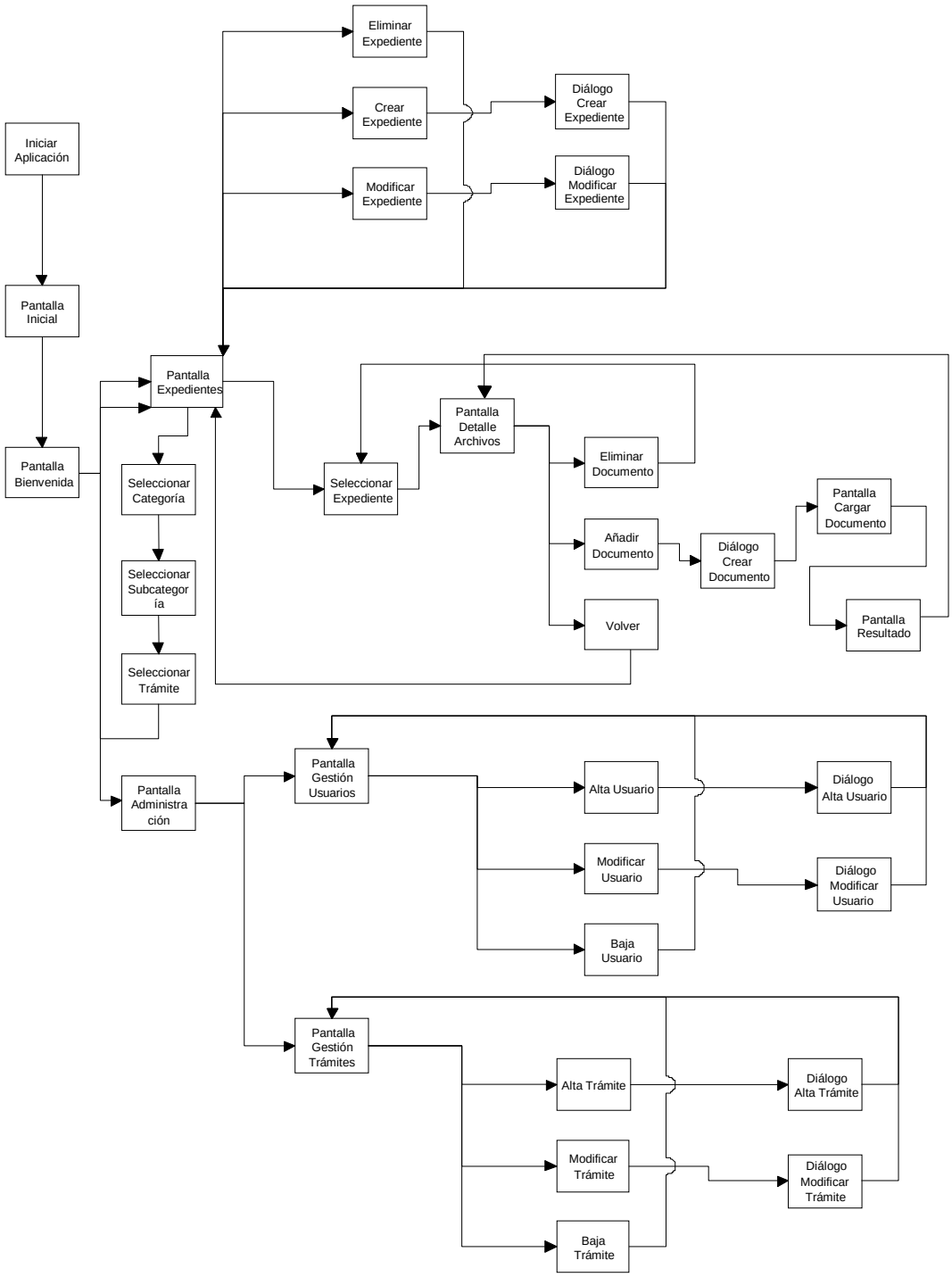


Figura 4. Diagrama de actividades de la aplicación

4.2.2. Rol Administrador Alfresco

Este usuario será el único que puede crear las carpetas asociadas a los expedientes y será el único que podrá gestionarlas. Además, mantiene las funcionalidades del perfil de Administrador.

Concretamente, debe poder hacer:

- Gestión de las carpetas asociadas a los expedientes: Alta, baja y modificación.
- Gestión de los documentos pertenecientes a un expediente: Visualización, descarga y borrado físico.
- Búsqueda de expedientes y documentos.

No se definirá este rol a nivel de la aplicación. Las acciones relativas a este perfil se realizarán con un usuario administrador de Alfresco a través de dicha herramienta.

4.3. Flujos principales de la aplicación

En este apartado se detallan los componentes que forman las diferentes partes de la aplicación.

La aplicación dispondrá de tres funcionalidades principales, como ya se adelantaba anteriormente:

1. Pantalla de bienvenida: Incluye la pantalla de bienvenida de la aplicación.
2. Pantalla resumen expedientes: Se proporcionará una herramienta para la gestión de expedientes en sus principales estados del ciclo de vida y hasta la disposición de los mismos.
3. Pantalla administración: Se facilitarán funcionalidades tanto para la gestión de usuarios como para la gestión de trámites.

La aplicación iniciará las tres funcionalidades y las englobará en una única pantalla, “Pantalla Inicial”, descrita en los siguientes apartados.

Desde esta pantalla de inicio, se iniciarán concurrentemente los flujos "Pantalla de bienvenida", "Pantalla resumen expedientes" y la "Pantalla administración".

A continuación, se analizarán todos los detalles de las diferentes acciones posibles, así como el detalle de las funcionalidades incluidas.

4.3.1. PANTALLA DE AUTENTICACIÓN

Esta pantalla se mostrará una vez se inicie la aplicación.

4.3.1.1. Restricciones

Cualquier usuario de la plataforma podrá acceder a la pantalla de autenticación.

4.3.1.2. Funcionalidades

Requerimento	Autenticación	Código	REQ-4.1.1
Descripción	Tras arrancar la aplicación, se mostrará la pantalla de autenticación, donde tras introducir el usuario y la contraseña se accederá a la "Pantalla Inicial". Si el usuario y/o la contraseña son incorrectos se informará con un mensaje de error y, obviamente, no se podrá acceder.		

4.3.2. PANTALLA INICIAL

A esta pantalla se accede tras pasar la autenticación.

4.3.2.1. Restricciones

Cualquier usuario que esté dado de alta en la aplicación, tras introducir correctamente su usuario y contraseña, podrá acceder a la pantalla principal de la aplicación.

4.3.2.2. Funcionalidades

Requerimento	Inicial	Código	REQ-4.2.1
Descripción	Una vez se muestre la "Pantalla Inicial", se iniciará de forma automática la "Pantalla de Bienvenida", la gestión de expedientes y la "Pantalla Administración".		

4.3.3. PANTALLA BIENVENIDA

A esta pantalla se accede tras pasar la autenticación o al pinchar sobre la pestaña Inicio.

4.3.3.1. Restricciones

Cualquier usuario que esté dado de alta en la aplicación podrá acceder a la pantalla de bienvenida.

4.3.3.2. Funcionalidades

Requerimento	Bienvenida	Código	REQ-4.3.1
Descripción	Se mostrará un mensaje de bienvenida al usuario.		

4.3.4. PANTALLA EXPEDIENTES

Esta pantalla contendrá el resumen de expedientes que se encuentran dentro de los trámites, para acceder a ella habrá que pinchar sobre la pestaña Expedientes de la “Pantalla Inicial”.

4.3.4.1. Restricciones

Se mostrarán todos los expedientes permitiendo el desplazamiento mediante una barra de desplazamiento lateral.

Los trámites se engloban en subcategorías, y las categorías contienen subcategorías. Tanto las categorías como las subcategorías estarán parametrizadas internamente en la aplicación, esto es, estarán predefinidas y no habrá ninguna herramienta para su gestión.

4.3.4.2. Funcionalidades

Requerimento	Expedientes	Código	REQ-4.4.1

Descripción	Se mostrarán tres barras desplegables para seleccionar la categoría, la subcategoría y el trámite. Nota: Las categorías y subcategorías que se mostrarán serán sólo las de los trámites accesibles por el usuario. Una vez seleccionado el trámite se mostrará el detalle de todos los expedientes que cumplan las condiciones. Por defecto, se mostrarán todos los expedientes. Las columnas a mostrar serán: Estado Nº de expediente Fecha apertura Título Interesado Selección

Requerimento	**Selección Expediente**	**Código**	**REQ-4.4.2**
Descripción	Se permitirá la selección de cualquier expediente pinchando sobre el Nº de expediente. Esta acción navegará a la “Pantalla Detalle de Archivos”.		

Requerimento	**Eliminar Expediente**	**Código**	**REQ-4.4.3**
Descripción	Se permitirá eliminar los expedientes seleccionados, pero internamente sólo se ocultará el expediente. Para ello, se seleccionarán los expedientes que se desean eliminar a través de *checkboxes* situados sobre la columna Selección y, posteriormente, actuando sobre el botón Eliminar. Una vez finalizada esta acción, se seguirá mostrando la pantalla de expedientes.		

Requerimento	**Crear Expediente**	**Código**	**REQ-4.4.4**

Descripción	El alta de nuevos expedientes estará disponible para todos los usuarios pulsando sobre el botón Crear. Esta acción permitirá realizar el alta de un expediente a partir de un formulario de datos (“Diálogo Crear Expediente”). La creación se hará siempre dentro de un tipo de trámite, por tanto, requerirá en primer lugar seleccionar un trámite a través de barras desplegables “Categorías”, “Subcategorías” y “Trámites”. El Nº de expediente se generará internamente. Una vez finalizada esta acción, se seguirá mostrando la pantalla de expedientes.

Requerimento	**Modificar Expediente**	**Código**	**REQ-4.4.5**
Descripción	Se permitirá modificar los principales parámetros de un expediente a través del “Diálogo Modificar Expediente”. Para ello, se pinchará sobre el botón situado en la columna Selección del expediente a modificar. No se permitirá modificar el Nº de expediente. Una vez finalizada esta acción, se seguirá mostrando la pantalla de expedientes.		

4.3.5. PANTALLA DETALLE ARCHIVOS

Pantalla donde se mostrará el detalle de todos los archivos dentro de un expediente.

Todos los elementos se podrán eliminar, pero internamente no se eliminarán, únicamente se ocultarán de cara al usuario.

No se contempla la posibilidad de recuperar un documento previamente eliminado, requerirá la intervención del “Administrador Alfresco”.

4.3.5.1. Restricciones

No se realizará paginación de los datos, por tanto, se mostrarán todos los archivos permitiendo el desplazamiento mediante una barra de desplazamiento lateral.

4.3.5.2. Funcionalidades

Requerimento	**Detalle Archivos**	**Código**	**REQ-4.5.1**

Descripción	Se mostrará el detalle de todos los documentos pertenecientes al expediente seleccionado. Las columnas a mostrar serán: Tipo Título Tipo documento Naturaleza Fecha documento Herramientas

Requerimento	**Eliminar Documento**	**Código**	**REQ-4.5.2**
Descripción	Se permitirá eliminar, internamente, cualquier documento visible por el usuario. Para ello, se pulsará sobre el botón Eliminar situado en la columna de Herramientas del documento a eliminar. Una vez finalizada esta acción, se seguirá mostrando la pantalla de detalle de archivos.		

Requerimento	**Añadir Documento**	**Código**	**REQ-4.5.3**
Descripción	El alta de nuevos documentos estará disponible para todos los usuarios pulsando sobre el botón Añadir documento. Esta acción permitirá realizar el alta de un documento a partir de un formulario de datos (“Diálogo Crear Documento”). Una vez finalizada esta acción, se navegará a la “Pantalla Cargar Documento”.		

Requerimento	**Volver**	**Código**	**REQ-4.5.4**
Descripción	Se permitirá volver a la pantalla de expedientes actuando sobre el botón Volver.		

4.3.6. PANTALLA CARGAR DOCUMENTO

Pantalla en la cual se seleccionará la ruta del documento que se desea subir.

4.3.6.1. Restricciones

El archivo se subirá físicamente al espacio, creado previamente en Alfresco, con nombre Nº de expediente.

Se mostrarán, tanto el Título como el Nº de expediente, pero no se permitirá modificar estos campos.

4.3.6.2. Funcionalidades

Requerimento	Carga Documento	Código	REQ-4.6.1
Descripción	Se permitirá seleccionar el archivo a subir a través de un botón Examinar. Tras esto, aparecerá la ruta del documento que se desea subir. La operación se llevará a cabo al pulsar sobre el botón Subir Fichero. Una vez finalizada esta acción se navegará a la “Pantalla Resultado”.		

4.3.7. PANTALLA RESULTADO

Pantalla donde se muestra el resultado de la carga de un documento.

4.3.7.1. Restricciones

Cualquier usuario que esté dado de alta en la aplicación, tras cargar un documento, accederá a la pantalla resultado.

4.3.7.2. Funcionalidades

Requerimento	Resultado Carga Documento	Código	REQ-4.7.1
Descripción	Se mostrará el resultado de la operación. Se permitirá volver a la pantalla de detalle de archivos actuando sobre el *link* Volver.		

4.3.8. PANTALLA ADMINISTRACIÓN

A esta pantalla se accede tras pulsar sobre la pestaña Administración de la “Pantalla Inicial”. Contendrá la gestión de usuarios y la gestión de trámites.

4.3.8.1. Restricciones

Cualquier usuario que esté dado de alta en la aplicación podrá acceder a la pantalla de administración.

4.3.8.2. Funcionalidades

Requerimento	Administración	Código	REQ-4.8.1
Descripción	Una vez se muestre la “Pantalla Administración”, se iniciará de forma automática la “Pantalla Gestión Usuarios” y la “Pantalla Gestión Trámites”.		

4.3.9. PANTALLA GESTIÓN USUARIOS

Pantalla donde se mostrará el detalle de todos los usuarios de la aplicación.

Todos los elementos se podrán eliminar, pero internamente no se eliminarán, únicamente se ocultarán de cara al usuario.

4.3.9.1. Restricciones

Cualquier usuario que esté dado de alta en la aplicación podrá acceder a la pantalla de gestión de usuarios.

No se realizará paginación de los datos, por tanto, se mostrarán todos los usuarios permitiendo el desplazamiento mediante una barra de desplazamiento lateral.

4.3.9.2. Funcionalidades

Requerimento	Gestión Usuarios	Código	REQ-4.9.1

Descripción	Se mostrará el detalle de todos los usuarios dados de alta en la aplicación. Las columnas a mostrar serán: Código Usuario Rol usuario Fecha creación usuario Selección

Requerimento	**Alta Usuario**	**Código**	**REQ-4.9.2**
Descripción	El alta de nuevos usuarios estará disponible para todos los usuarios pulsando sobre el botón Alta. Esta acción permitirá realizar el alta de un usuario a partir de un formulario de datos (“Diálogo Alta Usuario”). El Código se generará internamente. Una vez finalizada esta acción, se seguirá mostrando la pantalla de gestión de usuarios.		

Requerimento	**Modificar Usuario**	**Código**	**REQ-4.9.3**
Descripción	Se permitirá modificar los principales parámetros de un usuario a través del “Diálogo Modificar Usuario”. Para ello, se pinchará sobre el botón Modificación. El botón Modificación se mostrará cuando esté seleccionado sólo el usuario a modificar, esto es, si se selecciona más de uno, el botón se ocultará. No se permitirá modificar el Código. Una vez finalizada esta acción, se seguirá mostrando la pantalla de gestión de usuarios.		

Requerimento	**Baja Usuario**	**Código**	**REQ-4.9.4**

Descripción	Se permitirá eliminar los usuarios seleccionados, pero internamente sólo se ocultará el usuario. Para ello, se seleccionarán los usuarios que se desean eliminar a través de *checkboxes* situados sobre la columna Selección y, posteriormente, actuando sobre el botón Baja. En el caso que no haya ningún usuario seleccionado el botón Baja se ocultará. Una vez finalizada esta acción, se seguirá mostrando la pantalla de gestión de usuarios.

4.3.10. PANTALLA GESTIÓN TRÁMITES

Pantalla donde se mostrará el detalle de todos los trámites de la aplicación.

Todos los elementos se podrán eliminar, pero internamente no se eliminarán, únicamente se ocultarán de cara al usuario.

4.3.10.1. Restricciones

Cualquier usuario que esté dado de alta en la aplicación podrá acceder a la pantalla de gestión de trámites.

No se realizará paginación de los datos, por tanto, se mostrarán todos los trámites permitiendo el desplazamiento mediante una barra de desplazamiento lateral.

4.3.10.2. Funcionalidades

Requerimento	**Gestión Trámites**	**Código**	**REQ-4.10.1**
Descripción	Se mostrará el detalle de todos los trámites dados de alta en la aplicación. Las columnas a mostrar serán: Código trámite Tipología Subtipología Denominación trámite Selección		

Requerimento	Alta Trámite	Código	REQ-4.10.2
Descripción	El alta de nuevos trámites estará disponible para todos los usuarios pulsando sobre el botón Alta. Esta acción permitirá realizar el alta de un trámite a partir de un formulario de datos (“Diálogo Alta Trámite”). El Código trámite se generará internamente. La Tipología estará parametrizada internamente. Una vez finalizada esta acción, se seguirá mostrando la pantalla de gestión de trámites.		

Requerimento	Modificar Trámite	Código	REQ-4.10.3
Descripción	Se permitirá modificar los principales parámetros de un trámite a través del “Diálogo Modificar Trámite”. Para ello, se pinchará sobre el botón Modificación. El botón Modificación se mostrará cuando esté seleccionado sólo el trámite a modificar, esto es, si se selecciona más de uno, el botón se ocultará. No se permitirá modificar ni el Código trámite ni la Tipología. Una vez finalizada esta acción, se seguirá mostrando la pantalla de gestión de trámites.		

Requerimento	Baja Trámite	Código	REQ-4.10.4
Descripción	Se permitirá eliminar los trámites seleccionados, pero internamente sólo se ocultará el trámite. Para ello, se seleccionarán los trámites que se desean eliminar a través de *checkboxes* situados sobre la columna Selección y, posteriormente, actuando sobre el botón Baja. En el caso que no haya ningún trámite seleccionado el botón Baja se ocultará. Una vez finalizada esta acción, se seguirá mostrando la pantalla de gestión de trámites.		

4.4. Modelo de dominio

El modelo de dominio, también conocido como modelo conceptual, es una representación de las cosas, entidades, clases conceptuales u objetos del “mundo real” o dominio de interés. Se usa como base para el diseño de los objetos de software.

El modelo de dominio se representa con un conjunto de diagramas de clases.

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos.

Según el punto de vista, tiene puntos en común con el modelo de entidad-relación, ya que, este último, está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades, y de relaciones entre esos objetos.

El modelo entidad-relación es una herramienta para el modelado de datos de un sistema de información.

A continuación, se muestra un diagrama que representa ambos modelos. En él se muestran las clases conceptuales, las asociaciones entre las clases conceptuales, los atributos y las claves.

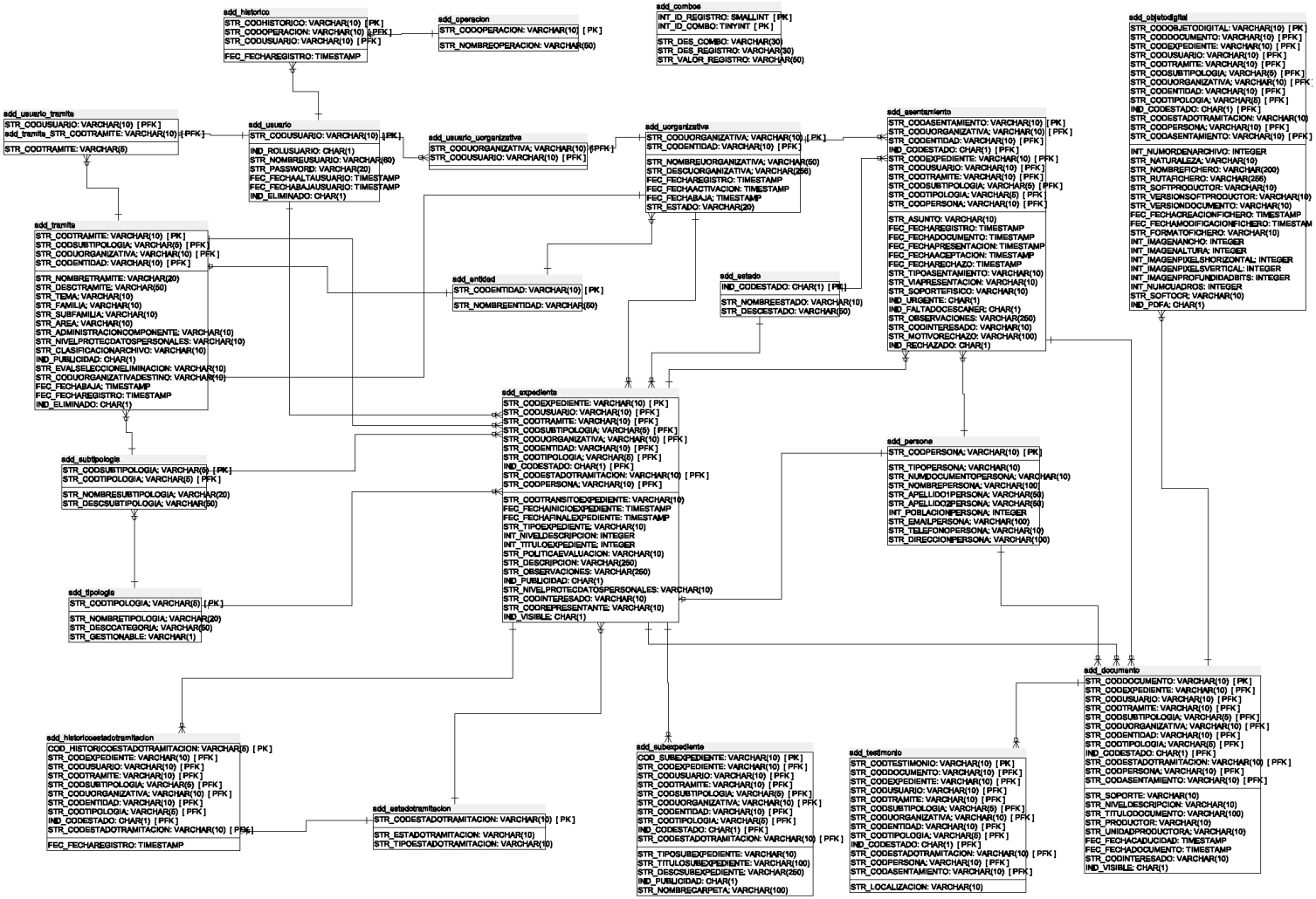


Figura 5. Diagrama modelo de dominio

5. Diseño

Este apartado describe en detalle cómo funciona la aplicación.

Esta sección se organiza de la siguiente forma:

- Diseño arquitectónico: Descripción de la arquitectura utilizada para la implementación de la solución.
- Diseño de las pantallas: Se mostrará el diseño de cada una de las pantallas y los diálogos modales de la aplicación.
- Código de ejemplo: Ejemplo de implementación de cada una de las capas.

5.1. Diseño arquitectónico

En este apartado se describen las diferentes capas utilizadas, los patrones de diseño y su uso a través de Struts. Finalmente, se muestra un diagrama de flujo de cómo viaja la información en la aplicación.

5.1.1. Modelo

El modelo comprende todos los objetos de negocio donde se implementa la lógica de negocio y donde se debe soportar todos los requisitos funcionales del sistema sin mezclarlo con partes correspondientes al workflow que corresponden al controlador.

Generalmente, los Action Beans siempre realizan las siguientes acciones:

1. Obtener los valores necesarios del Action Form, JavaBean, request o session.
2. Llamar a los objetos de negocio del modelo.
3. Analizar los resultados y según los mismos devolver el ActionForward correspondiente.

Veamos entonces un ejemplo de Action Bean:

```
public class LoginAction extends BaseAction{
      ...
```

```
protected ActionForward procesar(ActionMapping mapping,ActionForm
objetoModelo,HttpServletRequest request,HttpServletResponse response){
            try {
                  // Obteniendo atributos
                  LoginForm loginForm = (LoginForm) objetoModelo;
LoginResultado resultado = loginServicio.comprobarPasswordYUsuario(
loginForm.getCodUsuario(), loginForm.getPassword());
                  // Validando los parámetros
switch(resultado.getConfirmacion()){
                        case OK:
                        // Guardando el usuario en la sesión
request.getSession().setAttribute(USUARIO, resultado.getUsuario());
return mapping.findForward("ok_login");
                        ...
                        case KAO_USUARIO:
                        ...
                        case KAO_PASSWOR:

            }
request.setAttribute("codUsuario",
loginForm.getCodUsuario());
request.setAttribute("password", loginForm.getPassword());
                  return mapping.findForward("kao_login");
            }
            catch(Exception e){
                  log.error("################  ERROR LOGIN ACTION",e);
                  return mapping.findForward("kao_login");
            }
      }
...
}
```

Ejemplo 1. Action Bean

5.1.2. Vista

La vista comprende las JSP y los servlets involucrados en la generación de la interfaz de usuario. Struts provee soporte para construir aplicaciones multi-idioma, interacción con formularios y otras utilidades mediante la utilización de Tags especiales (TagLibraries).

Generalmente, una de las tareas que durante el desarrollo de una aplicación consume mucho trabajo es la interacción con formularios, ya sea para editar u obtener nueva información. Las comprobaciones, la gestión de errores, el volver a presentarle el mismo formulario al usuario con los valores que puso y los mensajes de error están soportados por Struts con tal de hacernos la vida un poco más fácil.

La idea es la siguiente: todo el trabajo de comprobaciones y generación de mensajes de error se implementa en los ActionForm y todo el trabajo de generación de interfaz en la/s JSP.

Veamos la receta:

1. Crear el ActionForm.
2. Crear la página JSP del formulario.
3. Declarar el ActionForm en struts-config.xml agregando en /struts-config/form-beans el tag <form-bean name="nombreForm" type="paquete.clase"/> y en la declaración del Action agregar los atributos name="nombreForm", scope="(request ó session)", e input="paginaForm.jsp".

   Ejemplo:

```
<struts-config>
  ...
  <form-beans>
  ...
   <form-bean name="loginForm"
type="es.alten.diputacion.add.modelo.struts.LoginForm"/>
  ...
  </form-beans>
  ...
  <action-mappings>
  ...
   <action path="/login"
                  scope="request"
            type="org.springframework.web.struts.DelegatingActionProxy"
                  validate="false"
                  name="loginForm">
      ...
      </action>
   ...
  <action-mappings>
  ...
</struts-config>
```

Ejemplo 2. Action Form

5.1.3. Controlador

El controlador comprende la funcionalidad involucrada desde que un usuario genera un estímulo (click en un link, envío de un formulario, etc.) hasta que se genera la interfaz de respuesta. Entre medio, llamará a los objetos de negocio del modelo para que resuelvan funcionalidad propia de la lógica de negocio y según el resultado de la misma ejecutará la JSP que deba generar la interfaz resultante.

Struts incluye un servlet que a partir de la configuración de struts-config.xml recibe las solicitudes del usuario, llama al Action Bean que corresponda y, según lo que éste retorne, ejecuta una JSP.

Por consiguiente, las tareas que se deben realizar son:

1. Escribir una clase Action.
2. Configurar el struts-config.xml para que incluya el nuevo action mapping y sus posibles forwards de salida.

   Por ejemplo:

```
<struts-config>
  ...
  <form-beans>
  ...
   <form-bean name="loginForm"
type="es.alten.diputacion.add.modelo.struts.LoginForm"/>
  ...
  </form-beans>
  ...
  <action-mappings>
  ...
   <action path="/login"
                 scope="request"
           type="org.springframework.web.struts.DelegatingActionProxy"
                 validate="false"
                 name="loginForm">
           <forward name="ok_login" path=".addPaginaBienvenida"/>
           <forward name="kao_login" path=".addLogin"/>
           <forward name="error" path=".addLogin"/>
           <forward name="inicio" path=".addLogin"/>
      </action>
   ...
  <action-mappings>
  ...
</struts-config>
```

Ejemplo 3.struts-config.xml

   Puesto que la acción está asociada a un formulario se debe definir un Form Bean, un Action Mapping con el Form Bean asociado y los forwards necesarios.

3. Incluir los Forms.

Los ActionForm Beans son clases que extienden ActionForm y que implementan métodos get y set para cada una de los inputs de un formulario de una página, y los métodos validate y reset.

Cuando un usuario completa un formulario y lo envía, el controlador busca en el scope especificado el ActionForm Bean correspondiente (todo esto configurado en el struts-config.xml) y si no lo encuentra lo crea. Luego realiza un set por cada input del formulario y finalmente llama al método validate. Si éste devolviera uno o más errores, el controlador llamaría a la JSP del formulario para que ésta lo volviera a generar (con los valores establecidos por el usuario) e incluyera el o los mensajes de error correspondientes. Si todo estuviese bien, llamaría al método perform del Action (también configurado en el struts-config.xml) pasándole el ActionForm Bean como parámetro para que sea utilizado para obtener los valores de los datos. Si bien el ActionForm tiene características que corresponden al modelo, los ActionForm pertenecen a la vista. Justamente uno de estos puntos comunes es la validación de datos y a fines de evitar la duplicación de funcionalidad, si desde un ActionForm debe realizar controles de validación que se hubiesen implementado en un objeto de negocio entonces se debería utilizar una instancia de éste para efectuarlos.

Al escribir un ActionForm debemos tener en cuenta los siguientes principios:

- No debe tener nada que corresponda a la lógica de negocio.
- No debería tener más que implementaciones de getters y setters (obligatoriamente un par por cada input del form; si el input se llama nombre entonces tendremos getNombre() y setNombre(String nombre)), y de los métodos reset y valídate.
- Debe ser un Firewall entre el usuario y el Action que detenga todo tipo de errores de incompletitud o inconsistencia.
- Si el formulario se desarrolla en varias páginas, el ActionForm y el Action deberán ser los mismos, lo que permitirá, entre otras cosas, que los input se puedan reorganizar en distintas páginas sin cambiar los ActionForm ni los Action.

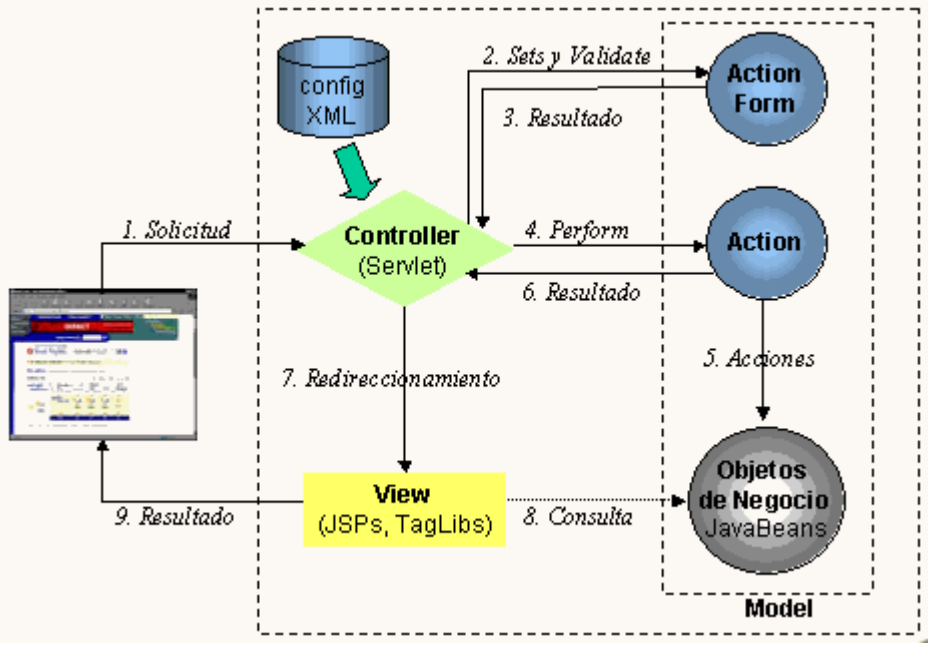


Figura 6. Funcionamiento ActionForm Beans

5.1.4. Diagrama flujo información

A continuación, se muestra, a través de un diagrama de flujo, cómo viaja la información en la aplicación, desde que el usuario interacciona con una página hasta que se le muestran los datos devueltos por la petición realizada.

