universidad del país vasco

euskal herriko unibertsitatea

Facultad de Informática **Informatika Fakultatea**

TITULACIÓN: Ingeniería Informática

Acceso Web para el gestor documental EPDM

Alumno/a: D./Dña. Julen Salgado Tomas

Director/a: D./Dña. Ana Rosa Sanchez Ortega

Proyecto Fin de Carrera, mayo de 2014

RESUMEN

Hoy en día, cabe destacar la gran envergadura de los proyectos en distintas áreas de la ingeniería. Cada vez es más necesario llevar una gestión adecuada de todos y cada uno de los documentos desarrollados a lo largo del ciclo de vida del proyecto. Para este fin, existen distintas herramientas que ofrecen una gestión automatizada. En el ámbito de la ingeniería mecánica, destaca la importancia de un buen control en el diseño de las piezas y ensamblajes.

Ibermática es una empresa que distribuye *SolidWorks*, un programa de diseño asistido para modelado mecánico que permite modelar piezas o conjuntos y crear, a partir de ellos, planos técnicos con información necesaria para la producción. Por otra parte, *SolidWorks Enterprise PDM* es una solución para el control de los datos de diseño creados con SolidWorks, en otras palabras, es un gestor documental que garantiza que toda la información se guarda de forma segura y permite mantener un control de versiones.

En este proyecto de Fin de Carrera se ha creado una versión web y otra para dispositivos móviles de *SolidWorks Enterprise PDM*, cuya finalidad es automatizar la gestión de datos de diseño. Actualmente existe una versión para la web que solamente funciona en exploradores que soportan Active X, es decir, Internet Explorer.

La nueva herramienta se apoya sobre la *API* de *SolidWorks* y se diferencia en que la parte del servidor se divide en dos capas. Una es un adaptador que se comunica con la base de datos mediante la *API* y ofrece un *Servicio Web* usando estándares como *JSON (JavaScript Object Notation)* para los datos transmitidos. La otra parte, es la interfaz web que se encarga de comunicarse con este adaptador y dar dinamismo al contenido.

Gracias a esta arquitectura, en un futuro será posible crear interfaces independientes que usen el servicio (adaptador), y hacer la herramienta multiplataforma desarrollando aplicaciones para la web, para teléfonos móviles o programas nativos.

AGRADECIMIENTOS

Gracias a todas las personas que han hecho posible que haya finalizado el proyecto. A Ana Rosa Sánchez como directora del proyecto por haber encaminado la gestión y la planificación del mismo. Al equipo de *Ibermática* por haber aportado gran variedad de soluciones prácticas a los problemas y haberme hecho sentir como un trabajador más de la empresa. A todos mis amigos del pueblo que siempre me han preguntado por el estado de mi trabajo o me han apoyado y a todos aquellos que me acompañaron en clase.

A toda mi familia por haber estado encima y porque sé que siempre estarán ahí. Especialmente a mis padres Enrique y Marga, por haber dejado dedicarme a lo que me gusta y a mi hermano Markel con el que he tenido la suerte de compartir uno de mis sueños.

Finalmente, a todas las personas maravillosas que he conocido y a mi novia Ainhoa que, a pesar de la distancia, ha sabido sacarme una sonrisa cada día. ¡Gracias!

ÍNDICE

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En este primer capítulo se presenta el contexto sobre el que se ha planteado el proyecto. El proyecto gira en torno a la implementación de un acceso web utilizando la *API* de *SolidWorks Enterprise PDM*, de modo que se expondrán los antecedentes de esta aplicación

1.1 CONTEXTO

SolidWorks es un programa de diseño asistido por computadora para modelos mecánicos. El programa permite modelar piezas o conjuntos y extraer planos técnicos o cualquier tipo de información relacionada. Es decir, el diseñador se abstrae de crear todos estos documentos técnicos ya que la aplicación lo realiza de manera automatizada.

Para los diseñadores, el hecho de trabajar en conjunto manejando numerosas piezas u ensamblajes resulta muy pesado. De ahí nace *SolidWorks Enterprise PDM*, cuya función principal es ofrecer una solución a la gestión de datos para pequeñas y grandes organizaciones.

Esta herramienta es muy usada en las empresas ya que permite reducir el tiempo de dedicación de los trabajadores aumentando así el rendimiento, fomentando el diseño innovador y reduciendo los costes. El área de ingeniería de Ibermática se encarga de dar soporte en esta herramienta de diseño y extender las funcionalidades de otros productos de SolidWorks en base a las necesidades de sus clientes principales.

1.2 ANTECEDENTES

1.2.1 SOLIDWORKS ENTERPRISE PDM

SolidWorks Enterprise PDM es un programa integrado completamente en el explorador de Windows que automatiza la gestión de datos de diseño. Utiliza un almacenamiento centralizado proporcionando acceso a distintos usuarios y permisos. Sincroniza los datos e integra un control de versiones para evitar pérdidas de datos.

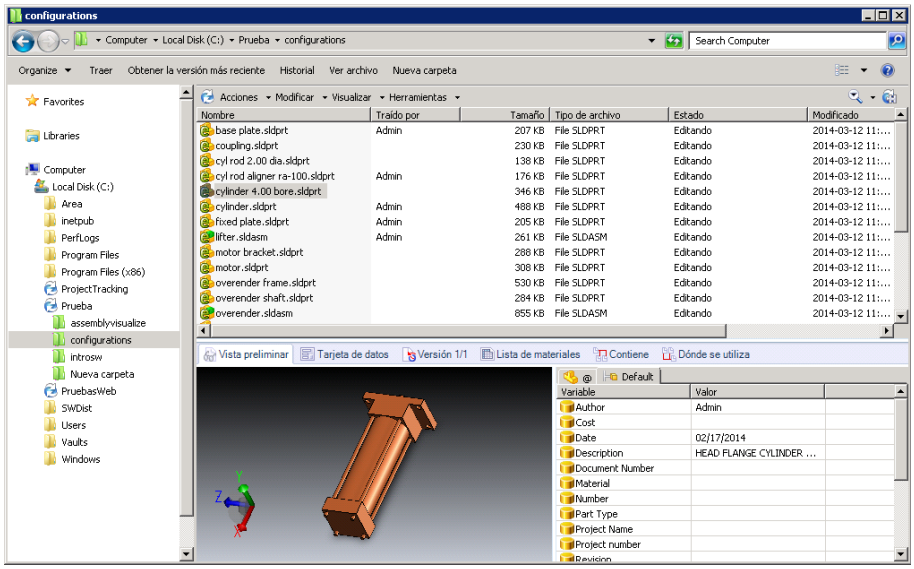

Ilustración 1: SolidWorks Enterprise PDM

Además, permite asignar flujos de trabajo mediante los cuales la empresa puede automatizar y controlar los procesos de desarrollo de sus productos. Se integra perfectamente con aplicaciones SolidWorks y es compatible con otras aplicaciones como CAD.

SolidWorks Enterprise PDM trabaja con almacenes y vistas locales. El servidor cuenta con un servidor de archivado donde guarda todos los almacenes con todos sus documentos y sus respectivas versiones. También administra una base de datos con el resto de la información necesaria. Los usuarios crean vistas locales en sus equipos que replican la información del almacén deseado en su sistema local. De esta forma, los usuarios son capaces de realizar cambios en los archivos locales antes de registrar y subir dichos cambios al servidor.

Entre las acciones más destacadas que ofrece al usuario se encuentran las siguientes:

- **Obtener versión más reciente:** Esta acción permite al usuario obtener la versión más reciente guardada en el servidor. Sobrescribe cualquier versión y cambio realizado en el archivo local.
- **Obtener versión:** Obtiene la versión seleccionada del archivo, sobrescribiendo el archivo local.
- **Traer:** Bloquea el archivo para que nadie pueda realizar cambios en el, solo el usuario que lo ha traído tendrá permisos de escritura. Los demás usuarios tendrán permisos de lectura pero no verán los cambios realizados hasta que el usuario que lo ha traído registre los cambios. Un archivo traído siempre estará traído en su versión más reciente, por ello, esta acción obtiene la última versión antes de bloquear el archivo.
- **Deshacer traer:** Desbloquea el archivo traído deshaciendo todos los cambios que se han hecho desde el último registro.
- **Registrar:** Registra todos los cambios que se han hecho en el archivo y crea una nueva versión del mismo, el cual será la versión más recuente. En caso de registrar un archivo traído en el que no se han realizado cambios, se ejecutará automáticamente la acción de deshacer traer.
- **Cambiar estado:** Cambia el estado del archivo a otro estado del mismo flujo de trabajo.

SolidWorks EPDM es capaz de administrar distintos flujos de trabajo creados por el usuario. Cada flujo de trabajo contiene estados y transiciones por los que pasará el archivo asociado. La siguiente ilustración muestra el flujo de trabajo predeterminado de *EPDM*.

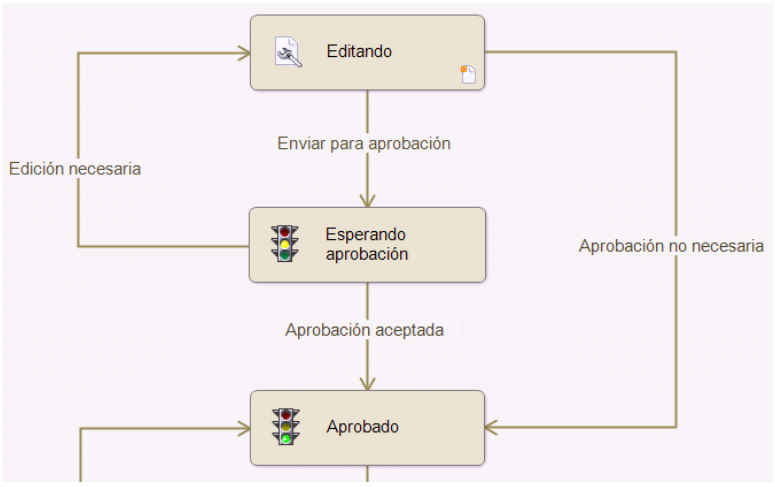

Ilustración 2: Flujo de trabajo predeterminado

Los archivos asociados a este flujo de trabajo podrán encontrarse en cualquiera de los estados definidos y el usuario será capaz de cambiar el estado mediante las transiciones. Además, ofrece la posibilidad de realizar determinadas acciones en los cambios de estado, tales como; imprimir el documento, cambiar la revisión, ejecutar un programa etc.

Otra característica interesante de esta herramienta es el uso de variables. Las variables son datos asociados a los archivos que se guardan en la base de datos. Es posible representar y cambiar el valor de dichas variables mediante el uso de tarjetas de datos. Por ejemplo, la siguiente ilustración muestra la tarjeta de datos asociada a un archivo Word.

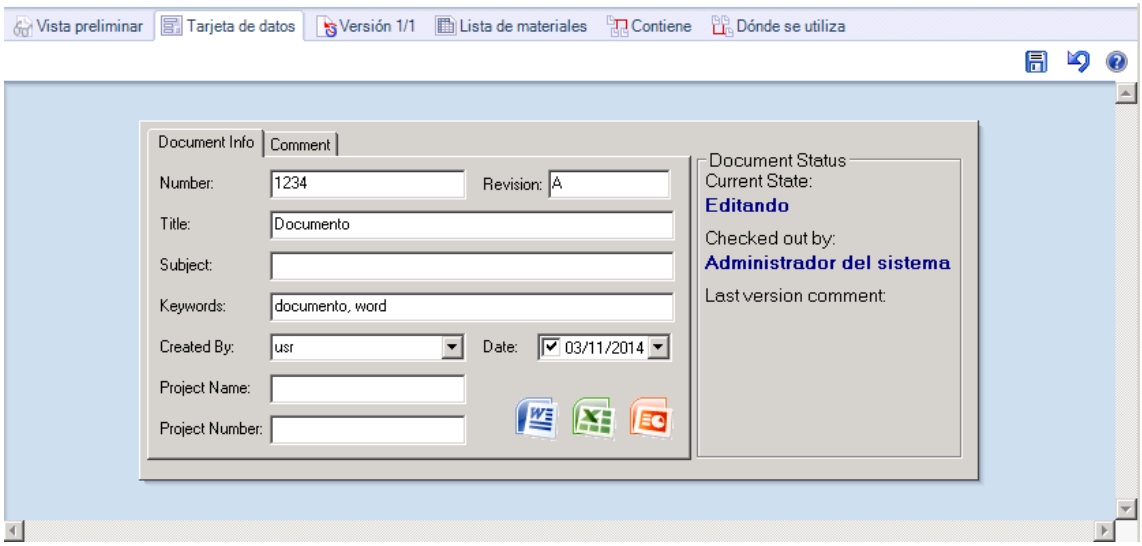

Ilustración 3: Tarjeta de datos EPDMIlustración 4

Como se puede ver, las tarjetas de datos están formadas por controles básicos que referencian a distintas variables. Cada cambio realizado en estos controles afectará en el valor de la variable, pero para poder modificar la tarjeta es necesario que el archivo esté traído, de lo contrario la tarjeta se muestra deshabilitada.

Lo interesante de esto es poder hacer búsquedas personalizadas filtrando los datos de todas estas variables. Por lo tanto, *SolidWorks Enterprise PDM* también ofrece la posibilidad de hacer búsquedas avanzadas mediante tarjetas de búsquedas parecidas a las tarjetas de datos.

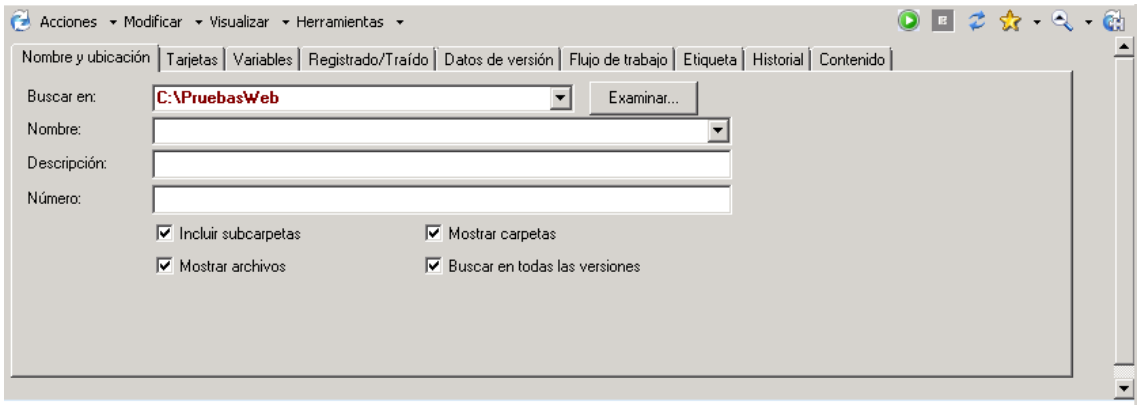

Ilustración 5: Tarjeta de búsqueda EPDM

Al igual que en las tarjetas convencionales, los controles de las tarjetas de búsqueda también están relacionadas con las variables y *EPDM* lanza una búsqueda filtrando el resultado por el valor de estas variables.

1.2.1 SOLIDWORKS EPDM WEB

SolidWorks Enterprise PDM incluye en la actualidad la posibilidad de instalar un acceso web en el servidor. Ofrece acciones suficientes para realizar las tareas básicas sobre almacenes de *PDM*.

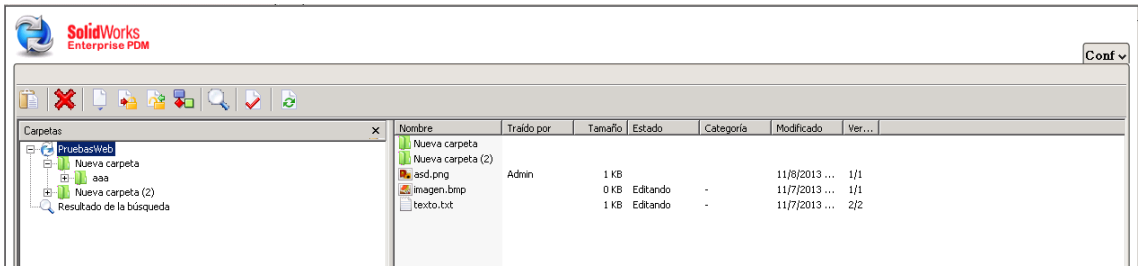

Ilustración 6: SolidWorks EPDM Web 1.0

El problema es que el uso de la tecnología *Active X*, limita el funcionamiento solamente al navegador *Internet Explorer* y por consiguiente, a los sistemas Microsoft Windows. Además, este software requiere una instalación de una versión diferente de PDM por cada máquina, por lo que el cambio de versiones llega a ser una tarea embarazosa.

1.3 NUESTRA PROPUESTA

El uso de esta aplicación conlleva las limitaciones expuestas anteriormente, que dificulta el trabajo a realizar tanto por el diseñador como por el administrador del sistema. Estas razones han sido suficientes para concluir que es necesaria la implementación de un nuevo sistema que corrija dichos problemas. Cada vez es mayor la influencia que la Web ha tenido sobre las nuevas tecnologías, y la compatibilidad de diferentes plataformas es un gran punto a favor en las aplicaciones.

Con este proyecto se creará un servicio que ofrecerá las funciones básicas del actual PDM mediante la web. Dicho servicio se implementará en la plataforma *Windows Communication Foundation*, que permitirá programar de forma rápida una aplicación distribuida a partir de una arquitectura basada en varios niveles. El desarrollo se hará con el lenguaje de programación *C#* estandarizado por *Microsoft*, el que nos permitirá acceder a los objetos y las funciones que nos ofrece la *API* de PDM.

Gracias a este servicio, cualquier interfaz web podrá consumir las funciones de PDM y dar dinamismo al contenido. Por lo tanto, también se creará un sitio web tanto para PC como para teléfonos móviles, que dará acceso a los almacenes situados en un servidor permitiendo la identificación simultánea de diferentes usuarios. De todas formas, se dividirá la implementación del servicio y de la página web para que, en un futuro, las nuevas aplicaciones puedan hacer uso del servicio y crear interfaces para otra plataforma.

1.4 ORGANIZACIÓN DEL DOCUMENTO

La estructura de esta memoria comienza, a partir de este punto, con el documento de objetivos del proyecto que se presenta en el capítulo 2 “Documento de objetivos del proyecto”. En él se muestran los objetivos que derivan del producto principal, la nueva versión Web de *Solidworks Enterprise PDM*. De la

misma forma, se presenta el alcance del proyecto, que determina lo que se tiene que hacer para lograr la aplicación web. Así, se expondrán las fases y tareas del proyecto en una Estructura de Descomposición de Trabajo, cada una con su respectiva descripción. También se presenta la planificación inicial que recoge las horas estimadas para cada tarea y un diagrama Gantt que muestra el tiempo de dedicación previsto para estas actividades a lo largo del tiempo. Por último, se realiza el análisis los riesgos que se deben tener en cuenta durante el desarrollo, del cual deriva el análisis de factibilidad para determinar si es posible llevar a cabo el proyecto.

En el capítulo 3 "Desarrollo técnico" se presenta el análisis, que recoge principalmente los requisitos del proyecto organizados en un diagrama de casos de uso. Se presenta un diagrama para cada tipo de usuario que hará uso de la aplicación. También se especifica cada uno de ellos mediante unas tablas que describen los flujos normales y alternativos. Finalmente, se muestra la arquitectura del sistema propuesta y la tecnología que se ha usado para llevar a cabo este proyecto.

En la siguiente sección, capítulo 4 “Diseño” se recoge el diseño que se ha seguido en la implementación de los servicios web que se alojarán en la parte del servidor. Se representará mediante diagramas de secuencia y el diagrama de clases general. Éstos muestran la relación que hay entre los objetos en la lógica de negocio y cómo interaccionan entre sí.

El capítulo 5 “Implementación” presenta las características más relevantes del desarrollo del proyecto, es decir, de la interfaz y la lógica de negocio. Se muestran los prototipos de la interfaz web y la interfaz móvil mediante diversas ilustraciones. Además, se resumen los problemas técnicos que se han tenido durante el desarrollo y las soluciones que se han propuesto para superarlos. Estas soluciones serán imprescindibles a la hora de tener que implantar el sistema en un cliente.

En el siguiente capítulo “Pruebas” se especifican las pruebas que se han realizado sobre el producto desarrollado. Inicialmente se presenta el diseño de las pruebas que se han ideado con el objetivo de encontrar errores potenciales que pudieran arruinar el funcionamiento de la aplicación. En el siguiente punto se muestran los resultados de las ejecuciones de dichas pruebas con su respectivo nivel de éxito.

A continuación, en el capítulo 7 "Conclusiones y líneas futuras" se analizan los objetivos del proyecto para determinar si estos se han cumplido. Además, se incluyen algunas conclusiones tanto generales como personales surgidas durante la realización del proyecto y tras su finalización. Finalmente, se sugieren algunas posibles mejoras que se podrían realizar al programa en futuros proyectos.

El último capítulo es el correspondiente a las "Referencias" en el que se muestran las fuentes consultadas para recabar la información necesaria para la realización del proyecto y de la presente memoria.

Y para finalizar esta memoria, se incluyen anexos, los cuales contienen las actas recogidos durante todas las reuniones realizadas tanto en la empresa como con la directora del proyecto, y otros documentos de ayuda como la firma de un *Applet* de *Java*.

2. DOCUMENTO DE OBJETIVOS DEL PROYECTO

En el presente capítulo, se expondrá el Documento de Objetivos del Proyecto, donde se definirá el alcance del proyecto y la planificación de las tareas a lo largo del ciclo de desarrollo. Además, se hará una evaluación exhaustiva de cada riesgo con su respectivo plan de contingencias. Finalmente, se incluye el análisis de factibilidad que valora si la realización del proyecto es posible.

2.1 OBJETIVOS DEL PROYECTO

La nueva versión de *PDM Web* permite la ejecución y el uso de *SolidWorks Enterprise PDM* en diferentes plataformas, ofreciendo los mismos servicios gracias a la adaptación a las nuevas tecnologías y estándares. Para ello, se han definido los siguientes objetivos:

- **Objetivo 1: Investigación de las nuevas tecnologías y estándares.**
  Se investigarán posibles tecnologías que puedan usarse para ofrecer un servicio multiplataforma. Tras analizar cada una de las tecnologías, se expondrán las propuestas que solucionen problemas de arquitectura para decidir el camino a seguir.
- **Objetivo 2: Implementación de un servicio web que ofrezca las funcionalidades de EPDM.**
  Se implementará un servicio web mediante una tecnología que soporte la API de *SolidWorks Enterprise PDM*. Se basará en tecnologías de Microsoft, ya que la API ofrece sus métodos en lenguajes de programación como *Visual Basic* y C#. De todos modos, los datos de entrada y salida de los servicios seguirán un formato estándar que soporte numerosas tecnologías.
- **Objetivo 3: Desarrollo de una interfaz web que actúe sobre los servicios web.**
  Se creará una interfaz web dinámica sobre nuevas tecnologías como *HTML5*, *JavaScript* y la librería *JQuery* que se comunicará con los servicios anteriormente creados mediante la técnica *Ajax*, ofreciendo al usuario una forma intuitiva de usar las funciones de *PDM*.
- **Objetivo 4: Desarrollo de una aplicación ligera para dispositivos móviles en HTML5 que explote los servicios.**
  Se creará un cliente ligero para dispositivos móviles que se encargará de visualizar solo la información más relevante. Al igual que la interfaz web, se utilizarán las mismas tecnologías pero el desarrollo se realizará sobre el *framework JQueryMobile*. A pesar de ser un cliente *EPDM*, esta aplicación contará con funciones más limitadas que la versión web.

2.2 ALCANCE DEL PROYECTO

Tras describir los objetivos del proyecto, se expone el alcance del proyecto. Teniendo en cuenta los objetivos del proyecto, al finalizar el proyecto se obtendrán: un servidor que ofrece servicios para acceder a las funcionalidades de *EPDM*, una aplicación web y una aplicación específica para dispositivos móviles.

Los productos obtenidos serán una primera versión temprana, por lo que no se realizará la implantación en ningún cliente. De hecho, tras lograr la primera versión, la empresa seguirá desarrollando y mejorando los servicios.

Para llevar a cabo este proyecto, se listarán las fases y tareas necesarias para desarrollar la aplicación final y se organizarán usando una *Estructura de Descomposición de Trabajo*.

2.2.1 ESTRUCTURA DE DESCOMPOSICIÓN DE TRABAJO

El propósito de la *Estructura de Descomposición de Trabajo* es organizar y definir el alcance total del proyecto. En él se especifican las tareas necesarias para llevar a cabo el proyecto, todo lo que no se muestra en el diagrama queda automáticamente fuera del mismo.