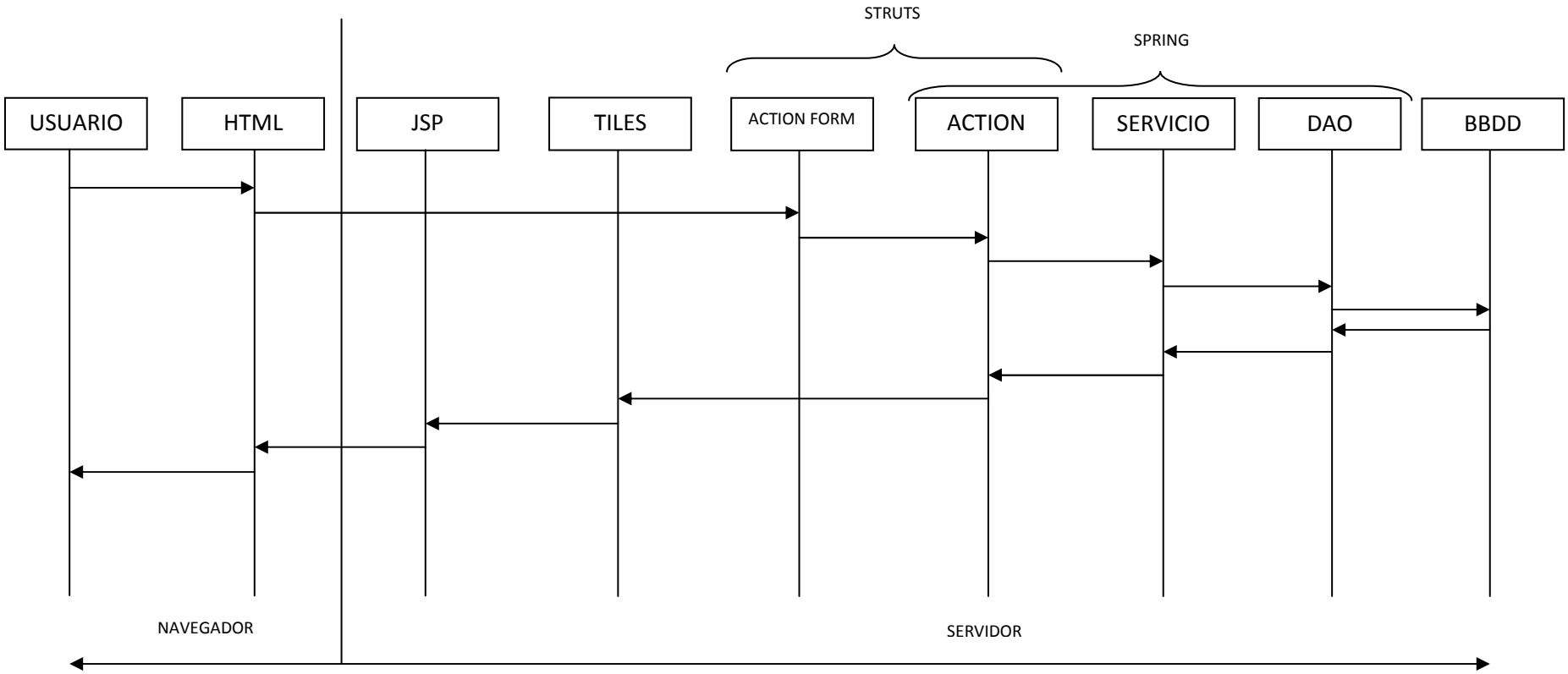


Figura 7. Diagrama flujo información

5.2. Diseño de las pantallas

En este apartado se muestra el diseño de cada una de las pantallas y diálogos modales que forman la aplicación.

5.2.1. PANTALLA DE AUTENTICACIÓN

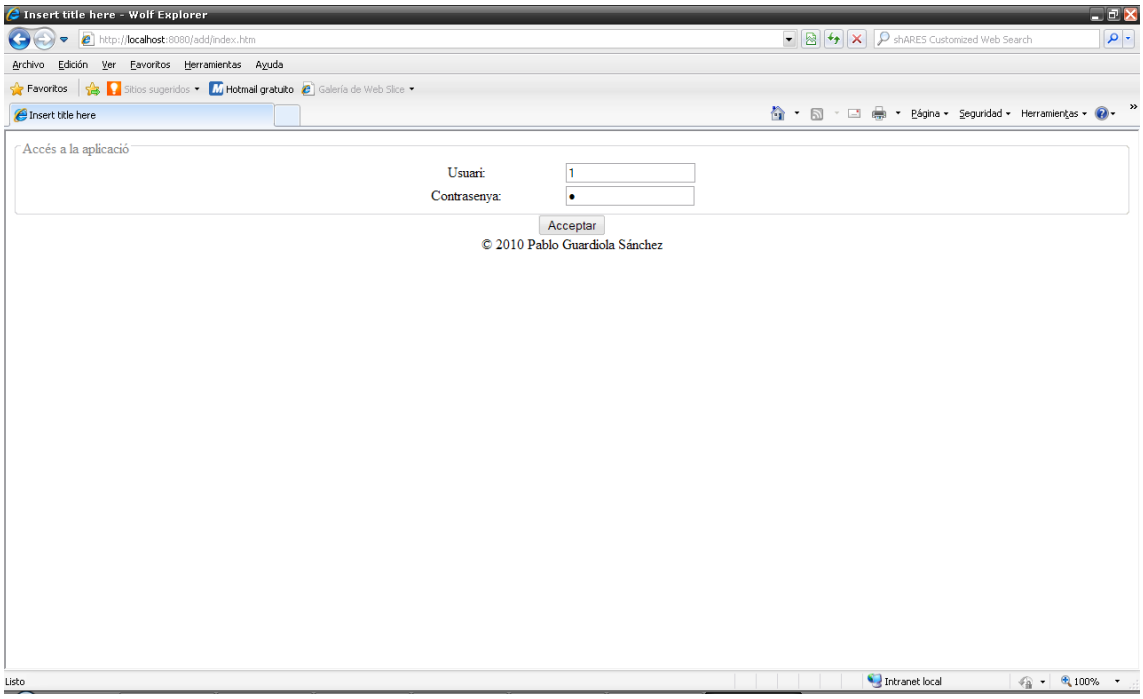


Figura 8. Diseño Pantalla Autenticación

5.2.2. PANTALLA BIENVENIDA

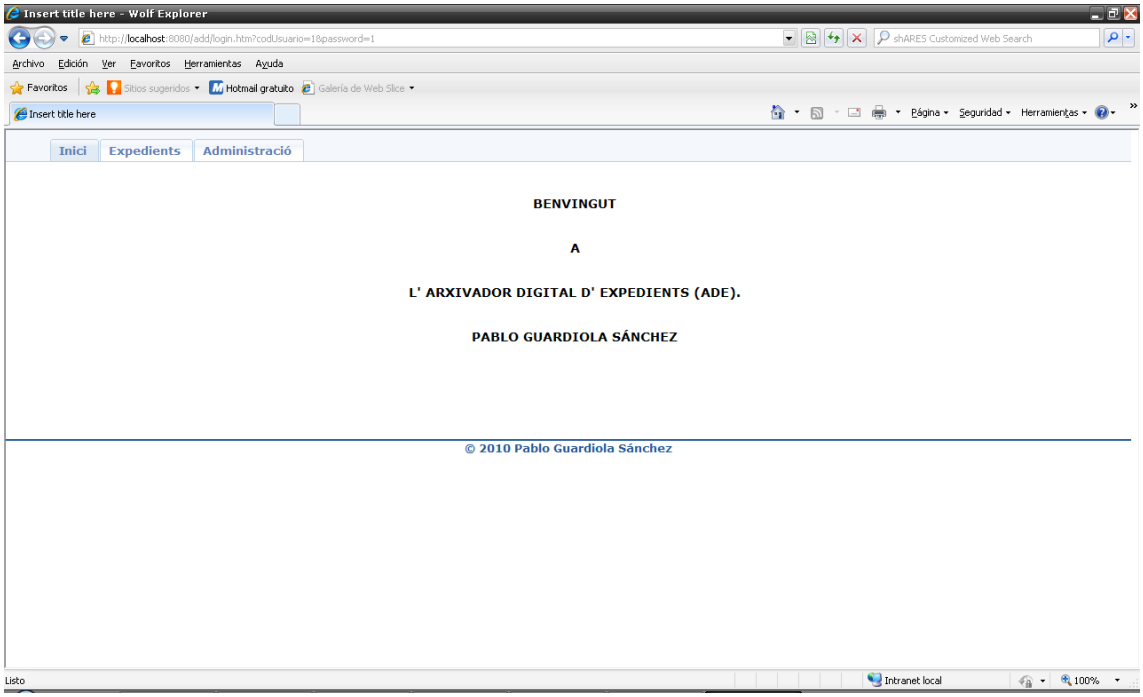


Figura 9. Diseño Pantalla Bienvenida

5.2.3. PANTALLA EXPEDIENTES

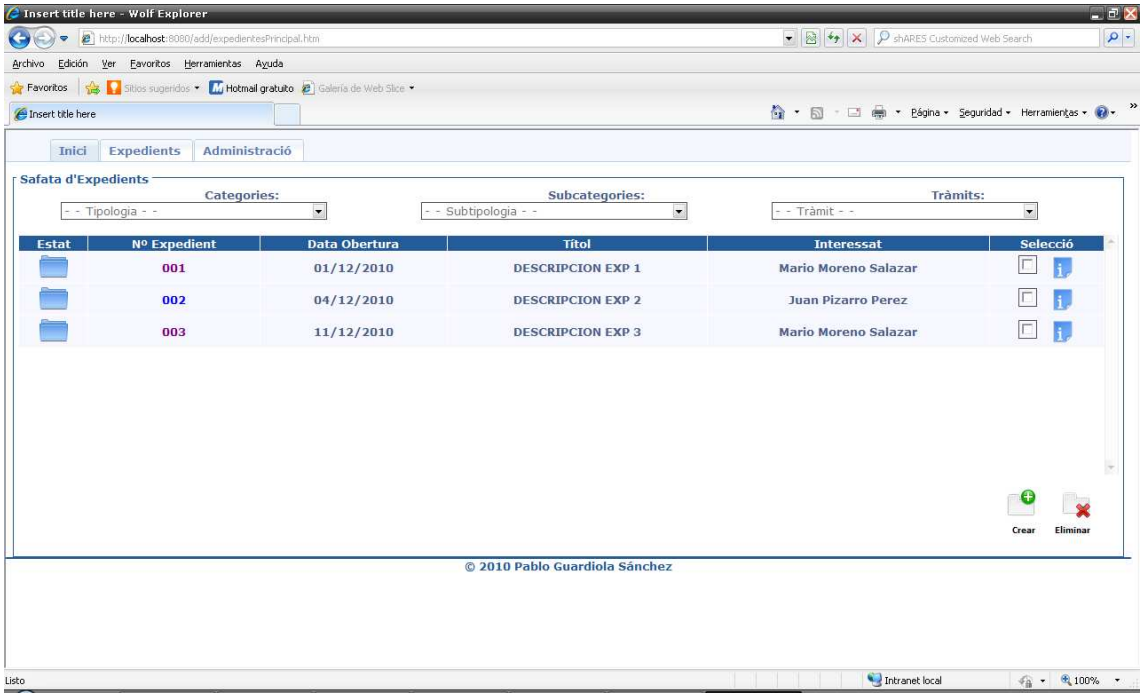


Figura 10. Diseño Pantalla Expedientes

5.2.3.1. DIÁLOGO CREAR EXPEDIENTE

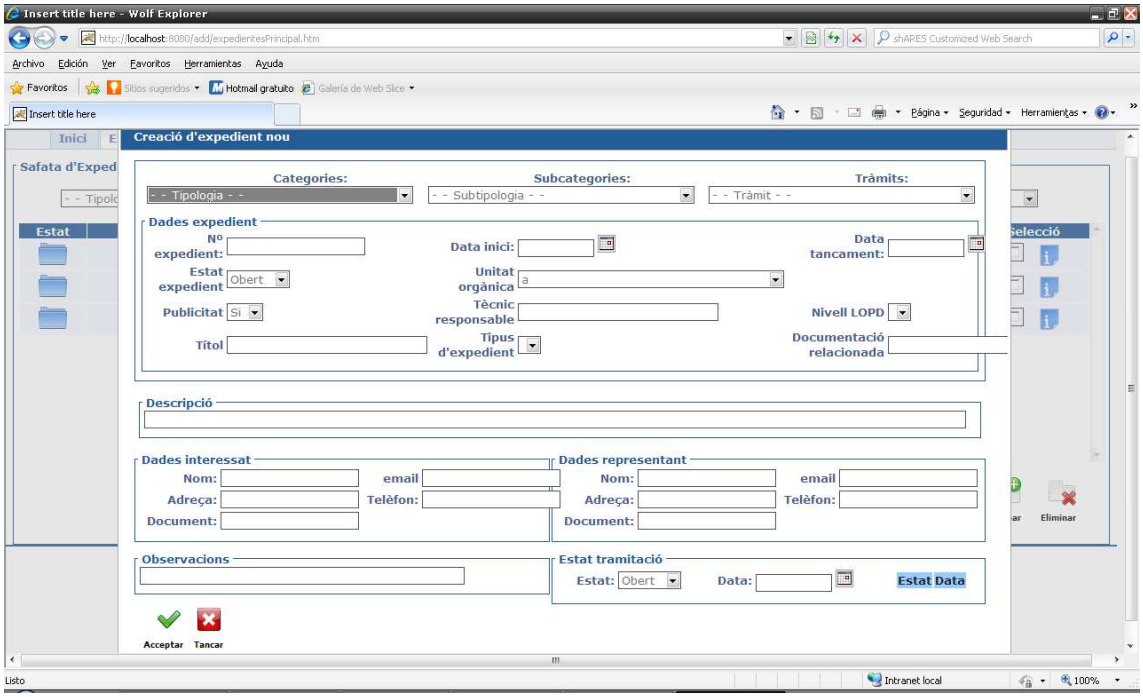


Figura 11. Diseño Diálogo Crear Expediente

5.2.3.2. DIÁLOGO MODIFICAR EXPEDIENTE

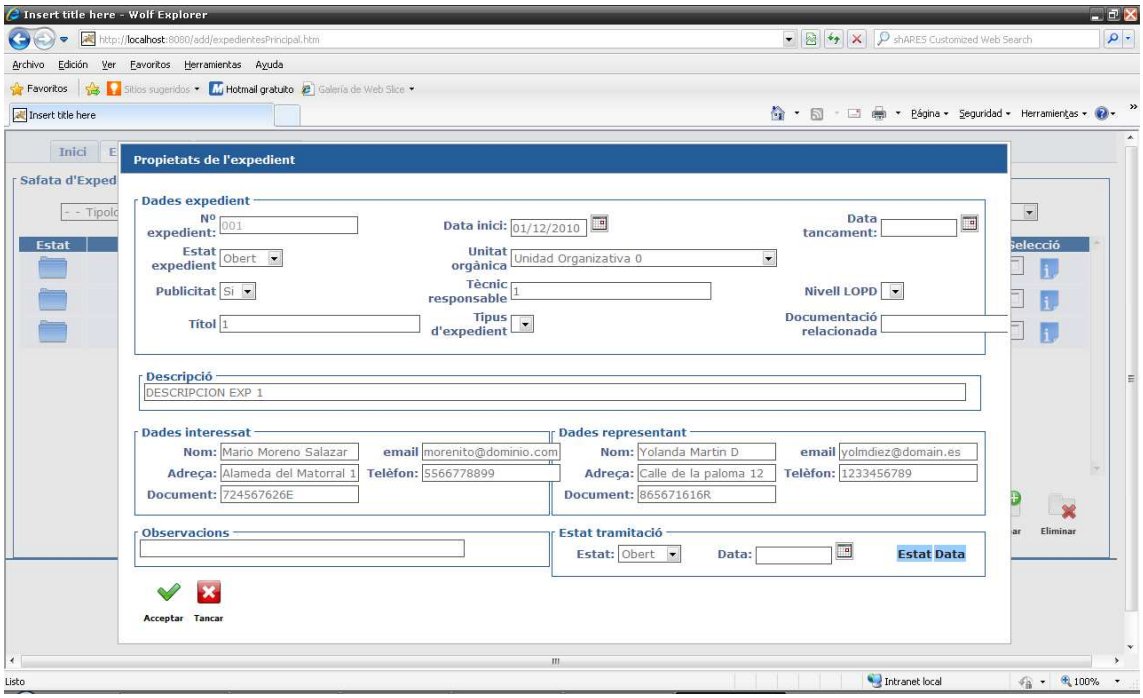


Figura 12. Diseño Diálogo Modificar Expediente

5.2.4. PANTALLA DETALLE ARCHIVOS

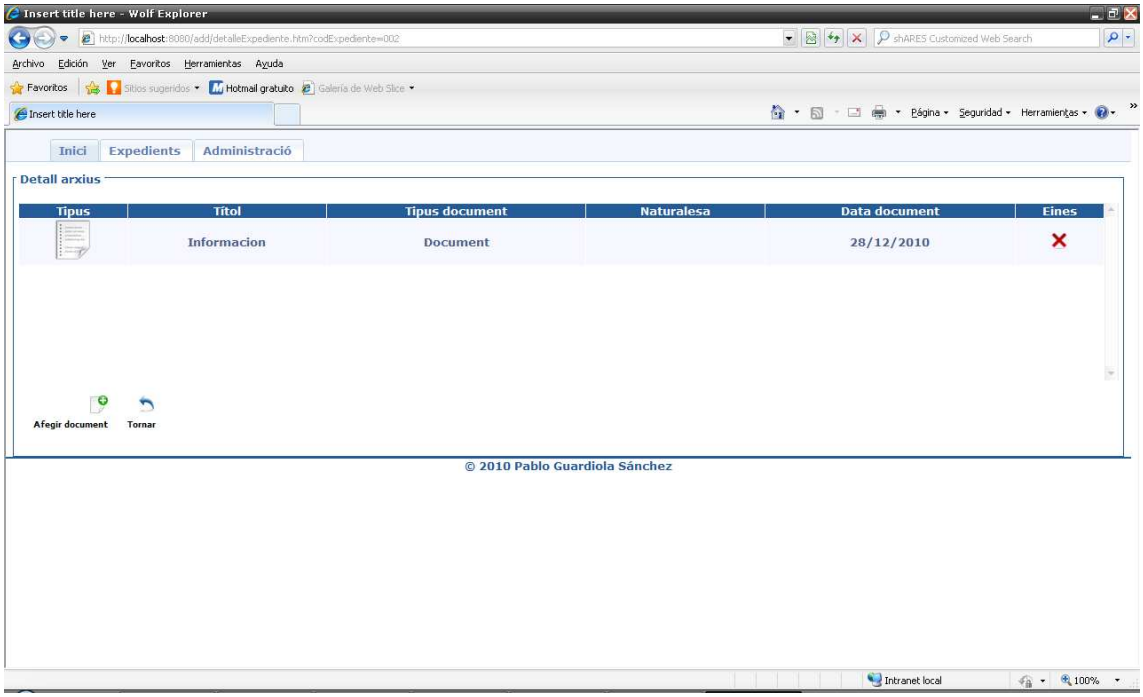


Figura 13. Diseño Pantalla Detalle Archivos

5.2.4.1. DIÁLOGO CREAR DOCUMENTO

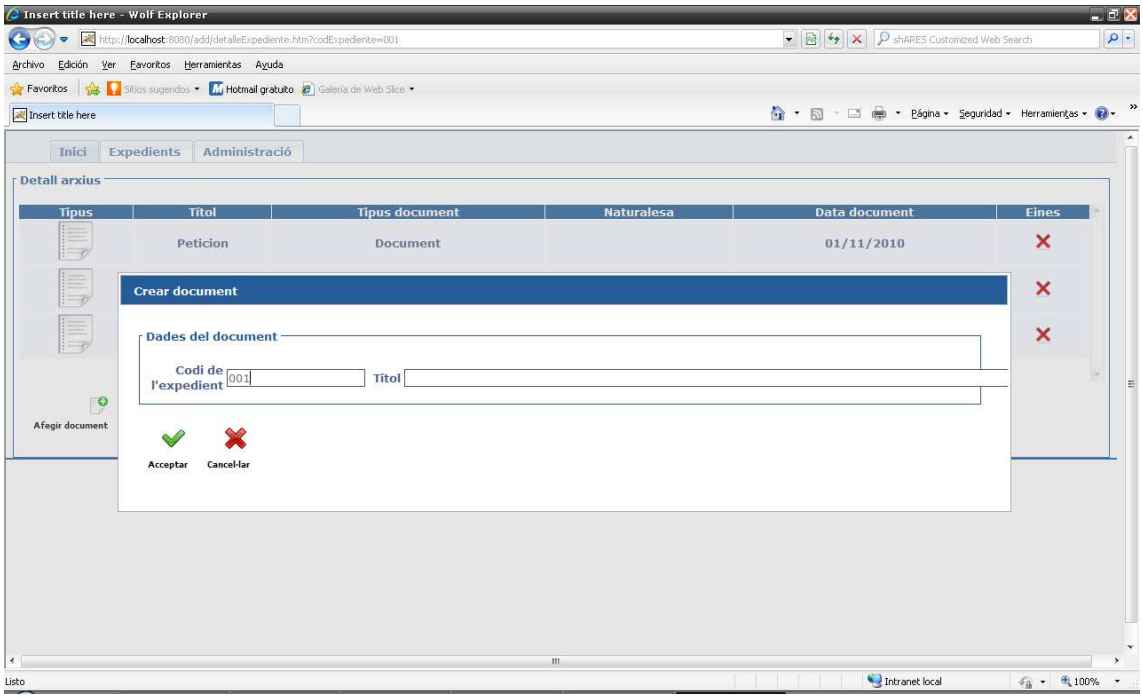


Figura 14. Diseño Diálogo Crear Documento

5.2.5. PANTALLA CARGAR DOCUMENTO

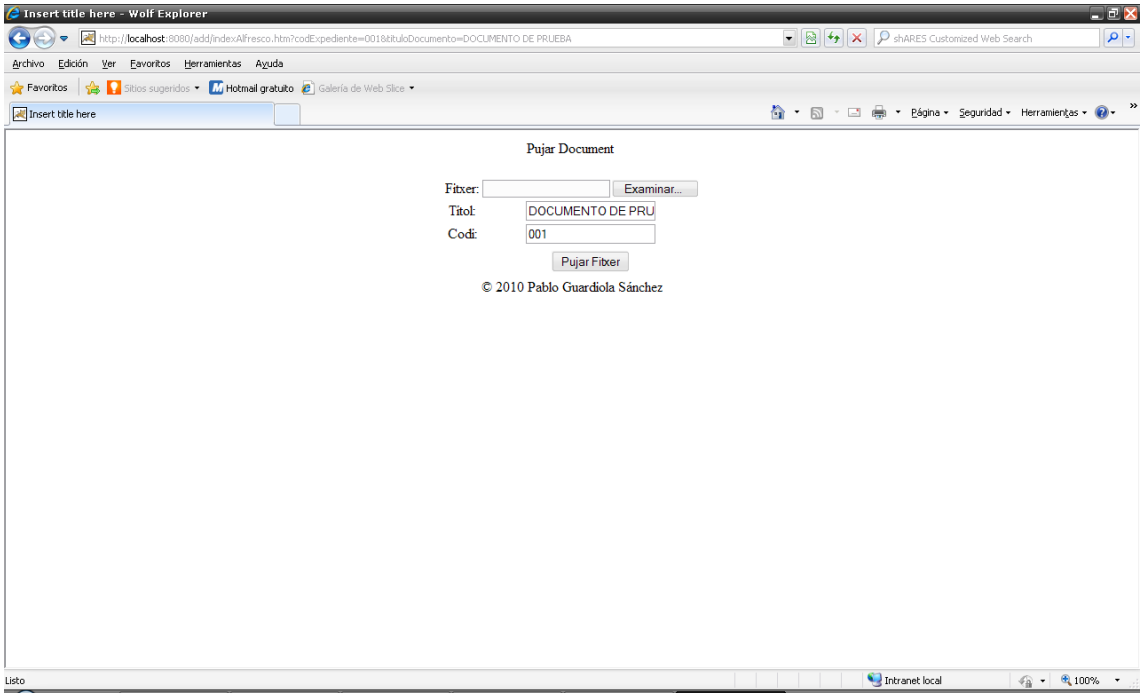


Figura 15. Diseño Pantalla Cargar Documento

5.2.6. PANTALLA RESULTADO

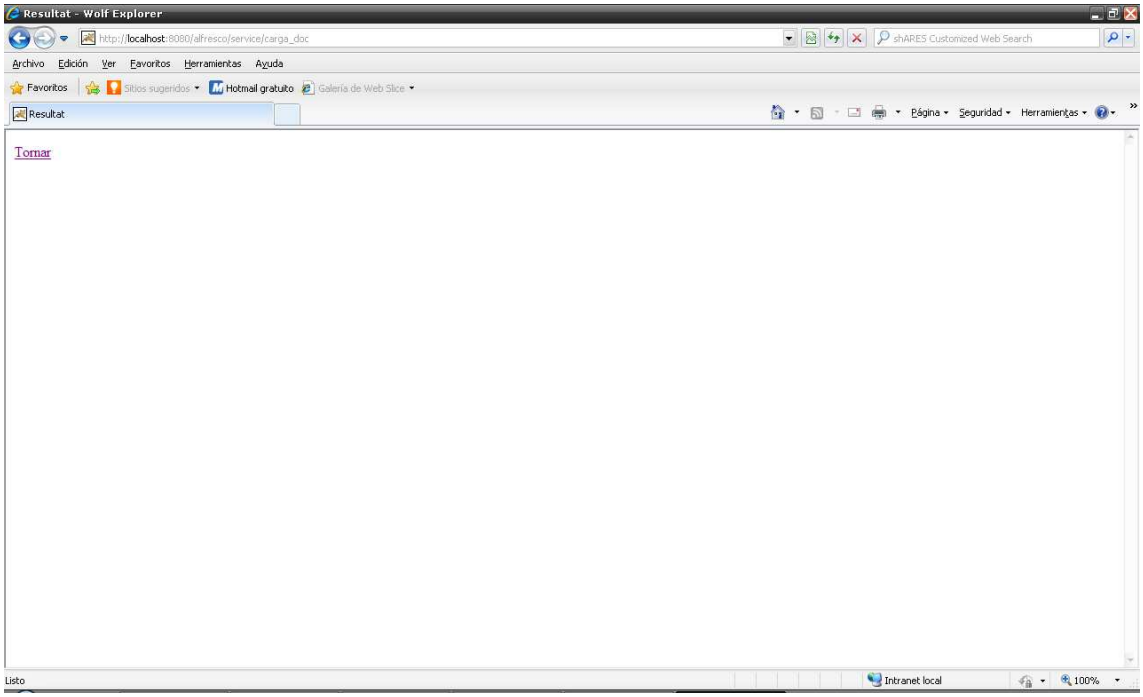


Figura 16. Diseño Pantalla Resultado

5.2.7. PANTALLA ADMINISTRACIÓN

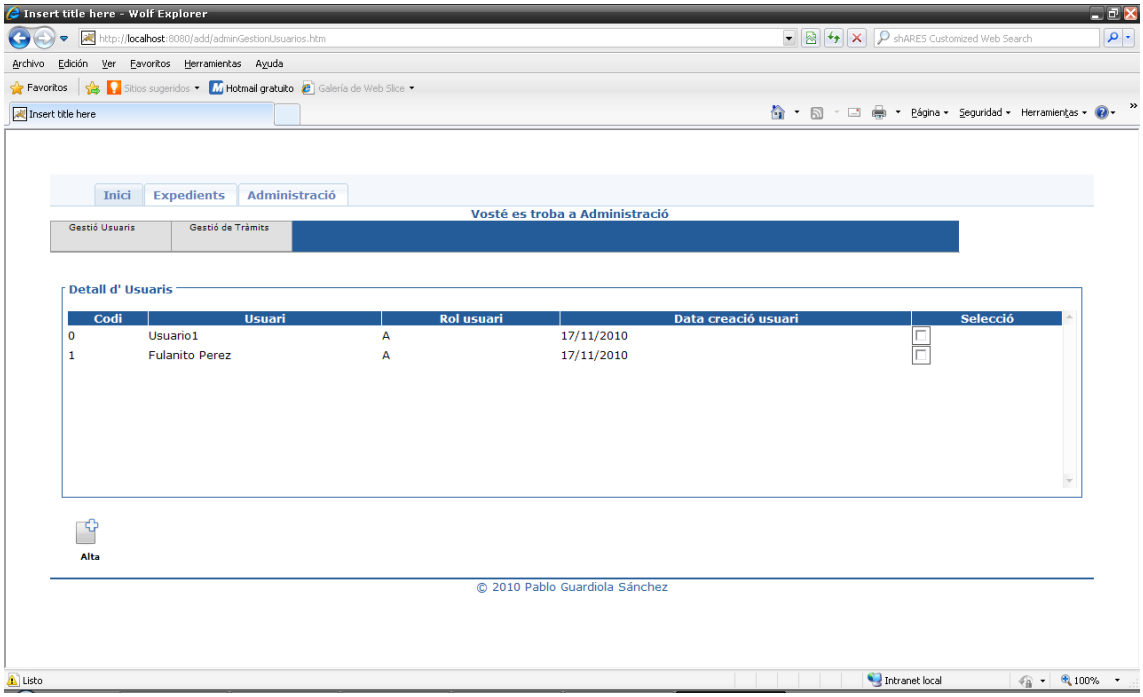


Figura 17. Diseño Pantalla Administración

5.2.8. PANTALLA GESTIÓN DE USUARIOS

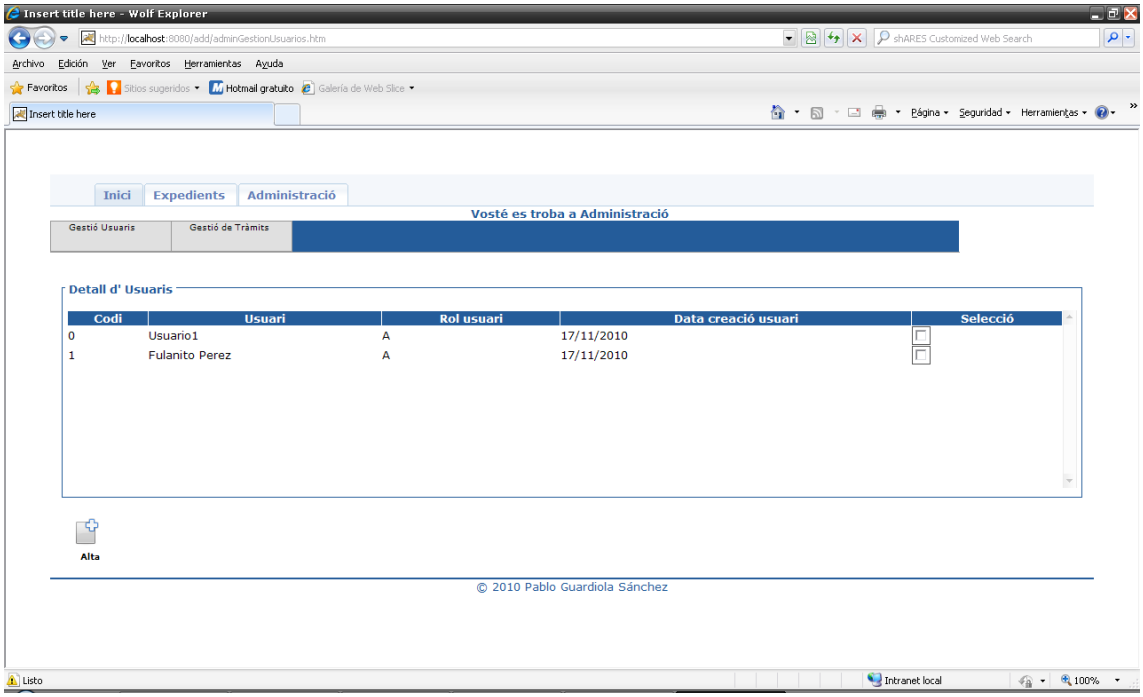


Figura 18. Diseño Pantalla Gestión Usuarios

5.2.8.1. DIÁLOGO ALTA DE USUARIO

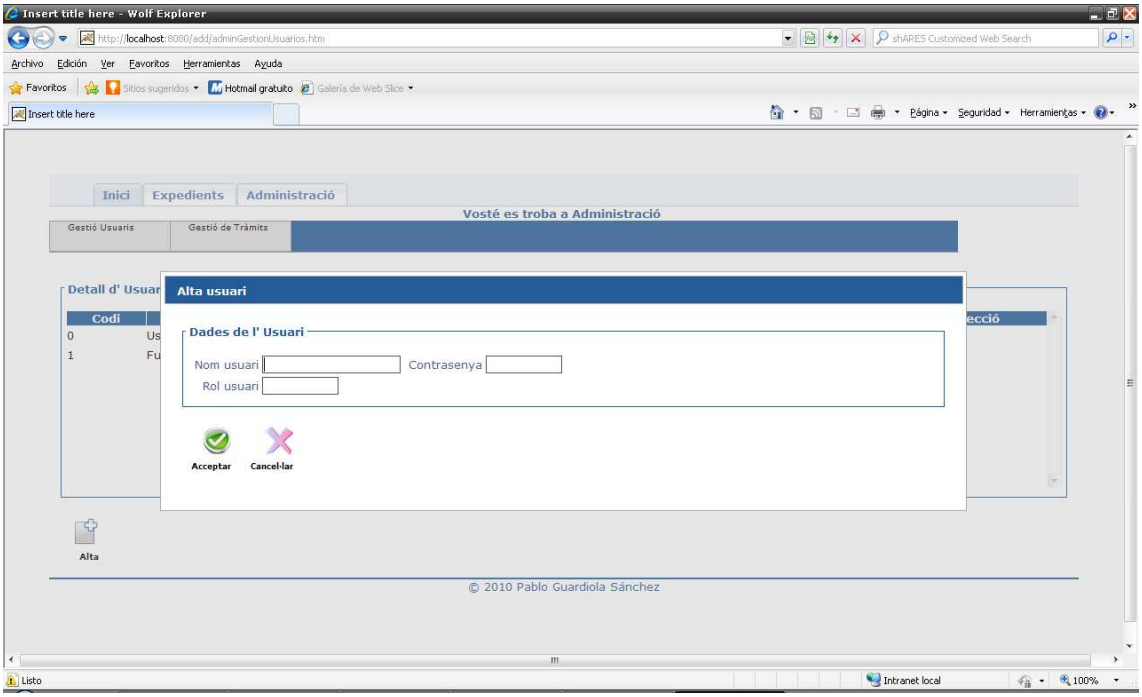


Figura 19. Diseño Diálogo Alta Usuario

5.2.8.2. DIÁLOGO MODIFICAR USUARIO

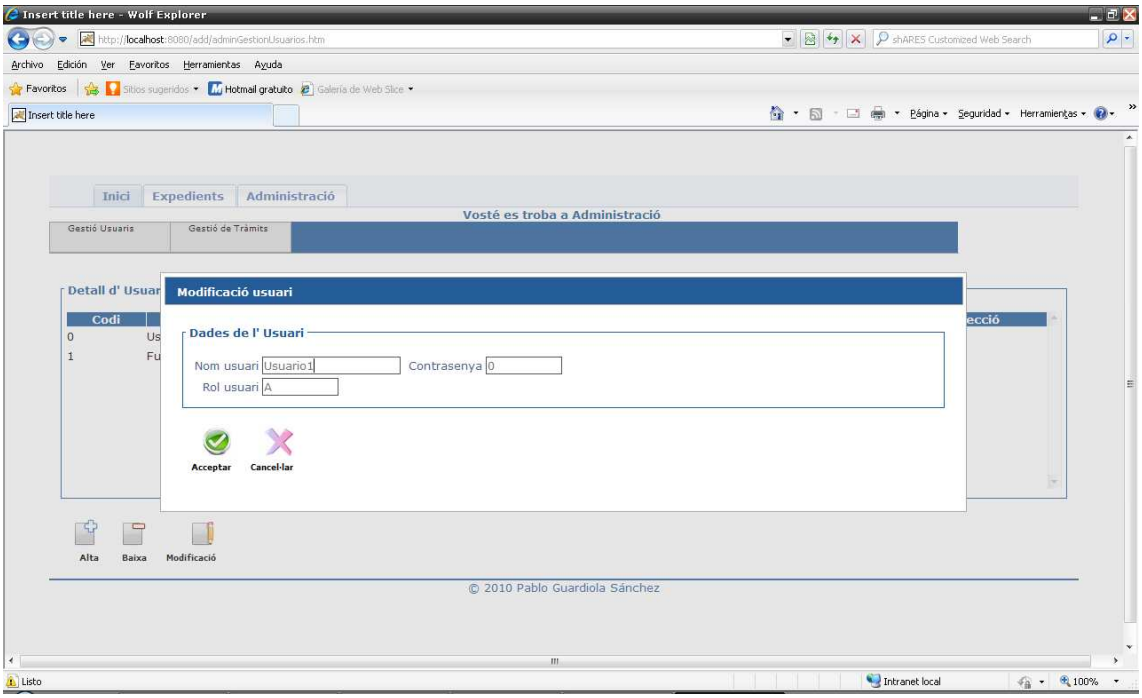


Figura 20. Diseño Diálogo Modificar Usuario

5.2.9. PANTALLA GESTIÓN DE TRÁMITES

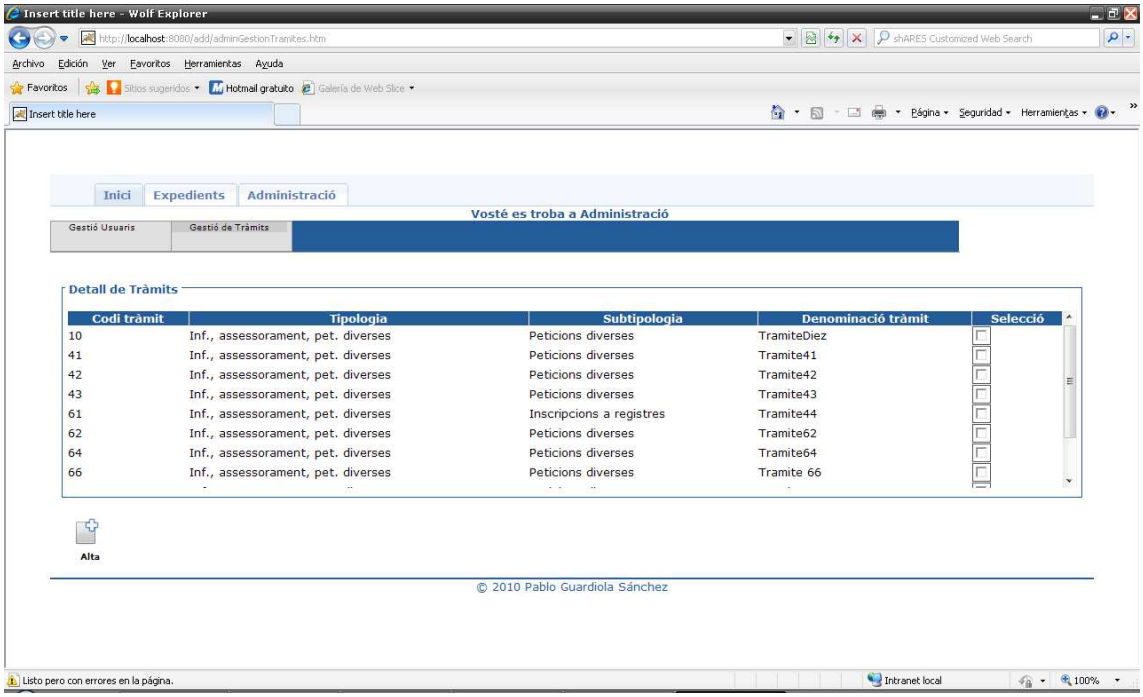


Figura 21. Diseño Pantalla Gestión Trámites

5.2.9.1. DIÁLOGO ALTA DE TRÁMITE

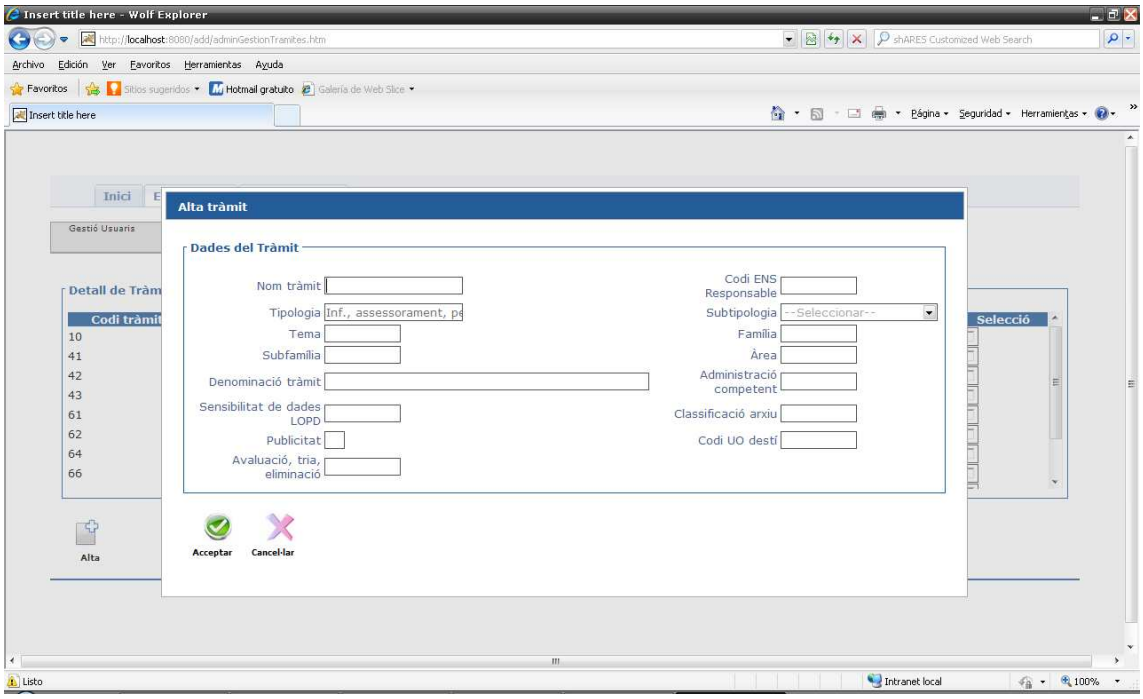


Figura 22. Diseño Diálogo Alta Trámite

5.2.9.2. DIÁLOGO MODIFICAR TRÁMITE

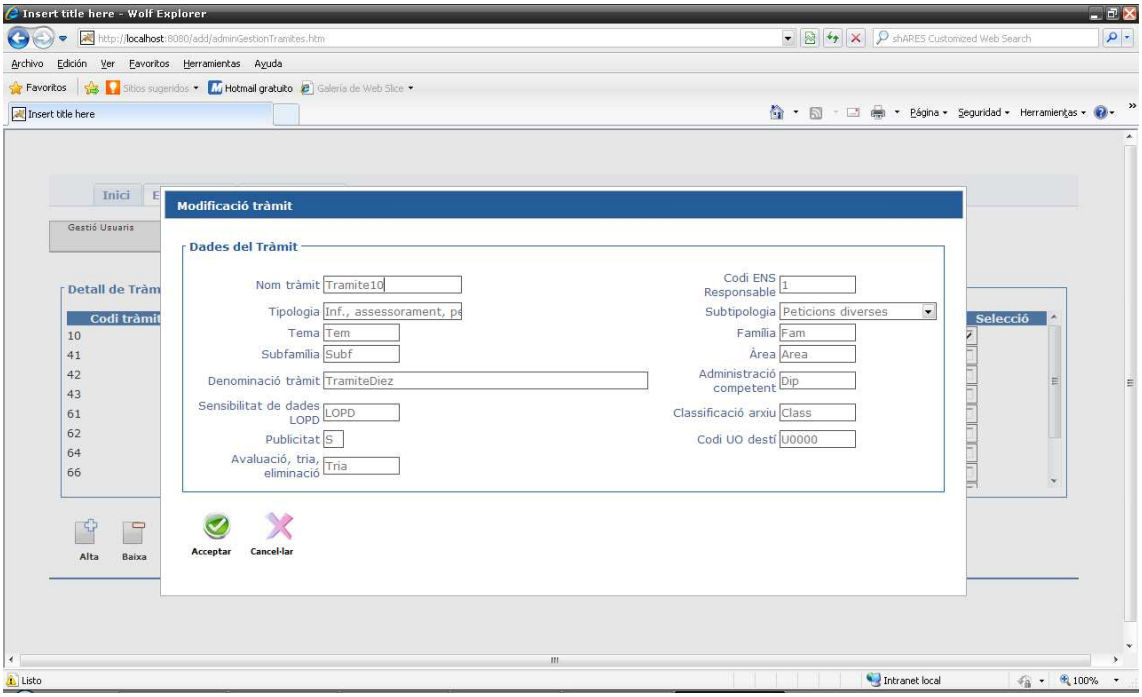


Figura 23. Diseño Diálogo Modificar Trámite

5.3. Código de ejemplo

Seguidamente se mostrará un diagrama de alto nivel sobre el funcionamiento general de la aplicación. A partir de este se mostrará un ejemplo de implementación de cada una de las capas.

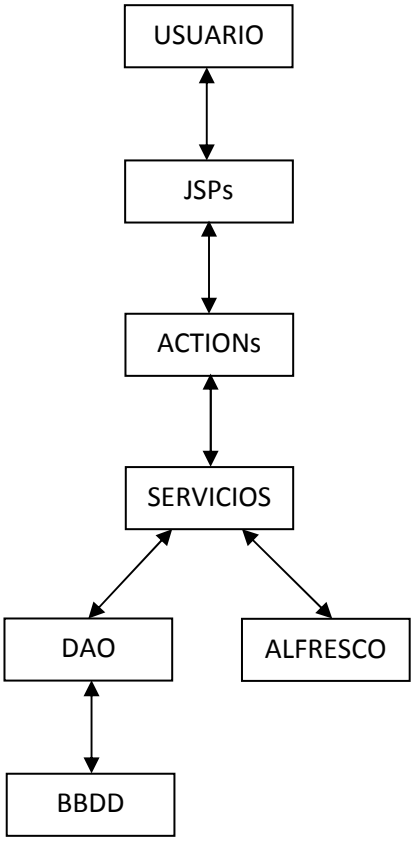


Figura 24. Funcionamiento general de las capas

5.3.1. JSP

Código ejemplo de la “Pantalla Gestión Trámites”.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c"  uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<script type="text/javascript"
src="/add/js/jquery1.4.4.min.js"></script>
<script type="text/javascript" src="/add/js/jquery-
ui1.8.6.min.js"></script>
<script type="text/javascript"
src="/add/js/dialogoAdministracionGestionTramite.js"></script>
<link rel="stylesheet" type="text/css" href="/add/css/jquery-
ui1.8.6.css"/>
```

```
<link rel="stylesheet" media="screen" type="text/css"
href="../estilo.css"/>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>Insert title here</title>
</head>
<body>
<div id="gestion_tramites">
<form action="bajaTramitesAction.htm"  method="post">
<div id="admin_gestion_tramites" class="principal" >
<div class="datos" >
<br class="espacio"/>
            <br class="espacio"/>
            <br class="espacio"/>
            <fieldset id="Detalle">
            <legend class="txtFieldset">Detall de Tràmits</legend>
            <br class="espacio"/>
            <input type="hidden" name="tramiteseleccionado"
id="tramiteseleccionado" value="">
             <div class="datagridMediano">
             <table cellpadding="0" cellspacing="1" border="0"
class="tablesorter" id="myTable">
              <thead>
                <tr>
                  <th align="center">Codi tràmit</th>
                  <th  align="center">Tipologia</th>
                  <th  align="center">Subtipologia</th>
                  <th  align="center">Denominació tràmit</th>
                  <th  align="center">Selecció</th>
                </tr>
              </thead>
              <tbody>
                  <c:forEach var="tramiteresultado"
items="${requestScope.listaTramites}">
                  <tr id="${tramiteresultado.tramite.codigoTramite}">
                        <td class="codTramite"><c:out
value="${tramiteresultado.tramite.codigoTramite}" /></td>
                        <td><div class="descTipologia"><c:out
value="${tramiteresultado.tipologia.descTipologia}" /></div></td>
                        <td><c:out
value="${tramiteresultado.subtipologia.descSubtipologia}" /></td>
                        <td>
                              <div class="descTramite"><c:out
value="${tramiteresultado.tramite.descTramite}" /></div>
                              <div class="nombreTramite"><input
type="hidden" value="<c:out
value="${tramiteresultado.tramite.nombreTramite}" />"/></div>
                              <div class="codigoEntidad"><input
type="hidden" value="<c:out
value="${tramiteresultado.tramite.codEntidad}" />"/></div>
                              <div class="codTipologia"><input
type="hidden" value="<c:out
value="${tramiteresultado.tipologia.codTipologia}" />"/></div>
                              <div class="temaTramite"><input
type="hidden" value="<c:out value="${tramiteresultado.tramite.tema}"
/>"/></div>
                              <div class="familiaTramite"><input
type="hidden" value="<c:out
value="${tramiteresultado.tramite.familia}" />"/></div>
```

```
                              <div class="subfamiliaTramite"><input
type="hidden" value="<c:out
value="${tramiteresultado.tramite.subfamilia}" />"/></div>
                              <div class="areaTramite"><input
type="hidden" value="<c:out value="${tramiteresultado.tramite.area}"
/>"/></div>
                              <div
class="administracionCompetente"><input type="hidden" value="<c:out
value="${tramiteresultado.tramite.administracionComponente}"
/>"/></div>
                              <div class="nivelProteccion"><input
type="hidden" value="<c:out
value="${tramiteresultado.tramite.nivelProteccion}" />"/></div>
                              <div class="clasificacionArchivo"><input
type="hidden" value="<c:out
value="${tramiteresultado.tramite.clasificacionArchivo}" />"/></div>
                              <div class="publicidad"><input
type="hidden" value="<c:out
value="${tramiteresultado.tramite.publicidad}" />"/></div>
                              <div class="codigoUODestino"><input
type="hidden" value="<c:out
value="${tramiteresultado.tramite.codUOrganizativaDestino}"
/>"/></div>
                              <div class="evalSelElim"><input
type="hidden" value="<c:out
value="${tramiteresultado.tramite.evalSeleccionEliminacion}"
/>"/></div>
                              <div class="codSubtipologia"><input
type="hidden" value="<c:out
value="${tramiteresultado.subtipologia.codSubtipologia}" />"/></div>
                              <div class="tramiteEliminado"><input
type="hidden" value="<c:out
value="${tramiteresultado.tramite.eliminado}" />" /></div>
                         </td>
                         <td><input class="classCheck" type="checkbox"
name="cheke2" id="cheke2" onClick="anyCheck2(this.form)"
value="${tramiteresultado.tramite.codigoTramite}"/></td>
                  </tr>
                  </c:forEach>
              </tbody>
            </table>
            </div>
            </fieldset>
      </div>
</div>
<div class="botonera">
<table border="0" cellpadding="10">
           <tr>
            <td><img id="abrir_alta_tramite"
src="/add/iconos/Add_documento.png" alt="Permet donar d'alta un
tràmit"><br></td>
            <td>
              <div id="boton_baja">
                <input type="image" src="/add/iconos/Borrar.png"
alt="Permet donar de baixa un tràmit" value="submit" style="border:
0px"><br>
                <span style="color: #000000; font-
size:9px"><b>Baixa</b></span>
              </div>
            </td>
```

```
            <td>
              <div id="boton_modificar">
                <img id="abrir_modificacion_tramite"
src="/add/iconos/Modificar_documento.png" alt="Permet modificar un
tràmit"><br>
              </div>
            </td>
            </tr>
</table>
</div>
</form>
<!--  Dialogo modal creacion-->
<div id="dialogo_crear_tramite" title="Alta tràmit">
      <div class="datos">
      <fieldset id="datosTramite">
      <legend class="txtFieldset">Dades del Tràmit</legend>
     <br class="espacio"/>
      <table>
       <tr>
         <td>
             <label for="nombretramite" class="txtLabel">Nom tràmit
</label>
         </td>
         <td>
          <input type="text" name="nombretramite" id="nombretramite"
size="20" value="" maxlength="20"/>
         </td>
         <td>
         <label for="codigoentidad" class="txtLabel">Codi ENS
Responsable </label>
         </td>
         <td>
         <input type="text" name="codigoentidad" id="codigoentidad"
size="10" value="" maxlength="10"/>
         </td>
        </tr>
        <tr>
         <td>
         <label for="tipologia" class="txtLabel">Tipologia </label>
         </td>
         <td>
            <input type="text" name="desctipologia" id="desctipologia"
readonly="readonly" value="<c:out
value="${requestScope.tipologiaGestionable.descTipologia}" />"/>
           <input type="hidden" name="tipologia" id="tipologia"
value="<c:out
value="${requestScope.tipologiaGestionable.codTipologia}" />"/>
         </td>
         <td>
            <label for="subtipologia" class="txtLabel">Subtipologia
</label>
         </td>
         <td>
<select id="subtipologia" name="subtipologia">
                  <option selected="selected" disabled="disabled">--
Seleccionar--</option>
                  <c:forEach
items="${requestScope.listaSubtipologiasGestionables}"
var="subtipologiagestionable">
```

```
                    <option
value="${subtipologiagestionable.codSubtipologia}"><c:out
value="${subtipologiagestionable.descSubtipologia}"/></option>
                    </c:forEach>
</select>
        </td>
       </tr>
       <tr>
         <td>
         <label for="tema" class="txtLabel">Tema </label>
         </td>
         <td>
         <input type="text" name="tema" id="tema" size="10" value=""
maxlength="10"/>
         </td>
         <td>
          <label for="familia" class="txtLabel">Família </label>
         </td>
         <td>
         <input type="text" name="familia" id="familia" size="10"
value="" maxlength="10"/>
         </td>
       </tr>
       <tr>
         <td>
          <label for="subfamilia" class="txtLabel">Subfamília </label>
         </td>
         <td>
         <input type="text" name="subfamilia" id="subfamilia"
size="10" value="" maxlength="10"/>
         </td>
         <td>
         <label for="area" class="txtLabel">Àrea </label>
         </td>
         <td>
         <input type="text" name="area" id="area" size="10" value=""
maxlength="10"/>
         </td>
       </tr>
       <tr>
       <td>
       <label for="denominaciontramite" class="txtLabel">Denominació
tràmit </label>
         </td>
         <td>
         <input type="text" name="denominaciontramite"
id="denominaciontramite" size="50" value="" maxlength="50"/>
         </td>
         <td>
         <label for="administracioncompetente"
class="txtLabel">Administració competent </label>
         </td>
         <td>
           <input type="text" name="administracioncompetente"
id="administracioncompetente" size="10" value="" maxlength="10"/>
         </td>
         </tr>
         <tr>
         <td>
```

```
          <label for="sensibilidaddatosLOPD"
class="txtLabel">Sensibilitat de dades LOPD </label>
          </td>
          <td>
           <input type="text" name="sensibilidaddatosLOPD"
id="sensibilidaddatosLOPD" size="10" value="" maxlength="10"/>
          </td>
          <td>
          <label for="clasificacionarchivo"
class="txtLabel">Classificació arxiu </label>
          </td>
          <td>
          <input type="text" name="clasificacionarchivo"
id="clasificacionarchivo" size="10" value="" maxlength="10"/>
          </td>
          </tr>
          <tr>
          <td>
           <label for="publicidad" class="txtLabel">Publicitat </label>
          </td>
          <td>
          <input type="text" name="publicidad" id="publicidad" size="1"
value="" maxlength="1"/>
          </td>
          <td>
          <label for="codigouodestino" class="txtLabel">Codi UO destí
</label>
          </td>
          <td>
          <input type="text" name="codigouodestino"
id="codigouodestino" size="10" value="" maxlength="10"/>
          </td>
          </tr>
          <tr>
          <td>
          <label for="evalselelim" class="txtLabel">Avaluació, tria,
eliminació </label>
          </td>
          <td>
           <input type="text" name="evalselelim" id="evalselelim"
size="10" value="" maxlength="10"/>
          </td>
          <td>
          </td>
          <td>
          </td>
          </tr>
        </table>
</fieldset>
</div>
<div class="botonera">
<table border="0" cellpadding="10">
            <tr>
              <td>
                <img id="aceptar_tramite" src="/add/iconos/Aceptar.jpg"
alt="Accepta" ><br>
                  <span style="color: #000000; font-
size:9px"><b>Acceptar</b></span>
              </td>
              <td>
```

```
                  <img id="cancelar_tramite"
src="/add/iconos/cancelar.jpg" alt="Tanca la finestra" ><br>
                  <span style="color: #000000; font-
size:9px"><b>Cancel•lar</b></span>
              </td>
             </tr>
      </table>
      <br class="espacio"/>
    </div>
</div>
<!--  Dialogo modal modificacion -->
<div id="dialogo_modificacion_tramite" title="Modificació tràmit">
<div class="datos">
            <fieldset id="moddatosTramite">
            <legend class="txtFieldset">Dades del Tràmit</legend>
            <br class="espacio"/>
            <table>
            <tr>
            <td>
             <label for="modnombretramite" class="txtLabel">Nom tràmit
</label>
            </td>
            <td>
            <input type="text" name="modnombretramite"
id="modnombretramite" size="20" value="" maxlength="20"/>
            <input type="hidden" name="modcodtramite"
id="modcodtramite" value=""/>
            </td>
            <td>
            <label for="modcodigoentidad" class="txtLabel">Codi ENS
Responsable </label>
            </td>
            <td>
            <input type="text" name="modcodigoentidad"
id="modcodigoentidad" size="10" value="" maxlength="10"/>
            </td>
            </tr>
            <tr>
            <td>
            <label for="modtipologia" class="txtLabel">Tipologia
</label>
            </td>
            <td>
            <input type="text" name="moddesctipologia"
id="moddesctipologia" readonly="readonly" value=""/>
            <input type="hidden" name="modtipologia" id="modtipologia"
value=""/>
            </td>
            <td>
             <label for="modsubtipologia" class="txtLabel">Subtipologia
</label>
            </td>
            <td>
            <select id="modsubtipologia" name="modsubtipologia">
                  <option selected="selected" disabled="disabled">--
Seleccionar--</option>
                        <c:forEach
items="${requestScope.listaSubtipologiasGestionables}"
var="subtipologiagestionable">
```

```
                                <option
value="${subtipologiagestionable.codSubtipologia}"><c:out
value="${subtipologiagestionable.descSubtipologia}"/></option>
                            </c:forEach>
            </select>
            </td>
            </tr>
            <tr>
            <td>
            <label for="modtema" class="txtLabel">Tema </label>
            </td>
            <td>
            <input type="text" name="modtema" id="modtema" size="10"
value="" maxlength="10"/>
            </td>
            <td>
            <label for="modfamilia" class="txtLabel">Família </label>
            </td>
            <td>
            <input type="text" name="modfamilia" id="modfamilia"
size="10" value="" maxlength="10"/>
            </td>
            </tr>
            <tr>
            <td>
            <label for="modsubfamilia" class="txtLabel">Subfamília
</label>
            </td>
            <td>
            <input type="text" name="modsubfamilia" id="modsubfamilia"
size="10" value="" maxlength="10"/>
            </td>
            <td>
            <label for="modarea" class="txtLabel">Àrea </label>
            </td>
            <td>
            <input type="text" name="modarea" id="modarea" size="10"
value="" maxlength="10"/>
            </td>
            </tr>
            <tr>
            <td>
            <label for="moddenominaciontramite"
class="txtLabel">Denominació tràmit </label>
            </td>
            <td>
            <input type="text" name="moddenominaciontramite"
id="moddenominaciontramite" size="50" value="" maxlength="50"/>
            </td>
            <td>
            <label for="modadministracioncompetente"
class="txtLabel">Administració competent </label>
            </td>
            <td>
            <input type="text" name="modadministracioncompetente"
id="modadministracioncompetente" size="10" value="" maxlength="10"/>
            </td>
            </tr>
            <tr>
            <td>
```

```
            <label for="modsensibilidaddatosLOPD"
class="txtLabel">Sensibilitat de dades LOPD </label>
            </td>
            <td>
            <input type="text" name="modsensibilidaddatosLOPD"
id="modsensibilidaddatosLOPD" size="10" value="" maxlength="10"/>
            </td>
            <td>
            <label for="modclasificacionarchivo"
class="txtLabel">Classificació arxiu </label>
            </td>
            <td>
            <input type="text" name="modclasificacionarchivo"
id="modclasificacionarchivo" size="10" value="" maxlength="10"/>
            </td>
            </tr>
            <tr>
            <td>
            <label for="modpublicidad" class="txtLabel">Publicitat
</label>
            </td>
            <td>
            <input type="text" name="modpublicidad" id="modpublicidad"
size="1" value="" maxlength="1"/>
            </td>
            <td>
            <label for="modcodigouodestino" class="txtLabel">Codi UO
destí </label>
            </td>
            <td>
            <input type="text" name="modcodigouodestino"
id="modcodigouodestino" size="10" value="" maxlength="10"/>
            </td>
            </tr>
            <tr>
            <td>
            <label for="modevalselelim" class="txtLabel">Avaluació,
tria, eliminació </label>
            </td>
            <td>
            <input type="text" name="modevalselelim"
id="modevalselelim" size="10" value="" maxlength="10"/>
            </td>
            <td>
            </td>
            <td>
            </td>
            </tr>
       </table>
       </fieldset>
</div>
<div class="botonera">
<table border="0" cellpadding="10">
            <tr>
             <td>
                  <img id="aceptar_modificacion_tramite"
src="/add/iconos/Aceptar.jpg" alt="Accepta" ><br>
                                  <span style="color: #000000; font-
size:9px"><b>Acceptar</b></span>
            </td>
```

```
            <td>
                  <img id="cancelar_modificacion_tramite"
src="/add/iconos/cancelar.jpg" alt="Tanca la finestra"><br/>
                              <span style="color: #000000; font-
size:9px"><b>Cancel•lar</b></span>
            </td>
            </tr>
      </table>
      <br class="espacio"/>
</div>
</div>
</div>
</body>
</html>
```

Ejemplo 4.adminGestionTramites.jsp

5.3.2. ACTION

Código ejemplo de la clase “AdministracionGestionTramitesAction”.