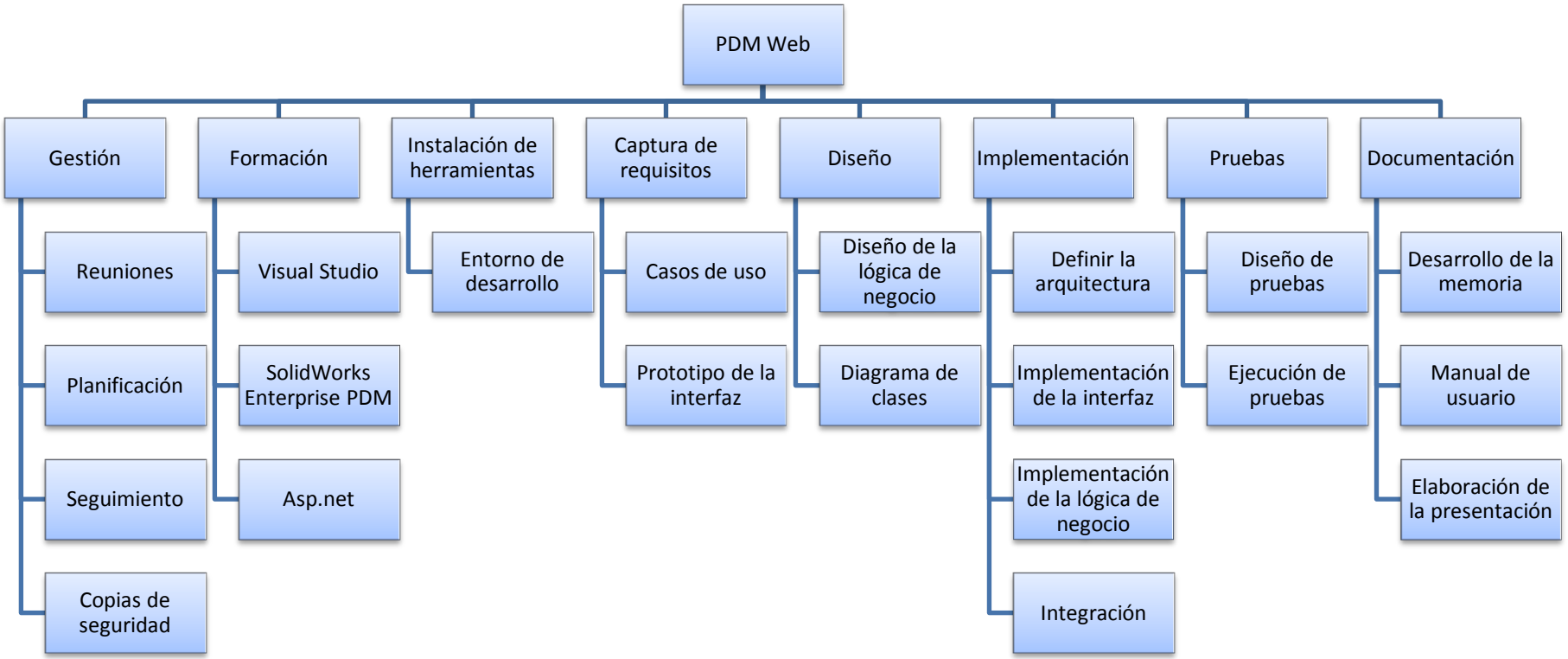

Ilustración 7: Estructura de Descomposición de Trabajo

2.2.2 FASES Y TAREAS

Esta sección recoge las distintas fases del proyecto así como las tareas que se realizarán en cada una de ellas.

GESTIÓN

La gestión del proyecto se llevará a cabo conforme a las siguientes tareas durante el ciclo completo del proyecto:

- **Reuniones:** Se exigirá la participación de las reuniones que se convoquen en la empresa para tomar aportar soluciones y alternativas más efectivas para avanzar en el proyecto. Las reuniones con la directora del proyecto se realizarán mensualmente en la Universidad, por otra parte las reuniones de la empresa se realizarán tras una convocatoria previa.
- **Planificación:** El proyecto partirá de una planificación inicial que tendrá como fruto, completar los objetivos del proyecto. Sin embargo, a lo largo del proyecto podrían ir cambiando los plazos planificados inicialmente.
- **Seguimiento del proyecto:** A lo largo del proyecto se realizará un seguimiento semanal de lo realizado hasta el momento y se harán cambios en la planificación en caso de ser necesario.
- **Copias de seguridad:** Regularmente se realizarán copias de seguridad para minimizar las desastrosas consecuencias que causaría la pérdida de datos parcial o completa.

FORMACIÓN

Teniendo en cuenta la tecnología inicial a usar y los conocimientos que se creen necesarios para llevar a cabo los objetivos, la formación se centrará principalmente en los siguientes temas:

- **Formación de Visual Studio 2010:** Se realizará una formación en esta herramienta mediante la práctica y trabajos externos asignados por la empresa. Esta tarea se refiere a la formación de la herramienta en sí, e incluso de los lenguajes de programación que soporta.
- **Estudio de SolidWorks Enterprise PDM:** La formación en esta área correrá por parte de la empresa que realizará un curso rápido de los principios básicos de *PDM*. Además, se deberá estudiar el libro de los fundamentos del *API* de programación.
- **Formación en Asp.net:** En un principio se estudiará y se comprobará la capacidad de esta tecnología para lograr los propósitos del proyecto. Sin embargo, más adelante puede que se usen otros métodos.

No se tiene en cuenta la formación de otras herramientas ajenas al propósito del proyecto, dado que se considera que no se necesita estudio adicional en ellas, tales como; *Microsoft Word* (para el desarrollo de los documentos), *Excel* (para la gestión del proeycto), *LibreOffice Draw* o *Dia* (siendo estas dos últimas, herramientas para el diseño de diagramas de flujo).

INSTALACIÓN DEL ENTORNO DE TRABAJO

Se procederá a instalar todas las herramientas que se cree que serán necesarias para el desarrollo del proyecto, entre ellas:

- **SolidWorks Enterprise PDM 2013:** para poder usar la *API* de programación y analizar sus funcionalidades para poder replicarlas en la aplicación web a desarrollar.
- **Visual Studio 2010:** para poder programar los servicios web que usen la *API* de *PDM* y desarrollar páginas dinámicas que se ejecuten en el servidor.
- **Aptana / Eclipse:** Aptana es un entorno basado en Eclipse para programar en puro HTML, CSS, JavaScript y la librería JQuery con el que se crearán las páginas web dinámicas en el cliente. Eclipse se usará para programar en Java y usar algunas de sus funciones en caso de necesitarlas.
- **Internet Information Services:** para proporcionar un servidor web en el que alojar todas las páginas y servicios web creados.
- **Internet Explorer, Google Chrome, Mozilla Firefox:** navegadores para poder visualizar el contenido creado en diferentes plataformas y mantener la compatibilidad.
- **Microsoft Office / Libre Office:** para desarrollar la parte de la documentación del proyecto y crear diagramas de diseño.

CAPTURA DE REQUISITOS

Al no haber un cliente específico de quien obtener los requisitos, la organización tomarás riendas para exigir las funcionalidades que deberá tener el producto y se harán reuniones para tomar decisiones al respecto.

- **Definición de los casos de uso:** Se definirán los casos de uso a partir de los requisitos de la aplicación. A lo largo del desarrollo del proyecto los requisitos podrán ir variando y será necesario definir nuevos casos de uso. Para esta tarea se usarán los diagramas de casos de uso.
- **Prototipo de la interfaz:** Antes de crear el programa se trabajará un prototipo inicial que recogerá los requisitos básicos del proyecto, sin reparar en cuestiones de diseño. De todas formas, el hecho de probar diversas tecnologías para lograr el producto requerirá realizar más de un prototipo según va avanzando el desarrollo.

DISEÑO

Antes de comenzar el desarrollo se realizará el diseño, que constará principalmente de las siguientes tareas:

- **Diseño de la lógica de negocio:** Se realizará el diseño de la lógica de negocio. Para ello, será necesario crear diagramas de secuencia para cada caso de uso para, posteriormente, facilitar la programación de los servicios.
- **Diagrama de clases:** Los casos de uso que se vayan a implementar se clasificarán en clases con diferente funcionalidad. Se mostrará un diagrama completo de todas ellas con sus respectivas relaciones.

IMPLEMENTACIÓN

Tras el diseño, se procederá a implementar lo que se ha diseñado. En el caso de tener que mejorar la aplicación incluyendo nuevas funcionalidades, se tendrá que volver a realizar el diseño antes de pasar a las tareas de implementación.

- **Definir la arquitectura:** Se especificará la arquitectura que se usará para desarrollar la aplicación; que incluirá módulos de datos, servidores y clientes.
- **Implementación de la interfaz:** En esta tarea se desarrollará la interfaz del programa, que será una página web desarrollada en HTML y JavaScript haciendo uso de los servicios.
- **Implementación de la lógica de negocio:** La lógica de negocio hace referencia a los servicios que se ofrecerán en un servidor. Estos servicios se desarrollarán en esta tarea mediante el lenguaje *C#* que será necesario para utilizar la *API* de *SolidWorks EPDM*.
- **Realizar la integración:** Esta tarea consistirá básicamente en integrar los dos módulos desarrollados para que la interfaz pueda comunicarse correctamente con el servidor y pueda consumir los servicios.

PRUEBAS

La fase de pruebas se realizará al final del desarrollo. Aunque a lo largo del proyecto se irán probando las funcionalidades que se implementan, no se tendrán en cuenta ya que estas no formarán parte del plan de pruebas. Para lograr una aplicación funcional se llevarán a cabo las siguientes tareas:

- **Diseño de las pruebas:** Se realizará el diseño de las pruebas que tendrán como objetivo encontrar errores en el programa.
- **Ejecución de las pruebas:** Se procederá a ejecutar las pruebas diseñadas anteriormente. Se considerarán pruebas exitosas aquellas que han encontrado problemas.

DOCUMENTACIÓN

Esta fase hace referencia a todas las tareas que tengan como objetivo crear la documentación del proyecto. Entre ellas:

- **Desarrollo de la memoria:** Se redactará la memoria durante todo el ciclo de desarrollo del proyecto, que recogerá y resumirá todos los aspectos del mismo.
- **Manual de usuario:** Se creará un manual que guiará al usuario final a utilizar correctamente la aplicación.
- **Elaborar la presentación:** En esta tarea se elaborará, en caso de requerirlo, la presentación que expondrá las ideas principales del proyecto ante un tribunal de la Universidad.

2.3 PLANIFICACIÓN

En el siguiente apartado se mostrará la planificación inicial de la que partirá el proyecto. Se expondrá la estimación del esfuerzo que se cree que requerirá cada una de las fases y tareas. También se planificará la fecha inicial y final de cada una; información que posteriormente se clasificará en un diagrama *Gantt*.

Dado que no se han establecido fechas de entrega, se han repartido las tareas a lo largo del tiempo de desarrollo del proyecto, teniendo en cuenta el esfuerzo que exigirá cada una de ellas.

2.3.1 ESTIMACIONES

Como ya se ha comentado, la siguiente tabla recoge el esfuerzo que se ha estimado para cada tarea. La estimación se ha realizado en horas, y también se incluyen las fechas de inicio y finalización para cada una de ellas.

Se han asignado las horas desde la experiencia del autor, teniendo en cuenta los proyectos desarrollados hasta el momento. Pero el hecho de ser un proyecto a desarrollar en una empresa, se puede apreciar, por ejemplo, que se invierten más horas desarrollando el producto que documentándolo.

Fases	Tareas	Fecha inicial	Fecha final	Horas estimadas
Gestión	Reuniones	14/10/2013	14/04/2014	15:00
	Planificación	14/10/2013	25/10/2013	6:00
	Seguimiento	14/10/2013	14/04/2014	25:00
	Copias de seguridad	14/10/2013	14/04/2014	10:00
Formación	Visual Studio	14/10/2013	01/11/2013	40:00
	SolidWorks Enterprise PDM	24/11/2013	04/12/2013	20:00
	Asp.net	01/12/2013	15/12/2013	16:00
Instalación	Entorno de desarrollo	20/10/2013	22/10/2013	8:00
Captura de requisitos	Casos de uso	12/11/2013	14/01/2014	8:00
	Prototipo de la interfaz	12/11/2013	01/02/2014	8:00
Diseño	Diseño de la lógica de negocio	15/11/2013	02/02/2014	24:00
	Diagrama de clases	15/11/2013	17/11/2013	16:00
Implementación	Definir la arquitectura	18/11/2013	20/11/2013	4:00
	Implementación de la interfaz	15/11/2013	01/04/2014	80:00
	Implementación de la lógica de negocio	20/11/2013	01/04/2014	160:00
	Integración	01/04/2014	02/04/2014	8:00
Pruebas	Diseño de pruebas	13/03/2014	15/03/2014	8:00
	Ejecución de pruebas	15/03/2014	30/03/2014	24:00
Documentación	Desarrollo de la memoria	01/04/2014	13/04/2014	32:00
	Manual de usuario	05/04/2014	08/04/2014	24:00
	Elaboración de la presentación	13/04/2014	14/04/2014	8:00
TOTAL		14/10/2013	14/04/2014	544:00

Tabla 1: Tabla de estimaciones

2.3.2 DIAGRAMA GANTT

Líder de proyecto: Julen
Fecha de inicio: 14/10/2013 lunes

ID	Tareas	Líder	Inicio	Fin	Duración (Días)	% Completado	Días de trabajo	Días completado	Días restantes
1	**Gestión**	Julen	14/10/2013	14/04/2014	183	0%	131	0	183
1.1	Reuniones		14/10/2013	14/04/2014	183	0%	131	0	183
1.2	Planificación		14/10/2013	25/10/2013	12	0%	10	0	12
1.3	Seguimiento		14/10/2013	14/04/2014	183	0%	131	0	183
1.4	Copias de seguridad		14/10/2013	14/04/2014	183	0%	131	0	183
2	**Formación**	Julen	14/10/2013	15/12/2013	63	0%	45	0	63
2.1	Visual Studio		14/10/2013	01/11/2013	19	0%	15	0	19
2.2	SolidWorks EPDM		24/11/2013	04/12/2013	11	0%	8	0	11
2.3	Asp.net		01/12/2013	15/12/2013	15	0%	10	0	15
3	**Instalación**	Julen	20/10/2013	22/10/2013	3	0%	2	0	3
3.1	Entorno de desarrollo		20/10/2013	22/10/2013	3	0%	2	0	3
4	**Captura requisitos**	Julen	12/11/2013	01/02/2014	82	0%	59	0	82
4.1	Casos de uso		12/11/2013	14/01/2014	64	0%	46	0	64
4.2	Prototipo de interfaz		12/11/2013	01/02/2014	82	0%	59	0	82
5	**Diseño**	Julen	15/11/2013	02/02/2014	80	0%	56	0	80
5.1	Lógica de negocio		15/11/2013	02/02/2014	80	0%	56	0	80
5.2	Diagrama de clases		15/11/2013	17/11/2013	3	0%	1	0	3
6	**Implementación**	Julen	15/11/2013	02/04/2014	139	0%	99	0	139
6.1	Definir arquitectura		18/11/2013	20/11/2013	3	0%	3	0	3
6.2	Interfaz		15/11/2013	01/04/2014	138	0%	98	0	138
6.3	Lógica de negocio		20/11/2013	01/04/2014	133	0%	95	0	133
6.4	Integración		01/04/2014	02/04/2014	2	0%	2	0	2
7	**Pruebas**	Julen	13/03/2014	30/03/2014	18	0%	12	0	18
7.1	Diseño		13/03/2014	15/03/2014	3	0%	2	0	3
7.2	Ejecución		15/03/2014	30/03/2014	16	0%	10	0	16
8	**Documentación**	Julen	01/04/2014	14/04/2014	14	0%	10	0	14
8.1	Memoria		01/04/2014	13/04/2014	13	0%	9	0	13
8.2	Manual de usuario		05/04/2014	08/04/2014	4	0%	2	0	4
8.3	Presentación		13/04/2014	14/04/2014	2	0%	1	0	2

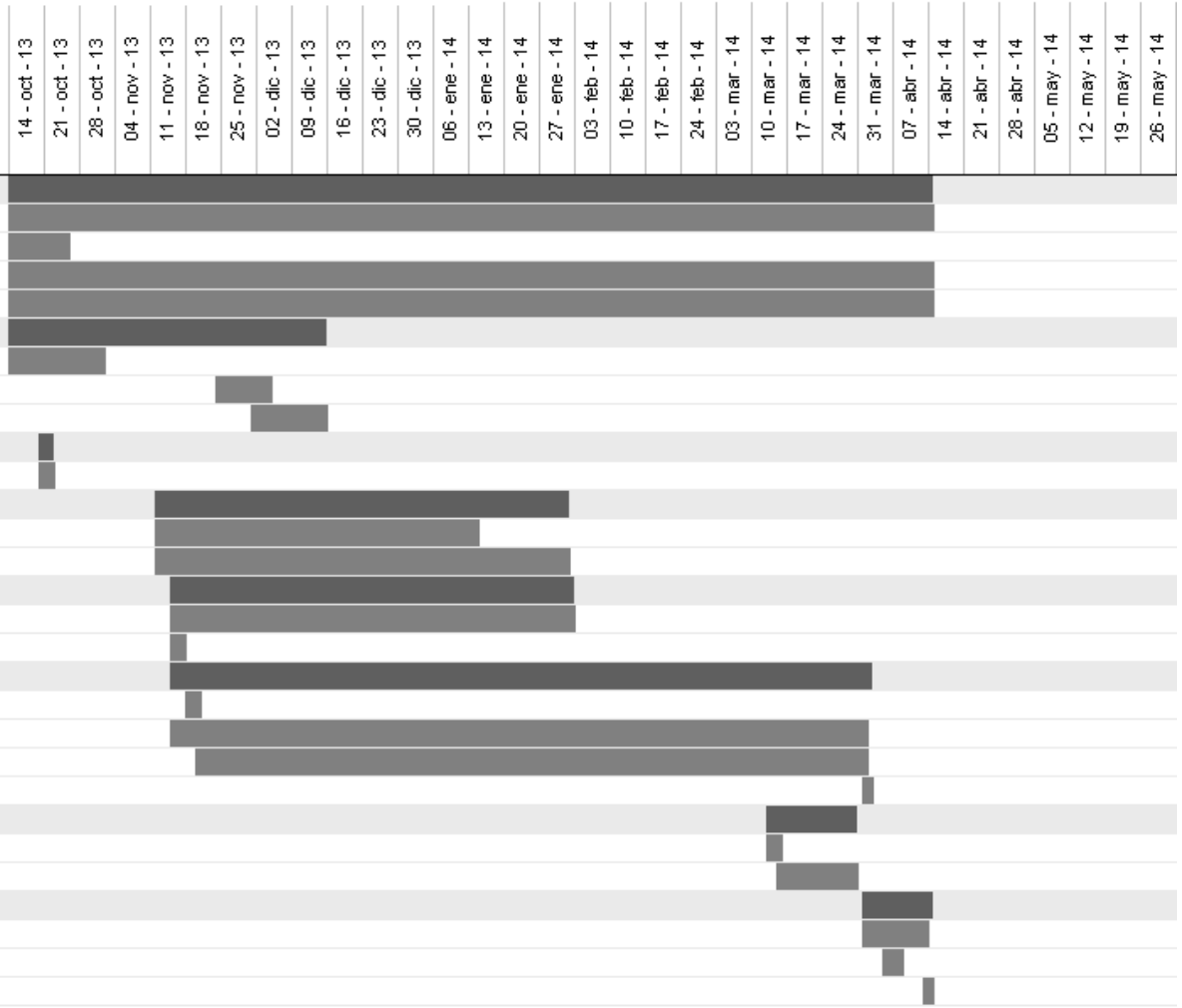

Ilustración 8: Diagrama Gantt

2.4 ANÁLISIS DE RIESGOS

Antes de analizar los riesgos, se presenta el Diagrama de Descomposición de Riesgos que organizará los mismos dependiendo de la categoría.

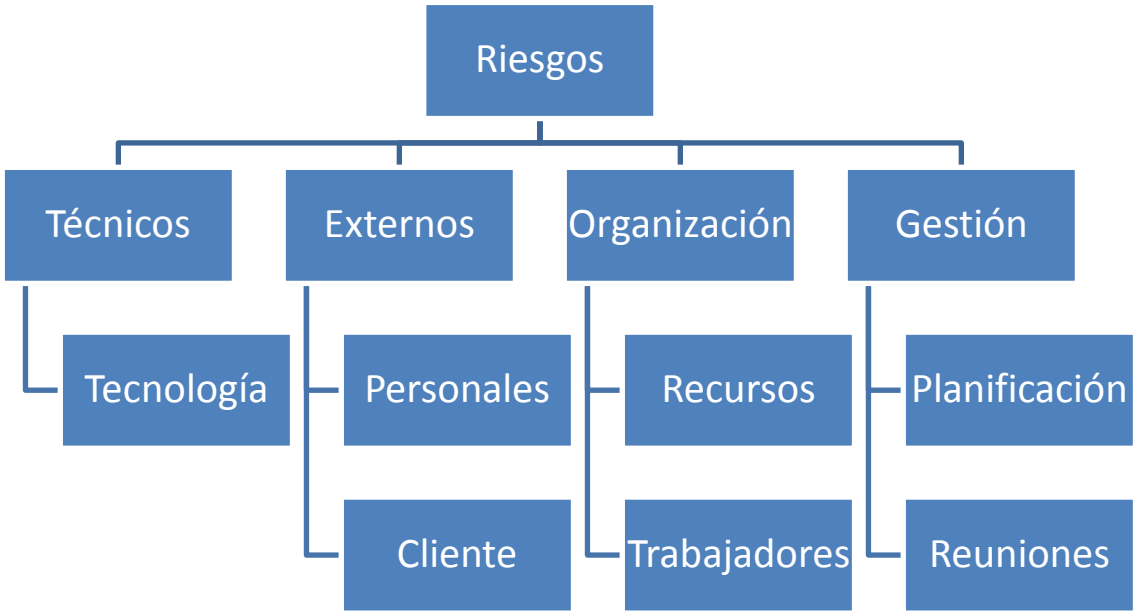

Ilustración 9: Diagrama de Descomposición de Riesgos

A continuación se muestra una tabla que resume la información de los riesgos que se deben tener en cuenta a lo largo del desarrollo del proyecto.

RIESGO	CATEGORÍA	NIVEL DE RIESGO	IMPACTO	PREVENCIÓN	SOLUCIÓN
Cambios de planificación	Gestión	Alto	Fuerte	Planificar dejando margen para los posibles cambios	Adaptar los plazos e invertir más horas
Selección de tecnología inadecuada	Técnico	Alto	Fuerte	Adelantarse a los posibles cambios	Invertir más horas
Enfermedades	Externo	Bajo	Fuerte	-	Invertir más horas
Pérdida de datos	Organización	Bajo	Fuerte	Realizar copias de seguridad regularmente	Invertir más horas
Caída de servidores	Externo	Bajo	Fuerte	-	Invertir más horas
Cambios en los requisitos	Organización	Alto	Medio	Desarrollar una arquitectura que minimice los cambios	Ajustar los plazos

RIESGO	CATEGORÍA	NIVEL DE RIESGO	IMPACTO	PREVENCIÓN	SOLUCIÓN
Falta de formación	Técnico	Alto	Medio	Seleccionar tecnología adecuada y dominada	Volver a invertir más tiempo en la formación
Ausencia de reuniones	Organización	Bajo	Medio	Avisar de la falta con antelación	Atrasar la reunión o estudiar el acta
Carga de trabajo externo	Externo	Medio	Leve	-	Adaptar los plazos

Tabla 2: Tabla de riesgos

La tabla se ha ordenado y coloreado dependiendo del impacto de cada riesgo. Para algunos riesgos no se ha definido un plan de prevención ya que pueden ser generados por factores externos que no se pueden evitar. Sin embargo, el riesgo de la "selección de tecnología adecuada" no depende de factores externos y es algo que hay que asumir. De hecho, el proyecto consiste en gran parte en investigar y probar las tecnologías que mejor se adapten al proyecto, y es por esto que se prevé y se asume de antemano el impacto que va a tener.

Se debe prestar especial atención a riesgos como los cambios de planificación o pérdidas de datos que requerirían, en caso de ocurrir, ajustar los plazos e invertir más tiempo. Por esta razón, se planificarán los plazos con un margen de tiempo por si algo sale mal.

También cabe destacar que el riesgo de los "cambios de requisitos" se suele categorizar como "externo", pero en este caso es la propia organización la que hace de cliente, por lo que se considera como riesgo interno de la organización.

2.5 ANÁLISIS DE FACTIBILIDAD

Una vez examinados los riesgos y los posibles problemas que podrían acarrear, se examina la disponibilidad de las herramientas y técnicas de los que se hará uso para lograr exitosamente los objetivos del proyecto.

El mayor riesgo se deriva de la investigación de diversas tecnologías y cambios de planificación, que como ya se ha comentado, es algo que va a ocurrir. Por esta razón, se ha decidido realizar una planificación con un amplio margen en caso de que se necesite invertir más tiempo en nuevas tecnologías. Para tomar esta decisión también se ha tenido en cuenta el tiempo de la fase de formación, que podría variar a lo largo del proyecto.

Finalmente, no se hará mención al coste económico que supondría el proyecto, ya que se hace uso de herramientas gratuitas o licenciadas proporcionadas por la empresa y son datos manejados por la misma. Teniendo en cuenta todo ello, se deduce que es posible llevar a cabo el proyecto.

3. DESARROLLO TÉCNICO

En el presente capítulo se analizan los requisitos principales del Proyecto de Fin de Carrera que se desarrollan en los siguientes apartados.

Para empezar, se definen los casos de uso mediante el uso de los *Diagramas de Casos de Uso* con su posterior descripción y especificación de los flujos de eventos. A continuación se presenta una arquitectura del sistema óptima sobre la que partirá el desarrollo del proyecto. Para ello también se expondrá una breve descripción de la tecnología utilizada para lograr los objetivos.

3.1 CASOS DE USO

En esta sección se presentarán los casos de uso organizados en diagramas por cada uno de los distintos clientes que harán uso de las funcionalidades proporcionadas. En este caso, teniendo en cuenta los objetivos del proyecto, se distinguen dos tipos de usuario que hacen referencia a las distintas interfaces que se van a implementar:

- **Cliente web:** Se refiere al cliente que accederá a las funcionalidades de EPDM mediante el acceso web desde un ordenador.
- **Cliente móvil:** Serán los que accederán a algunas funcionalidades limitadas de EPDM mediante dispositivos móviles. En esencia, será un subconjunto de casos de uso del cliente web.

3.1.1 DIAGRAMA DE CASOS DE USO

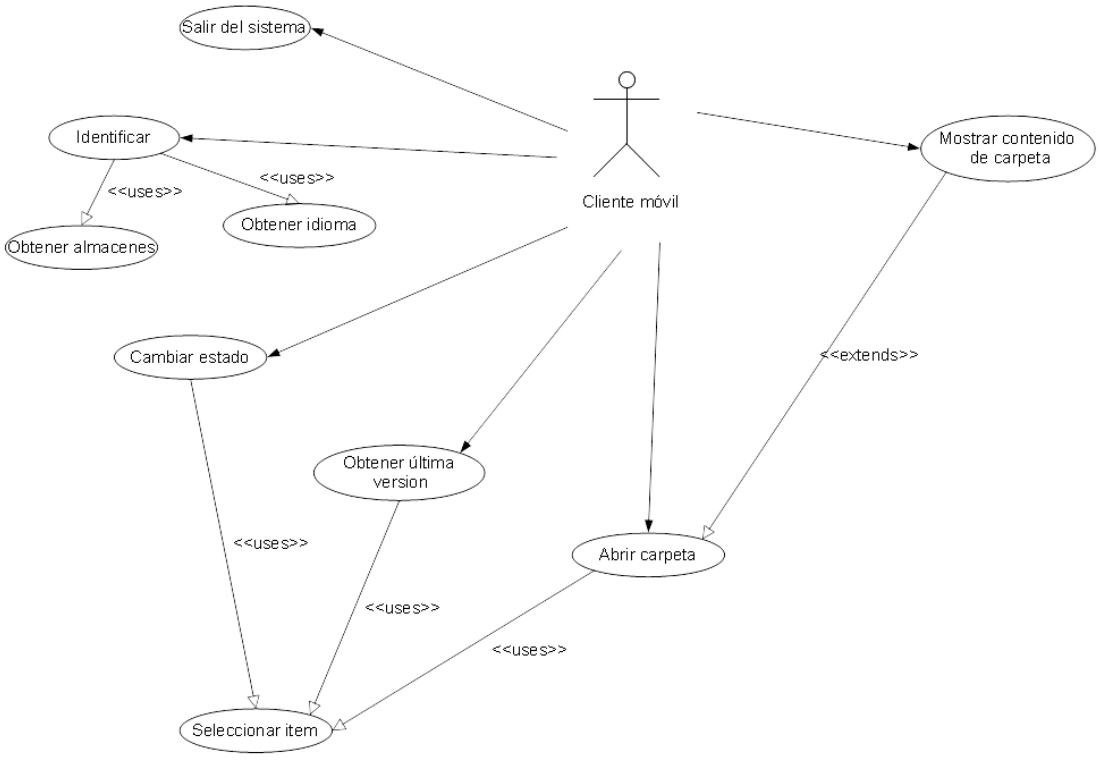

Ilustración 10: Casos de uso del cliente móvil

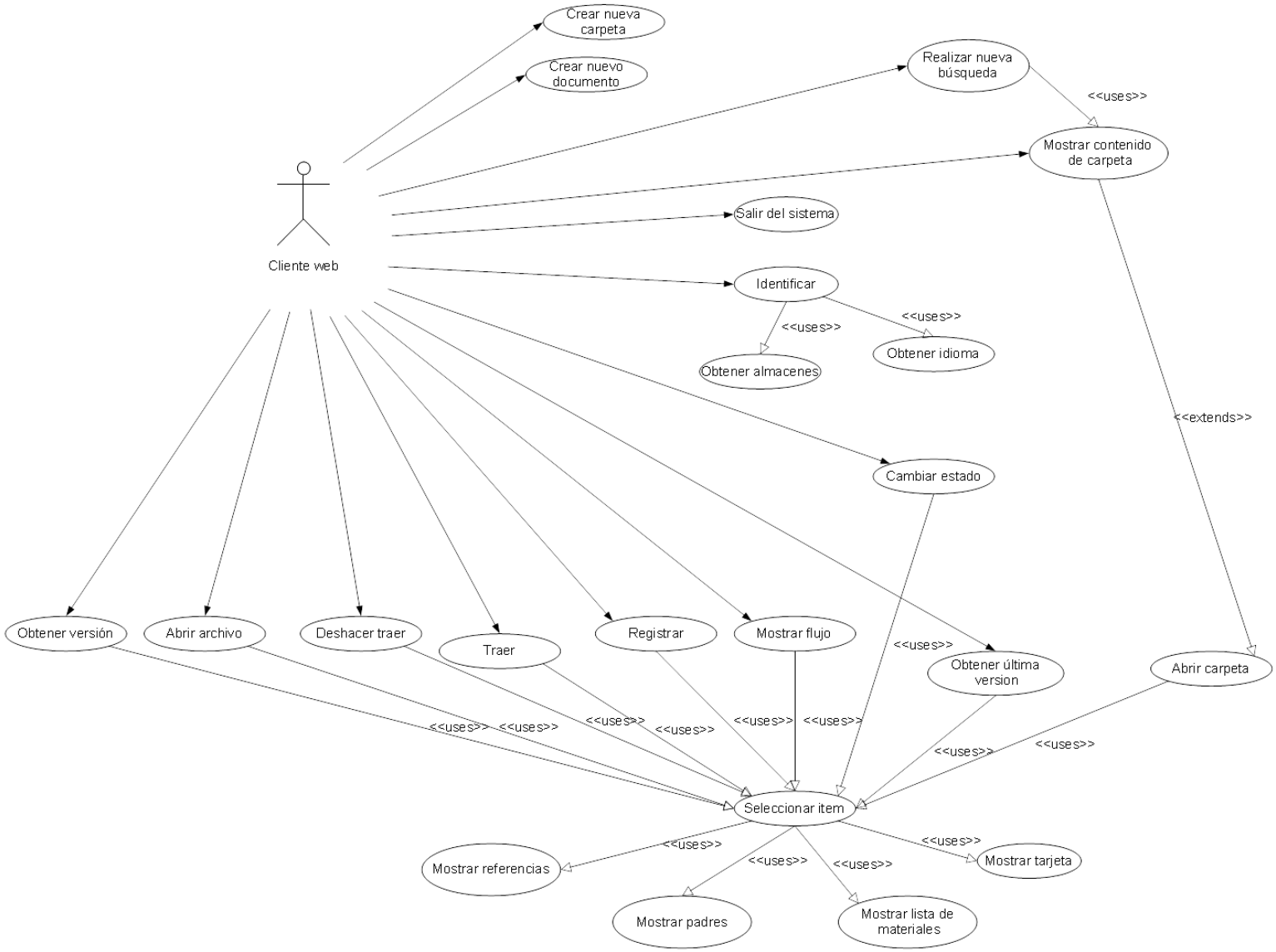

Ilustración 11: Casos de uso del cliente web

3.1.2 ESPECIFICACIÓN DE CASOS DE USO

Nombre	Obtener almacenes	
Descripción	Permite al usuario obtener la lista de almacenes en los que puede identificarse	
Actores	Cliente, Sistema	
Precondiciones	El usuario no está identificado	
Flujo normal	Cliente	Sistema
		1. El sistema devuelve la lista de almacenes del servidor.
Flujo alternativo		
Poscondiciones	El usuario obtiene la lista de almacenes	

Tabla 3: Caso de uso - Obtener almacenes

Nombre	Identificar	
Descripción	Permite al usuario identificarse en el sistema mediante un nombre de usuario y contraseña.	
Actores	Cliente, Sistema	
Precondiciones		
Flujo normal	Cliente	Sistema
	1. El cliente introduce el usuario y contraseña los que quiere acceder al sistema.	2. El sistema intenta la identificación con los datos proporcionados por el usuario.
		3. El sistema identifica con éxito al usuario y termina el caso de uso.
Flujo alternativo		2. El sistema falla en la identificación e informa al usuario con un mensaje y vuelve al paso 1.
Poscondiciones	El usuario es identificado en el sistema	

Tabla 4: Caso de uso - Identificación

Nombre	Salir del sistema	
Descripción	Permite al usuario salir del sistema donde se ha identificado con éxito.	
Actores	Cliente, Sistema	
Precondiciones	El usuario está identificado en el sistema.	
Flujo normal	Cliente	Sistema
Flujo alternativo		
Poscondiciones	El usuario sale del sistema.	

Tabla 5: Caso de uso - Salir del sistema

Nombre	Obtener idioma	
Descripción	Obtiene el idioma del usuario identificado	
Actores	Cliente, Sistema	
Precondiciones	El usuario está identificado en el sistema.	
Flujo normal	Cliente	Sistema
		1. El sistema devuelve el idioma asociado al usuario de EPDM
Flujo alternativo		
Poscondiciones	Se conoce el idioma del usuario para mostrar el contenido en su idioma.	

Tabla 6: Caso de uso - Obtener idioma

Nombre	Mostrar contenido de carpeta	
Descripción	Permite al usuario mostrar el contenido de la carpeta seleccionada	
Actores	Cliente, Sistema	
Precondiciones	El usuario está identificado en el sistema	
Flujo normal	Cliente	Sistema
	1. El cliente selecciona la carpeta de la que obtener el contenido.	2. El sistema devuelve una lista de archivos y carpetas con sus propiedades.
Flujo alternativo		
Poscondiciones	El usuario obtiene el contenido de la carpeta con la capacidad de interactuar sobre él.	

Tabla 7: Caso de uso - Mostrar contenido de carpeta

Nombre	Crear nueva carpeta	
Descripción	Permite al usuario crear una carpeta dentro de la carpeta en la que se encuentra.	
Actores	Cliente, Sistema	
Precondiciones	El usuario se encuentra dentro de la vista de una carpeta.	
Flujo normal	Cliente	Sistema
	2. El cliente ingresa el nombre de la nueva carpeta.	1. El sistema muestra un cuadro donde escribir el nombre de la nueva carpeta.
		3. El sistema crea la nueva carpeta en la visualiza en la pantalla.
Flujo alternativo		3. El sistema falla porque el usuario ha ingresado: - Un nombre vacío. - Caracteres especiales. - Nombre de carpeta existente. El sistema muestra un mensaje de error y vuelve al paso 1.
Poscondiciones	Se crea una nueva carpeta	

Tabla 8: Caso de uso - Nueva carpeta

Nombre	Crear nuevo documento	
Descripción	Permite al usuario crear un nuevo documento dentro de la carpeta en la que se encuentra.	
Actores	Cliente, Sistema	
Precondiciones	El usuario se encuentra dentro de la vista de una carpeta.	
Flujo normal	Cliente	Sistema
	2. El cliente ingresa el nombre del nuevo archivo	1. El sistema muestra un cuadro donde escribir el nombre del nuevo archivo
		3. El sistema crea el nuevo documento y lo visualiza en pantalla.
Flujo alternativo		3. El sistema falla porque el usuario ha ingresado: - Un nombre vacío. - Caracteres especiales. - Nombre de archivo existente. El sistema muestra un mensaje de error y vuelve al paso 1.
Poscondiciones	Se crea un nuevo documento	

Tabla 9: Caso de uso - Nuevo documento

Nombre	Realizar nueva búsqueda	
Descripción	Permite al usuario buscar elementos mediante el uso de tarjetas de búsqueda	
Actores	Cliente, Sistema	
Precondiciones		
Flujo normal	Cliente	Sistema
	2. El cliente selecciona la tarjeta de búsqueda deseada y acepta.	1. El sistema muestra una lista desplegable con las tarjetas de búsqueda disponibles.
	4. El usuario rellena los campos necesarios del formulario y realiza la petición de búsqueda.	3. El sistema muestra el formulario asociado a la tarjeta de búsqueda.
		5. El sistema realiza una búsqueda con los parámetros recibidos y muestra el contenido en pantalla.
Flujo alternativo		
Poscondiciones	El usuario obtiene el contenido de la búsqueda	

Tabla 10: Caso de uso - Nueva búsqueda

Nombre	Mostrar tarjeta	
Descripción	Permite al usuario visualizar la tarjeta asociada al elemento seleccionado	
Actores	Cliente, Sistema	
Precondiciones	Se debe seleccionar un archivo o carpeta	
Flujo normal	Cliente	Sistema
	2. El cliente selecciona la configuración deseada.	1. El sistema muestra las configuraciones disponibles para el archivo seleccionado.
		3. El sistema muestra los controles de la tarjeta en un formulario.
Flujo alternativo		
Poscondiciones	El usuario obtiene la vista de la tarjeta	

Tabla 11: Caso de uso - Mostrar tarjeta

Nombre	Seleccionar ítem	
Descripción	Permite al usuario seleccionar un elemento de la carpeta para poder interactuar sobre él	
Actores	Cliente, Sistema	
Precondiciones		
Flujo normal	Cliente	Sistema
		1. El sistema marca el elemento y hace uso de “Mostrar tarjeta” para el elemento seleccionado.
Flujo alternativo		
Poscondiciones	El elemento queda seleccionado para posteriores acciones	

Tabla 12: Caso de uso - Seleccionar ítem

Nombre	Abrir carpeta	
Descripción	Permite al usuario abrir y obtener el contenido de la carpeta seleccionada	
Actores	Cliente, Sistema	
Precondiciones	La carpeta debe estar seleccionada	
Flujo normal	Cliente	Sistema
	1. El cliente selecciona la carpeta que quiere abrir.	2. El sistema devuelve una lista de archivos y carpetas con sus propiedades.
Flujo alternativo		
Poscondiciones	El usuario obtiene el contenido de la carpeta con la capacidad de interactuar sobre él.	

Tabla 13: Caso de uso - Abrir carpeta

Nombre	Abrir archivo	
Descripción	Permite al usuario abrir el archivo seleccionado con el programa que el sistema operativo tiene asociado.	
Actores	Cliente, Sistema	
Precondiciones	Un archivo debe estar seleccionado	
Flujo normal	Cliente	Sistema
	1. El cliente selecciona el archivo que desea abrir.	2. El sistema abre el documento con el programa asociado.
Flujo alternativo		2. Si el archivo no existe, el sistema descargará la última versión del archivo y lo volverá a intentar.
Poscondiciones	El usuario obtiene la ejecución de un programa externo con el documento abierto y listo para la edición.	

Tabla 14: Caso de uso - Abrir archivo

Nombre	Obtener última versión	
Descripción	Permite al usuario descargar la última versión del archivo seleccionado.	
Actores	Cliente, Sistema	
Precondiciones	Un archivo debe estar seleccionado	
Flujo normal	Cliente	Sistema
	1. El cliente selecciona el archivo que desea obtener.	2. El sistema obtiene el archivo y envía los datos al cliente.
Flujo alternativo	3. El cliente selecciona de la estructura los archivos que desea obtener.	2. Si el archivo tiene referencias, el sistema muestra una estructura de árbol con los archivos y sus últimas versiones
		4. El sistema obtiene las últimas versiones de los archivos seleccionados y los envía al cliente.
Poscondiciones	El usuario obtiene los datos del archivo y los guarda en el sistema de archivos local.	

Tabla 15: Caso de uso - Obtener última versión

Nombre	Obtener versión	
Descripción	Permite al usuario descargar una versión en concreto del archivo seleccionado.	
Actores	Cliente, Sistema	
Precondiciones	Un archivo debe estar seleccionado	
Flujo normal	Cliente	Sistema
	1. El cliente selecciona el archivo que desea obtener.	2. El sistema obtiene el historial de versiones del archivo y lo muestra en un cuadro.
	3. El cliente selecciona la versión del archivo que desea descargar.	4. El sistema obtiene la versión seleccionada del archivo y envía los datos al cliente.
Flujo alternativo	5. El cliente selecciona en la estructura los archivos que desea obtener.	4. Si el archivo tiene referencias, el sistema muestra una estructura de árbol con los archivos y sus versiones referenciadas.
		6. El sistema obtiene las versiones referenciadas de los archivos seleccionados y los envía al cliente.
Poscondiciones	El usuario obtiene los datos del archivo y los guarda en el sistema de archivos local.	

Tabla 16: Caso de uso - Obtener versión

Nombre	Traer	
Descripción	Permite al usuario traer el archivo bloqueándolo para que nadie más pueda editarlo.	
Actores	Cliente, Sistema	
Precondiciones	Un archivo debe estar seleccionado y no debe estar traído por nadie	
Flujo normal	Cliente	Sistema
	1. El cliente selecciona el archivo que desea traer.	2. Trae el archivo seleccionado.
Flujo alternativo		2. Si el archivo no existe en el sistema de archivos local o no es la última versión, obtiene la última versión antes de traer.
Poscondiciones	El usuario bloquea el archivo para su edición.	

Tabla 17: Caso de uso – Traer

Nombre	Deshacer traer	
Descripción	Permite al usuario deshacer la operación de traer	
Actores	Cliente, Sistema	
Precondiciones	Un archivo debe estar seleccionado y traído por él	
Flujo normal	Cliente	Sistema
	1. El cliente selecciona el archivo que desea desbloquear.	2. El sistema deshace el traer en el archivo seleccionado y lo desbloquea sin hacer ningún cambio.
Flujo alternativo		
Poscondiciones	El archivo seleccionado queda desbloqueado.	

Tabla 18: Caso de uso - Deshacer traer

Nombre	Registrar	
Descripción	Permite al usuario registrar un archivo existente para guardar los cambios en el servidor creando una nueva versión.	
Actores	Cliente, Sistema	
Precondiciones	Un archivo debe estar seleccionado y traído él	
Flujo normal	Cliente	Sistema
	1. El cliente selecciona el archivo que desea registrar.	2. El sistema sube el archivo seleccionado al servidor y lo registra en una nueva versión. El archivo queda desbloqueado.
Flujo alternativo		2. Si no se ha hecho ningún cambio en el archivo, el archivo solo se desbloquea sin incrementar la versión.
Poscondiciones	Se incrementa la versión del archivo seleccionado y queda desbloqueado.	

Tabla 19: Caso de uso – Registrar

Nombre	Cambiar estado	
Descripción	Permite al usuario cambiar el estado del archivo seleccionado	
Actores	Cliente, Sistema	
Precondiciones	Un archivo debe estar seleccionado y no debe estar traído por nadie	
Flujo normal	Cliente	Sistema
	1. El cliente selecciona el archivo cuyo estado desea cambiar.	2. El sistema muestra un recuadro con los posibles cambios de estado.
	3. El cliente selecciona el nuevo estado del archivo.	4. El sistema cambia de estado el archivo.
Flujo alternativo		
Poscondiciones	El archivo seleccionado se encuentra en el nuevo estado.	

Tabla 20: Caso de uso - Cambiar estado

Nombre	Mostrar lista de materiales	
Descripción	Permite al usuario visualizar la lista de materiales de un archivo	
Actores	Cliente, Sistema	
Precondiciones	Un archivo debe estar seleccionado	
Flujo normal	Cliente	Sistema
	1. El cliente selecciona el archivo cuyos materiales quiera visualizar.	2. El sistema muestra un recuadro con la lista de materiales y sus propiedades.
Flujo alternativo		
Poscondiciones	El usuario obtiene la lista de materiales del archivo.	

Tabla 21: Caso de uso - Lista de materiales

Nombre	Mostrar referencias	
Descripción	Permite al usuario visualizar el árbol de referencias de un archivo	
Actores	Cliente, Sistema	
Precondiciones	Un archivo debe estar seleccionado	
Flujo normal	Cliente	Sistema
	1. El cliente selecciona el archivo cuyas referencias quiera visualizar.	2. El sistema muestra un recuadro con la lista de referencias organizada en un árbol.
Flujo alternativo		
Poscondiciones	El usuario obtiene el árbol de referencias	

Tabla 22: Caso de uso - Mostrar referencias

Nombre	Mostrar padres	
Descripción	Permite al usuario visualizar los padres de un archivo	
Actores	Cliente, Sistema	
Precondiciones	Un archivo debe estar seleccionado	
Flujo normal	Cliente	Sistema
	1. El cliente selecciona el archivo cuyos padres quiera visualizar.	2. El sistema muestra un recuadro con la lista de padres organizada en un árbol.
Flujo alternativo		
Poscondiciones	El usuario obtiene la lista de padres en un árbol	

Tabla 23: Caso de uso - Mostrar padres

Nombre	Mostrar flujo	
Descripción	Permite al usuario visualizar el flujo de trabajo asociado a un archivo y su estado actual	
Actores	Cliente, Sistema	
Precondiciones	Un archivo debe estar seleccionado	
Flujo normal	Cliente	Sistema
	1. El cliente selecciona el archivo cuyo flujo desea visualizar	2. El sistema muestra un cuadro con el diagrama del flujo de trabajo y su estado actual marcado.
Flujo alternativo		
Poscondiciones	El usuario obtiene el diagrama del flujo de trabajo asociado al archivo	

Tabla 24: Caso de uso - Mostrar flujo

3.2 DIAGRAMA DE CLASES DEL DOMINIO

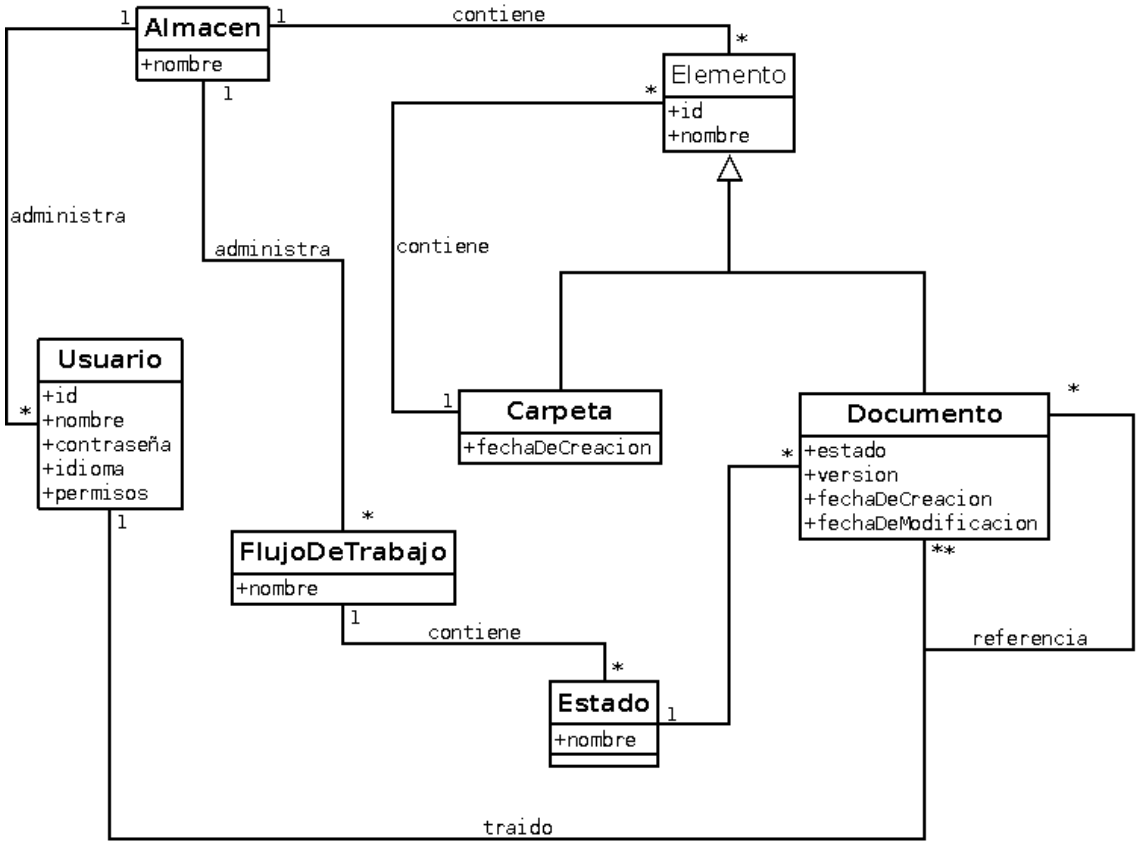

Ilustración 12: Diagrama de clases del dominio

A partir de los casos de uso, se construye el Diagrama de Clases del Dominio. Este diagrama muestra una aproximación simplificada de los objetos sobre los que trabaja el *API* de *SolidWorks Enterprise PDM*. Estos objetos ya están implementados en la *API* y su propia base de datos los almacena, por esta razón solo se debe preocuparse de realizar correctamente las llamadas a las funciones de la *API*.

3.3 DIAGRAMAS DE SECUENCIA DEL SISTEMA

Tras definición y especificación completa de los casos de uso, se muestran los *Diagramas de Secuencia del Sistema* para cada uno de los casos de uso. Estos diagramas en *UML* muestran gráficamente los eventos que fluyen de los actores al sistema.

Cada uno de los eventos constituye una función que posteriormente se diseñará y se implementará. En ciertos casos de use, dichos eventos se repiten y solo es necesario implementarlos una vez como ya se verá más adelante.