```
package es.alten.diputacion.add.web.struts;
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.log4j.Logger;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import es.alten.diputacion.add.modelo.Subtipologia;
import es.alten.diputacion.add.modelo.Tipologia;
import
es.alten.diputacion.add.modelo.resultado.TramiteConTipologiaGestionabl
eResultado;
import es.alten.diputacion.add.modelo.struts.BaseForm;
import
es.alten.diputacion.add.servicios.AdministracionTramiteServicio;
public class AdministracionGestionTramitesAction extends BaseAction{
      Logger log =
Logger.getLogger(AdministracionGestionTramitesAction.class);
      private AdministracionTramiteServicio
administracionTramiteServicio;
      @Override
      protected void delegarError(BaseForm objetoModelo,
                  HttpServletRequest request, HttpServletResponse
response) {
            // TODO Auto-generated method stub
      }
      @Override
      protected ActionForward procesar(ActionMapping mapping,
                  ActionForm objetoModelo, HttpServletRequest request,
                  HttpServletResponse response) {

            try {
                  List<TramiteConTipologiaGestionableResultado>
```

```
listaTramites =
administracionTramiteServicio.obtenerTramitesConTipologiaGestionable()
;
                  request.setAttribute("listaTramites", listaTramites);
                  Tipologia tipologiaGestionable =
administracionTramiteServicio.obtenerTipologiaGestionable();
                  request.setAttribute("tipologiaGestionable",
tipologiaGestionable);
                  List<Subtipologia> listaSubtipologiasGestionables =
administracionTramiteServicio.obtenerSubtipologiasporTipologiaGestiona
ble();
                  for(Subtipologia subtipologiagest :
listaSubtipologiasGestionables){
            log.info("##########Descripcion:"+
subtipologiagest.getDescSubtipologia());
                  }
            request.setAttribute("listaSubtipologiasGestionables",
listaSubtipologiasGestionables);
                  log.info("############################  OK");
                  return mapping.findForward("actionOK") ;
            } catch (Exception e) {
                  log.error("############################  ERROR",e);
                  // TODO Auto-generated catch block
                  return mapping.findForward("error") ;
            }
      }
      public void setAdministracionTramiteServicio(
                  AdministracionTramiteServicio
administracionTramiteServicio) {
            this.administracionTramiteServicio =
administracionTramiteServicio;
      }
      public AdministracionTramiteServicio
getAdministracionTramiteServicio() {
            return administracionTramiteServicio;
      }
}
```

Ejemplo 5. AdministracionGestionTramitesAction.java

5.3.3. SERVICIO

Código de ejemplo de la clase “AdministracionTramitesServicioImpl”.

```
package es.alten.diputacion.add.servicios.impl;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import org.springframework.transaction.annotation.Transactional;
import es.alten.diputacion.add.abstraccion.SubtipologiaDao;
import es.alten.diputacion.add.abstraccion.TipologiaDao;
import es.alten.diputacion.add.abstraccion.TramiteDao;
import es.alten.diputacion.add.modelo.Subtipologia;
import es.alten.diputacion.add.modelo.Tipologia;
import es.alten.diputacion.add.modelo.Tramite;
```

```
import
es.alten.diputacion.add.modelo.resultado.TramiteConTipologiaGestionabl
eResultado;
import
es.alten.diputacion.add.servicios.AdministracionTramiteServicio;
@Transactional()
public class AdministracionTramitesServicioImpl implements
AdministracionTramiteServicio {
	private TramiteDao tramiteDAO;
	private SubtipologiaDao subtipologiaDAO;
	private TipologiaDao tipologiaDAO;
	public TramiteDao getTramiteDAO() {
		return tramiteDAO;
	}
	public void setTramiteDAO(TramiteDao tramiteDAO) {
		this.tramiteDAO = tramiteDAO;
	}
	public SubtipologiaDao getSubtipologiaDAO() {
		return subtipologiaDAO;
	}
	public void setSubtipologiaDAO(SubtipologiaDao subtipologiaDAO)
{
		this.subtipologiaDAO = subtipologiaDAO;
	}
	public TipologiaDao getTipologiaDAO() {
		return tipologiaDAO;
	}
	public void setTipologiaDAO(TipologiaDao tipologiaDAO) {
		this.tipologiaDAO = tipologiaDAO;
	}
	@Override
	public List<TramiteConTipologiaGestionableResultado>
obtenerTramitesConTipologiaGestionable() {
		List
<TramiteConTipologiaGestionableResultado>listaResultadoTramitesGestion
ables = new ArrayList<TramiteConTipologiaGestionableResultado>();
		List <Tramite> tramites =
tramiteDAO.consultarTramitesTipologiaGestionable();
		for (Tramite tramite : tramites){
			TramiteConTipologiaGestionableResultado
tramiteGestionableResultado = new
TramiteConTipologiaGestionableResultado();
			Subtipologia subtipologia =
subtipologiaDAO.consultaSubtipologiaPorTramite(
tramite.getCodSubtipologia());
		tramiteGestionableResultado.setSubtipologia(subtipologia);
			Tipologia tipologia =
tipologiaDAO.consultaTipologiaPorSubtipologia(
subtipologia.getCodTipologia());
			tramiteGestionableResultado.setTipologia(tipologia);
			tramiteGestionableResultado.setTramite(tramite);
	listaResultadoTramitesGestionables.add(
tramiteGestionableResultado);
		}
		Collections.sort(listaResultadoTramitesGestionables);
		return listaResultadoTramitesGestionables;
	}
	@Override
	public Tipologia obtenerTipologiaGestionable() {
```

```
            Tipologia tipologiagestionable =
tipologiaDAO.selectTipologiaGestionable();
            return tipologiagestionable;
      }
      @Override
      public List<Subtipologia>
obtenerSubtipologiasporTipologiaGestionable() {
            List<Subtipologia> subtipologiasgestionables =
subtipologiaDAO.selectSubtipologiaPorTipologiaGestionable(
tipologiaDAO.selectTipologiaGestionable().getCodTipologia());
            return subtipologiasgestionables;
      }
      @Override
      public TramiteConTipologiaGestionableResultado
altaTramite(Tramite tramite) {
            Tramite tramiteResultadoDAO = new Tramite();
            tramiteResultadoDAO = tramiteDAO.insertTramite(tramite);
            TramiteConTipologiaGestionableResultado tramiteResultado =
new TramiteConTipologiaGestionableResultado();
            tramiteResultado.setTramite(tramiteResultadoDAO);
            Subtipologia subtipologia =
subtipologiaDAO.consultaSubtipologiaPorTramite(
tramiteResultadoDAO.getCodSubtipologia());
            tramiteResultado.setSubtipologia(subtipologia);
            Tipologia tipologia =
tipologiaDAO.consultaTipologiaPorSubtipologia(
subtipologia.getCodTipologia());
            tramiteResultado.setTipologia(tipologia);
            return tramiteResultado;
      }
      @Override
      public TramiteConTipologiaGestionableResultado
modificacionTramite(Tramite tramite) {
            Tramite tramiteResultadoDAO = new Tramite();
            tramiteResultadoDAO = tramiteDAO.updateTramite(tramite);
            TramiteConTipologiaGestionableResultado tramiteResultado =
new TramiteConTipologiaGestionableResultado();
            tramiteResultado.setTramite(tramiteResultadoDAO);
            Subtipologia subtipologia =
subtipologiaDAO.consultaSubtipologiaPorTramite(
tramiteResultadoDAO.getCodSubtipologia());
            tramiteResultado.setSubtipologia(subtipologia);
            Tipologia tipologia =
tipologiaDAO.consultaTipologiaPorSubtipologia(
subtipologia.getCodTipologia());
            tramiteResultado.setTipologia(tipologia);
            return tramiteResultado;
      }
      @Override
      public Tramite buscarTramitePorCodigoTramite(String id_tramite)
{
            return tramiteDAO.buscarTramitePorIdTramite(id_tramite);
      }
      @Override
      public void borrarTramitePorCodigoTramite(String idTramite) {
            tramiteDAO.deleteTramite(idTramite);
      }
}
```

Ejemplo 6. AdministracionTramitesServicioImpl.java

5.3.4. DAO

Código de ejemplo de la clase “TramiteDaoImpl”.

```
package es.alten.diputacion.add.abstraccion.jdbc;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import java.util.Map;
import org.apache.log4j.Logger;
import org.springframework.dao.DataAccessException;
import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import es.alten.diputacion.add.abstraccion.TramiteDao;
import es.alten.diputacion.add.modelo.Tramite;
import es.alten.diputacion.add.modelo.TramiteExpandido;
import es.alten.diputacion.add.util.AddJdbcUtils;
import es.alten.diputacion.add.util.SecuenciadorProvicional;
import es.alten.diputacion.add.util.TipoCondicion;
public class TramiteDaoImpl implements TramiteDao{
      Logger log = Logger.getLogger(TramiteDaoImpl.class);
      public static final String INSERT_SQL="INSERT INTO ADD_TRAMITE
(STR_CODTRAMITE, STR_NOMBRETRAMITE, STR_DESCTRAMITE, " +
                  "STR_CODSUBTIPOLOGIA, STR_TEMA, STR_FAMILIA,
STR_SUBFAMILIA, STR_AREA, STR_ADMINISTRACIONCOMPONENTE, " +
                  "STR_NIVELPROTECDATOSPERSONALES,
STR_CLASIFICACIONARCHIVO, IND_PUBLICIDAD,
STR_EVALSELECCIONELIMINACION, " +
                  "STR_CODENTIDAD, STR_CODUORGANIZATIVADESTINO,
FEC_FECHAREGISTRO, FEC_FECHABAJA, IND_ELIMINADO) " +
                  "VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
      private JdbcTemplate jdbcTemplate;
      @SuppressWarnings("unchecked")
      public List<Tramite> consultarTramitesUsuario(String codUsuario)
{
            return this.jdbcTemplate.query("SELECT * FROM ADD_TRAMITE,
" +
                        "ADD_USUARIO_TRAMITE WHERE
ADD_USUARIO_TRAMITE.STR_CODUSUARIO = '" + codUsuario +
                        "'AND
ADD_USUARIO_TRAMITE.STR_CODTRAMITE=ADD_TRAMITE.STR_CODTRAMITE AND
ADD_TRAMITE.IND_ELIMINADO='N'" , new RowMapperTramite());
      }
      public Tramite consultaTramitePorCodigo(String elegido) {
            return (Tramite) this.jdbcTemplate.queryForObject("SELECT *
FROM ADD_TRAMITE WHERE STR_CODTRAMITE=? AND IND_ELIMINADO='N'" , new
Object[] { elegido }, new RowMapperTramite ());
      }
      @SuppressWarnings("unchecked")
      public List<Tramite> consultarTramitesTipologiaGestionable() {
            return this.jdbcTemplate.query("SELECT ADD_TRAMITE.* FROM
ADD_TRAMITE, ADD_SUBTIPOLOGIA, ADD_TIPOLOGIA " +
                        "WHERE ADD_TIPOLOGIA.STR_GESTIONABLE = 'S' AND
ADD_TIPOLOGIA.STR_CODTIPOLOGIA=ADD_SUBTIPOLOGIA.STR_CODTIPOLOGIA " +
```

```
                        "AND
ADD_SUBTIPOLOGIA.STR_CODSUBTIPOLOGIA=ADD_TRAMITE.STR_CODSUBTIPOLOGIA
AND ADD_TRAMITE.IND_ELIMINADO='N'" , new RowMapperTramite());
      }
      private class RowMapperTramite implements RowMapper {
            public Object mapRow(ResultSet rs, int rowNum) throws
SQLException {
                  Tramite tramite = new Tramite();
            tramite.setCodigoTramite(rs.getString("STR_CODTRAMITE"));
      tramite.setNombreTramite(rs.getString("STR_NOMBRETRAMITE"));
            tramite.setDescTramite(rs.getString("STR_DESCTRAMITE"));
      tramite.setCodSubtipologia(rs.getString("STR_CODSUBTIPOLOGIA"));
            tramite.setTema(rs.getString("STR_TEMA"));
            tramite.setFamilia(rs.getString("STR_FAMILIA"));
            tramite.setSubfamilia(rs.getString("STR_SUBFAMILIA"));
            tramite.setArea(rs.getString("STR_AREA"));
      tramite.setAdministracionComponente(
rs.getString("STR_ADMINISTRACIONCOMPONENTE"));
      tramite.setNivelProteccion(
rs.getString("STR_NIVELPROTECDATOSPERSONALES"));
            tramite.setPublicidad(rs.getString("IND_PUBLICIDAD"));
      tramite.setEvalSeleccionEliminacion(
rs.getString("STR_EVALSELECCIONELIMINACION"));
            tramite.setCodEntidad(rs.getString("STR_CODENTIDAD"));
      tramite.setCodUOrganizativaDestino(
rs.getString("STR_CODUORGANIZATIVADESTINO"));
      tramite.setClasificacionArchivo(
rs.getString("STR_CLASIFICACIONARCHIVO"));
            tramite.setFechaBaja(rs.getDate("FEC_FECHABAJA"));
            tramite.setFechaRegistro(rs.getDate("FEC_FECHAREGISTRO"));
            tramite.setEliminado(rs.getString("IND_ELIMINADO"));
            return tramite;
            }
      }
      @SuppressWarnings("unchecked")
      public List<Tramite> consultarTodasTramites() {
                  return this.jdbcTemplate.query( "SELECT * FROM
ADD_TRAMITE " +
                              "WHERE ADD_TRAMITE.IND_ELIMINADO = 'N'
ORDER BY STR_NOMBRETRAMITE ASC", new RowMapperTramite());
      }
      public JdbcTemplate getJdbcTemplate() {
            return jdbcTemplate;
      }
      public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
            this.jdbcTemplate = jdbcTemplate;
      }
      public Tramite insertTramite(Tramite tramite) {
            try {
                  int codigoTramite =
SecuenciadorProvicional.getSecuencia();
                  String codTramite = new
Integer(codigoTramite).toString();
                  String sql="INSERT INTO ADD_TRAMITE (STR_CODTRAMITE,
STR_NOMBRETRAMITE, STR_DESCTRAMITE, STR_CODSUBTIPOLOGIA, " +
                              "STR_TEMA, STR_FAMILIA, STR_SUBFAMILIA,
STR_AREA, STR_ADMINISTRACIONCOMPONENTE,
STR_NIVELPROTECDATOSPERSONALES, " +
```

```
                              "STR_CLASIFICACIONARCHIVO,
IND_PUBLICIDAD, STR_EVALSELECCIONELIMINACION, STR_CODENTIDAD,
STR_CODUORGANIZATIVADESTINO, " +
                              "FEC_FECHAREGISTRO, FEC_FECHABAJA,
IND_ELIMINADO) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)";
                  Object[] parametros = new Object[]{
                              codigoTramite,
                              tramite.getNombreTramite(),
                              tramite.getDescTramite(),
                              tramite.getCodSubtipologia(),
                              tramite.getTema(),
                              tramite.getFamilia(),
                              tramite.getSubfamilia(),
                              tramite.getArea(),
                              tramite.getAdministracionComponente(),
                              tramite.getNivelProteccion(),
                              tramite.getClasificacionArchivo(),
                              tramite.getPublicidad(),
                              tramite.getEvalSeleccionEliminacion(),
                              tramite.getCodEntidad(),
                              tramite.getCodUOrganizativaDestino(),
                              tramite.getFechaRegistro(),
                              tramite.getFechaBaja(),
                              tramite.getEliminado()};
                  jdbcTemplate.update(sql,parametros);
                  return buscarTramitePorIdTramite(codTramite);
            } catch (DataAccessException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
                  return null;
            }
      }
      public Tramite buscarTramitePorIdTramite(String id_tramite) {
            try {
                  return
(Tramite)this.jdbcTemplate.queryForObject("SELECT * FROM ADD_TRAMITE "
+
                        " WHERE STR_CODTRAMITE = ? AND
ADD_TRAMITE.IND_ELIMINADO='N'", new Object[] { id_tramite }, new
RowMapperTramite());
            } catch (EmptyResultDataAccessException e) {
                  return null;
            }
      }
      @SuppressWarnings("unchecked")
      public List<Tramite> consultaTramitesByCodigoSubtipologia(String
codSubtipologia) {
            return this.jdbcTemplate.query("SELECT * FROM ADD_TRAMITE "
+
                        "WHERE STR_CODSUBTIPOLOGIA = ?" +
                        " ORDER BY STR_CODTRAMITE ",
new Object []{codSubtipologia},new RowMapperTramite());
      }
      public Tramite updateTramite(Tramite tramite) {
            try {
                  String sql="UPDATE ADD_TRAMITE SET " +
                  "STR_CODTRAMITE=?," +
                  "STR_NOMBRETRAMITE=?," +
                  "STR_DESCTRAMITE=?," +
                  "STR_CODSUBTIPOLOGIA=?," +
```

```
                  "STR_TEMA=?," +
                  "STR_FAMILIA=?," +
                  "STR_SUBFAMILIA=?," +
                  "STR_AREA=?," +
                  "STR_ADMINISTRACIONCOMPONENTE=?," +
                  "STR_NIVELPROTECDATOSPERSONALES=?," +
                  "STR_CLASIFICACIONARCHIVO=?," +
                  "IND_PUBLICIDAD=?," +
                  "STR_EVALSELECCIONELIMINACION=?," +
                  "STR_CODENTIDAD=?," +
                  "STR_CODUORGANIZATIVADESTINO=?," +
                  "FEC_FECHAREGISTRO=?," +
                  "FEC_FECHABAJA=?," +
                  "IND_ELIMINADO=?" +
                  " WHERE STR_CODTRAMITE=?";
                  Object[] parametros = new Object[]{
                              tramite.getCodigoTramite(),
                              tramite.getNombreTramite(),
                              tramite.getDescTramite(),
                              tramite.getCodSubtipologia(),
                              tramite.getTema(),
                              tramite.getFamilia(),
                              tramite.getSubfamilia(),
                              tramite.getArea(),
                              tramite.getAdministracionComponente(),
                              tramite.getNivelProteccion(),
                              tramite.getClasificacionArchivo(),
                              tramite.getPublicidad(),
                              tramite.getEvalSeleccionEliminacion(),
                              tramite.getCodEntidad(),
                              tramite.getCodUOrganizativaDestino(),
                              tramite.getFechaRegistro(),
                              tramite.getFechaBaja(),
                              tramite.getEliminado(),
                              tramite.getCodigoTramite()};
                  jdbcTemplate.update(sql,parametros);
            }
            catch(DataAccessException e){
                  e.getMessage();
                  e.getClass().getName();
            }
            String codTram=tramite.getCodigoTramite();
            return buscarTramitePorIdTramite(codTram);
      }
      public void deleteTramite(String idTramite) {
            try {
                  this.jdbcTemplate.update("UPDATE ADD_TRAMITE SET
IND_ELIMINADO='S', FEC_FECHABAJA=SYSDATE() WHERE STR_CODTRAMITE=?",
new Object[]{idTramite});
            } catch (DataAccessException e) {
                  e.printStackTrace();
            }
      }
}
```

Ejemplo 7. TramiteDaoImpl.java

6. Resultados

En este apartado vamos a mostrar un ejemplo de ejecución del funcionamiento de nuestra aplicación a través de capturas de la misma.

6.1. Preparación

El primer paso consiste en arrancar el programa *XAMPP*. Este programa combina muchos paquetes en uno sólo. Particularmente nos interesan dos, el servidor web Apache y MySQL para gestionar la base de datos con phpMyAdmin.

Figura 25. XAMPP

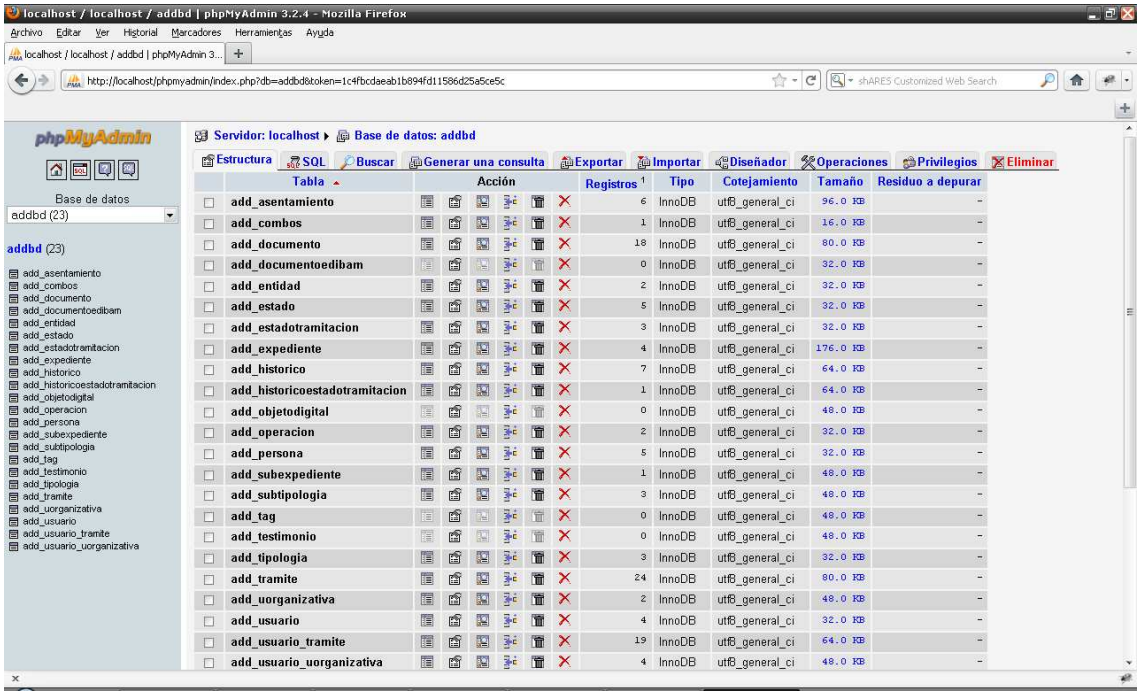


Figura 26. phpMyAdmin

Tras esto, se debe arrancar el contenedor de servlets y JSPs, Tomcat, con el siguiente comando:

\apache-tomcat-6.0.29\bin> catalina.bat jpda start

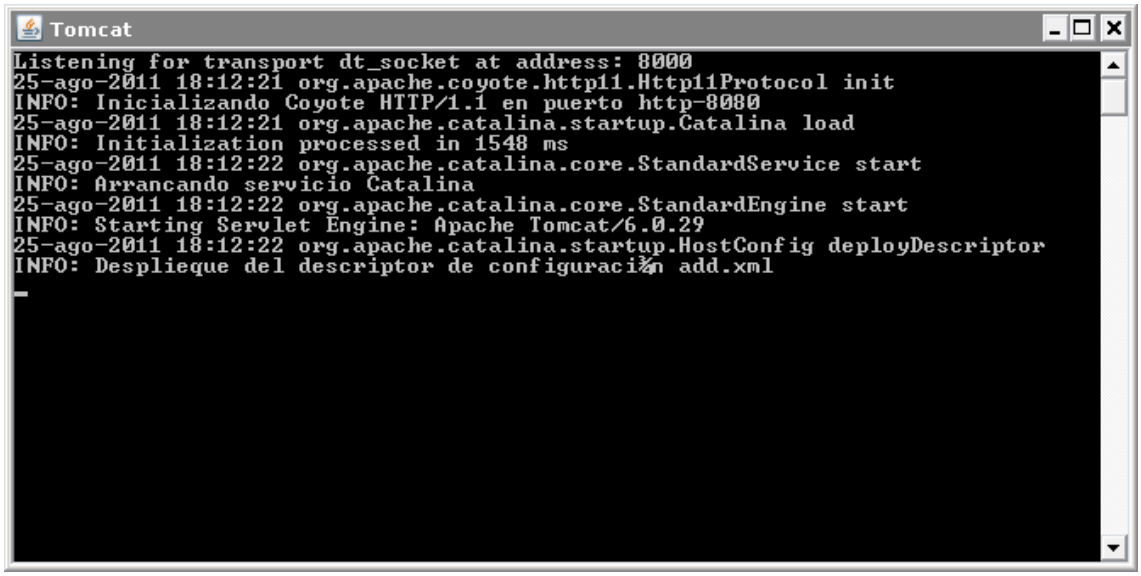


Figura 27. Tomcat

Con esto Tomcat despliega los .war contenidos en la carpeta \apache-tomcat-6.0.29\webapps y arranca, las respectivas aplicaciones.

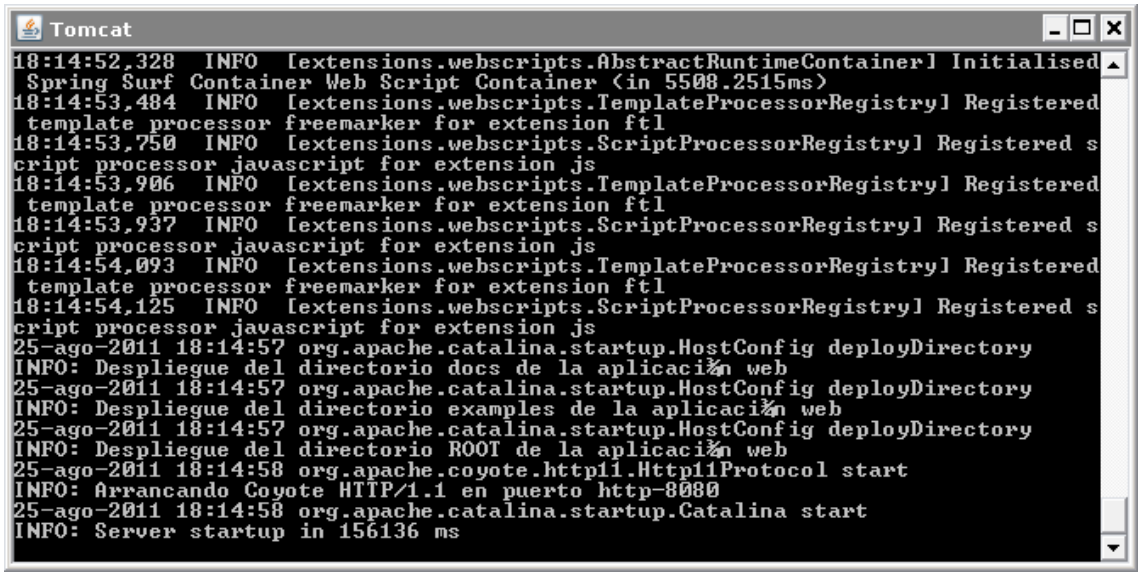


Figura 28. Despliegue aplicaciones sobre Tomcat

6.2. Ejecución

Una vez aquí, se debe abrir el navegador Explorer y escribir en la barra de direcciones http://localhost:8080/add para arrancar nuestra aplicación.

Tras esto, en primer lugar, aparece la “Pantalla Autenticación”.

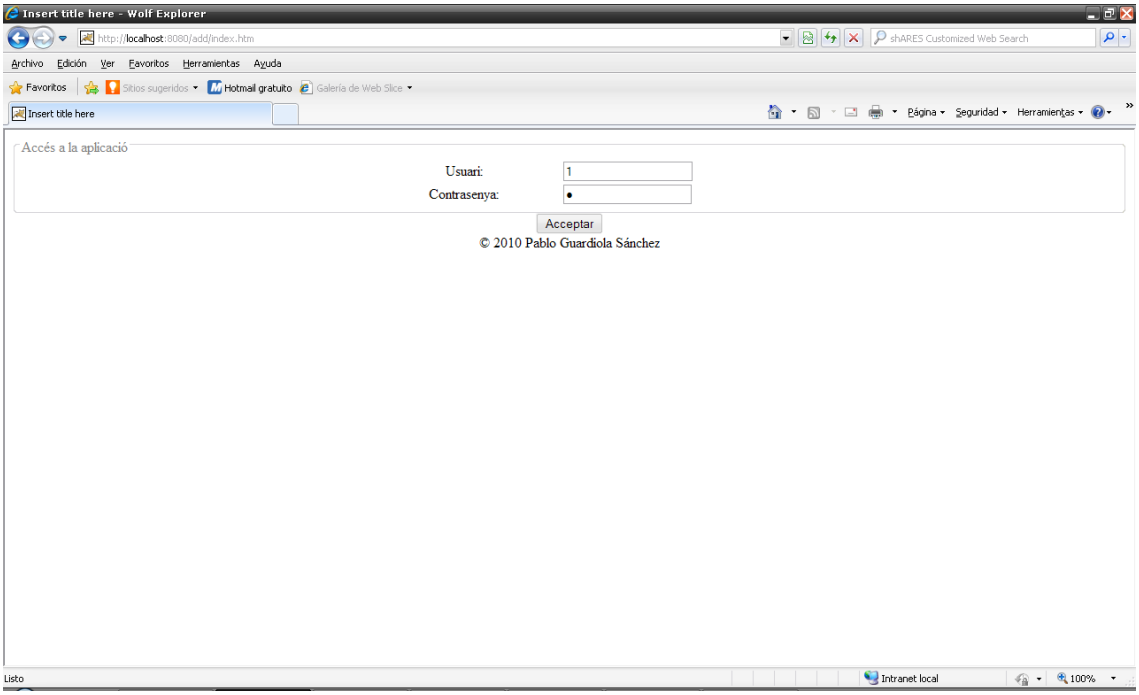


Figura 29. Pantalla Autenticación

Si la contraseña no es correcta, aparece un mensaje de error indicándonoslo.

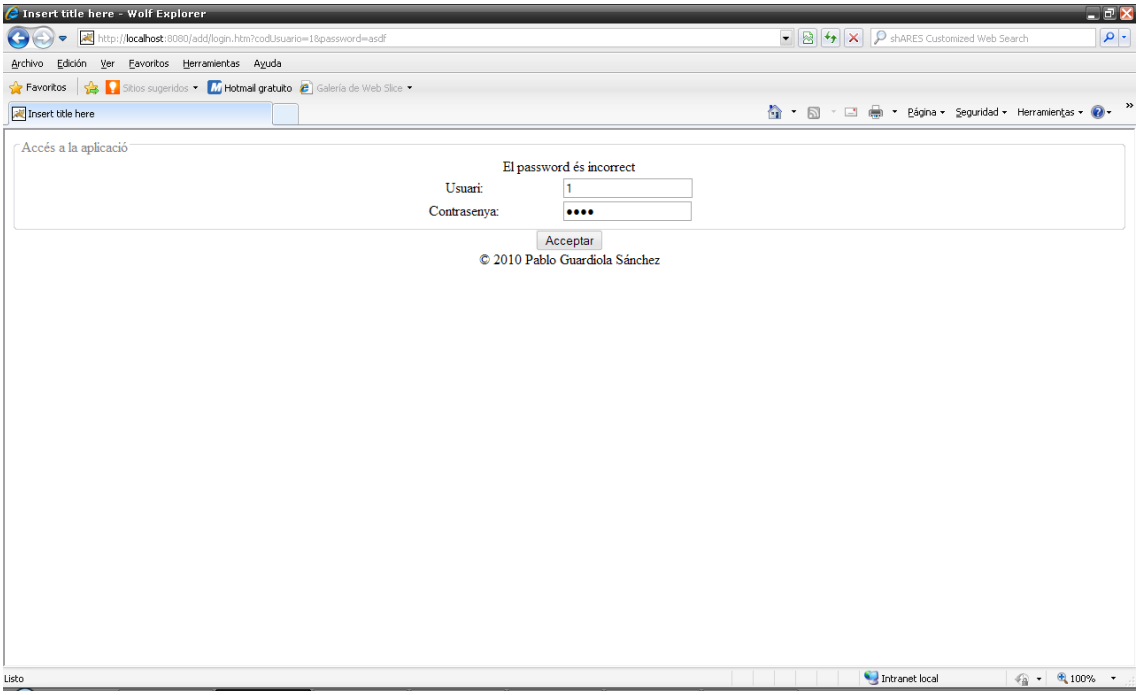


Figura 30. Pantalla Autenticación contraseña incorrecta

Cuando se introduce un usuario y contraseña que se encuentra en la base de datos, se accede a la “Pantalla Inicial”.

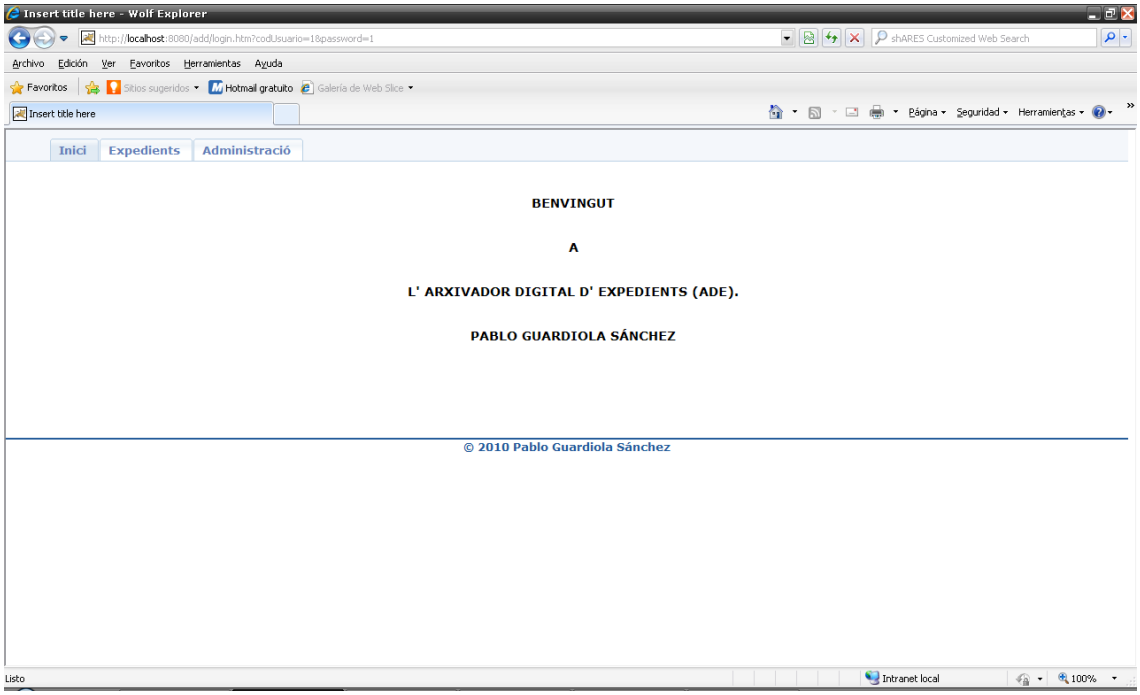


Figura 31. Pantalla Inicial

Se puede observar que la “Pantalla Inicial” está compuesta de tres pestañas, *Inicio*, *Expedientes* y *Administración*. La pestaña *Inicio* es la seleccionada por defecto y carga la “Pantalla de Bienvenida”, que muestra un mensaje de bienvenida al usuario.

Al pinchar sobre la pestaña *Expedientes* se carga la “Pantalla de Expedientes”.

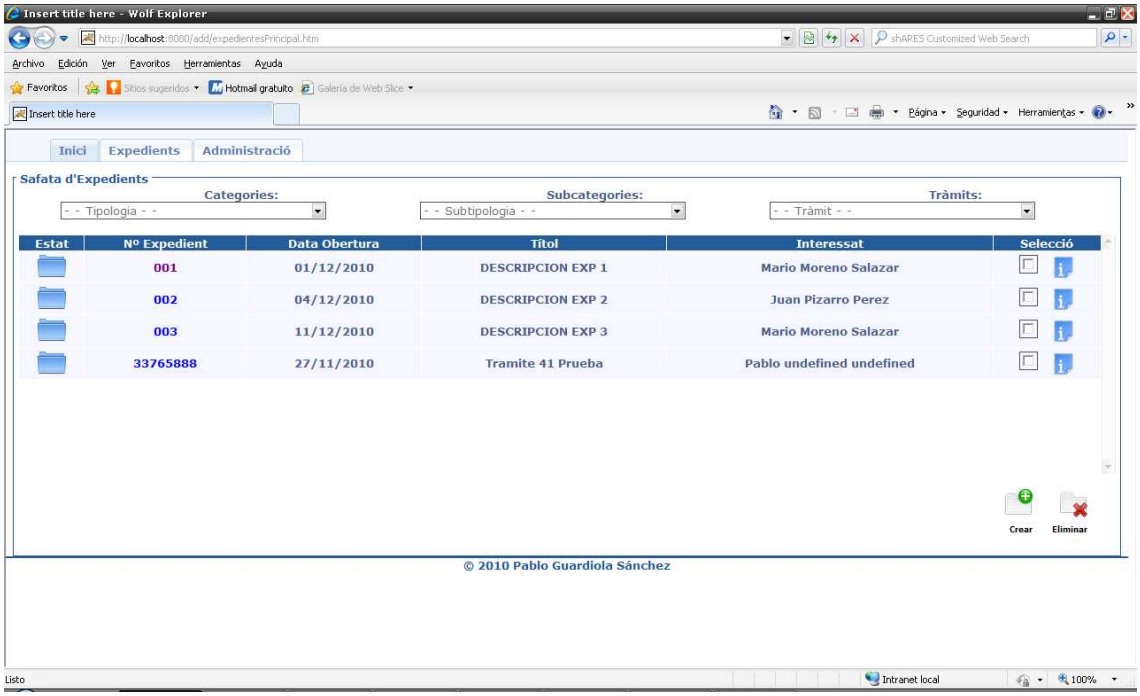


Figura 32. Pantalla Expedientes

Por defecto se cargan todos los expedientes registrados. En cambio, si se selecciona un trámite concreto, a través de las barras desplegables de *Categorías*, *Subcategorías* y *Trámites*, aparecen solamente los expedientes asociados a dicho trámite.

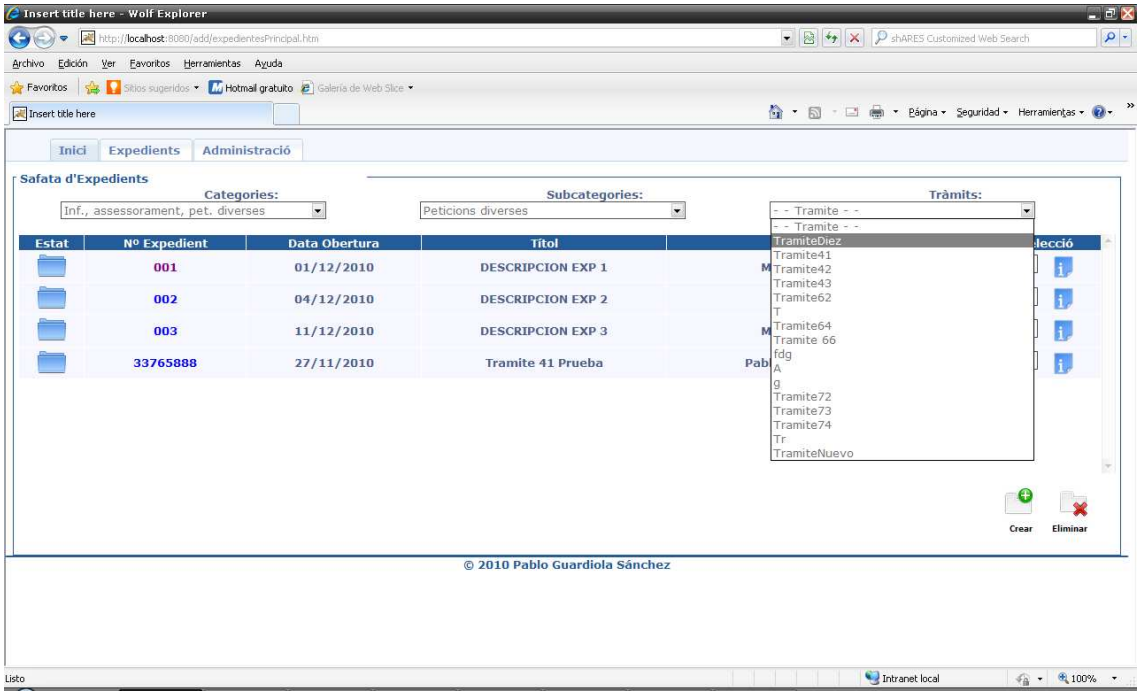


Figura 33. Selección de un trámite

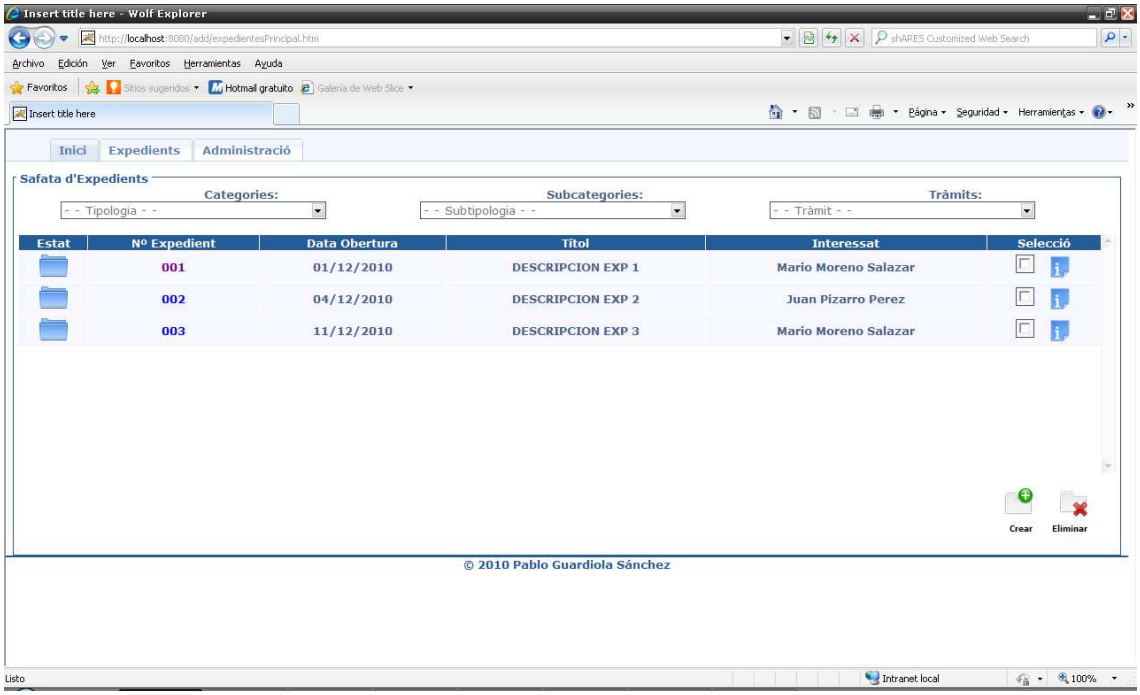


Figura 34. Expedientes asociados a un trámite concreto

Si accedemos a la base de datos, se comprueba que para los expedientes *001*, *002* y *003* el trámite asociado es el *10*.

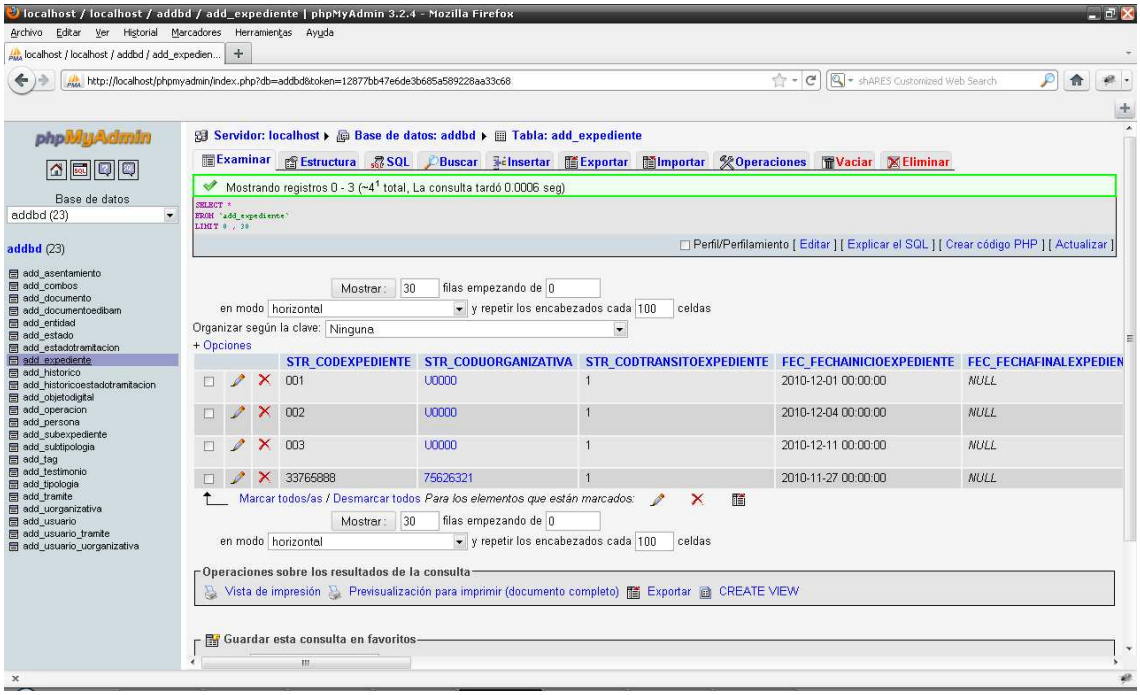


Figura 35. Tabla de expedientes trámite asociado

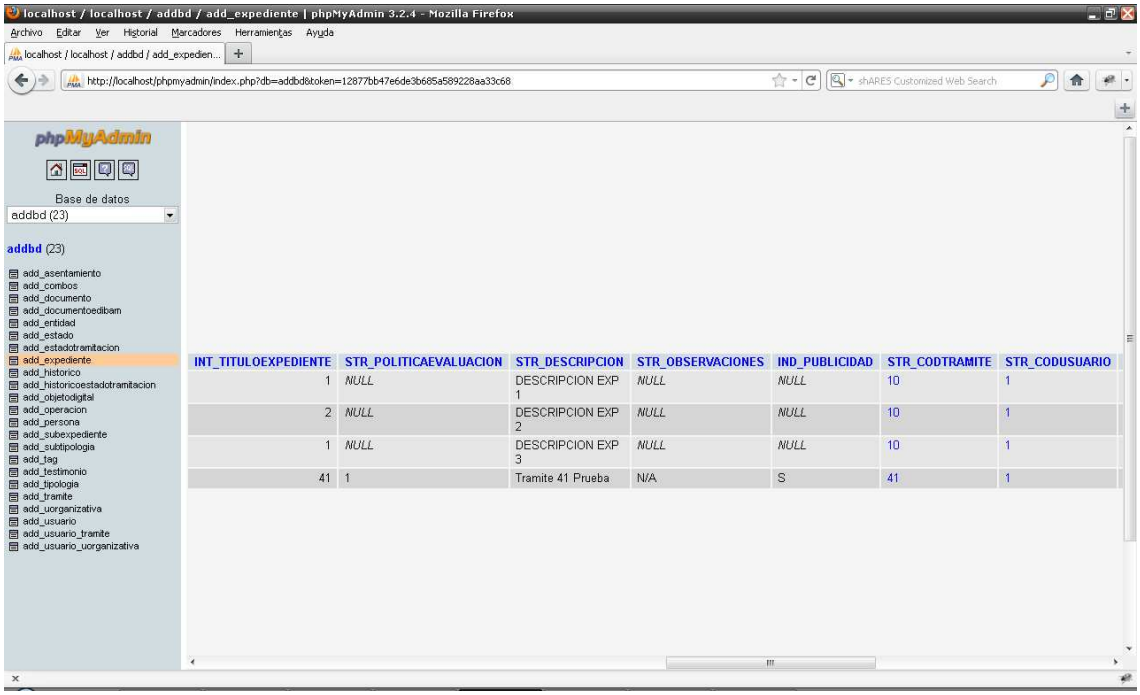


Figura 36. Tabla de expedientes trámite asocioado (cont.)

En este punto, se pueden crear nuevos expedientes, modificar y/o borrar ya existentes.

Para ello, si se pulsa sobre el botón *Crear* aparece un diálogo modal (“Diálogo Crear Expediente). Este diálogo es un formulario que hay que rellenar con los datos del expediente a crear. Mencionar que el número de expediente se genera internamente.

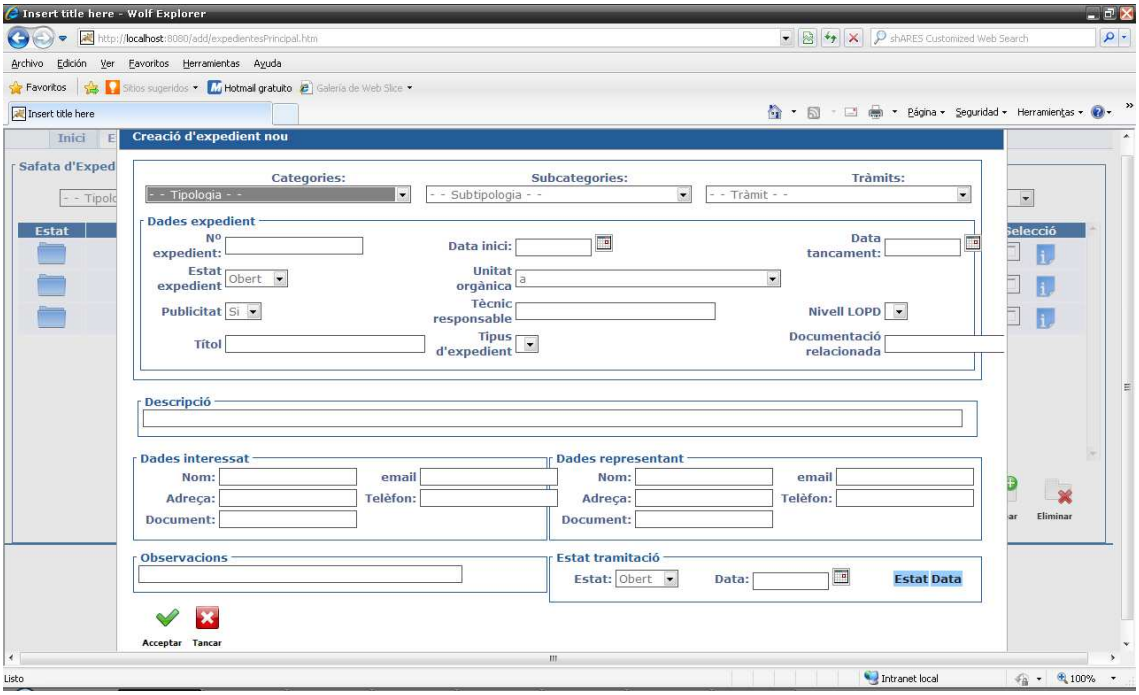


Figura 37. Diálogo Crear Expediente

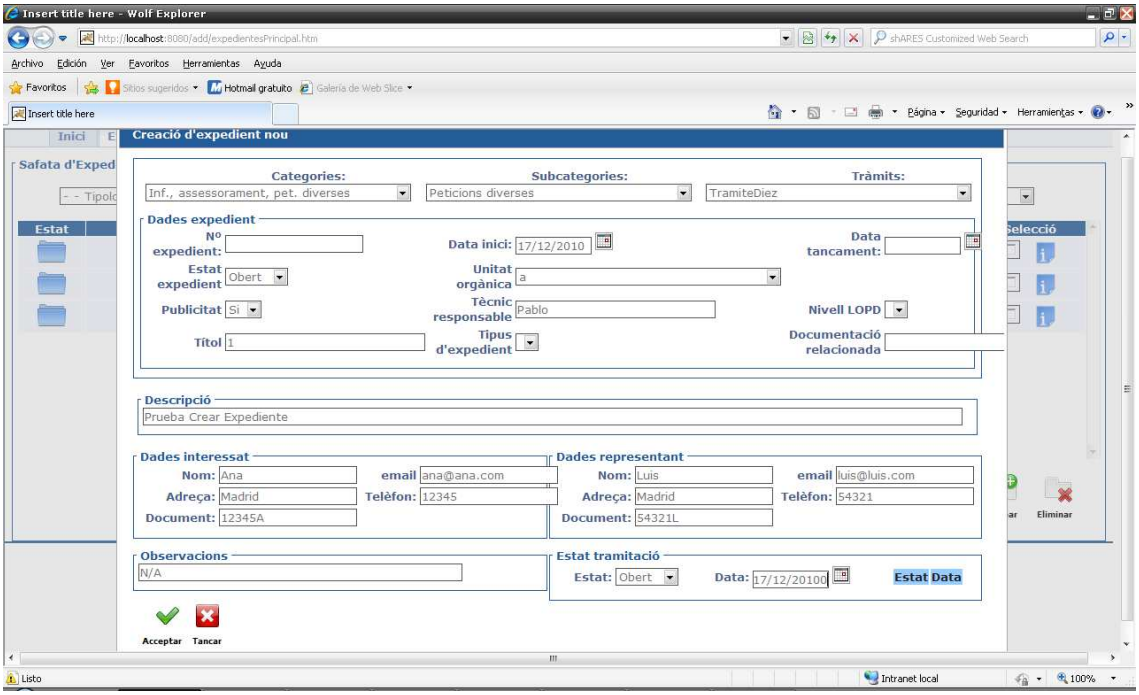


Figura 38. Formulario Crear Expediente con datos