Ilustración 13: DSS - Obtener almacenes

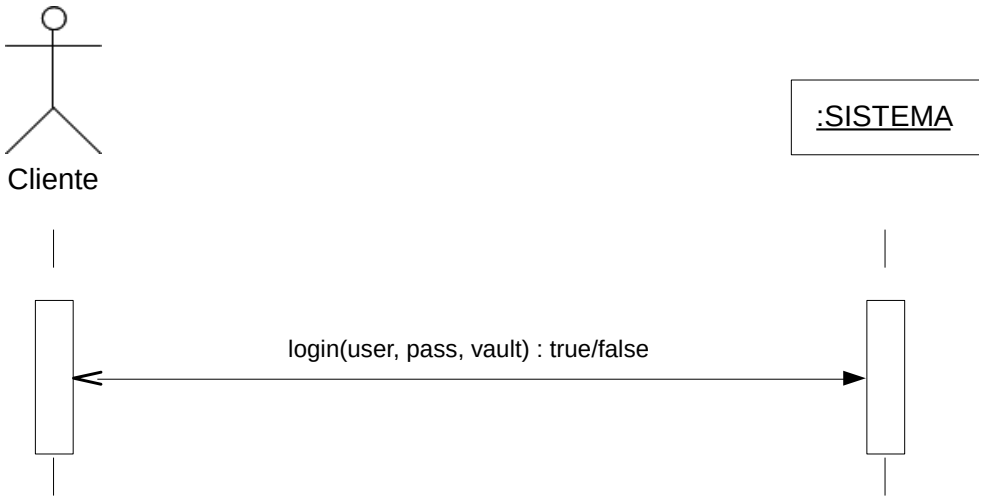

Ilustración 14: DSS – Identificar

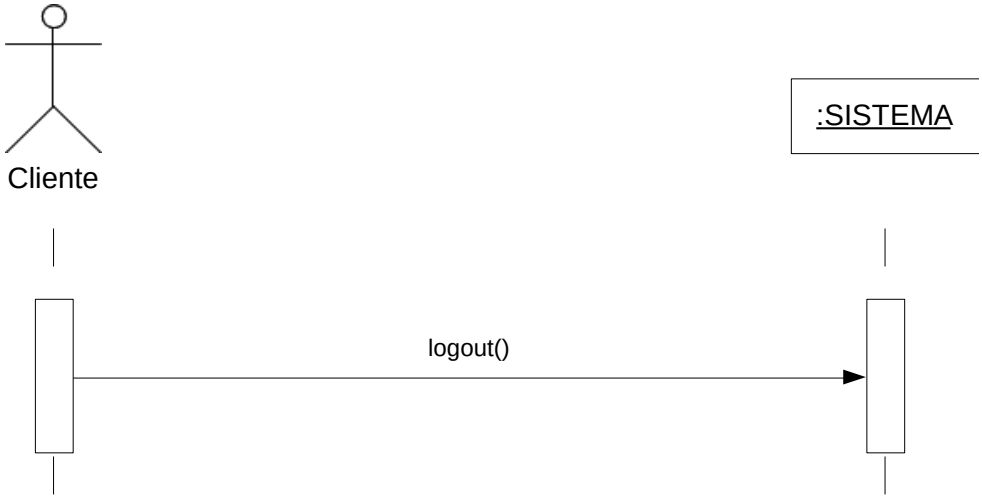

Ilustración 15: DSS - Salir del sistema

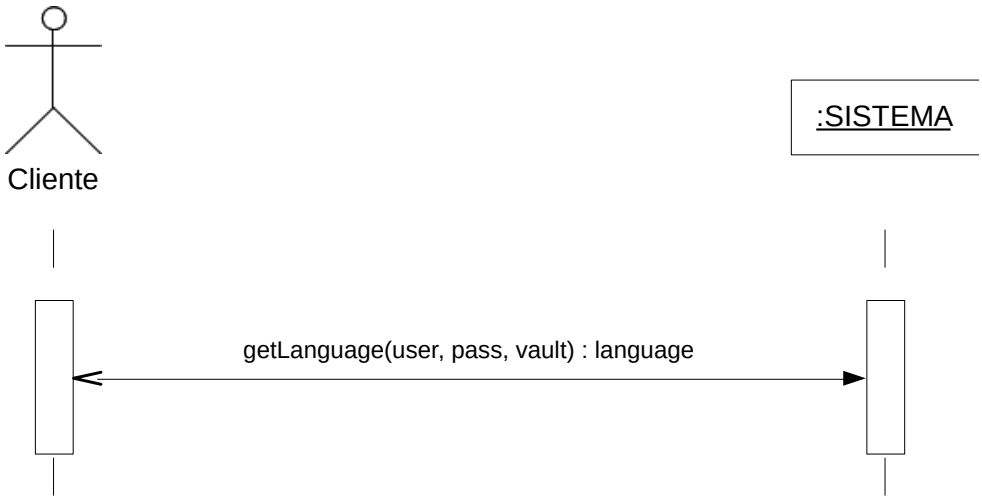

Ilustración 16: DSS – Obtener idioma

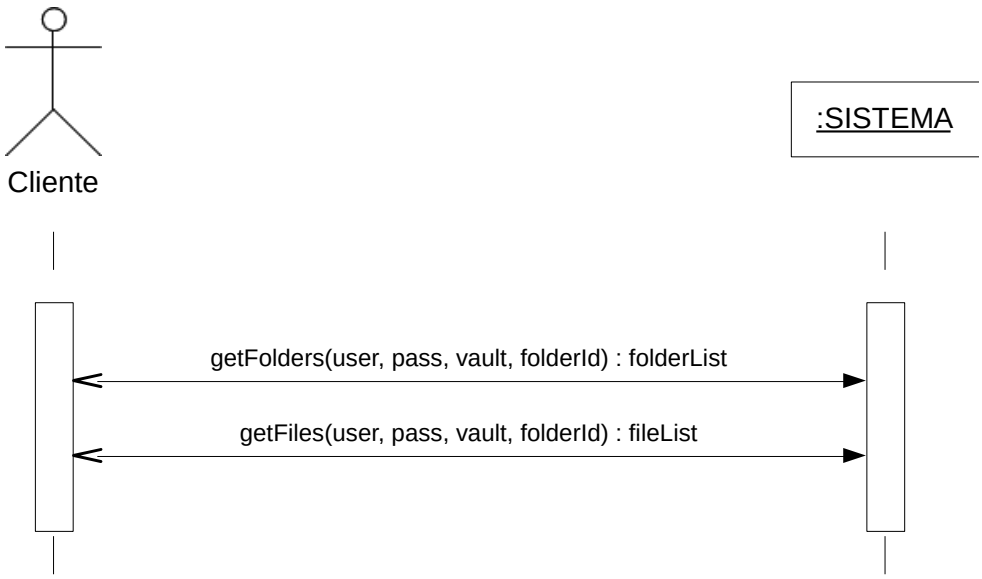

Ilustración 17: DSS - Mostrar contenido de carpeta

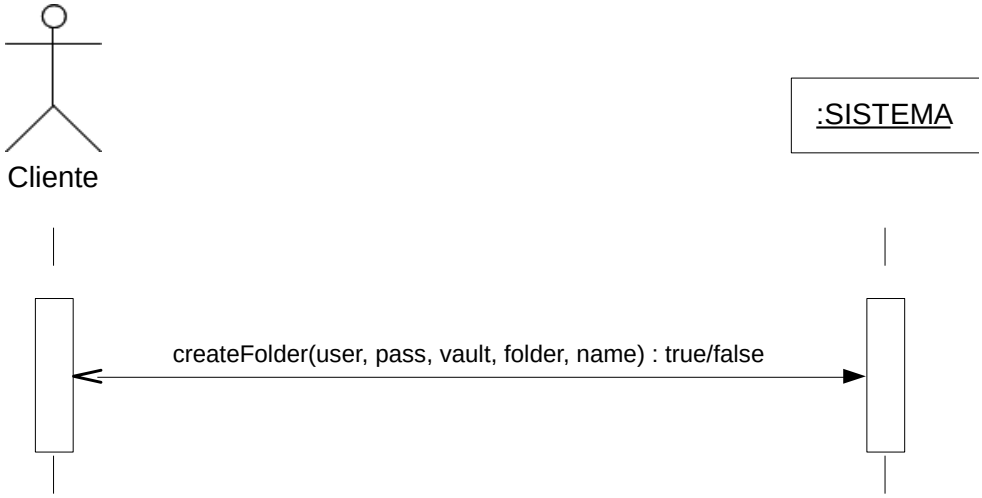

Ilustración 18: DSS – Crear nueva carpeta

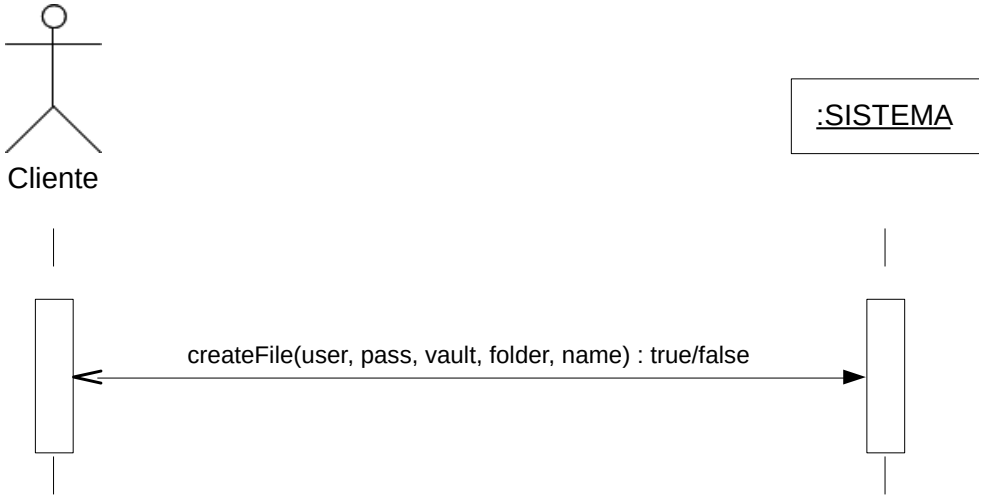

Ilustración 19: DSS – Crear nuevo documento

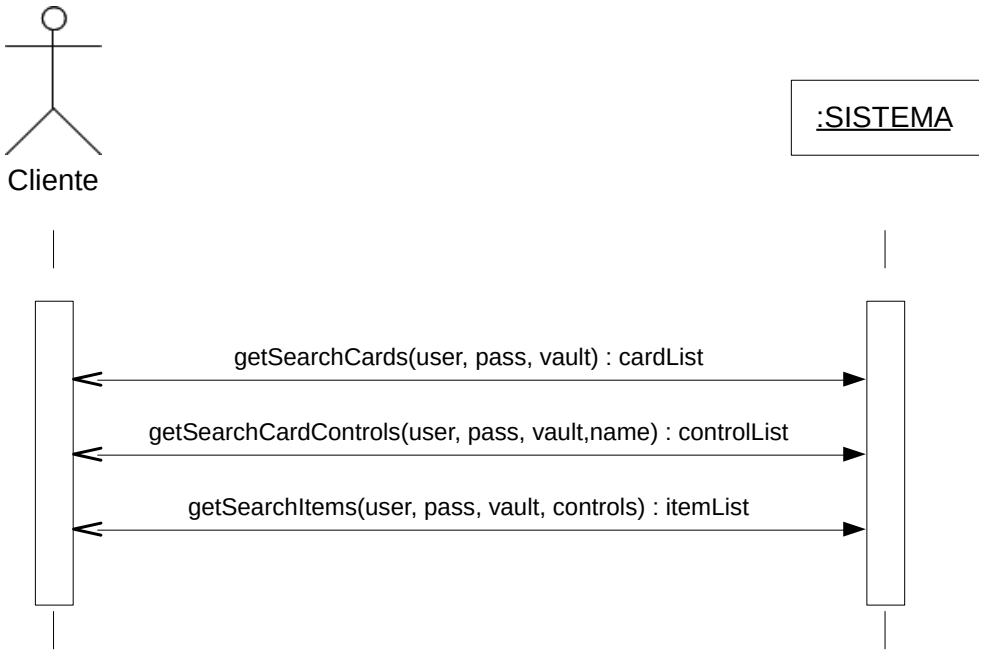

Ilustración 20: DSS – Realizar nueva búsqueda

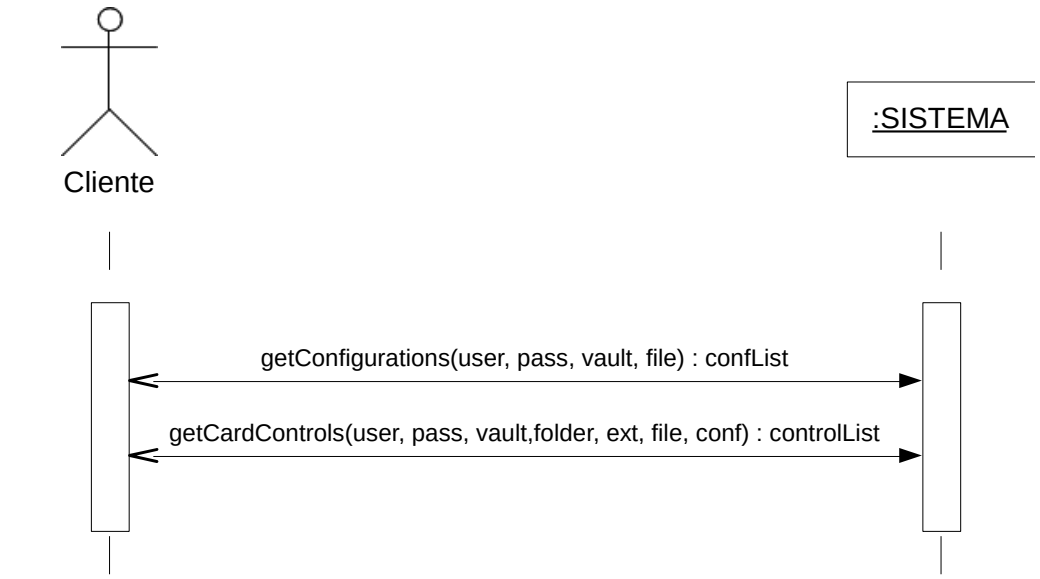

Ilustración 21: DSS - Mostrar tarjeta

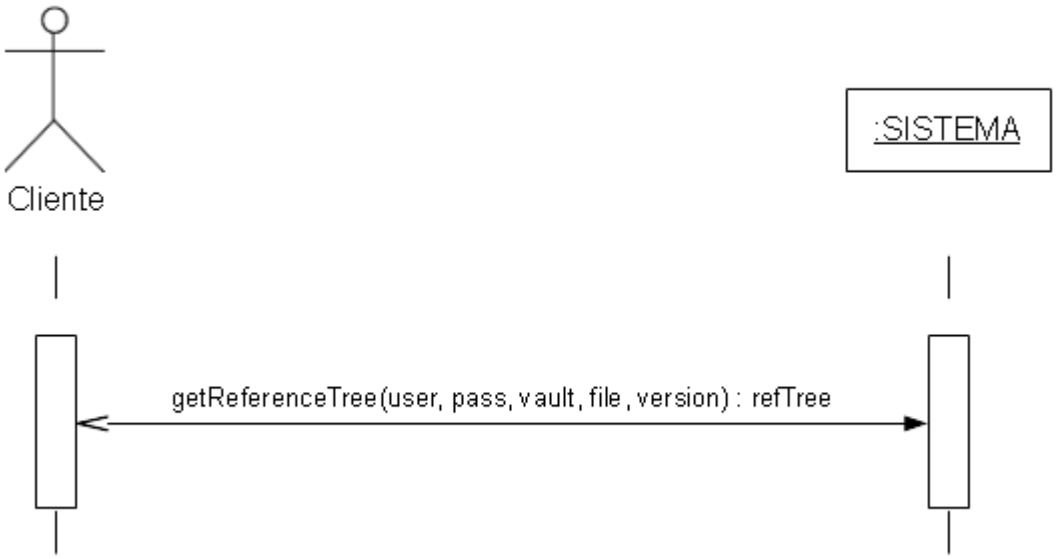

Ilustración 22: DSS - Mostrar referencias

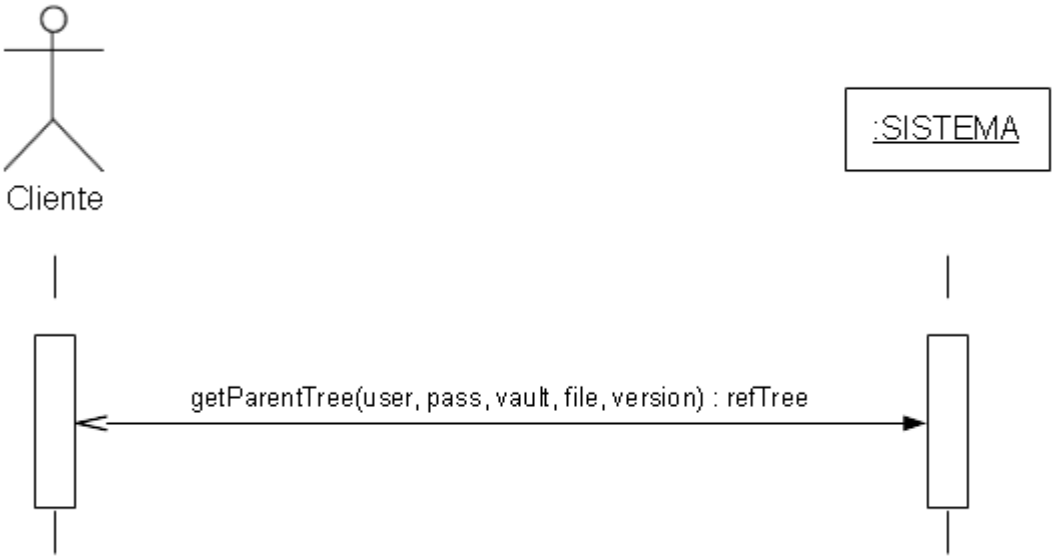

Ilustración 23: DSS - Mostrar padres

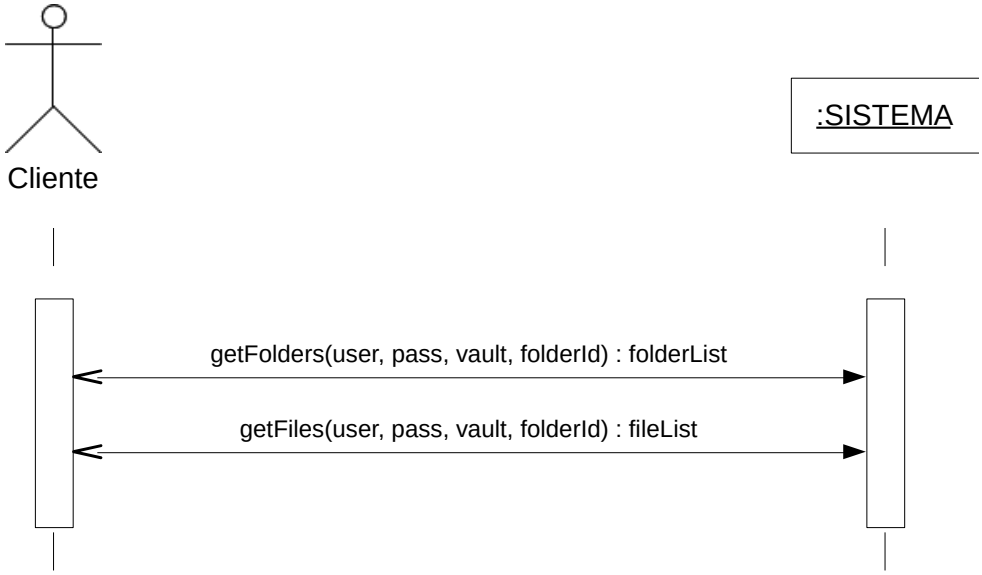

Ilustración 24: DSS - Abrir carpeta

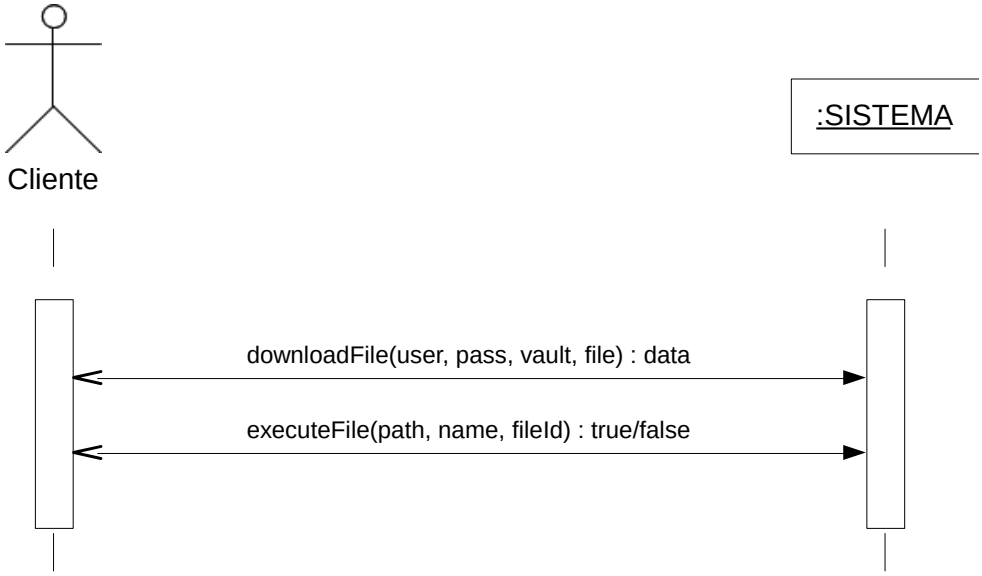

Ilustración 25: DSS - Abrir archivo

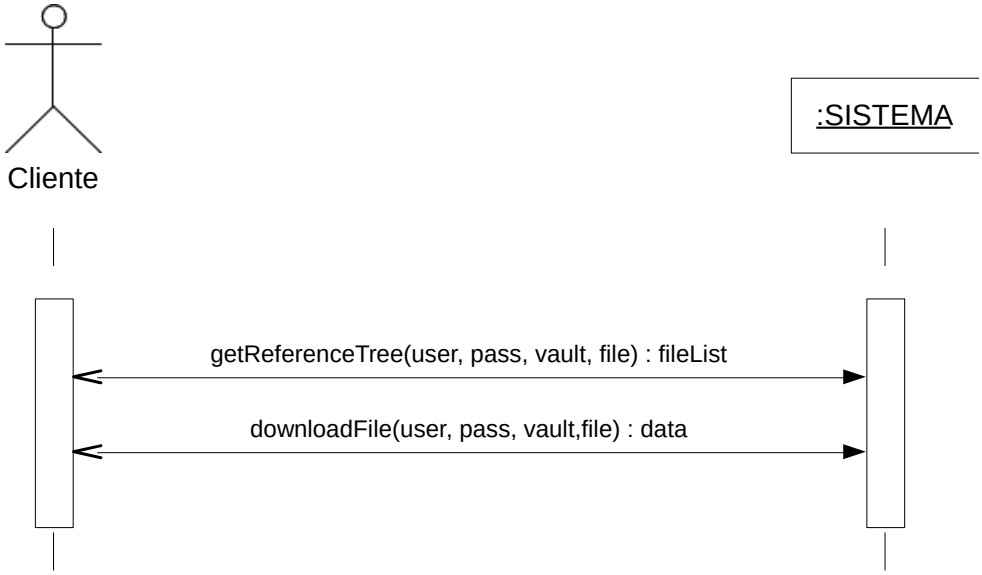

Ilustración 26: DSS - Obtener última versión

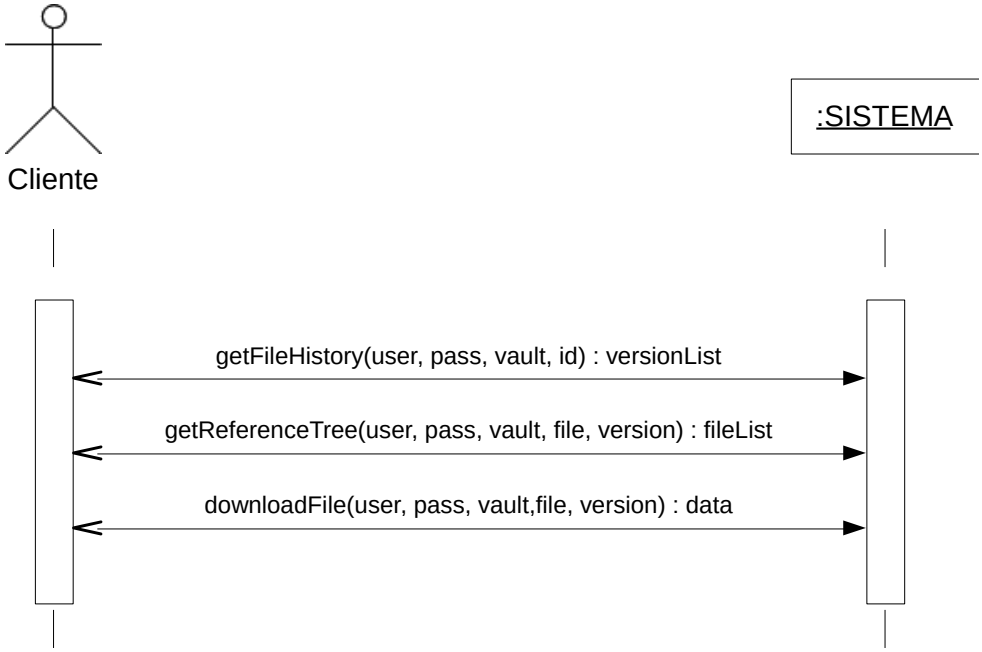

Ilustración 27: DSS - Obtener versión

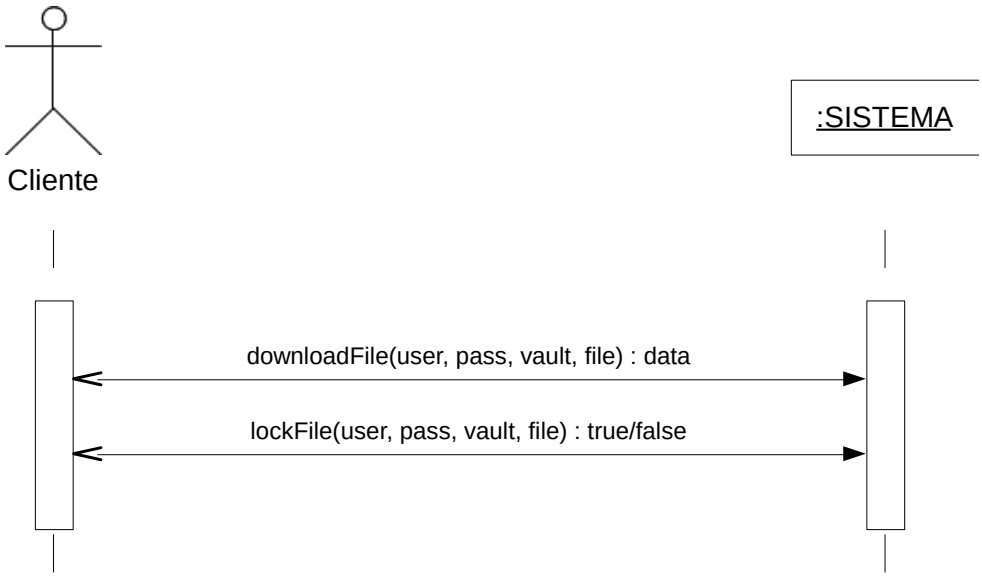

Ilustración 28: DSS - Traer

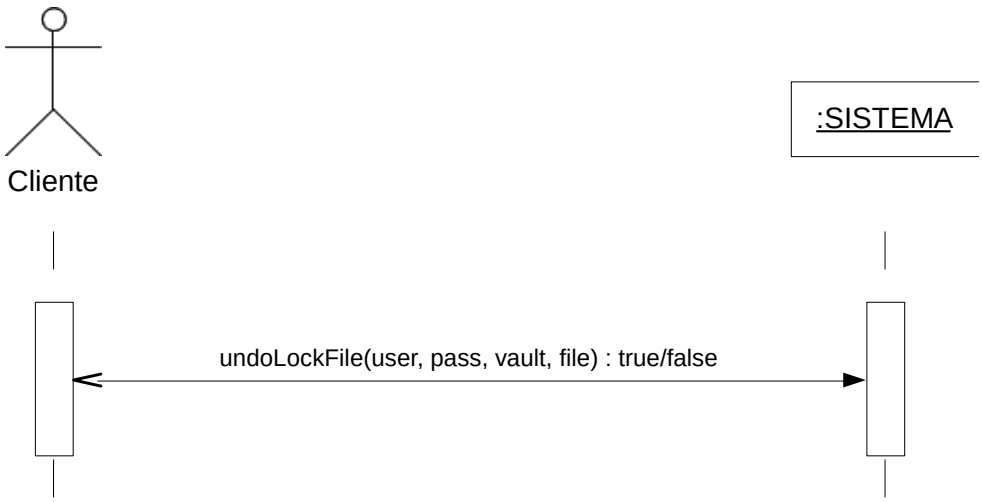

Ilustración 29: DSS - Deshacer traer

Ilustración 30: DSS – Registrar

Ilustración 31: DSS - Cambiar estado

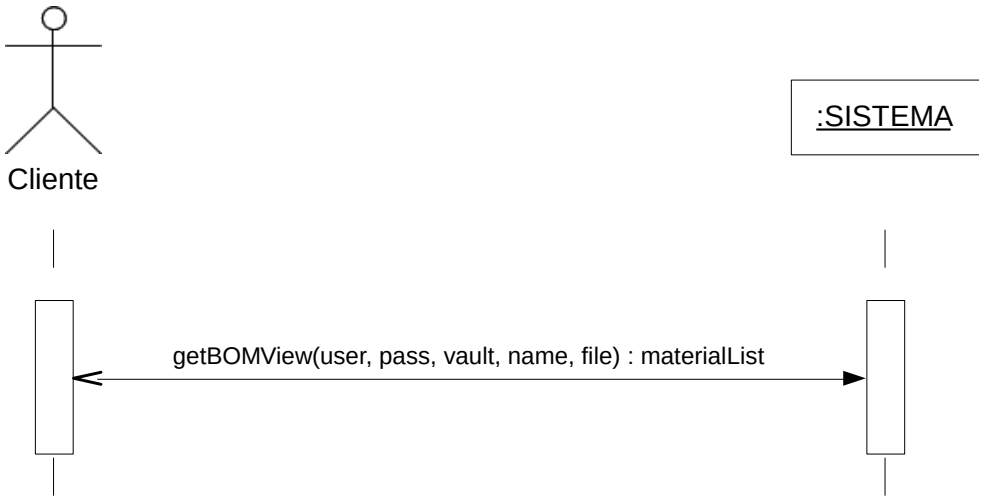

Ilustración 32: DSS – Mostrar lista de materiales

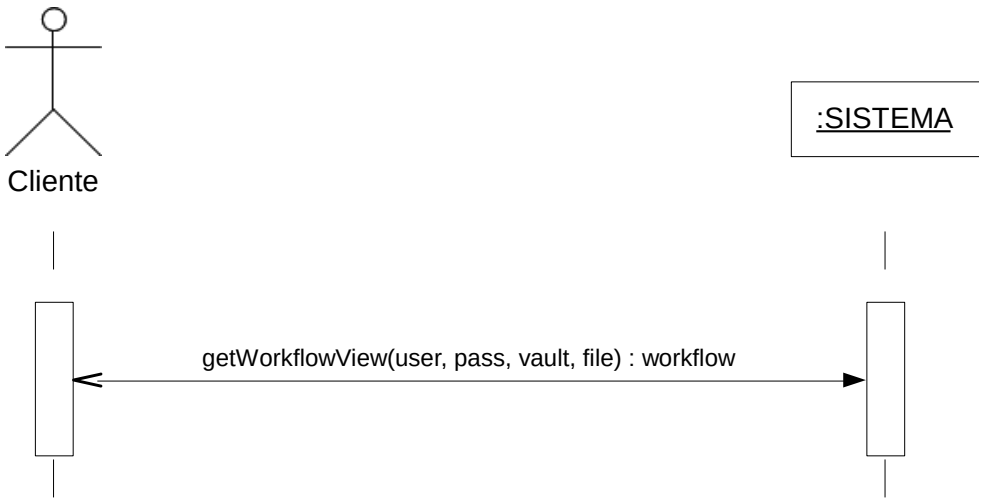

Ilustración 33: DSS - Mostrar flujo

3.4 ARQUITECTURA DEL SISTEMA

Aunque en un principio la idea era utilizar una arquitectura de tres niveles, se ha decidido por separar la arquitectura en cuatro niveles. Concretamente, se han separado el proxy y el servidor para que puedan comunicarse desde distintas máquinas. De hecho, por cuestiones de seguridad y otros problemas de acceso, se ha querido alojar el servidor web (IIS) en una máquina distinta de donde se encuentra el servidor *PDM*.

Esta arquitectura permite alojar el servidor *PDM*, por ejemplo, en una red privada a la cual solamente el proxy tiene acceso directo. Así, los usuarios deberán realizar las peticiones al proxy que ofrecerá una interfaz con los servicios necesarios y decidirá hacia dónde redirigir las llamadas.

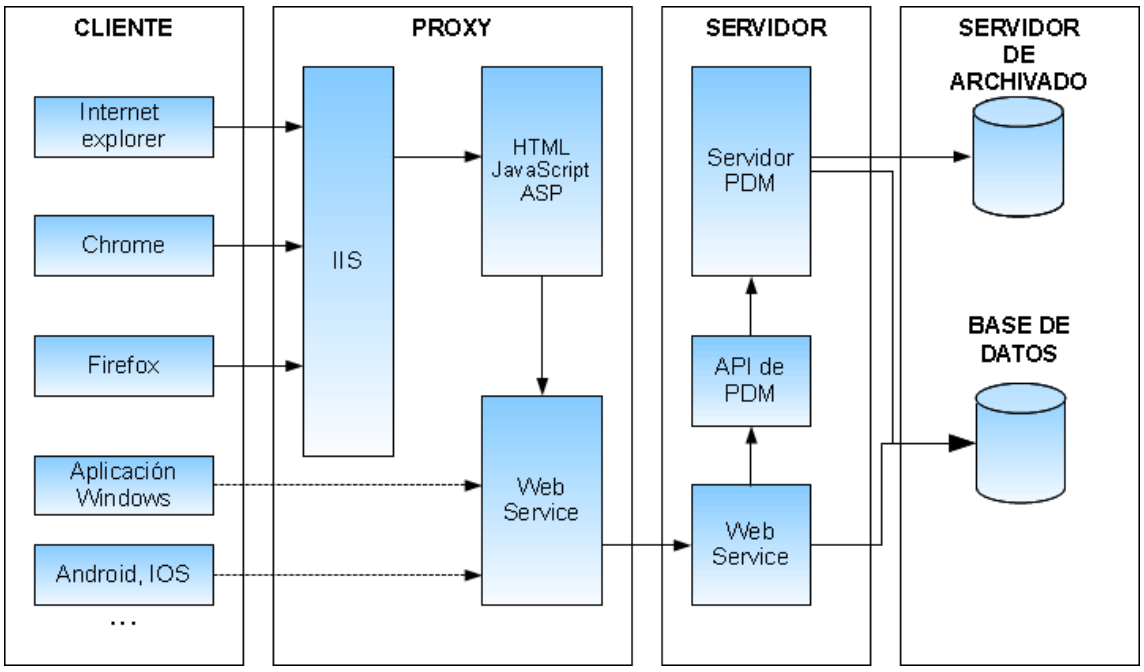