Al pulsar sobre el botón *Aceptar*, se crea en la base de datos el nuevo expediente y aparece, automáticamente, en la bandeja de expedientes.

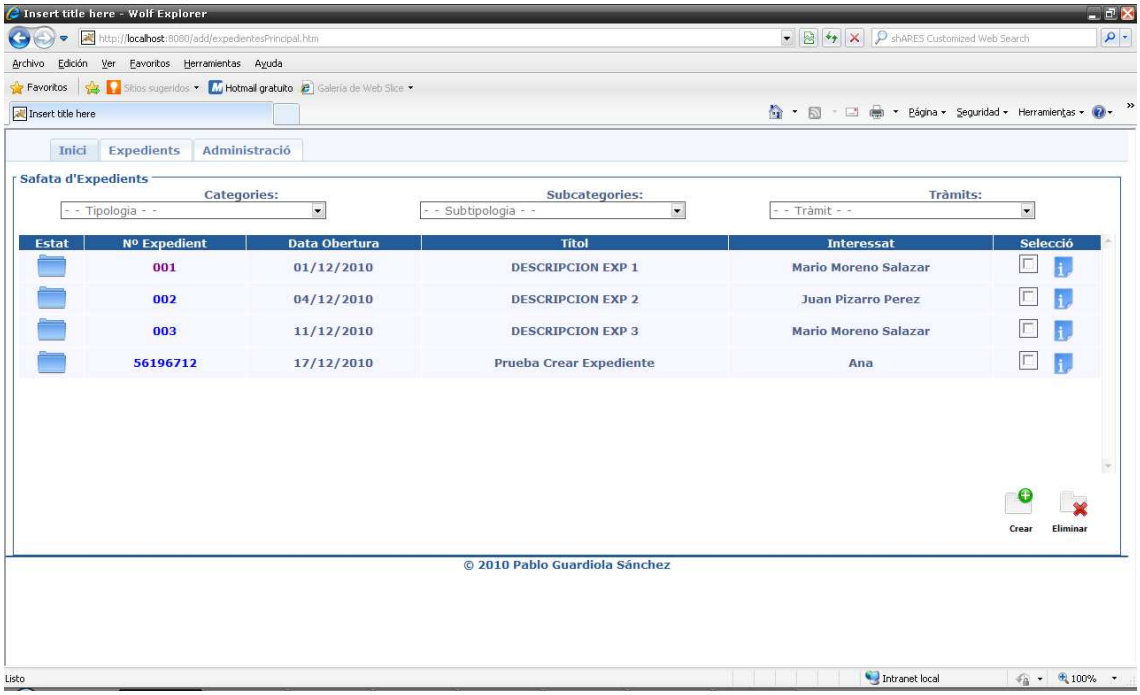


Figura 39. Resultado Crear Expediente

Para modificar un expediente se debe pulsar sobre el icono situado en la columna *Selección* del expediente en cuestión. Tras ello, aparecerá un diálogo modal ("Diálogo Modificar Expediente"). Este diálogo muestra el formulario con los datos del expediente y permite modificar algunos. Por ejemplo, el número de expediente no está permitido modificarlo.

Modificamos la descripción del expediente a 'Prueba Modificar Expediente'.

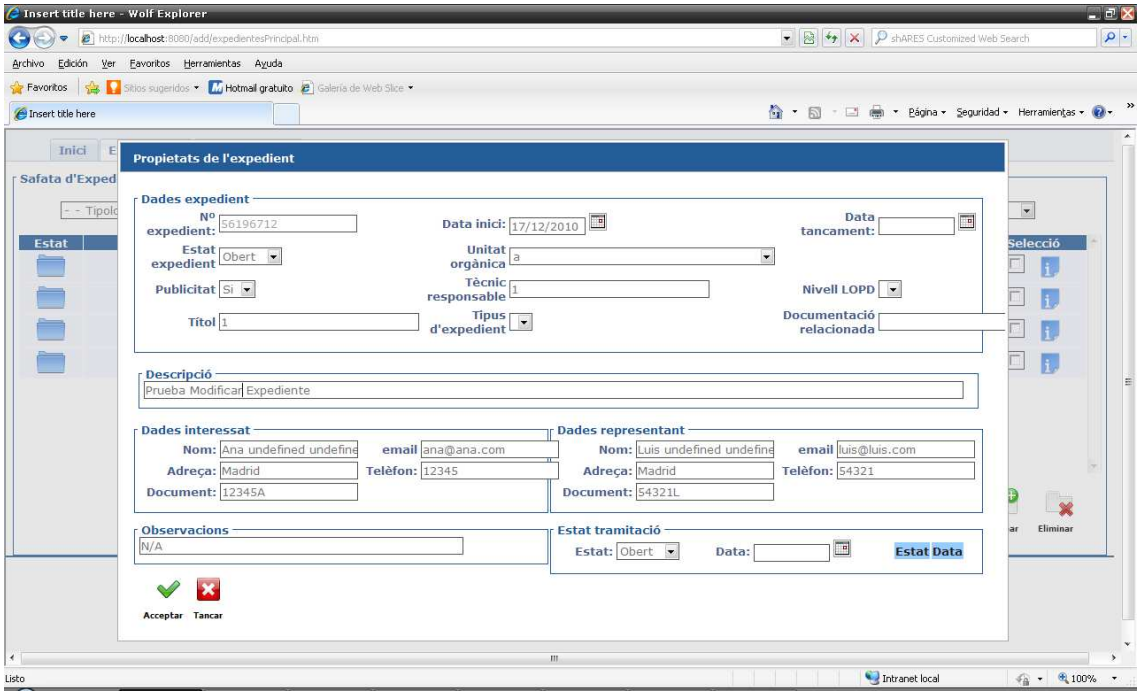


Figura 40. Diálogo Modificar Expediente

Tras pulsar el botón *Aceptar*, al igual que en el caso de crear un expediente, se actualizan los datos tanto en la base de datos como en la pantalla de expedientes.

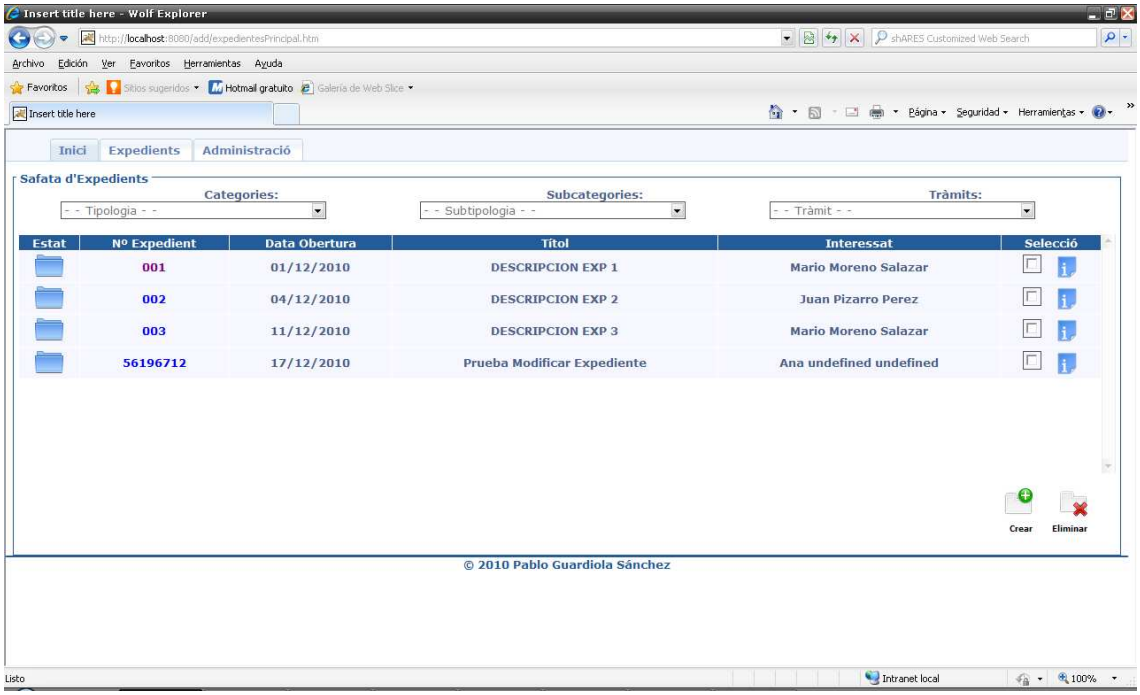


Figura 41. Resultado Modificar Expediente

Para eliminar expedientes, en primer lugar se deberán seleccionar los expedientes a eliminar. Para ello, se debe hacer click sobre el checkbox situado en la columna

Selección del expediente que se desea eliminar y pulsar sobre el botón *Eliminar*. Apuntar que la aplicación permite bajas masivas, esto es, si se selecciona más de un expediente se eliminan todos de una vez.

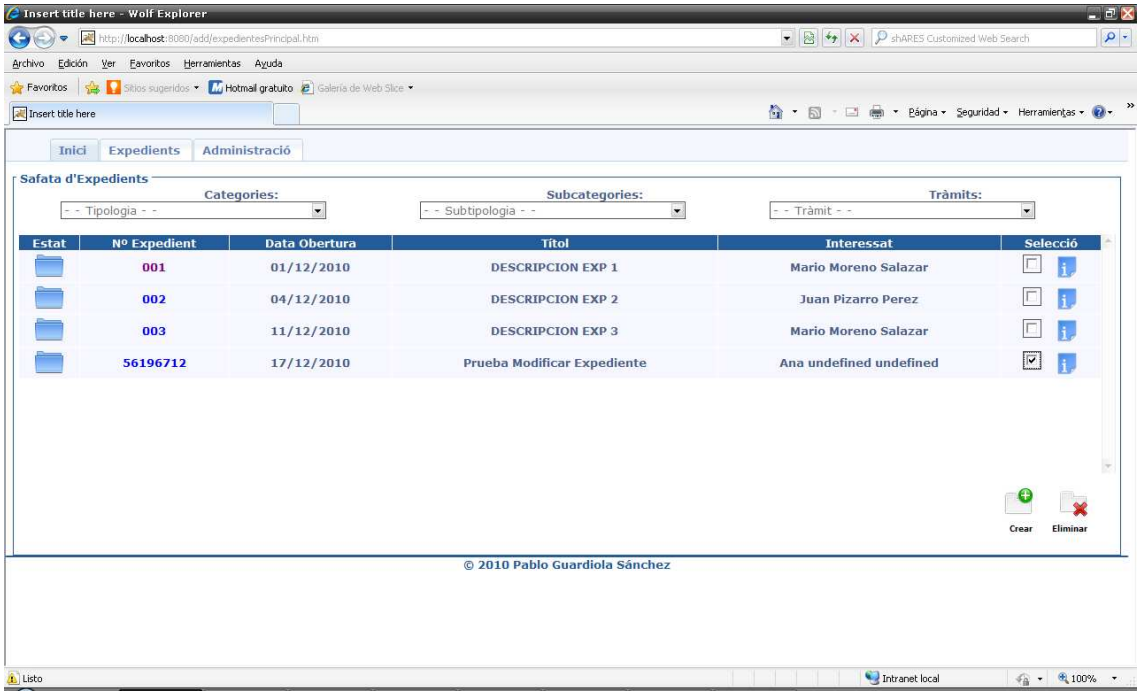


Figura 42. Selección de expediente a eliminar

Tras pulsar el botón *Eliminar* aparece la bandeja de expedientes actualizada. Mencionar que no se efectúa un borrado físico sobre la base de datos, es un borrado lógico, esto es, se marca a 'N' el campo 'IND_VISIBLE' de la tabla.

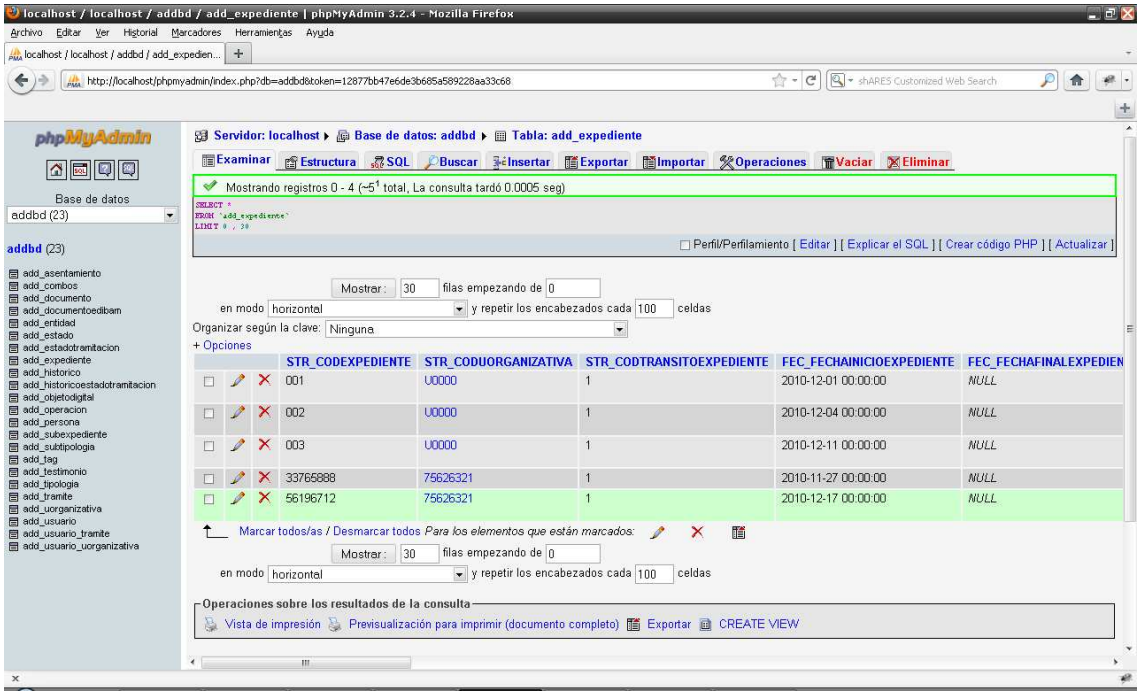


Figura 43. Expediente eliminado

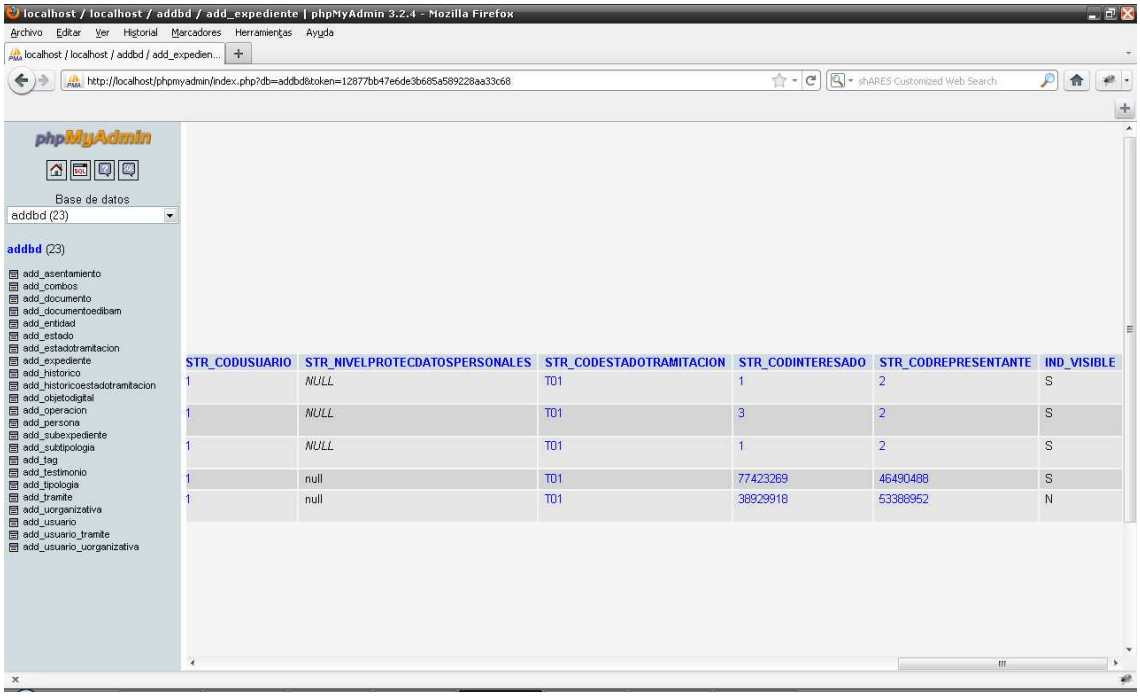


Figura 44. Expediente eliminado (cont.)

Pulsando sobre el *link* situado en la columna Nº Expediente de la bandeja de expedientes, se navega a la “Pantalla Detalle Archivos”. En esta pantalla se muestran los diferentes archivos asociados a dicho expediente. En ella se pueden añadir documentos y eliminarlos.

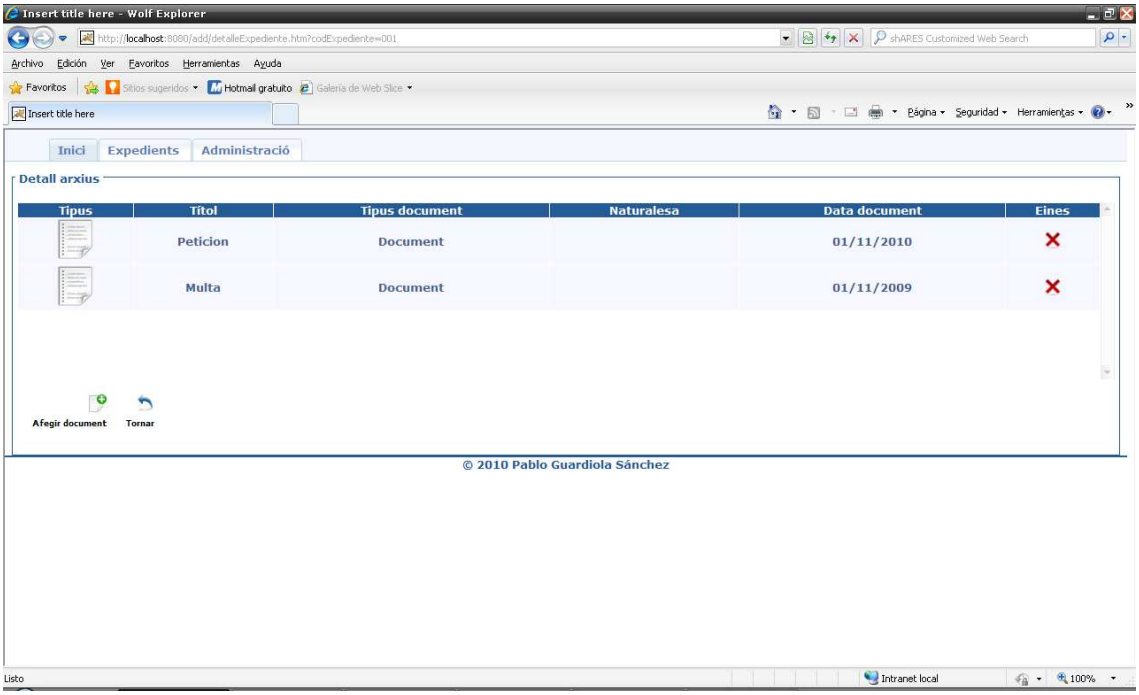


Figura 45. Pantalla Detalle Archivos

Para añadir un nuevo documento se debe pulsar sobre el botón *Añadir documento*, tras esto, aparece el “Diálogo Crear Documento”.

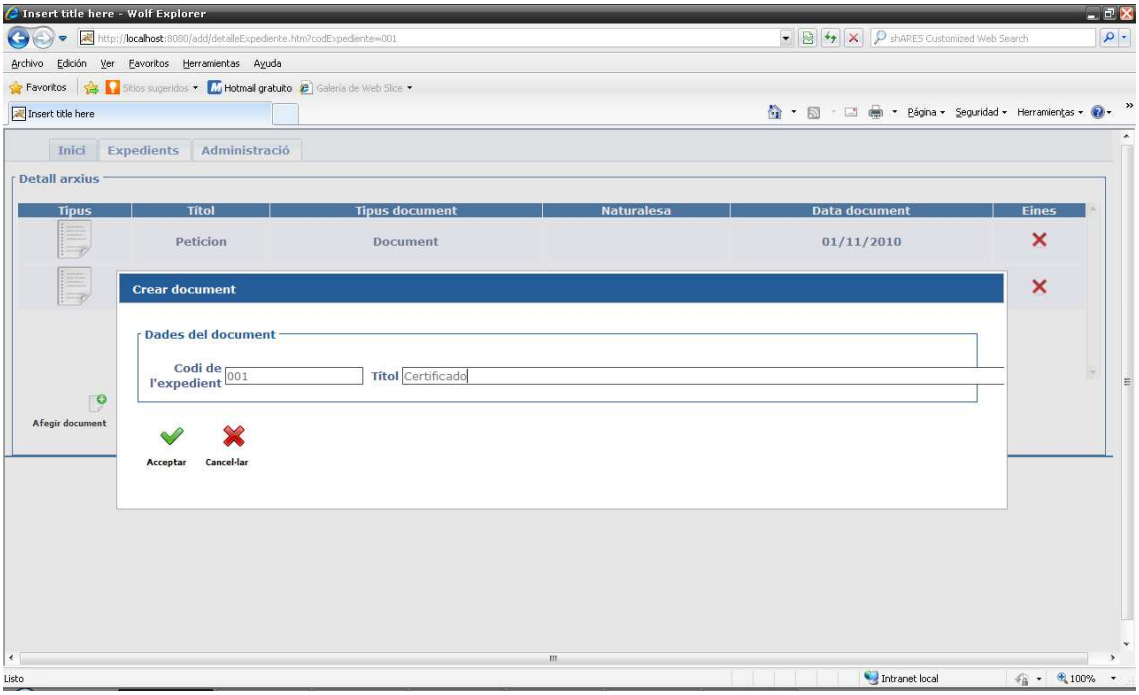


Figura 46. Diálogo Crear Documento

En este punto se debe escribir el *Título* del documento y pulsar sobre el botón *Aceptar*. Esta acción navega a la “Pantalla Cargar Documento”.

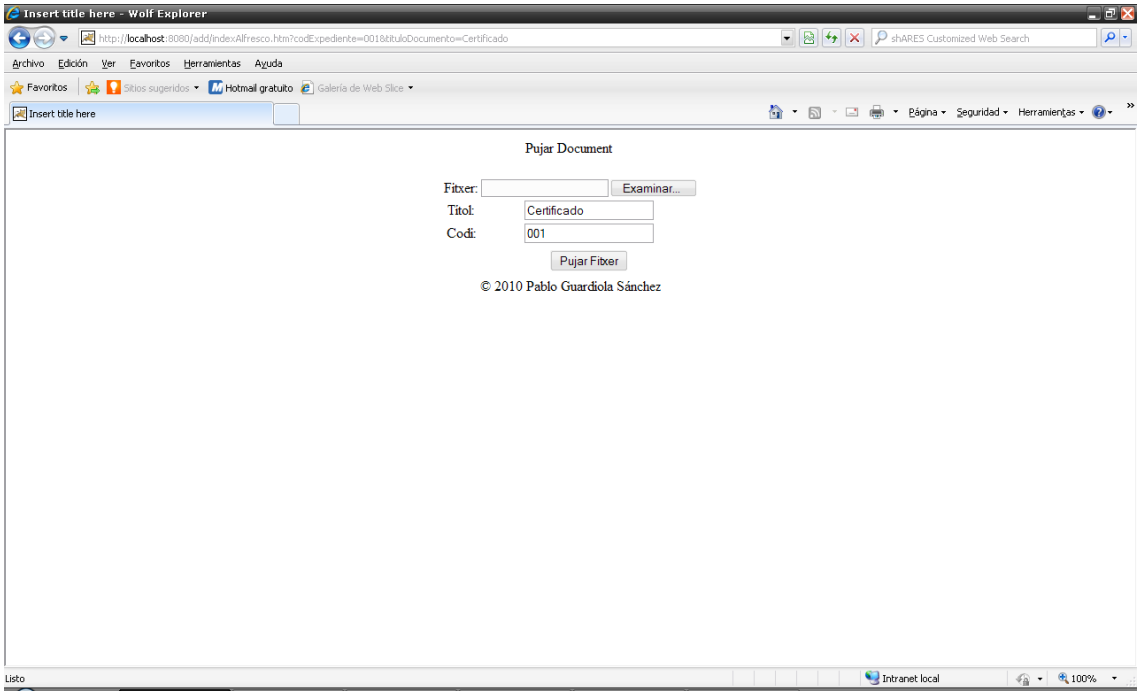


Figura 47. Pantalla Cargar Documento

En esta pantalla se debe seleccionar el fichero que se desea subir a través del botón *Examinar*. Tras ello, se pulsa sobre el botón *Subir Fichero*.

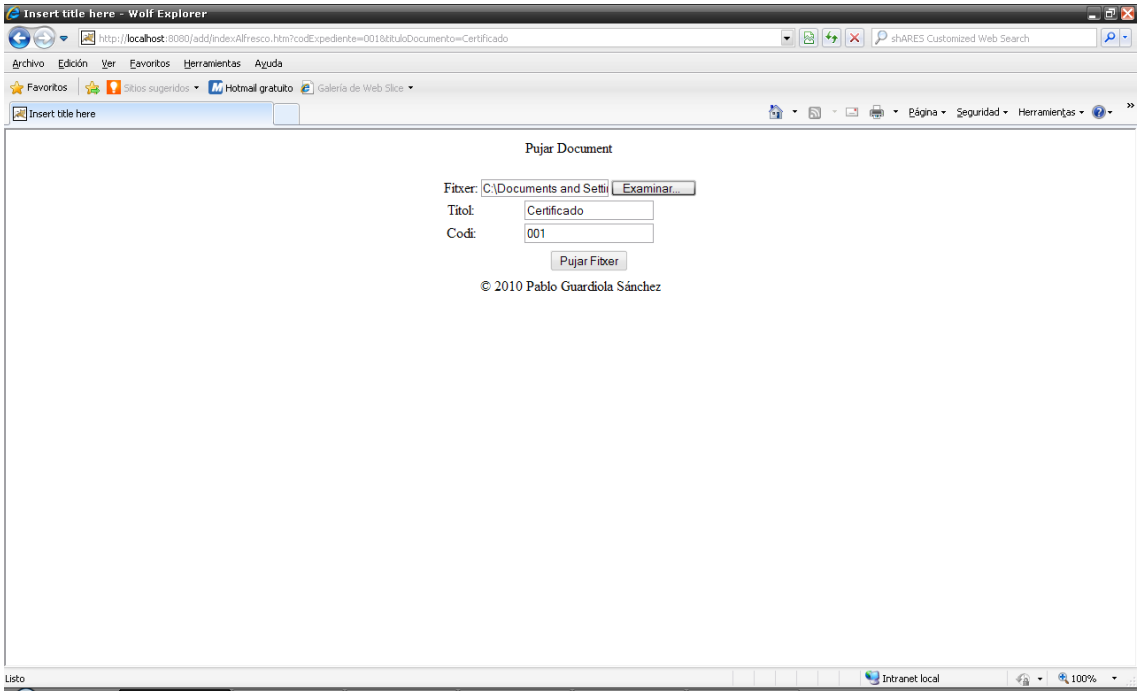


Figura 48. Selección de fichero a subir

Esta operativa sube el fichero a Alfresco, concretamente, lo sube al espacio, creado previamente, asociado al expediente en cuestión.

Tras acceder con el perfil de “Administrador Alfresco” en Alfresco, se observa que la operación se ha efectuado correctamente.

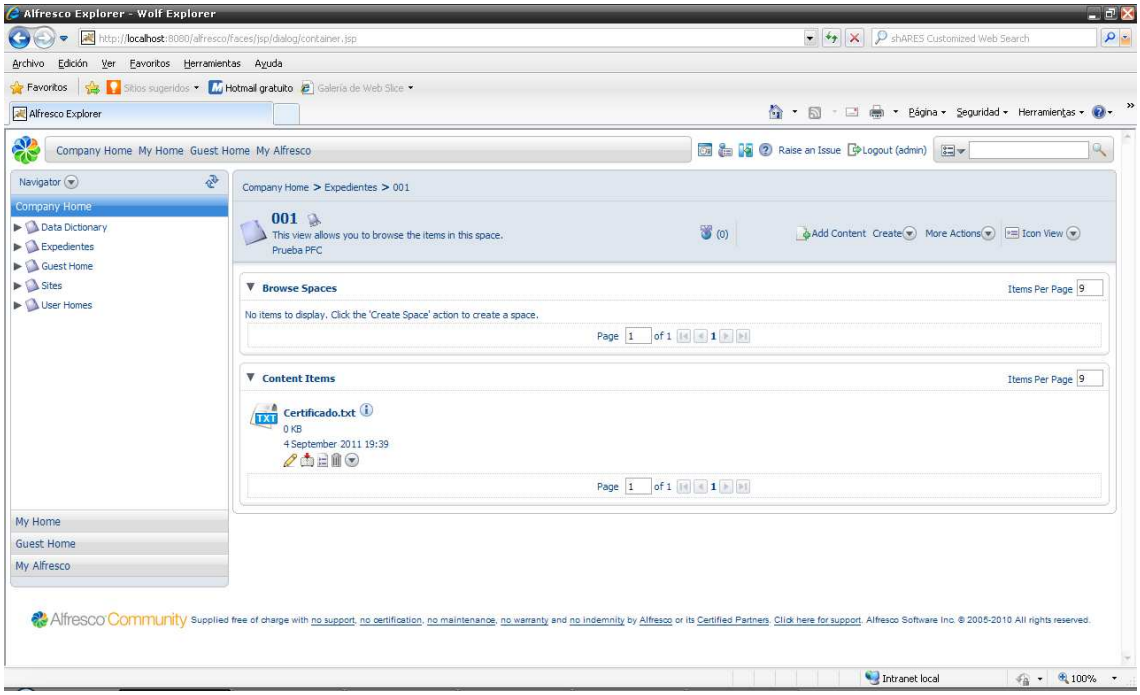


Figura 49. Archivo subido en Alfresco

Puesto que todo ha ido bien se muestra la “Pantalla Resultado”. En esta aparece un *link* que nos permite volver a la “Pantalla Detalle Archivos”.

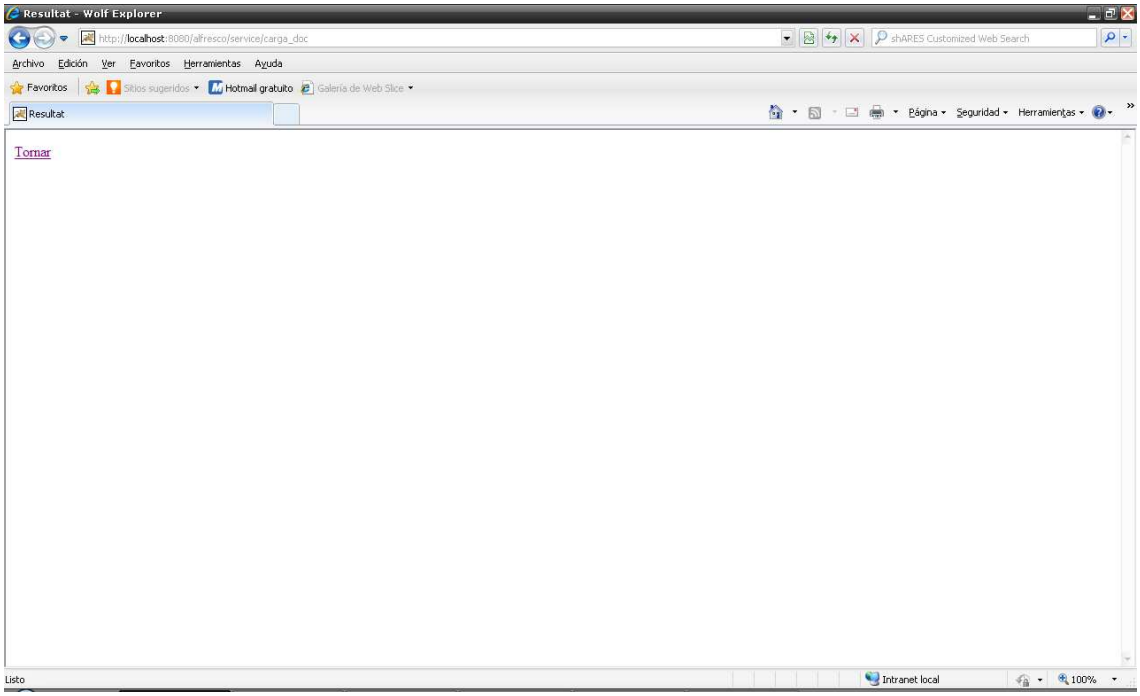


Figura 50. Pantalla Resultado

Si pulsamos sobre el *link Volver* se observa que se ha añadido el documento y queda reflejado en la “Pantalla Detalle Archivos”.

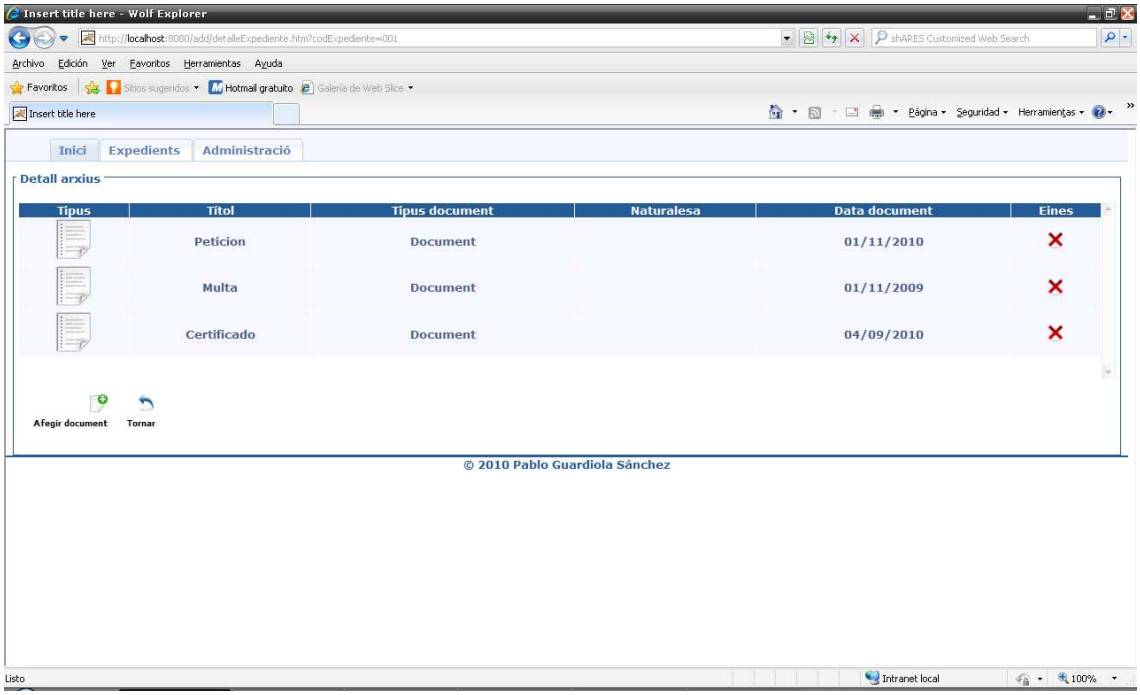


Figura 51. Archivo subido en la aplicación

Para eliminar un documento se debe pulsar sobre el botón *Eliminar* situado en la columna *Herramientas* del documento que se quiere borrar.

Tras ello se actualiza la “Pantalla Detalle Archivos” con los cambios realizados. Internamente se guarda la fecha de borrado en la base de datos.

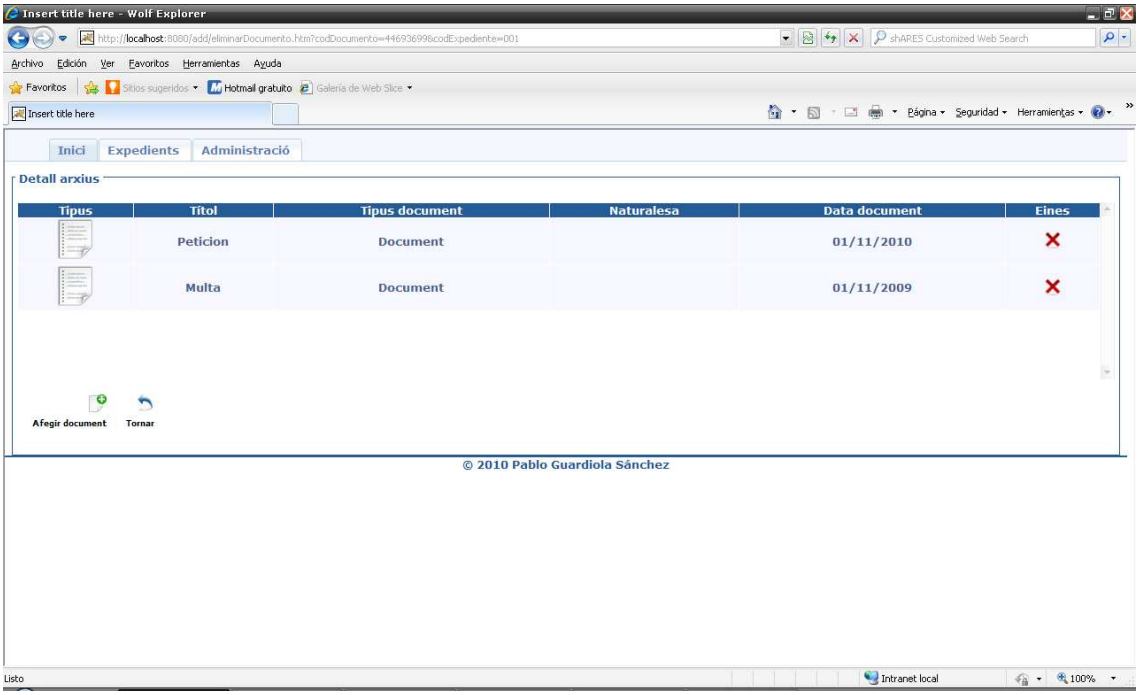


Figura 52. Archivo eliminado

Al igual que en los casos anteriores, se produce un borrado lógico sobre la base de datos. Además, no se borra el fichero físico, por si se quisiera recuperar en otro momento. Para borrarlo definitivamente, habría que hacerlo directamente sobre Alfresco con el perfil “Administrador Alfresco”.

Por otro lado, la “Pantalla Administración” está compuesta por dos pestañas, que se corresponden con la “Pantalla Gestión Usuarios” y la “Pantalla Gestión Trámites”, respectivamente.

La “Pantalla Gestión Usuarios” se carga por defecto al pulsar sobre la pestaña *Administración*. En ella se pueden dar de alta nuevos usuarios, modificar usuarios ya existentes y darlos de baja.

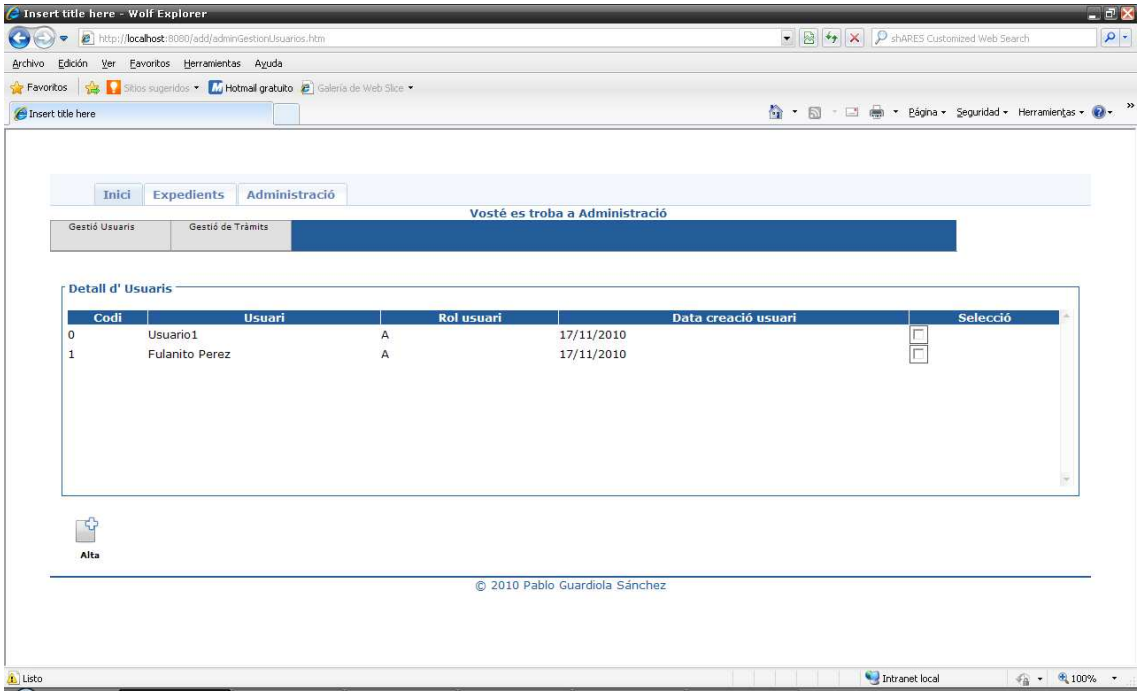


Figura 53. Pantalla Gestión Usuarios

Para dar de alta un usuario se debe pulsar sobre el botón *Alta*. Tras esto, aparece el "Diálogo Alta Usuario".

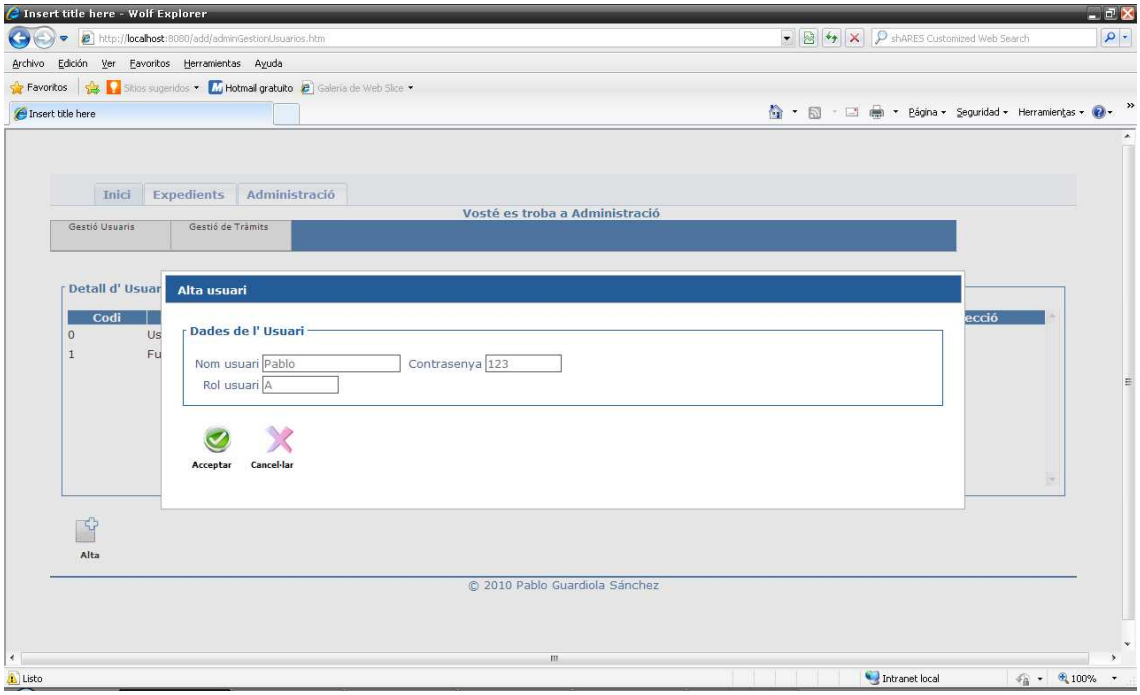


Figura 54. Formulario Alta Usuario con datos

Después de rellenar los datos y tras pulsar el botón *Aceptar* se actualizan los datos tanto en la base de datos como en la "Pantalla Gestión Usuarios".

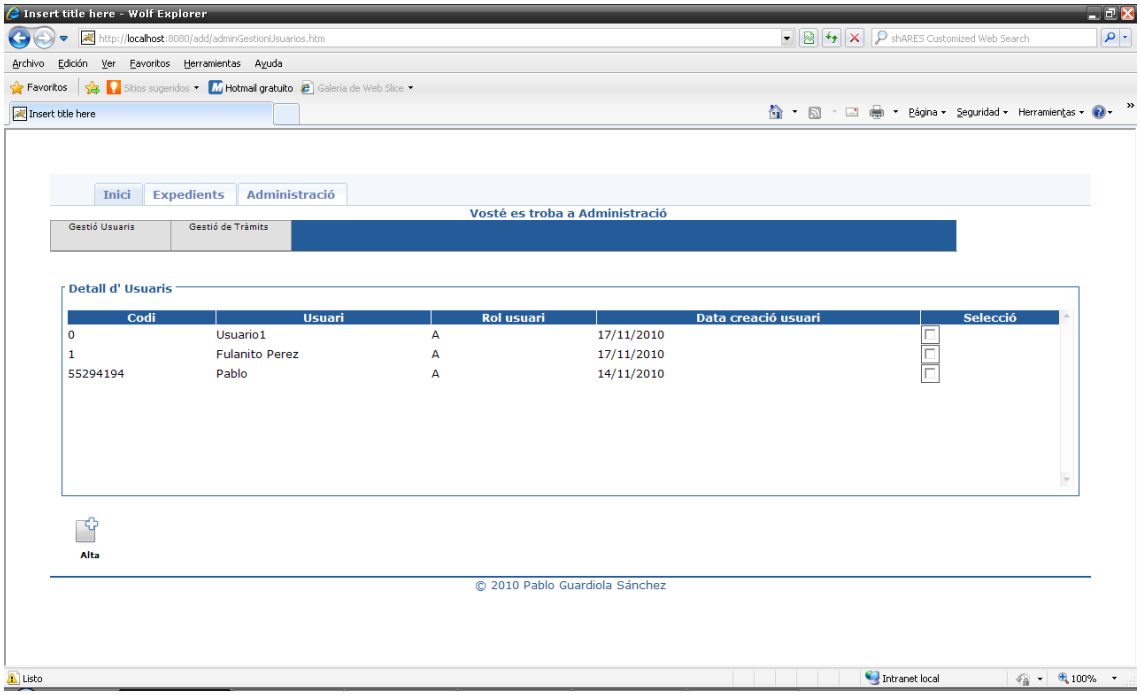


Figura 55. Resultado Crear Usuario

Para modificar los atributos de un usuario, en primer lugar, se debe seleccionar el usuario a modificar a través del *checkbox* situado en la columna *Selección*, tras esto, aparece el botón *Modificar* y al pulsar sobre él se muestra el "Diálogo Modificar Usuario".

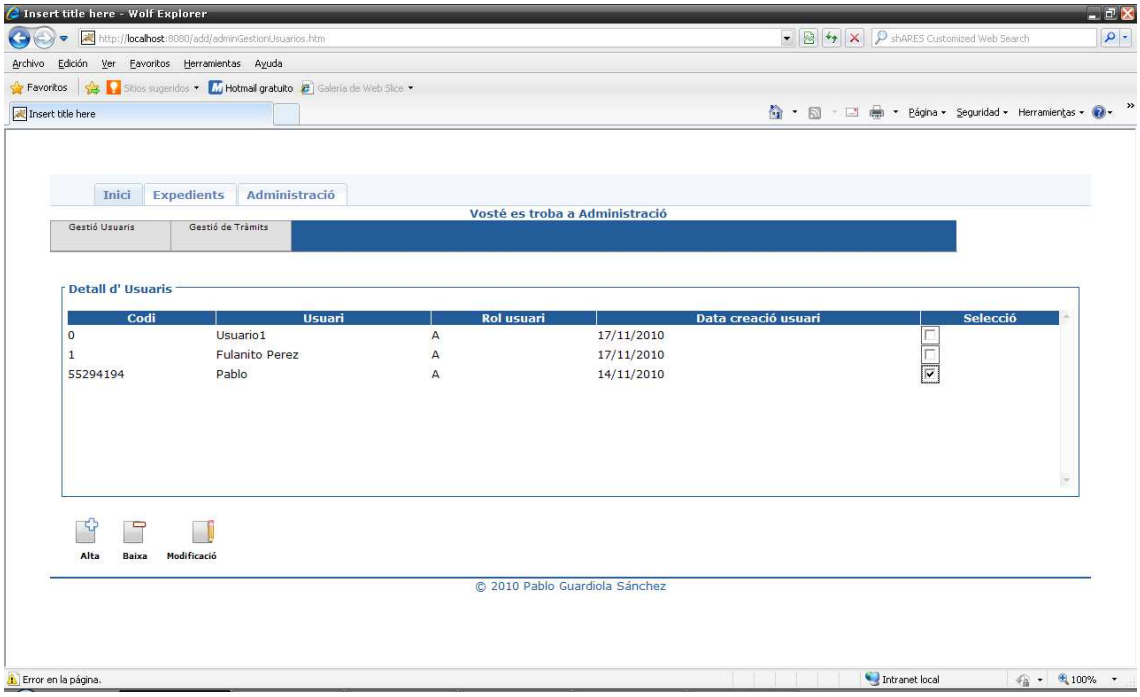


Figura 56. Selección de usuario a modificar

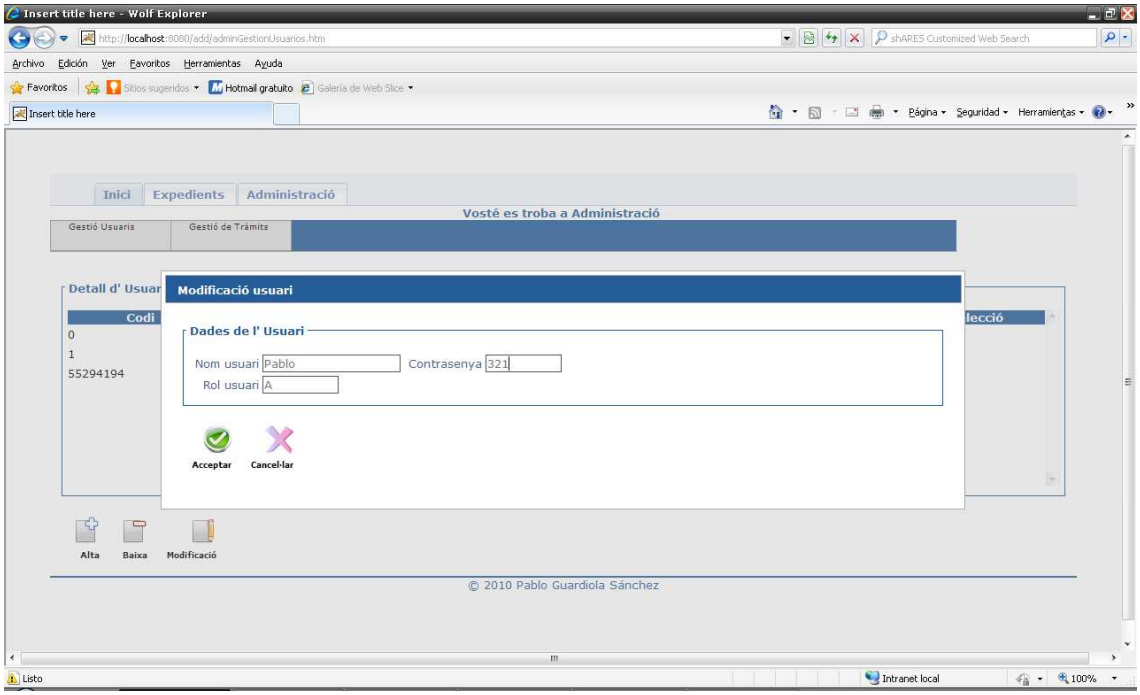


Figura 57. Formulario Modificar Usuario con datos

En el ejemplo, se ha modificado la contraseña a '321', al pulsar sobre el botón *Aceptar* se actualizan los cambios realizados.

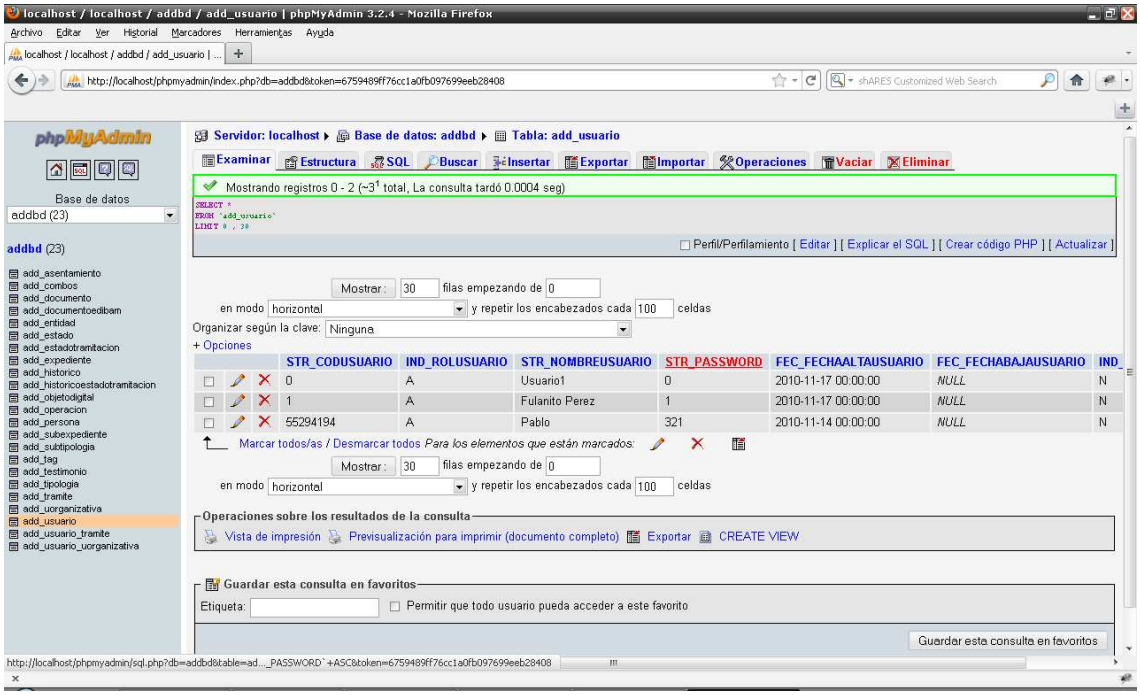


Figura 58. Usuario modificado

Para eliminar un usuario, se deben seleccionar los usuarios a eliminar a través de la columna *Selección* y pulsar sobre el botón *Baja*. Mencionar que como en casos

anteriores se permiten bajas masivas. También se guarda internamente en la base de datos la fecha de baja del usuario.

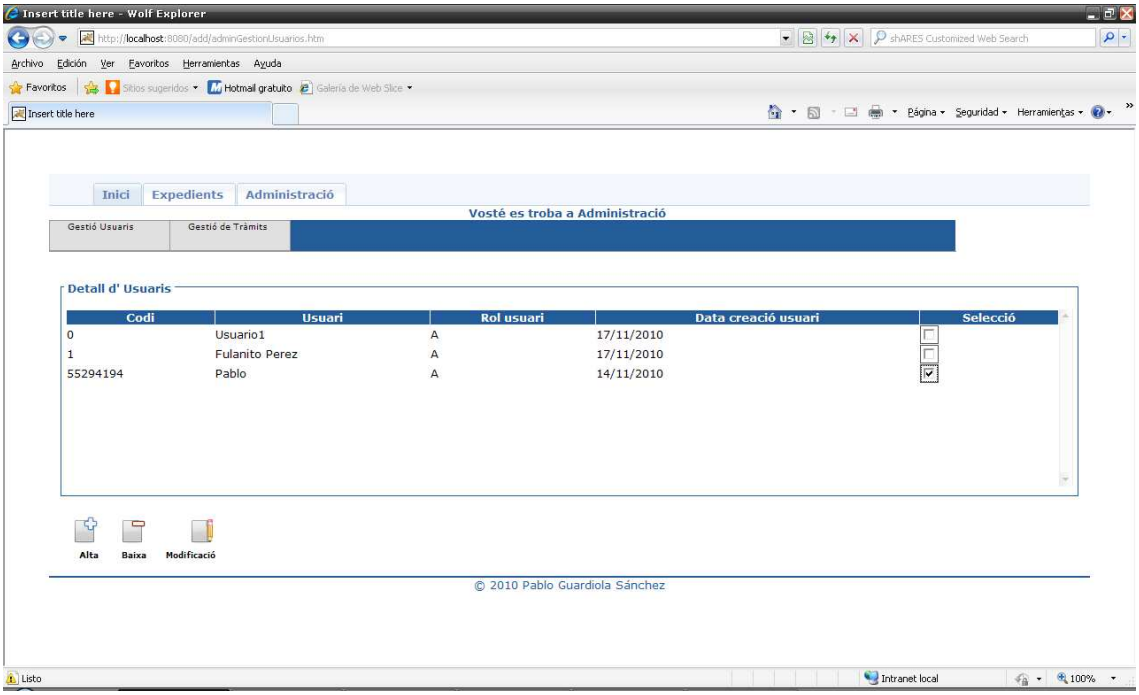


Figura 59. Selección de usuario a eliminar

Al igual que en casos anteriores, se produce un borrado lógico, en este caso se marca a ‘S’ el campo ‘IND_ELIMINADO’ de la tabla.

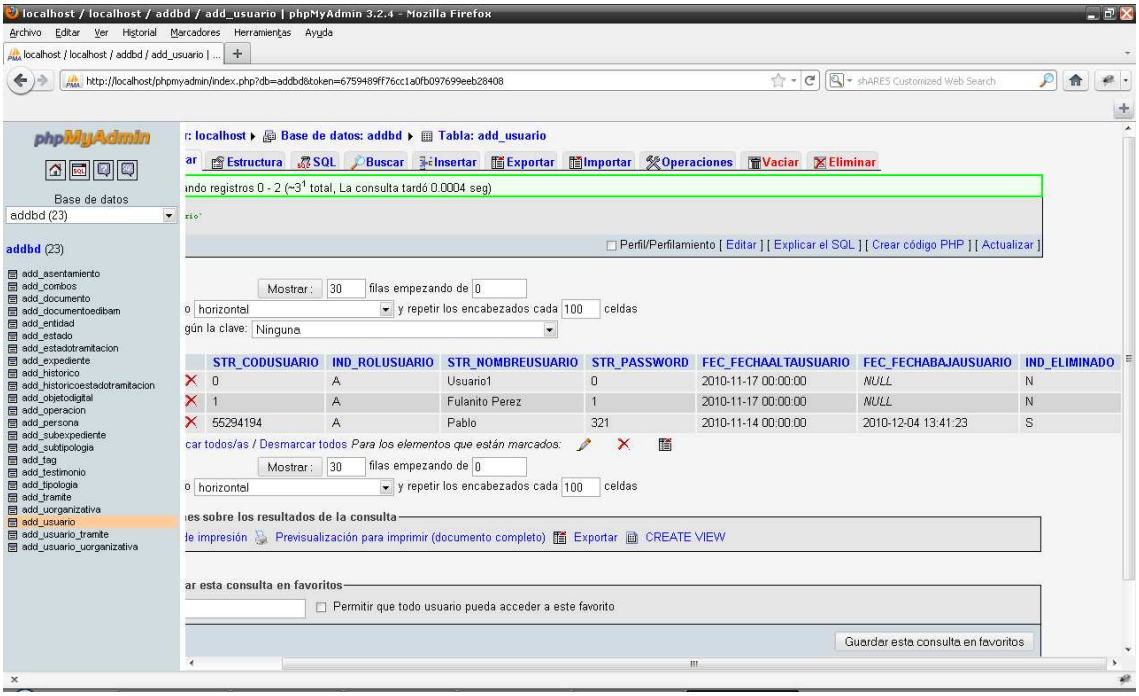


Figura 60. Usuario eliminado

Para acceder a la “Pantalla Gestión Trámites” se debe pulsar sobre la pestaña *Gestión de Trámites*. Esta pantalla permite el mantenimiento de los trámites de la aplicación, esto es, darlos de alta, modificarlos y eliminarlos. Tiene funcionamiento similar a la “Pantalla Gestión Usuarios”.

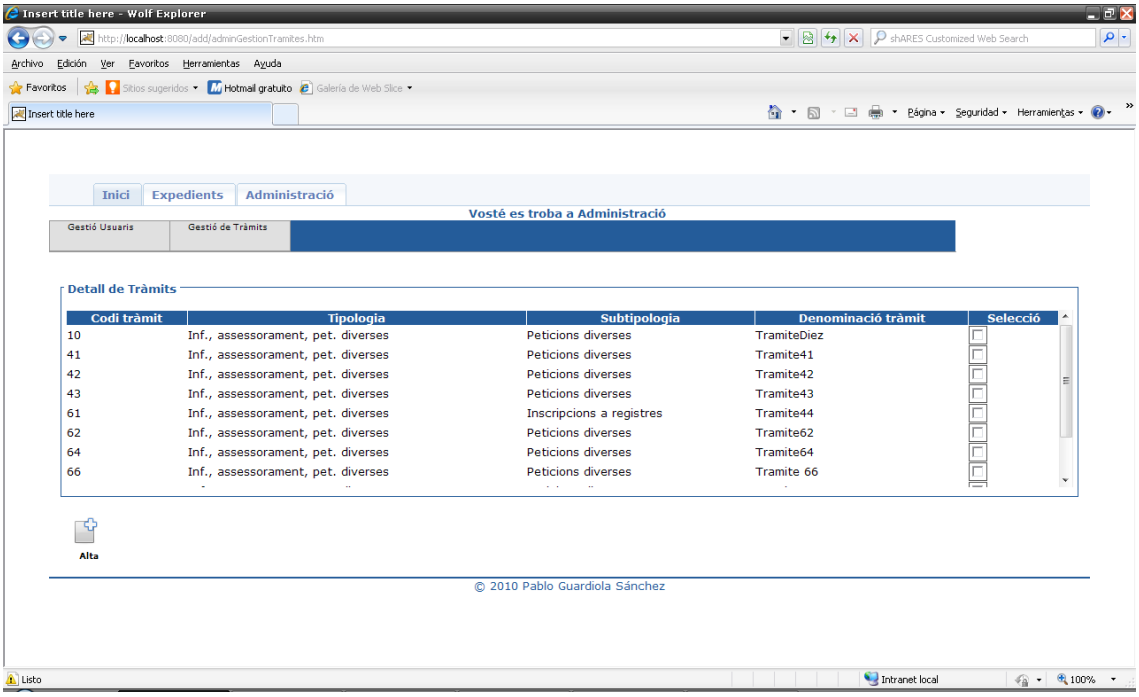


Figura 61. Pantalla Gestión Trámites

Al pulsar sobre el botón *Alta*, aparece el “Diálogo Alta Trámite”.

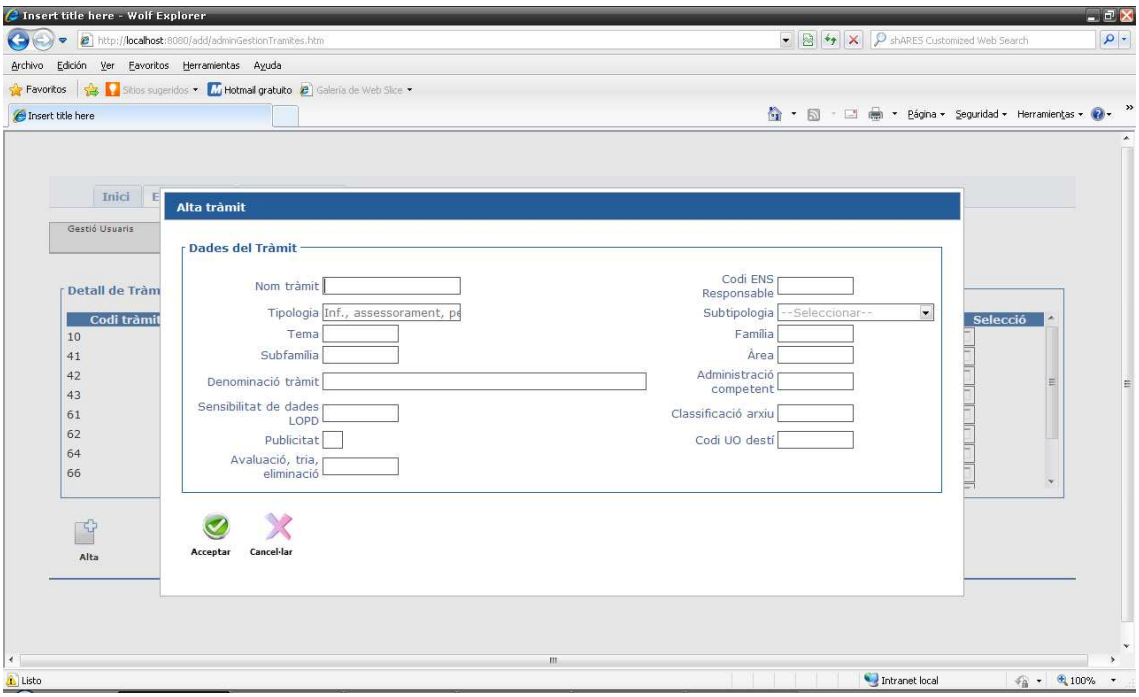


Figura 62. Diálogo Alta Trámite

Tras rellenar el formulario con los datos del nuevo trámite y pulsar sobre el botón *Aceptar* se añade el trámite al catálogo de trámites de la aplicación.

Para modificar un trámite, al igual que para los usuarios, se debe seleccionar tan sólo un trámite a través de la columna *Selección*, con ello se muestra el botón *Modificar* y, tras pulsarlo, aparece el “Diálogo Modificar Trámite”.

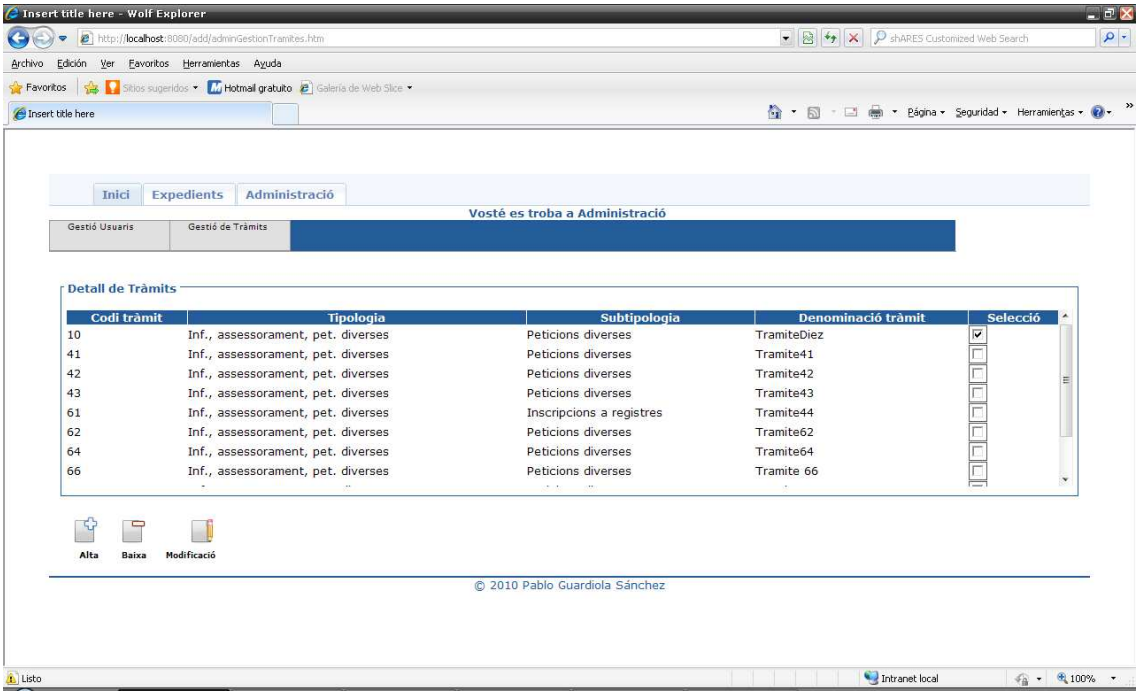


Figura 63. Selección trámite a modificar

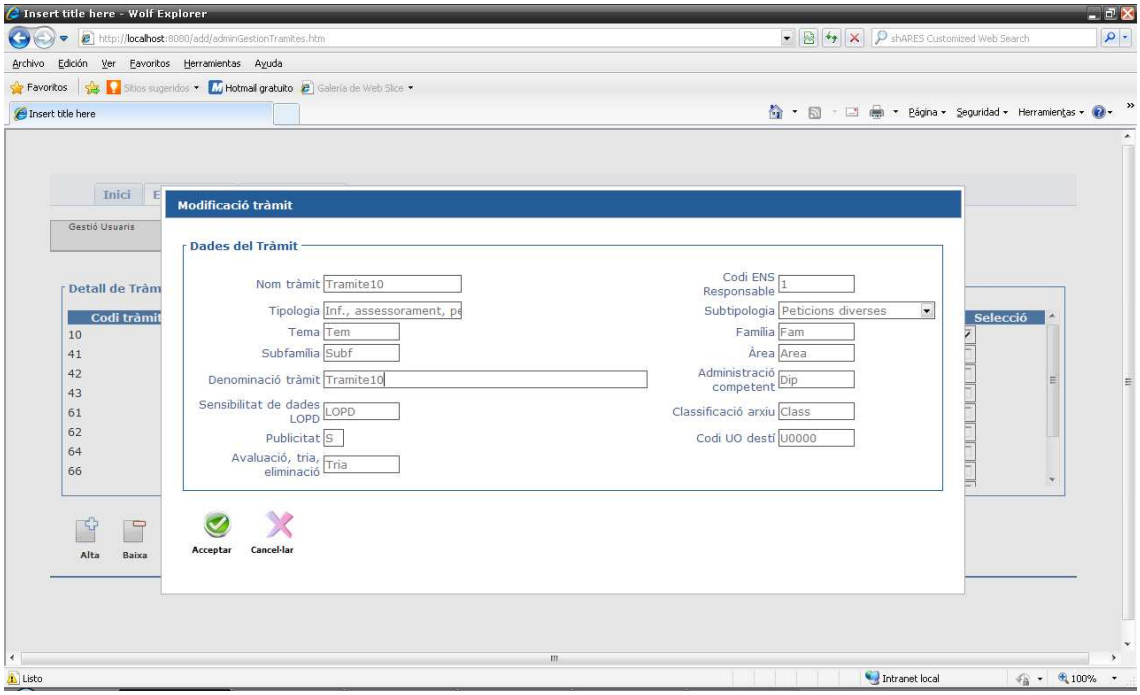


Figura 64. Formulario Modificar Trámite con datos

En el ejemplo, se cambia la descripción del trámite de 'TramiteDiez' a 'Tramite10'. Al pulsar sobre el botón *Aceptar* se actualizan los cambios sobre la aplicación.

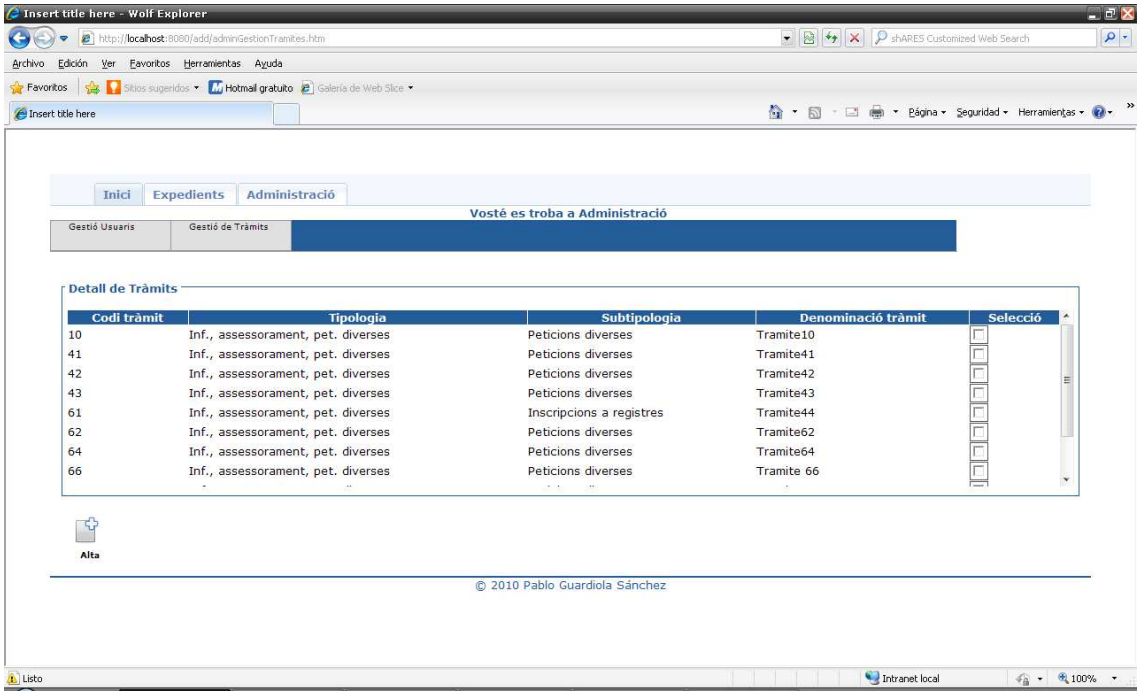


Figura 65. Resultado Modificar Trámite

Para eliminar un trámite, basta con seleccionar el/los trámites a dar de baja y pulsar sobre el botón Baja.

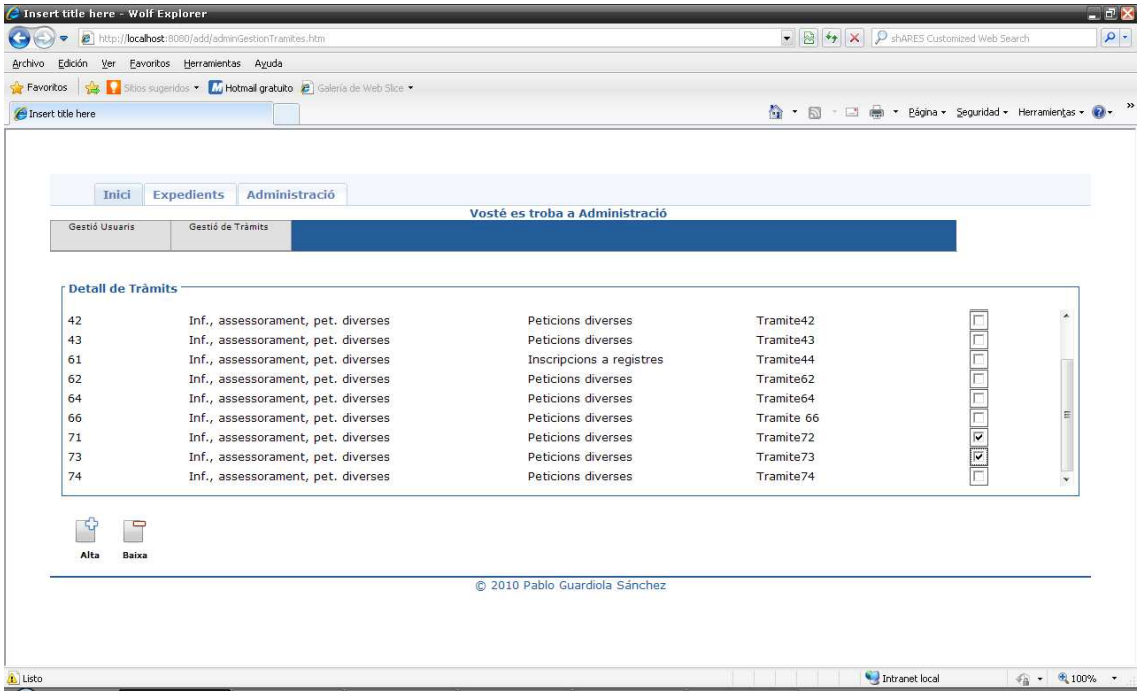


Figura 66. Selección trámites a eliminar

Al igual que en el resto de la aplicación el borrado es lógico, en este caso, se marca a 'S' el campo 'IND_ELIMINADO' de la tabla y se inserta la fecha de baja.

En el ejemplo se observa que los trámites '71' y '73' se han actualizado correctamente.

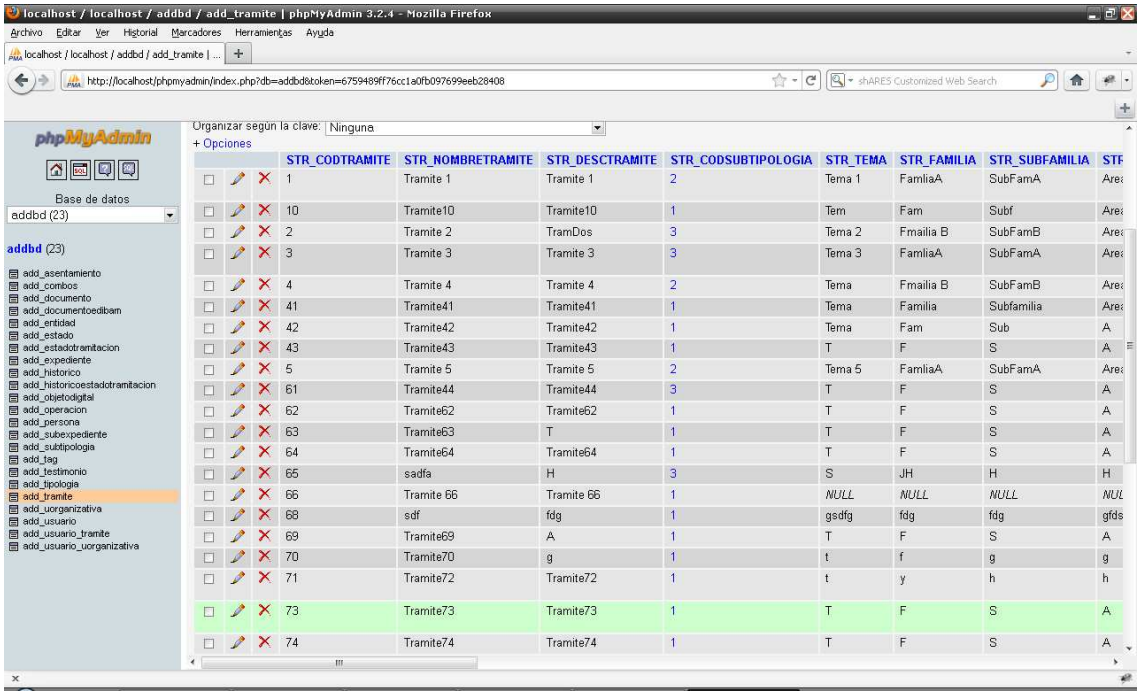
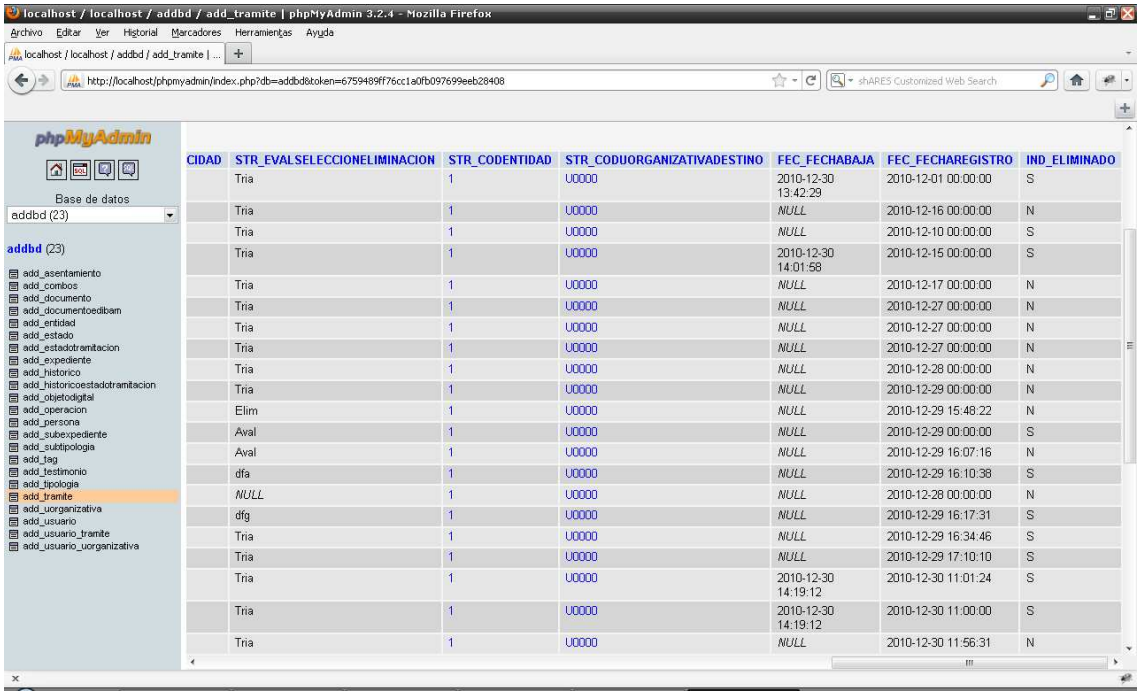


Figura 67. Trámites eliminados



CIDAD	STR_EVALSELECCIONELIMINACION	STR_CODENTIDAD	STR_CODUORGANIZATIVADESTINO	FEC_FECHABAJA	FEC_FECHAREGISTRO	IND_ELIMINADO
	Tria	1	U0000	2010-12-30 13:42:29	2010-12-01 00:00:00	S
	Tria	1	U0000	NULL	2010-12-16 00:00:00	N
	Tria	1	U0000	NULL	2010-12-10 00:00:00	S
	Tria	1	U0000	2010-12-30 14:01:58	2010-12-15 00:00:00	S
	Tria	1	U0000	NULL	2010-12-17 00:00:00	N
	Tria	1	U0000	NULL	2010-12-27 00:00:00	N
	Tria	1	U0000	NULL	2010-12-27 00:00:00	N
	Tria	1	U0000	NULL	2010-12-27 00:00:00	N
	Tria	1	U0000	NULL	2010-12-28 00:00:00	N
	Tria	1	U0000	NULL	2010-12-29 00:00:00	N
	Elim	1	U0000	NULL	2010-12-29 15:48:22	N
	Aval	1	U0000	NULL	2010-12-29 00:00:00	S
	Aval	1	U0000	NULL	2010-12-29 16:07:16	N
	dfa	1	U0000	NULL	2010-12-29 16:10:38	S
	NULL	1	U0000	NULL	2010-12-28 00:00:00	N
	dfg	1	U0000	NULL	2010-12-29 16:17:31	S
	Tria	1	U0000	NULL	2010-12-29 16:34:46	S
	Tria	1	U0000	NULL	2010-12-29 17:10:10	S
	Tria	1	U0000	2010-12-30 14:19:12	2010-12-30 11:01:24	S
	Tria	1	U0000	2010-12-30 14:19:12	2010-12-30 11:00:00	S
	Tria	1	U0000	NULL	2010-12-30 11:56:31	N

Figura 68. Trámites eliminados (cont.)

7. Usos futuros

Este proyecto está centrado en la gestión de expedientes a través de Alfresco, así pues, uno de los posibles usos futuros o ampliaciones del proyecto sería integrar Alfresco completamente en la aplicación y que fuera trasparente al usuario. Entre otras cosas, implicaría que al crear un expediente se creara directamente el espacio asociado en Alfresco.

Otra propuesta interesante sería poder visualizar y descargar los archivos asociados a un expediente directamente desde la aplicación, así como, crear un buscador de estos y no tener que usar el que incorpora Alfresco.

8. Conclusiones

Considero la posibilidad de realizar el proyecto fin de carrera en una empresa como una gran oportunidad por múltiples razones:

Uno de los puntos más importantes que saco de esta experiencia, es el ser consciente de las múltiples tecnologías que se emplean para poder llevar a cabo un proyecto real. Cómo podemos seleccionar dentro de cada una de ellas las funcionalidades requeridas y, tras fusionarlas, ver cómo conviven en un proyecto único.

Me ha resultado muy interesante y útil para mi futuro aprender tanto el desarrollo de tecnologías que hasta ahora eran desconocidas para mí, tales como Struts y Spring, como la optimización de su uso al seleccionar solamente los módulos que resultan útiles en cada caso. Aprendes a adaptar las tecnologías a tus necesidades, gestionando mejor su uso y, a la vez, aprendes nuevas utilidades de las ya conocidas.

Otro de los puntos a resaltar es el hecho de realizar un trabajo que va a ser utilizado en la vida real. Por un lado, aumenta notablemente la motivación ya que el objetivo no se queda como hasta ahora en la obtención de una calificación. La responsabilidad crece, tienes que responder a un equipo profesional del cual formas parte lo que resulta tanto novedoso como inquietante, en definitiva es un reto. Por otro lado, es muy satisfactorio aplicar los conocimientos que has ido adquiriendo durante la carrera y, de este modo, empezar a ver los frutos de tu desarrollo académico.