Ilustración 34: Arquitectura del sistema

También cabe destacar que la aplicación web desarrollada para dispositivos móviles accede y obtiene las páginas directamente del servidor web (IIS) del proxy. Si se fuera a desarrollar una aplicación nativa tanto para *Android* tanto como para *IOS*, las llamadas se harían directamente a los servicios alojados en el proxy.

4. DISEÑO

El siguiente capítulo presenta los diagramas de secuencia y el diagrama de clases que se ha definido para implementar los servicios web.

Mientras que los casos de uso son funciones que ofrece la interfaz de usuario al cliente web tanto como al cliente móvil, los diagramas de secuencia muestran el diseño que se ha seguido a la hora de implementar los servicios web. Los casos de uso realizan llamadas directas a los servicios que se definen a continuación.

4.1 DIAGRAMAS DE SECUENCIA

Para empezar, se mostrará el caso usado en la fase de identificación que obtiene la lista de almacenes para ofrecer al usuario la posibilidad de identificarse en uno de ellos.

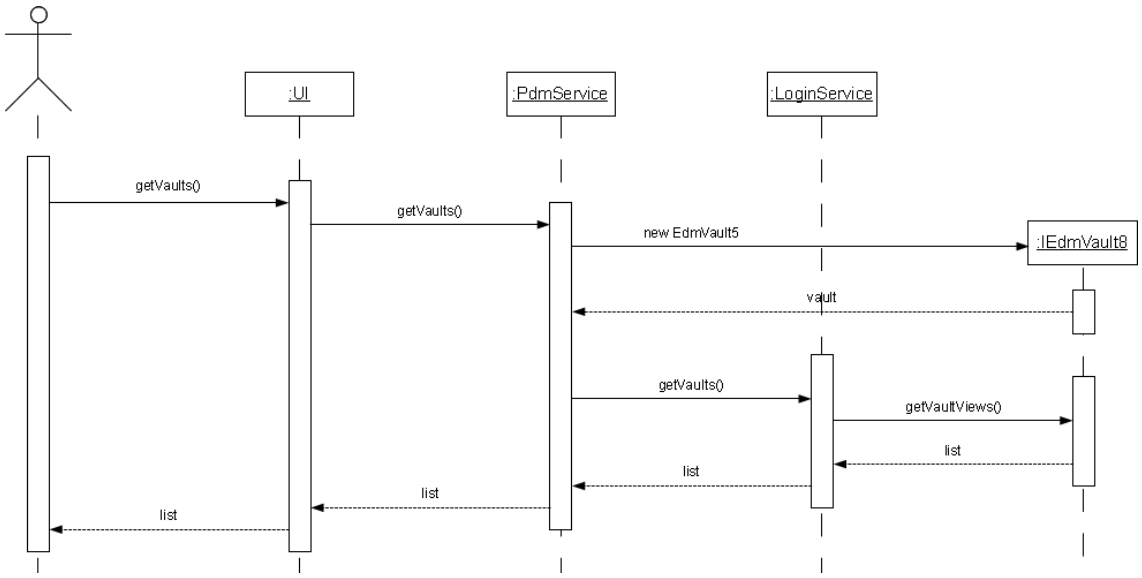

Ilustración 35: Diagrama de secuencia - Obtener almacenes

Una vez obtenido la información de los almacenes, se presenta el diseño del caso más importante de todos, la identificación del usuario y la entrada al sistema. Cabe destacar que es necesario utilizar esta función antes de cualquier otra que actúe sobre el servidor PDM. Es por ello que todas las funciones requieran el usuario y contraseña como parámetro.

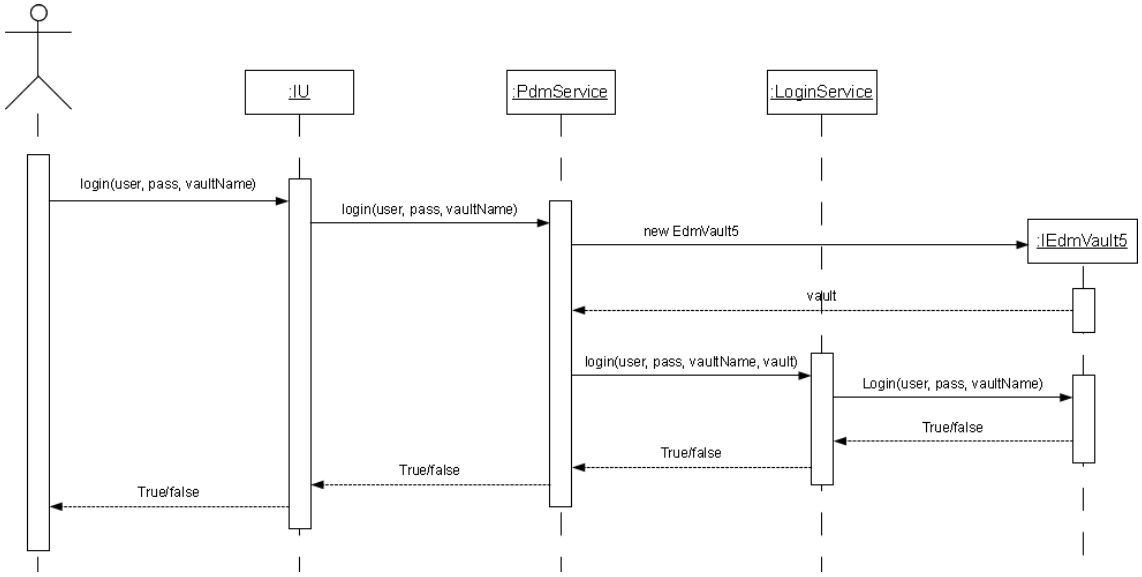

Ilustración 36: Diagrama de secuencia – Identificar

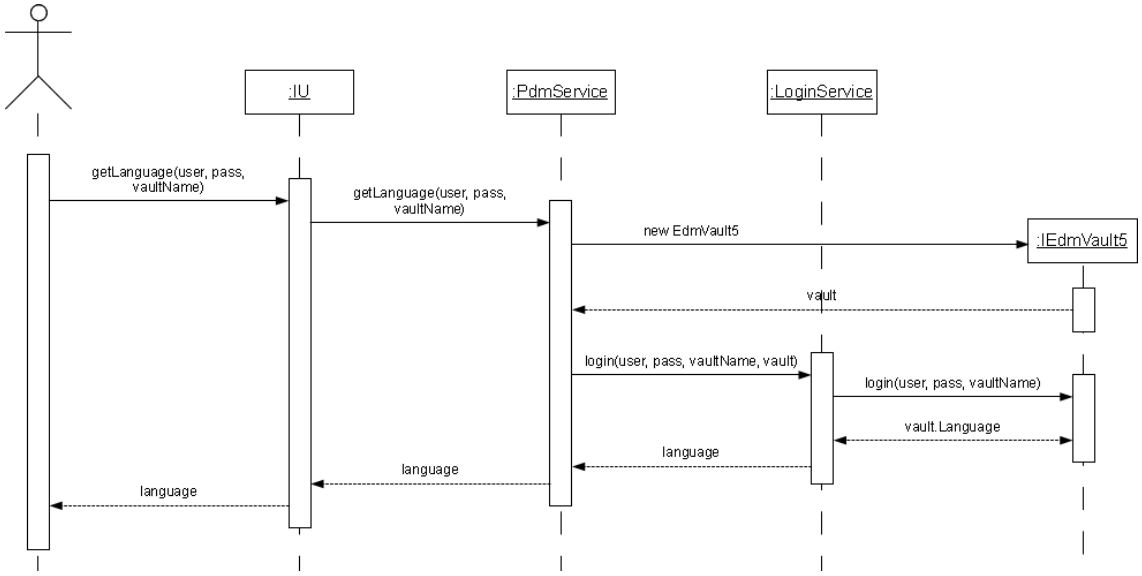

Ilustración 37: Diagrama de secuencia - Obtener idioma

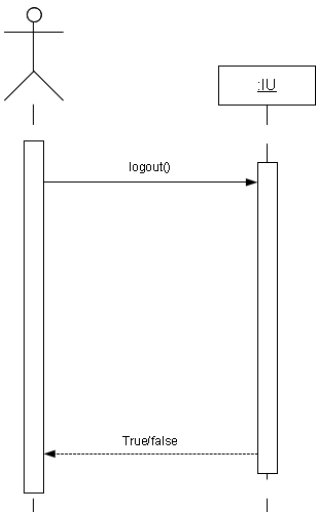

Ilustración 38: Diagrama de secuencia - Salir del sistema

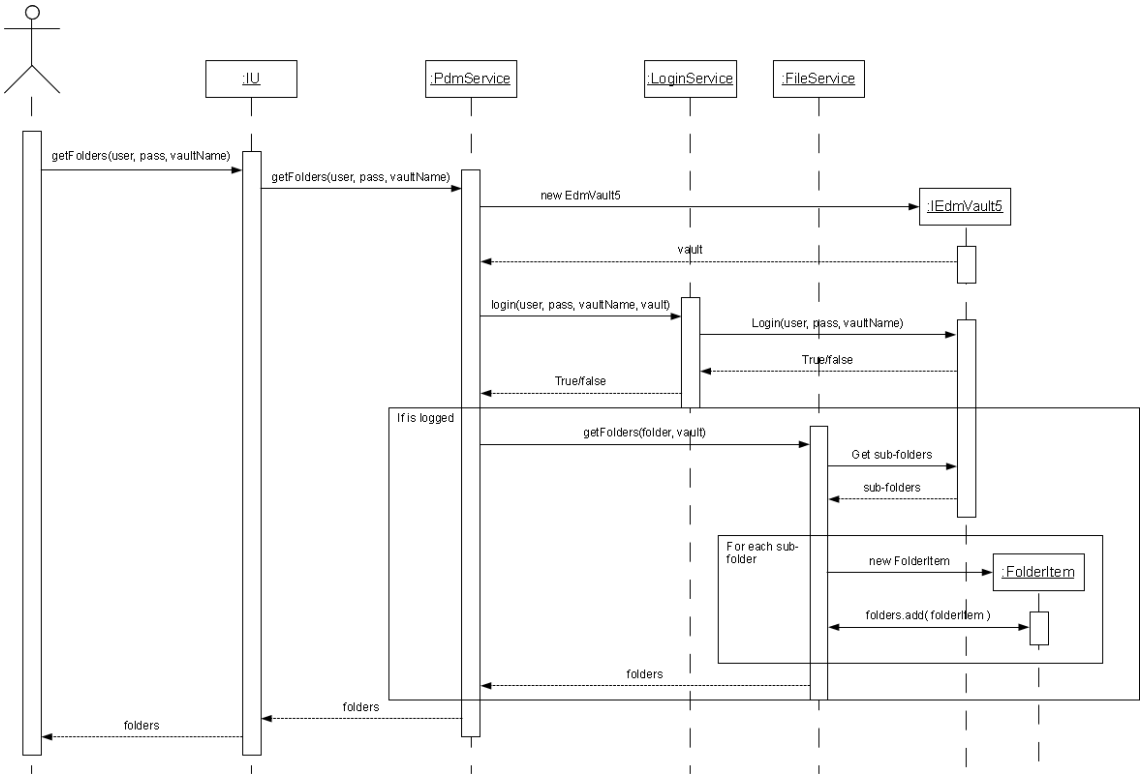

Ilustración 39: Diagrama de secuencia - Mostrar carpetas (completo)

Como se puede observar en la *Ilustración 39*, antes de realizar la llamada a la clase *FileService* (que es la que interesa), el servicio *PdmService* realiza la identificación de usuario. Debido al gran tamaño de los diagramas, a partir de este punto se simplificarán los casos y se omitirá el paso de la identificación. Pero se debe tener en cuenta que realmente la primera acción es siempre identificar al usuario. Se puede ver como ejemplo la *Ilustración 40* que recoge el diagrama de **Mostrar carpetas** simplificado.

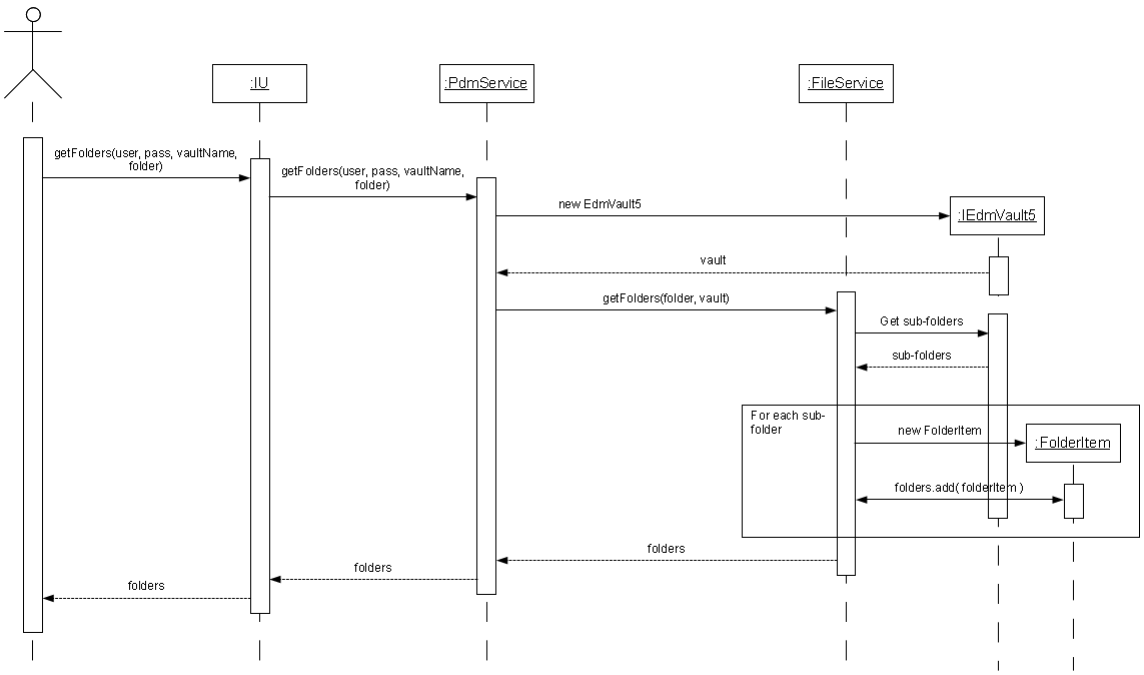

Ilustración 40: Diagrama de secuencia - Mostrar carpetas

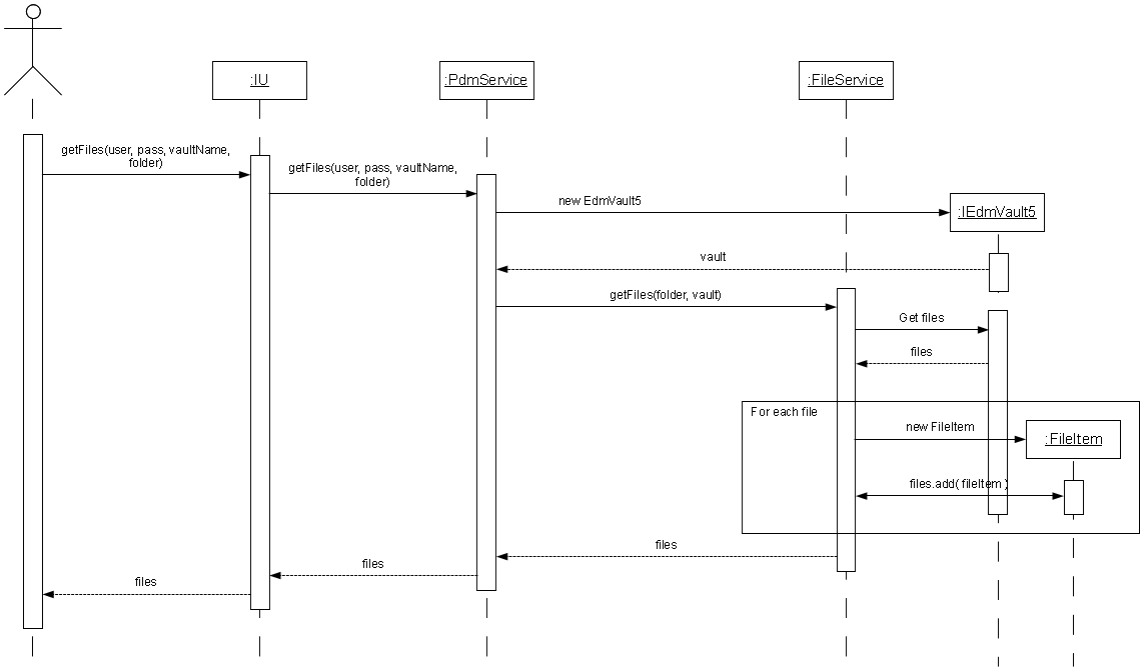

Ilustración 41: Diagrama de secuencia - Mostrar archivos

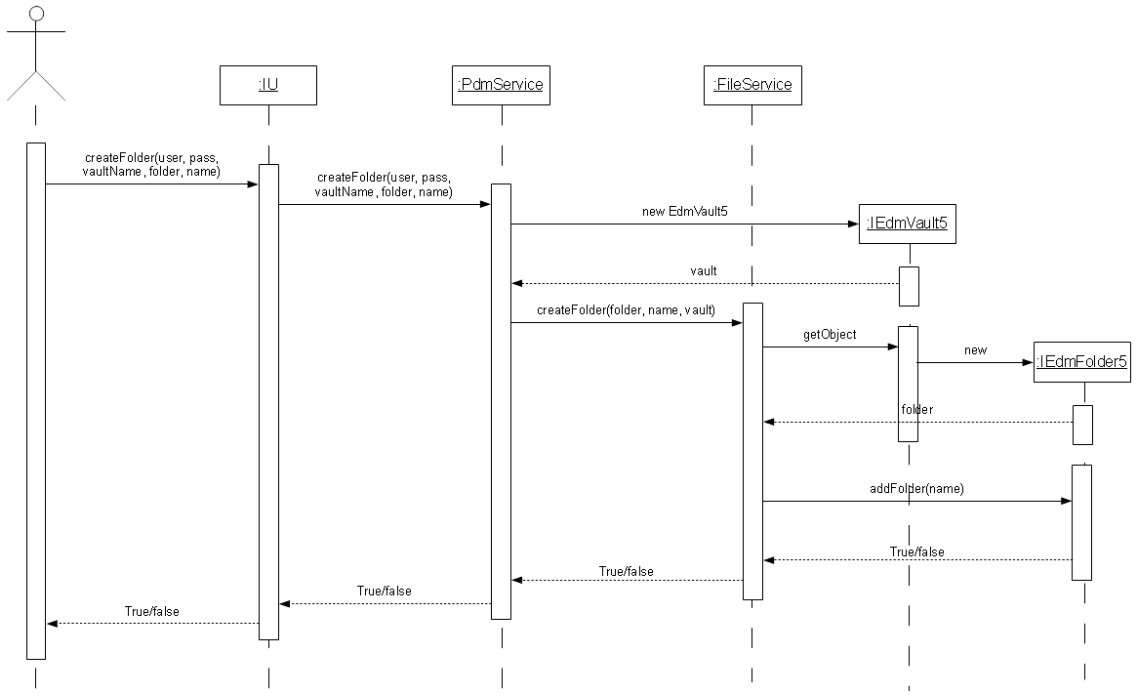

Ilustración 42: Diagrama de secuencia - Crear carpeta

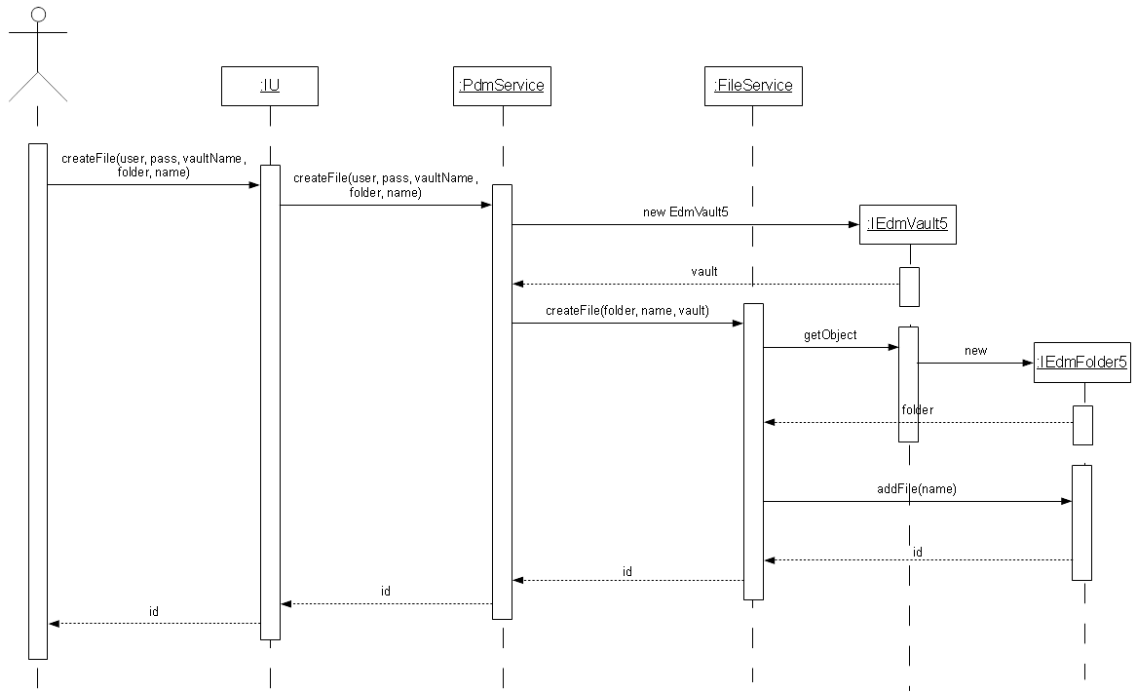

Ilustración 43: Diagrama de secuencia - Crear archivo

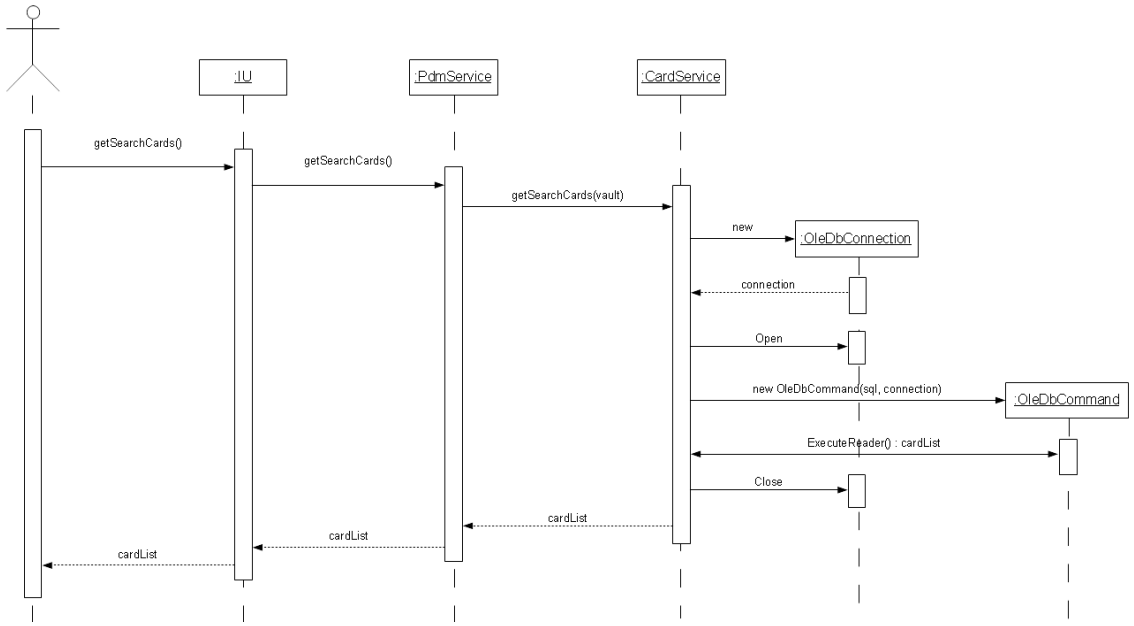

Ilustración 44: Diagrama de secuencia - Obtener tarjetas de búsqueda

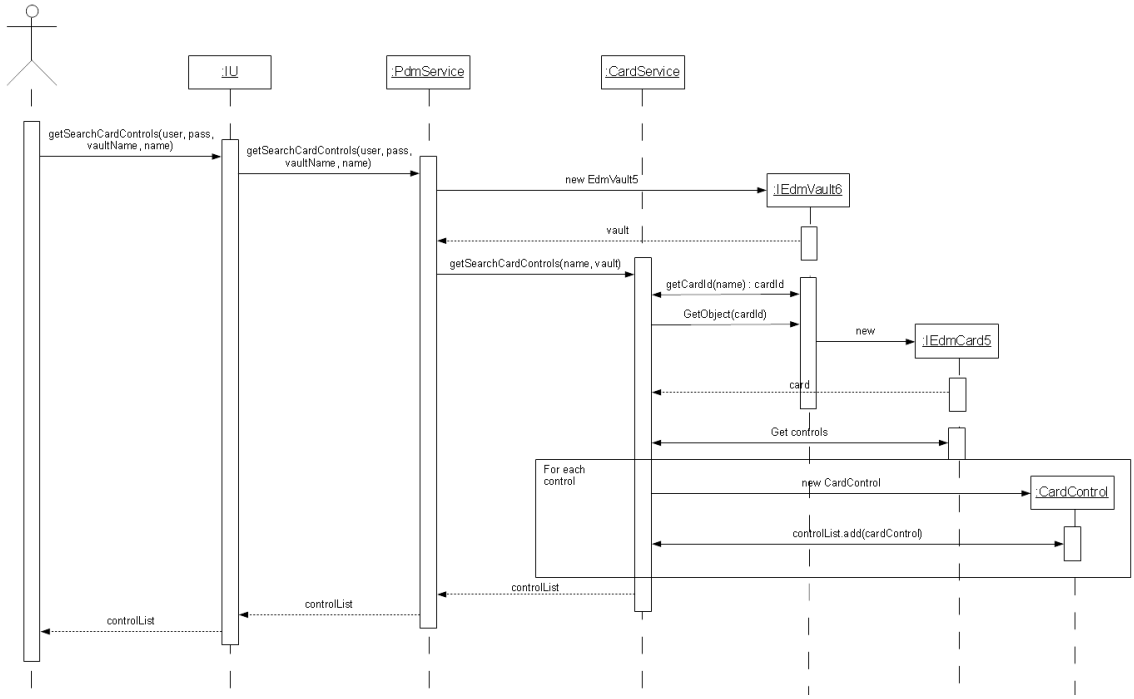

Ilustración 45: Diagrama de secuencia - Obtener controles de tarjeta de búsqueda

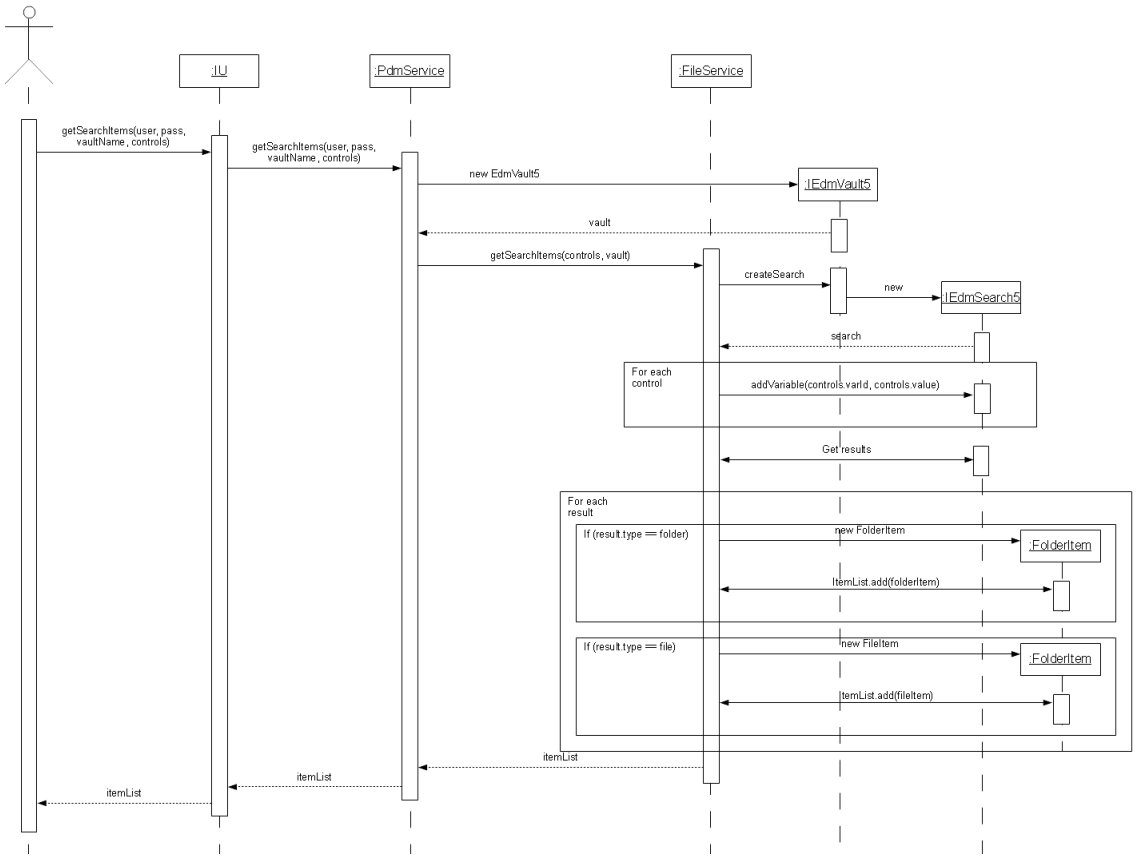

Ilustración 46: Diagrama de secuencia - Obtener ítems encontrados

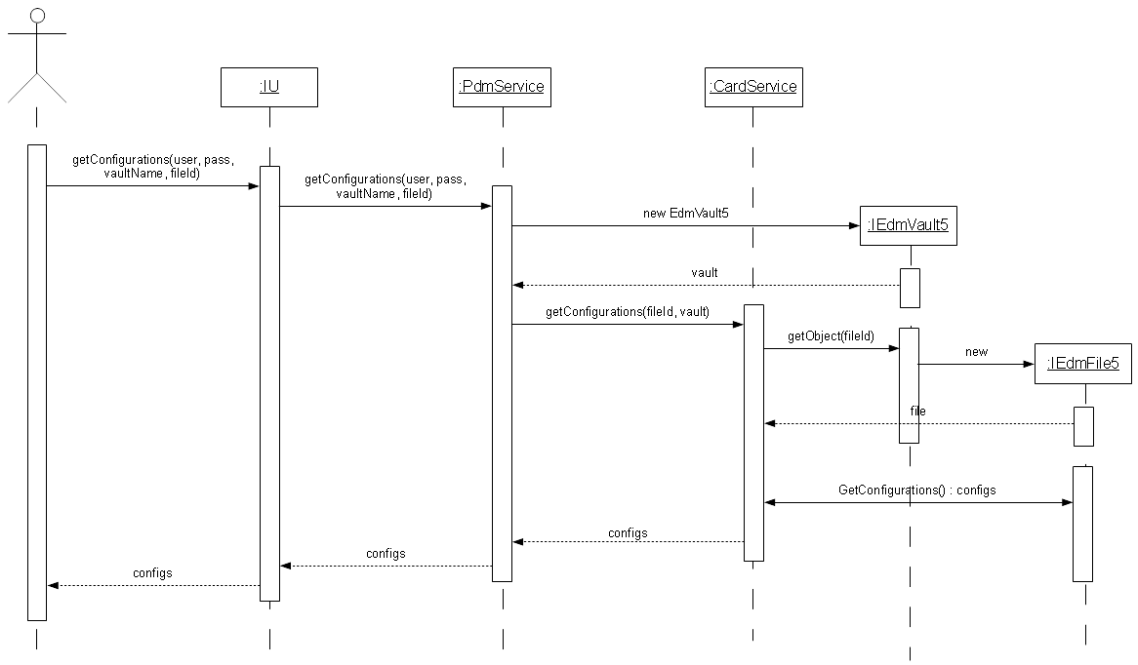

Ilustración 47: Diagrama de secuencia - Obtener configuraciones

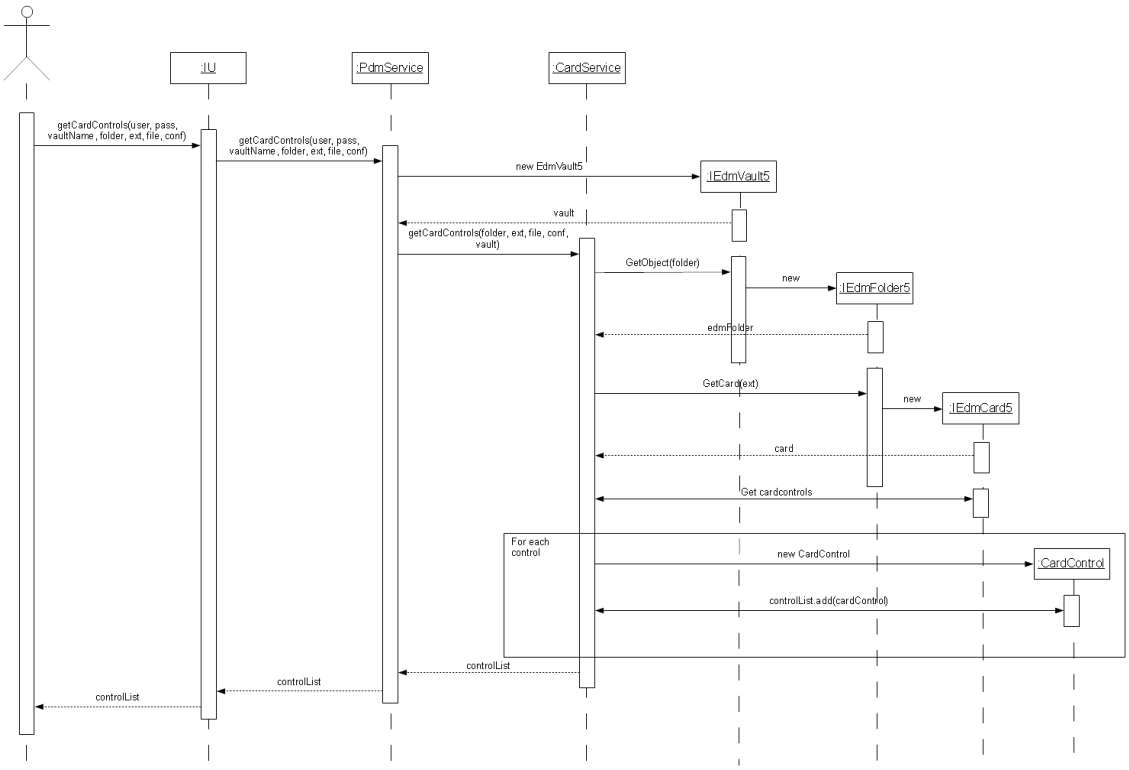

Ilustración 48: Diagrama de secuencia - Obtener controles de tarjeta

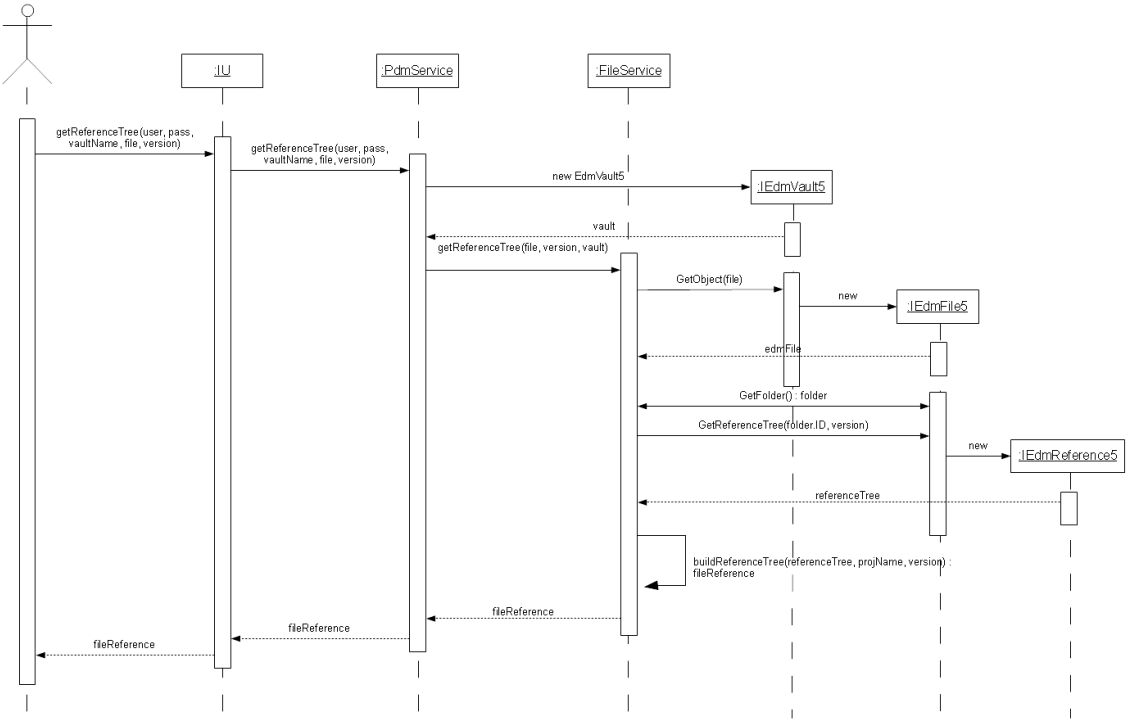

Ilustración 49: Diagrama de secuencia - Obtener árbol de referencias

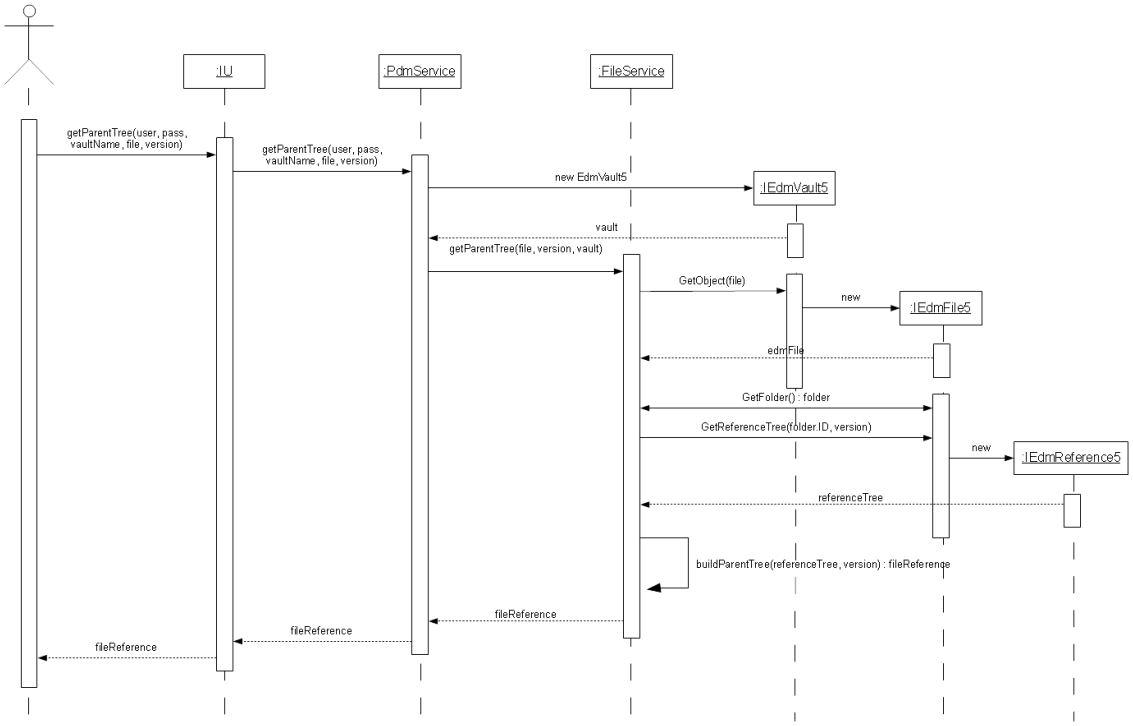

Ilustración 50: Diagrama de secuencia - Obtener árbol de padres

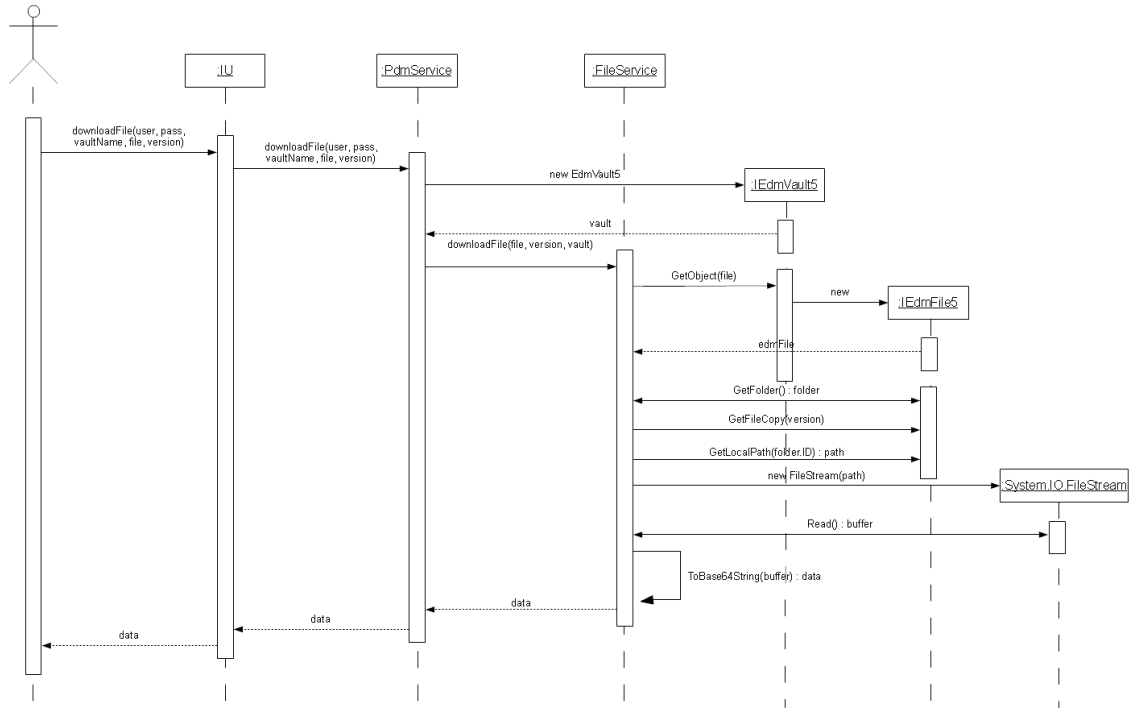

Ilustración 51: Diagrama de secuencia - Descargar archivo

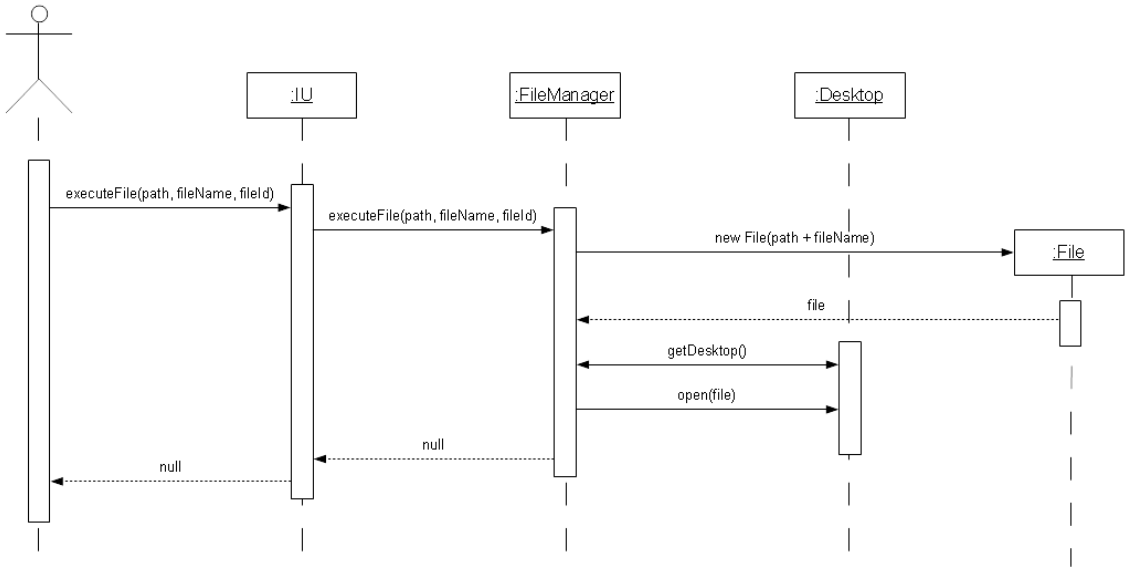

Ilustración 52: Diagrama de secuencia - Ejecutar archivo

A pesar de encontrarse entre el resto de los servicios, esta última función se ha implementado utilizando Java, ya que se requiere el acceso a los recursos de la máquina local para poder abrir un archivo con un programa asociado.