Por último y no menos importante, considero que es un buen trampolín entre la vida estudiantil y laboral. Aprendes cómo está estructurada una empresa y cuál es su metodología de trabajo, tanto en tu caso personal como viendo las situaciones a las que se enfrentan tus compañeros, que teóricamente serán similares a las que tendrás que afrontar tú en el futuro. Es una buena forma de adquirir confianza y aprender a desenvolverte en el mundo empresarial antes de finalizar la carrera.

9. Referencias

- Presentando la plataforma de administración electrónica e-Dibam

http://blog.eyeos.org/es/2010/05/25/presentando-la-plataforma-de-administracion-electronica-e-dibam/

- Assistència en administració electrònica - Projecte e-dibam – Funcionalitats

http://www.diba.es/web/eadministracio/eadministracio/edibam/funcionalitats

- Dossier informatiu

http://www.diba.es/documents/190796/201633/assistenciagovernlocal-eadministracio-fitxers-dossier_informatiu_egovern-pdf.pdf

- Gestión de Expedientes

http://www.squidoo.com/gestion-expedientes

- Gestión documental

http://es.wikipedia.org/wiki/Gesti%C3%B3n_documental

- Software de gestión documental

http://es.wikipedia.org/wiki/Software_de_gesti%C3%B3n_documental

- Aplicación web

http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web

- e-Administración

http://es.wikipedia.org/wiki/E-Administraci%C3%B3n

- Sede electrónica

http://es.wikitel.info/wiki/Sede_electr%C3%B3nica

- Eclipse

http://es.wikipedia.org/wiki/Eclipse_(software)

- Java_EE

http://es.wikipedia.org/wiki/Java_EE

- Struts

http://es.wikipedia.org/wiki/Apache_Struts

- Spring

http://www.genbetadev.com/java-j2ee/spring-framework-introduccion

- Maven

http://biblioteca.uct.cl/tesis/miguel-garrido/tesis.pdf

http://www.di.uniovi.es/~dflanvin/docencia/dasdi/teoria/Transparencias/03.%20Aplicaciones%20Empresariales%20J2EE.pdf

http://es.wikipedia.org/wiki/Maven

- Tomcat

http://es.wikipedia.org/wiki/Tomcat

- MySQL

http://www.esepestudio.com/articulo/desarrollo-web/bases-de-datos-mysql/Que-es-MySQL.htm

- Alfresco

http://es.wikipedia.org/wiki/Alfresco

- Java

http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

- JSP

http://es.wikipedia.org/wiki/JavaServer_Pages

- JSTL

http://es.wikipedia.org/wiki/JavaServer_Pages_Standard_Tag_Library

- JavaScript

http://es.wikipedia.org/wiki/JavaScript

- jQuery

http://es.wikipedia.org/wiki/JQuery

- AJAX

http://es.wikipedia.org/wiki/AJAX

- JSON

http://es.wikipedia.org/wiki/JSON

- CSS

http://es.wikipedia.org/wiki/Hojas_de_estilo_en_cascada

- Modelo entidad-relación

http://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n

- Modelo de dominio

http://www-2.dc.uba.ar/materias/isoft1/is1-2005_2/apuntes/SlidesDC.pdf

A. Anexo I: Configuración Eclipse

1. Requisitos previos

Tener instalado Eclipse Java EE IDE for Web Developers (Helios Service Release 1).

2. Integración de Maven con Eclipse

Maven nos facilita la creación del proyecto en los IDEs más habituales. Vamos a ver aquí como usar Maven para montar nuestro proyecto Maven en Eclipse.

Para crear nuestro proyecto eclipse, nos basta con ejecutar el comando mvn eclipse:eclipse en el directorio raíz de nuestro proyecto. Esto creará los ficheros .project y .classpath que definen un proyecto de Eclipse. Una vez creados los ficheros, nos vamos a Eclipse e importamos el proyecto. En Eclipse seleccionamos "File" → "Import" → "General" → "Existing projects into workspace", nos aparecerá una ventana en la que podemos seleccionar el directorio raíz de nuestro proyecto y al aceptar, aparecerá nuestro proyecto listo para importar. Lo seleccionamos y aceptamos.

Una vez importado el proyecto, hay una cosa que debemos hacer una sola vez en nuestro workspace de Eclipse, que es definir una variable que apunte al sitio donde Maven guarda los jars que se descarga de internet. Esto se hace en "Window" → "Preferences" → "java" → "Build path" → "Classpath variables". Pulsamos el botón "new" para crear una nueva variable de nombre M2_REPO y que apunte al sitio donde Maven tiene los jars. $HOME/.m2/repository es la ubicación por defecto.

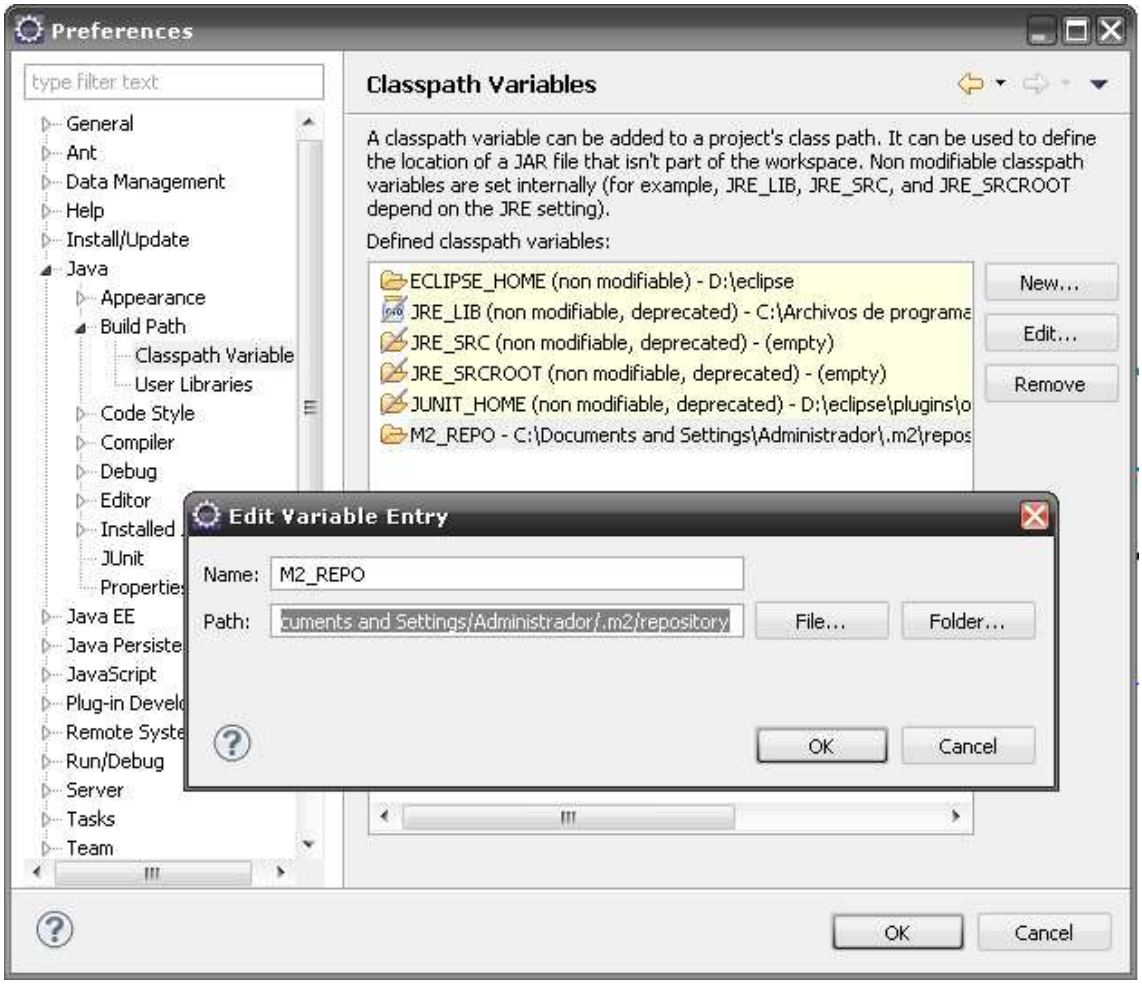


Figura 69. Configuración repositorio Maven en Eclipse

Si todo va bien, debemos ver como se compila nuestro proyecto y no hay ningún problema de compilado.

Si a lo largo del trabajo sobre nuestro proyecto añadimos más dependencias en el pom.xml, debemos volver a crear nuestros proyectos Eclipse.

mvn eclipse:clean borrará los ficheros del proyecto y mvn eclipse:eclipse volverá a crearlos. Luego, en Eclipse, debemos refrescar nuestro proyecto.

B. Anexo II: Ejemplo de compilación

En primer lugar, se debe abrir una ventana de línea de comandos y situarse en la carpeta donde se encuentra el proyecto.

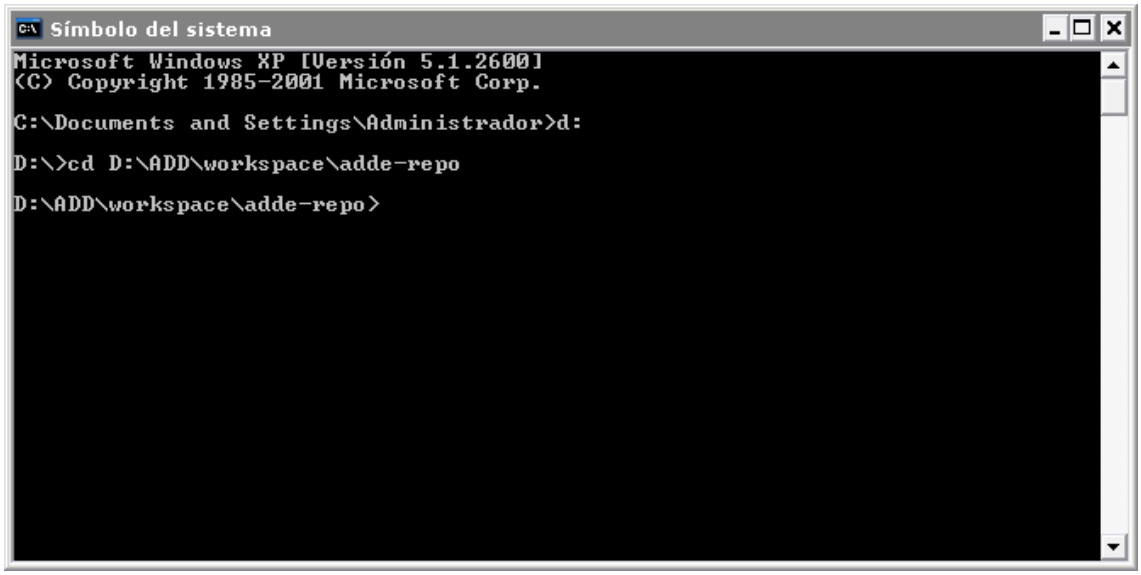


Figura 70. Ventana de línea de comandos

Seguidamente, compilaremos la parte común del proyecto, esta incluye las clases elementales del modelo y algunas fuciones útiles. Para ello, hay que situarse en la carpeta donde se encuentra esta parte y escribir la siguiente instrucción:

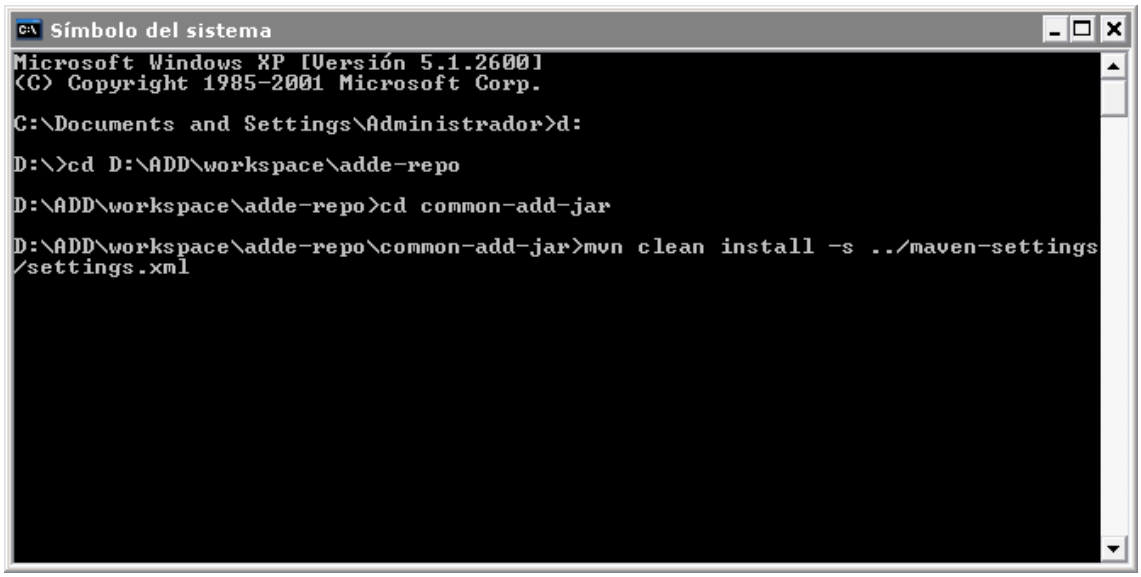


Figura 71. Instrucción de compilado de parte común

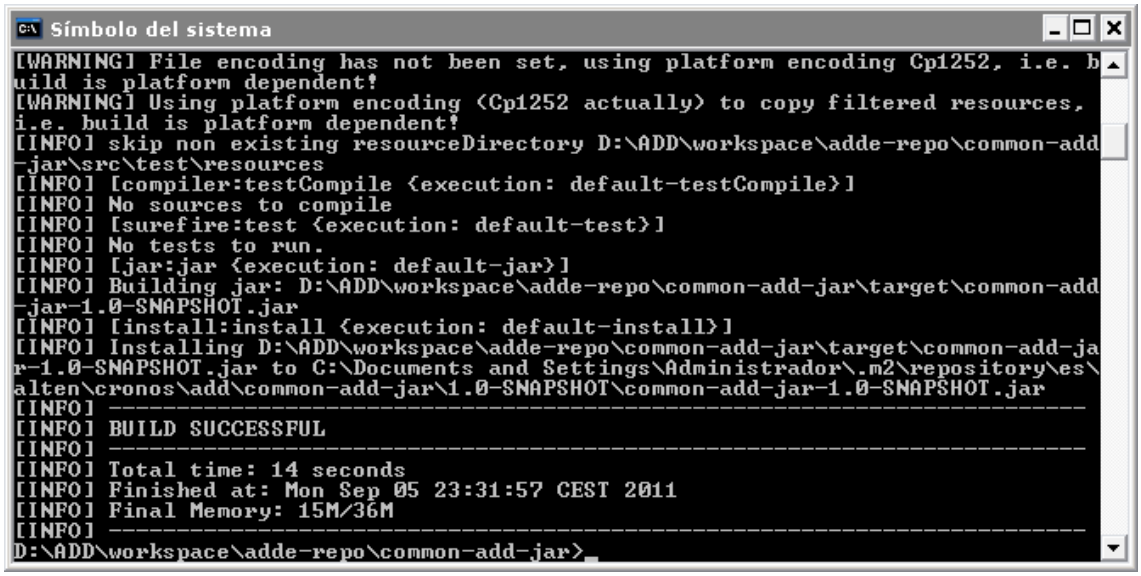


Figura 72. Resultado compilación parte común

Mencionar que en la carpeta maven-settings se encuentra el fichero settings.xml, donde configurar el usuario y la contraseña con los que vamos a acceder al repositorio de Maven.

Por último, se debe compilar el resto del proyecto incluyendo el .jar creado en el paso anterior. Para esto, hay que situarse en la carpeta donde se encuentra el resto del proyecto y ejecutar la siguiente instrucción:

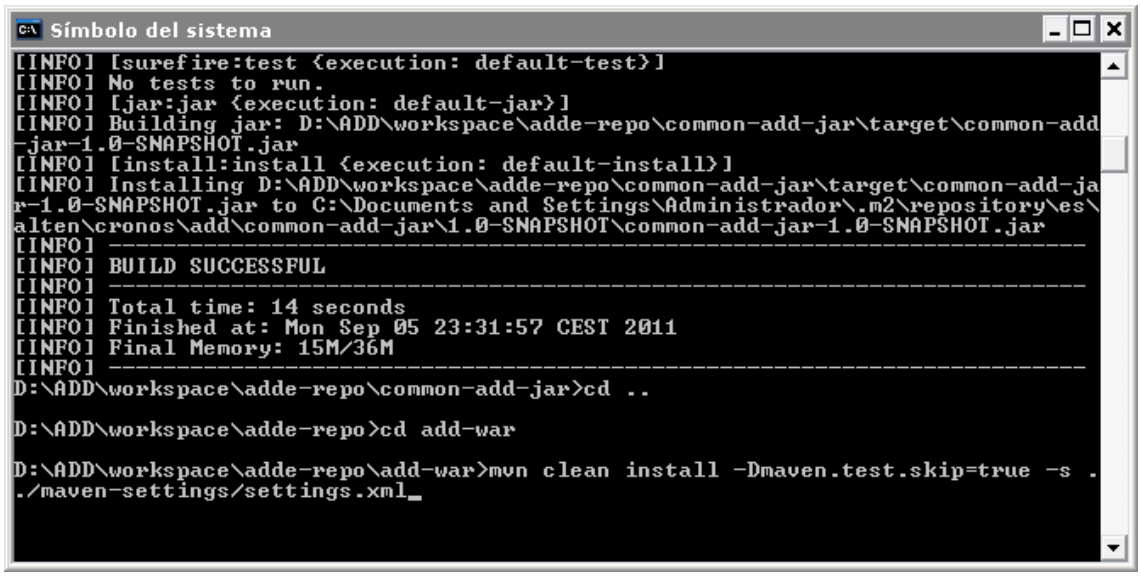


Figura 73. Instrucción de compilado del proyecto

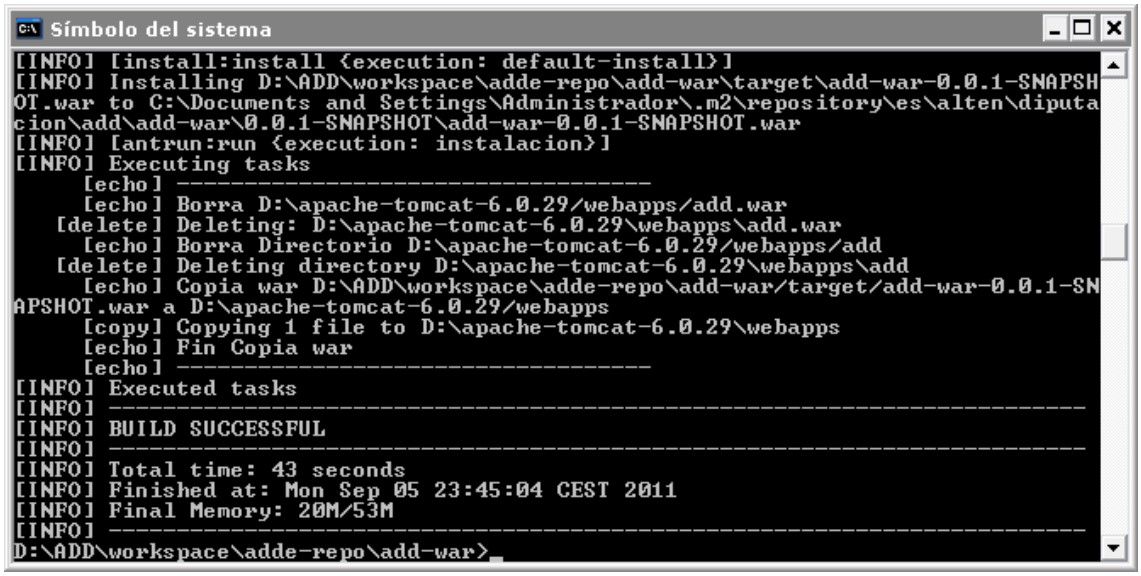


Figura 74. Resultado compilación proyecto

Se observa que tras la compilación se deja el .war del proyecto automáticamente en la carpeta webapps del servidor web con soporte de servlets y JSP's Tomcat.