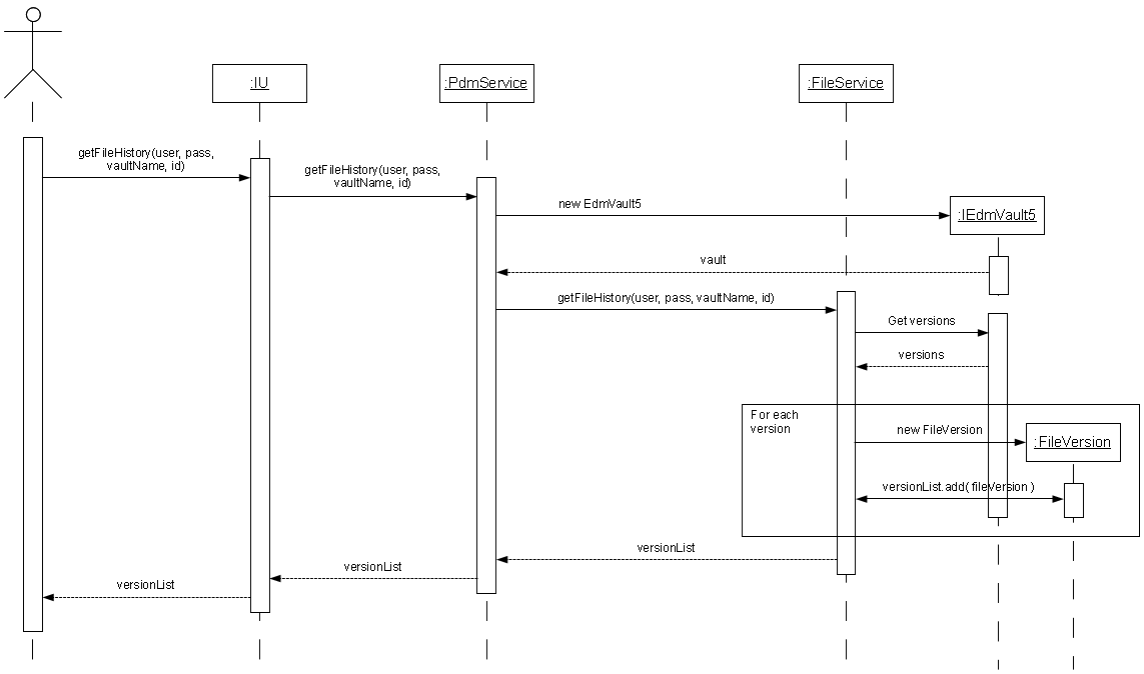

Ilustración 53: Diagrama de secuencia – Obtener histórico de archivo

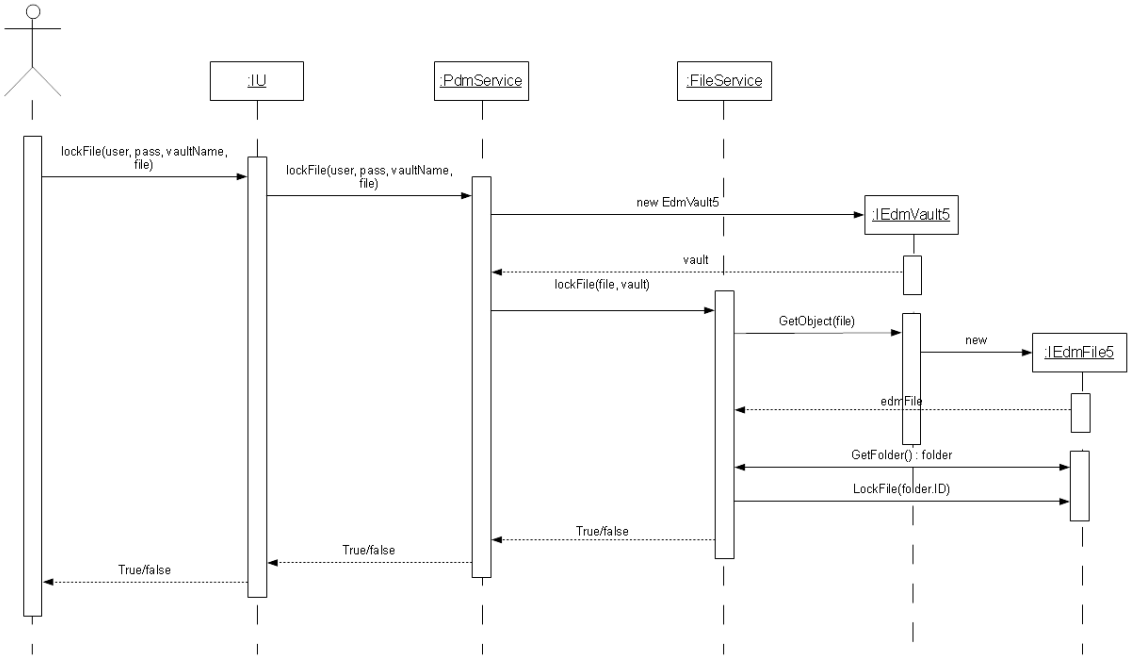

Ilustración 54: Diagrama de secuencia - Traer archivo

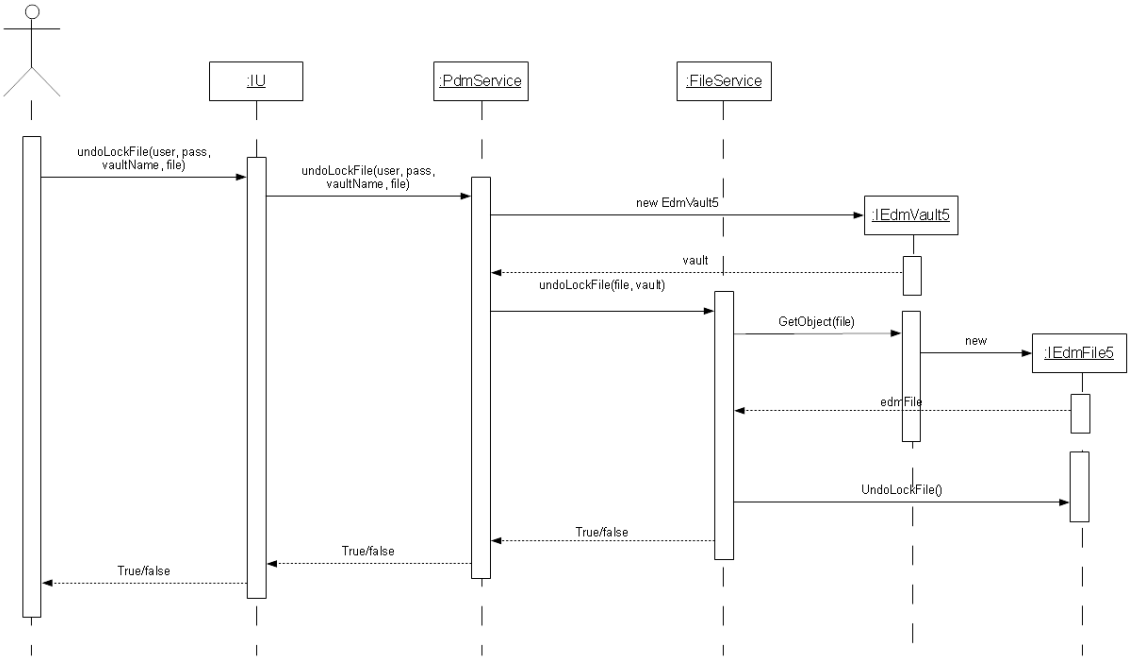

Ilustración 55: Diagrama de secuencia - Deshacer traer

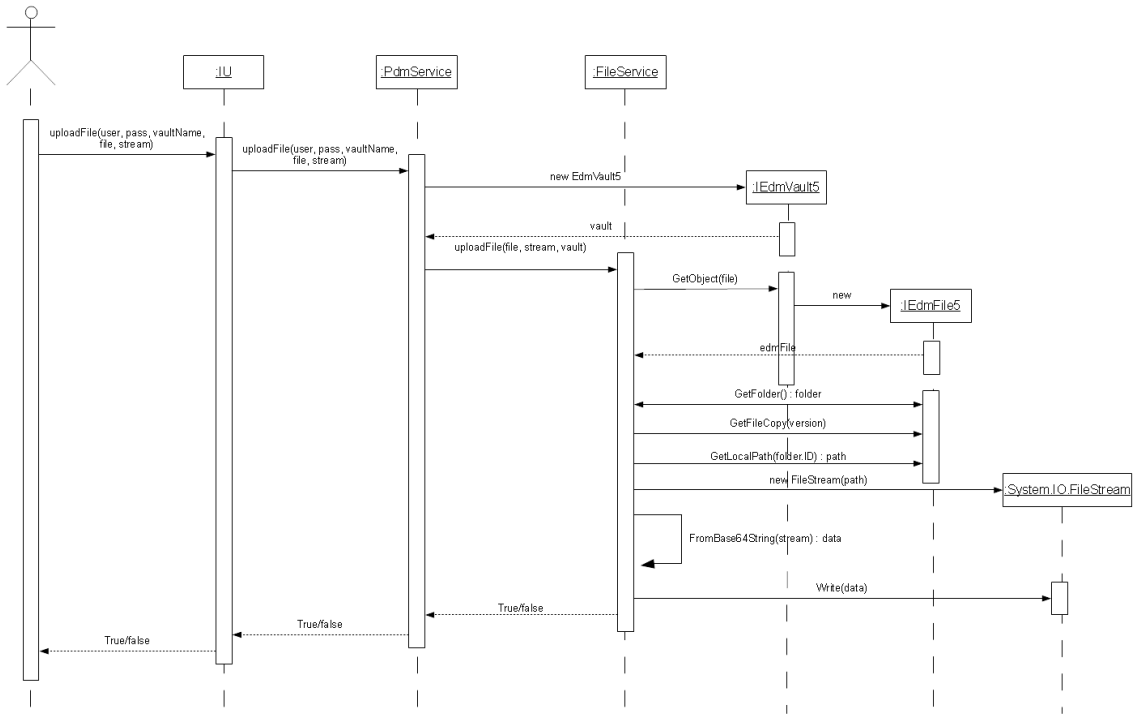

Ilustración 56: Diagrama de secuencia - Subir archivo

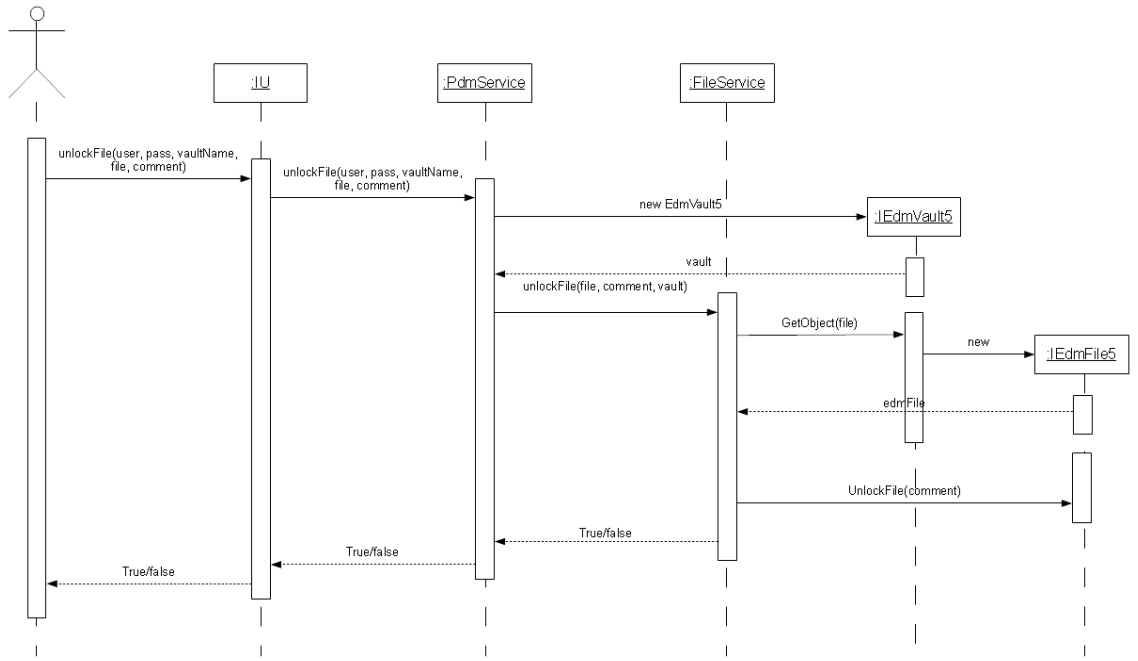

Ilustración 57: Diagrama de secuencia - Registrar archivo

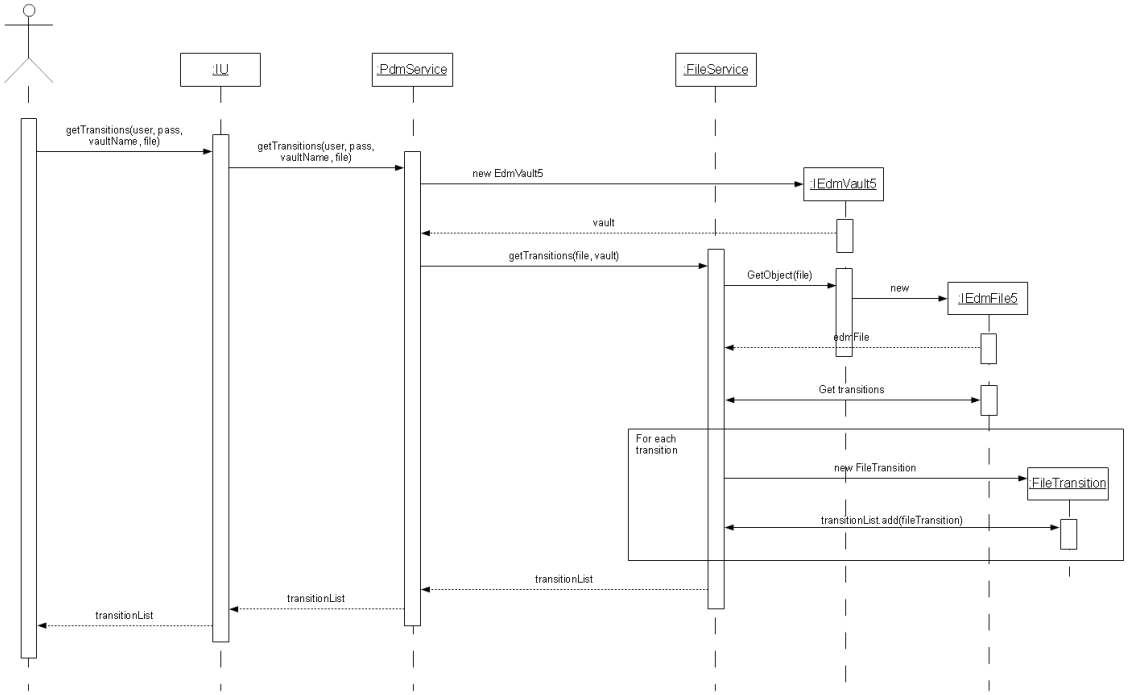

Ilustración 58: Diagrama de secuencia - Obtener transiciones

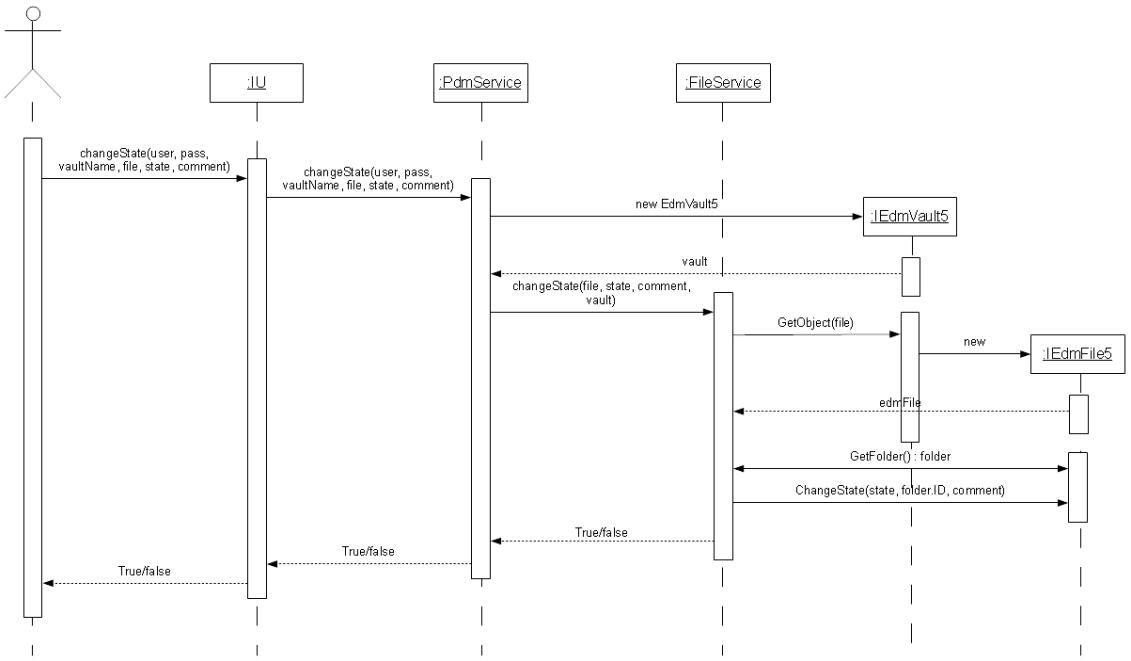

Ilustración 59: Diagrama de secuencia - Cambiar estado

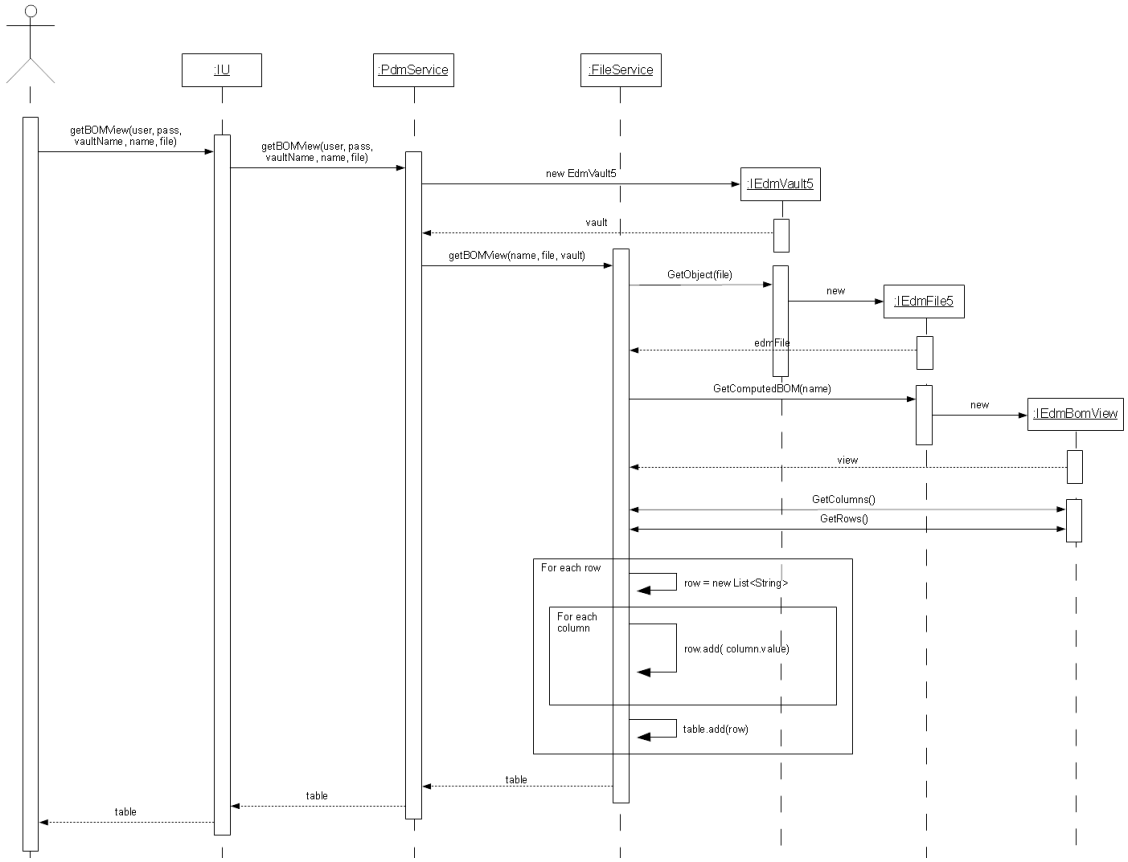

Ilustración 60: Diagrama de secuencia - Obtener lista de materiales

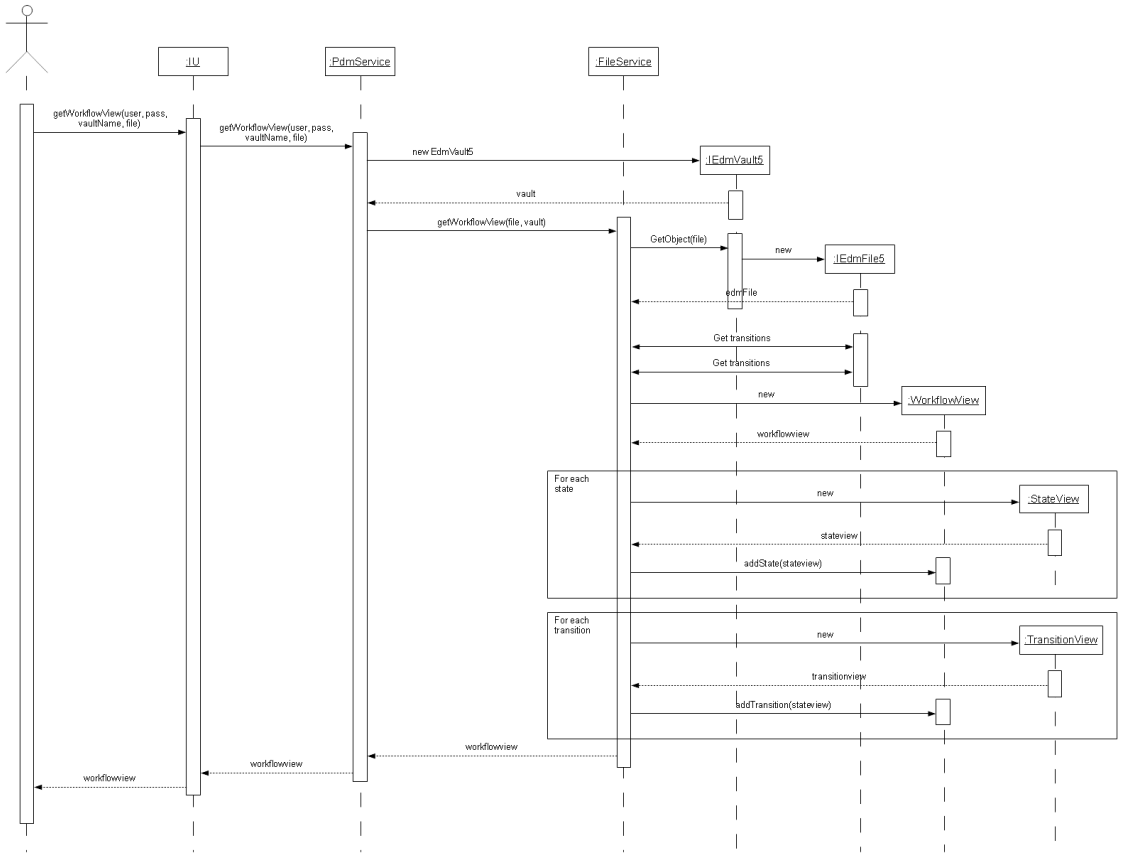

Ilustración 61: Diagrama de secuencia - Obtener vista del flujo de trabajo

4.2 DIAGRAMA DE CLASES

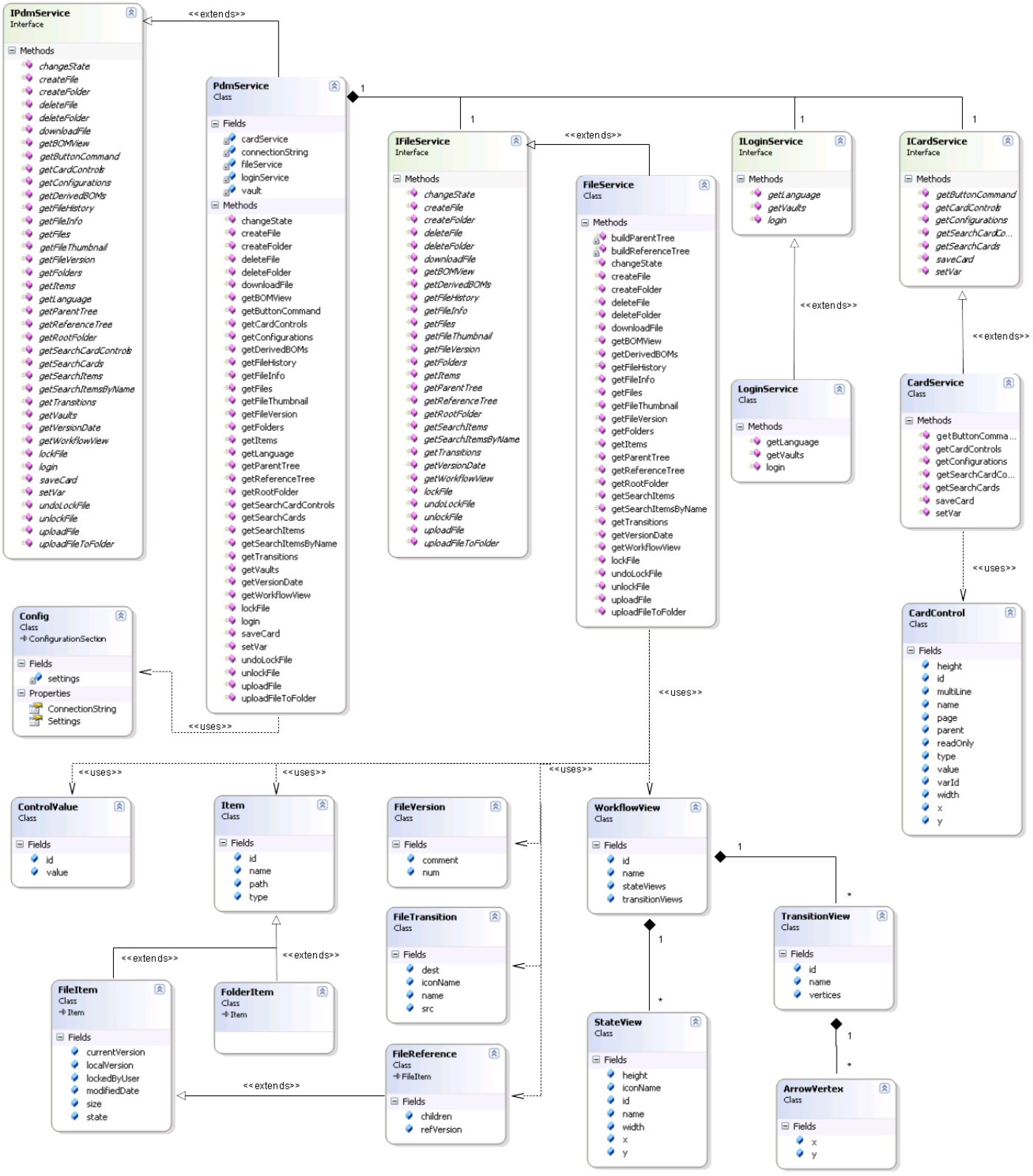

Ilustración 62: Diagrama de clases

El Diagrama de Clases muestra la estructura y los objetos que se implementan en los servicios web. Son el resultado del diseño y ofrecen métodos con los que el usuario podrá interactuar con la funcionalidad ofrecida por *SolidWorks Enterprise PDM*. Estos métodos son llamados desde una interfaz web con *JavaScript*, de modo que toda la información transmitida es serializada en formato *JSON* en la implementación de la clase **PdmService**.

5. IMPLEMENTACIÓN

En el presente capítulo se expondrán los detalles de la implementación tanto de la interfaz como de la lógica de negocio. Además, se analizarán los problemas surgidos durante el desarrollo del proyecto para que en un futuro sirva como manual para el desarrollador.

5.1 CAPA DE PRESENTACIÓN

En este Proyecto de Fin de Carrera, se han desarrollado dos interfaces de usuario para dos clientes distintos, que en esencian, consumen los mismos servicios programados en C#. A pesar de ello, cada uno ofrece distintas funcionalidades dependiendo de la plataforma para la que se ha desarrollado.

Las dos interfaces se han realizado para la Web, y se han desarrollado usando *HTML5* y *JavaScript*. Cabe destacar la importancia que ha tenido la librería *JQuery* para el manejo de los objetos *DOM*. Además, la interfaz reducida para los dispositivos móviles se ha desarrollado usando el *framework JQueryMobile*, que ofrece una *API* intuitiva de crear aplicaciones web para estos dispositivos de forma rápida y sin preocuparse por el aspecto.

5.1.1 INTERFAZ WEB

La interfaz de *PdmWeb* ha ido evolucionando de forma gradual empezando de un diseño totalmente básico y terminando con un aspecto similar a las herramientas web más utilizadas hoy en día. En la siguiente ilustración se puede apreciar el primer borrador de la interfaz que ofrecía la funcionalidad de visualizar y descargar el archivo seleccionado.

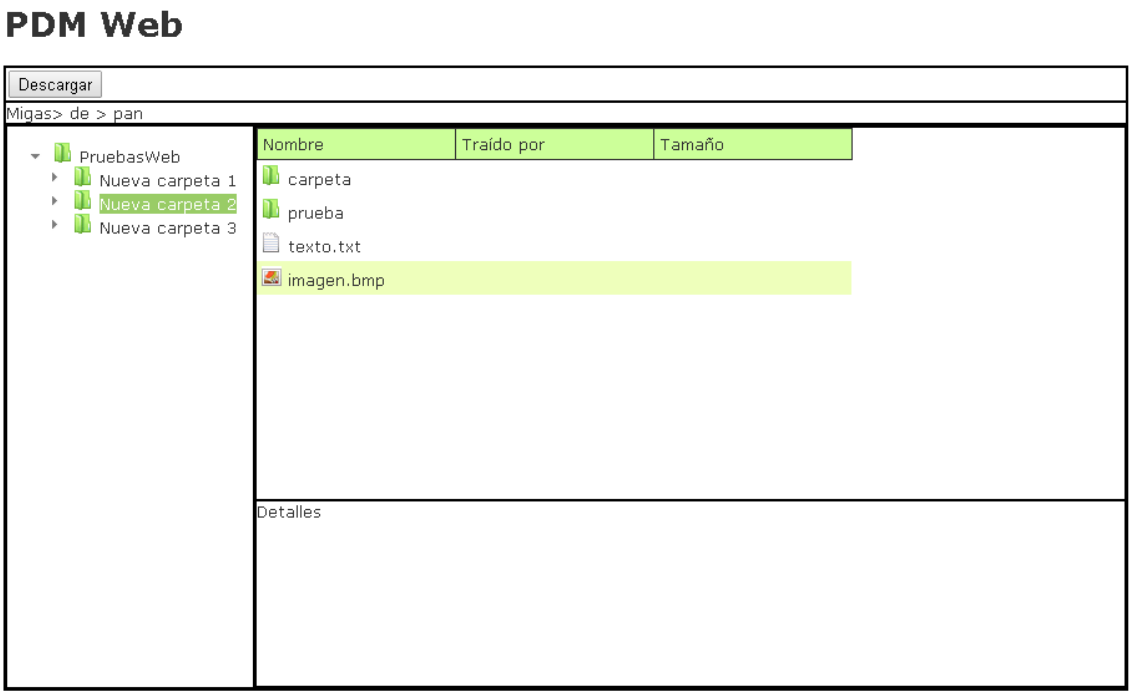

Ilustración 63: Evolución PdmWeb

Poco a poco la interfaz ha ido mejorando gracias a las recomendaciones de otros miembros del equipo hasta lograr una interfaz adecuada para las funciones que ofrece la aplicación web. En las siguientes imágenes se procede a describir cada parte de la nueva interfaz.

Ilustración 64: Pantalla de identificación

En la *Ilustración 64* se puede observar la primera pantalla por la que se accede al resto de las funcionalidades, es la pantalla de identificación. Ofrece al usuario la posibilidad de identificarse en el sistema proporcionando un nombre de usuario y contraseña. También es posible seleccionar el almacén en el que desea identificarse, por defecto es la primera de la lista.

Si los datos introducidos no son correctos, la aplicación muestra un mensaje de error para que el usuario pueda volver a intentarlo. De lo contrario, la identificación se realizara satisfactoriamente y la aplicación redirigirá al usuario a la pantalla principal.

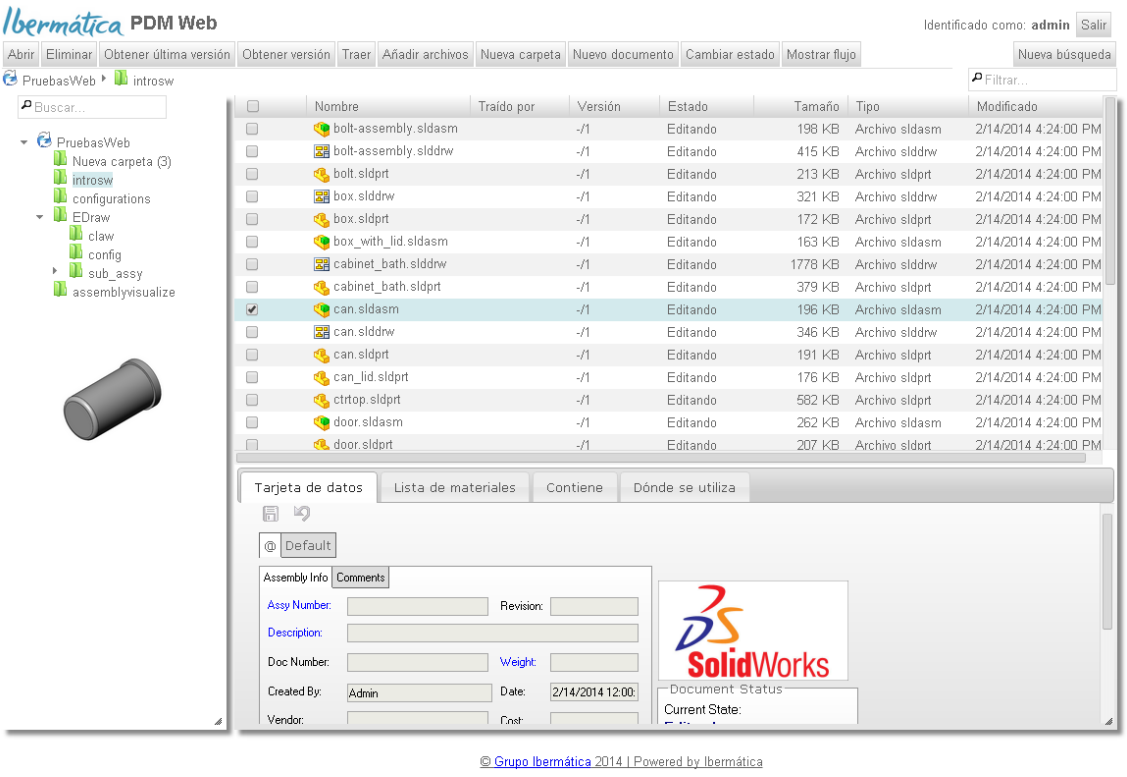

Ilustración 65: PdmWeb - Pantalla principal

La pantalla principal se divide principalmente en varias vistas, cuyos tamaños se pueden adaptar clicando y arrastrando los bordes. Además, cabe mencionar que el idioma de la interfaz se obtiene de la configuración del usuario de *PDM*, de este modo, un usuario configurado en inglés verá el contenido en su idioma.

La parte superior cuenta con una barra de acciones que cambian dependiendo de los archivos seleccionados y sus propiedades. Se trata de las funcionalidades básicas que ofrece *PdmWeb*. Además, se puede salir del sistema para volver a la pantalla de identificación.

Ilustración 66: PdmWeb - Barra de acciones

A la izquierda se encuentra la vista que permite visualizar y navegar por las carpetas del almacén. Debajo de las carpetas se muestra la vista previa del archivo que ha sido seleccionado.

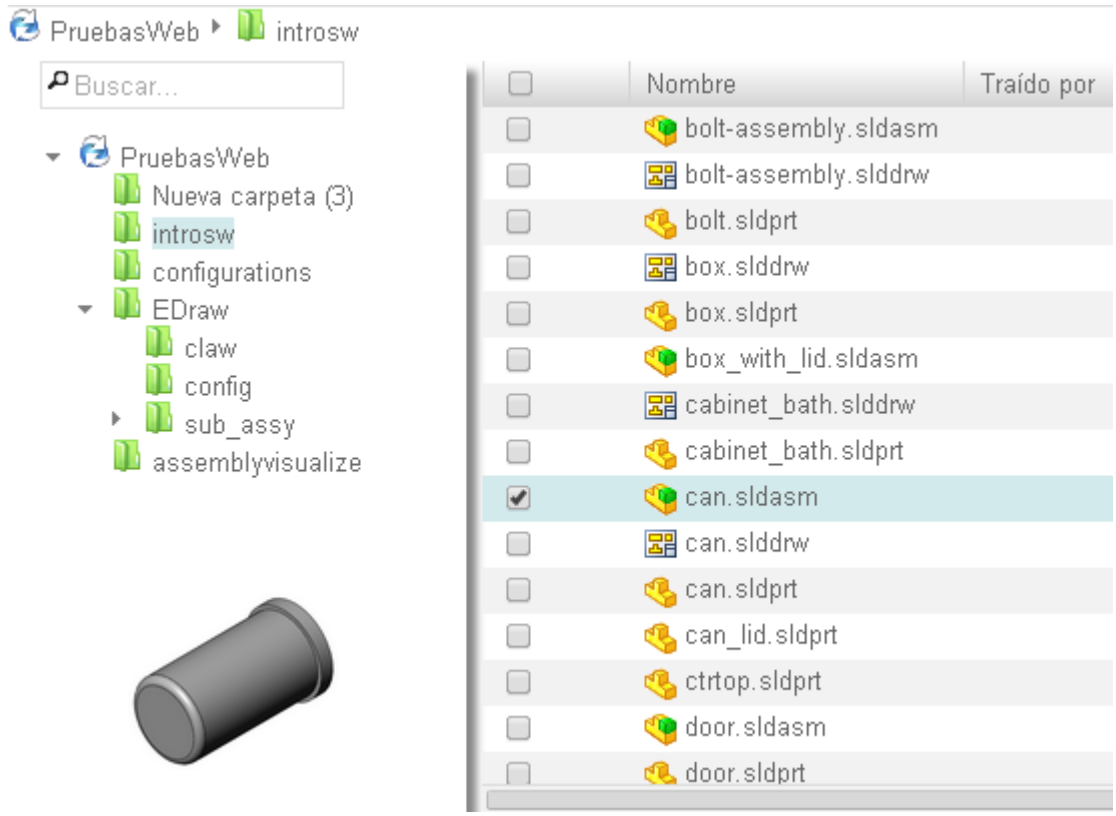

Ilustración 67: PdmWeb - Navegación

En la parte central, se muestra el contenido de la carpeta seleccionada y se ofrece la posibilidad de seleccionar más de un archivo para operar sobre ellos. Se muestra una columna por cada propiedad del elemento, y también es posible mostrar u ocultarlas haciendo clic derecho sobre la cabecera de las columnas. Además, el clic derecho sobre los archivos despliega un menú contextual que permite realizar las mismas acciones que se ofrecen en la barra superior para tener un acceso rápido.

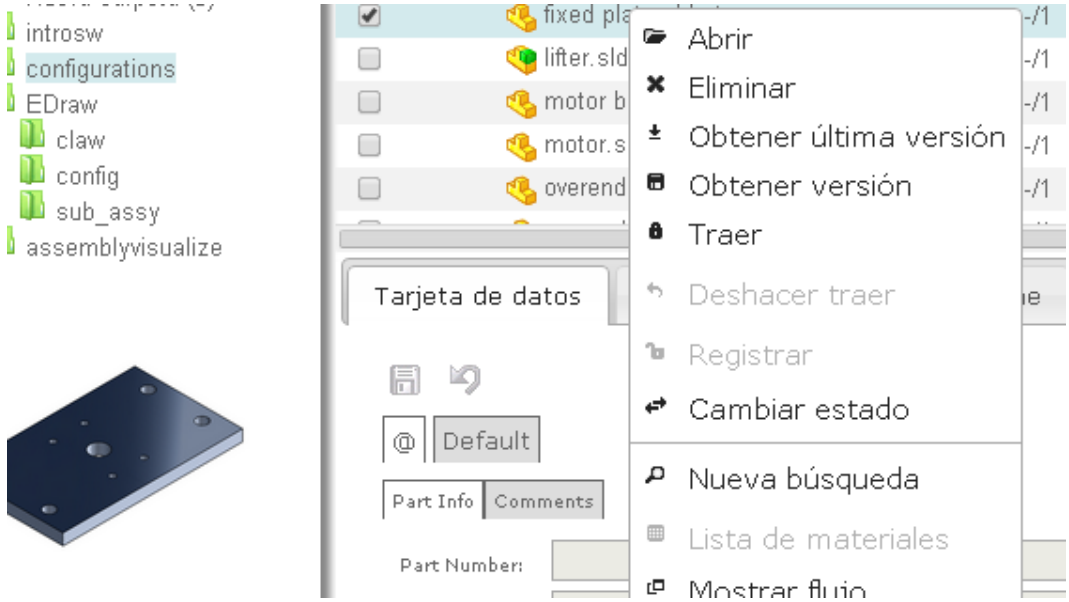

Ilustración 68: PdmWeb - Menú contextual

Finalmente, en la parte inferior se encuentra la vista que muestra los detalles del archivo seleccionado. Entre ellos: la tarjeta de datos, la lista de materiales y dos formas distintas de ver las referencias del archivo. En la siguiente imagen se puede ver la tarjeta de datos asociada a un tipo de archivo en concreto. En este caso se trata de una tarjeta de prueba que se ha creado con el editor de *PDM* para ver que es capaz de mostrarla correctamente.

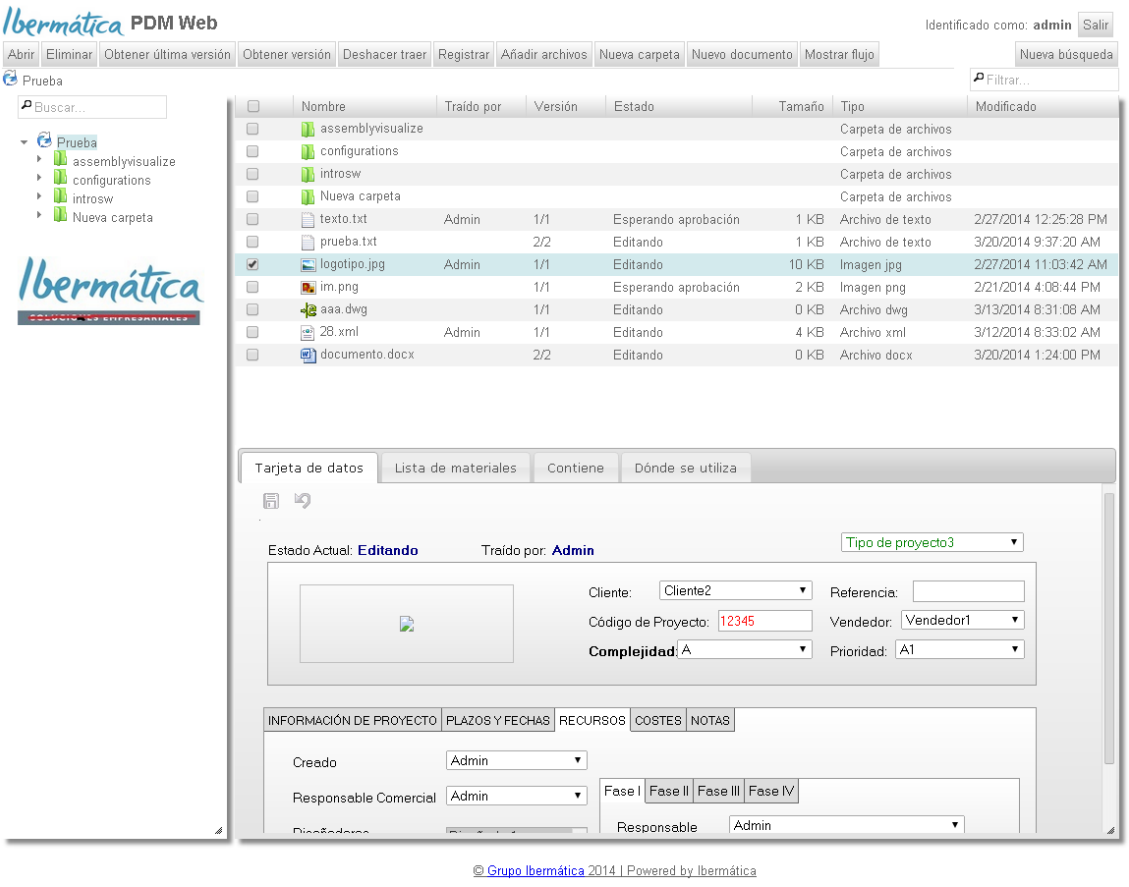

Ilustración 69: PdmWeb - Tarjeta de datos

Además, para los tipos de archivo que admiten configuraciones (como son los de SolidWorks), también se muestra la tarjeta para distintas configuraciones. En la siguiente *Ilustración 70* se pueden apreciar, sobre la tarjeta, las pestañas que permiten visualizar cada configuración.

Ilustración 70: PdmWeb - Tarjeta de datos SLDPRT

La segunda pestaña "Lista de materiales", como su nombre indica, muestra la lista de materiales del ensamblaje o plano seleccionado. Lo plasma en una estructura de árbol con las propiedades para cada elemento.

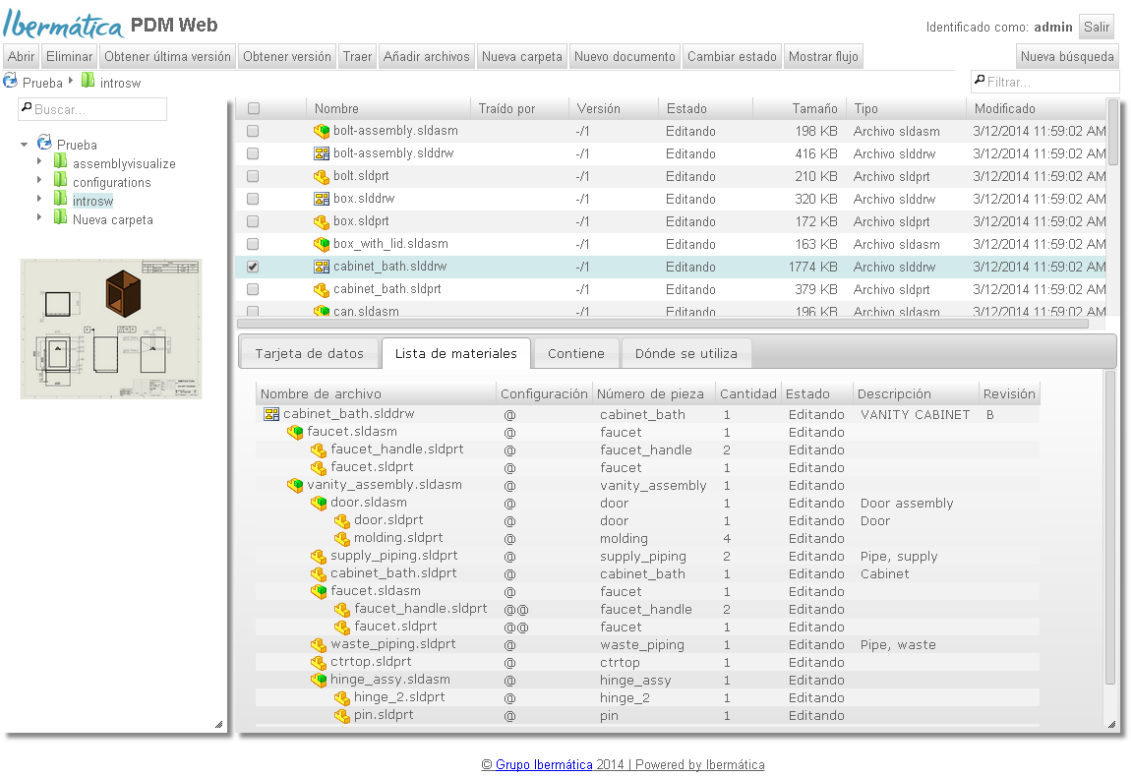

Ilustración 71: PdmWeb - Lista de materiales

Las siguientes pestañas son las que muestran las referencias del archivo, también, en una estructura de árbol. La pestaña “Contiene”, es prácticamente la misma que la lista de materiales, es decir, el elemento seleccionado es la raíz y de él derivan todas las referencias de forma recursiva.

Sin embargo, la pestaña de “Dónde se utiliza”, es de lógica inversa. En este caso, el árbol muestra sus referencias padre de forma recursiva hasta llegar a la raíz (generalmente un plano *.slddrw*). En la siguiente *Ilustración 72* se puede apreciar que la pieza seleccionada es utilizada en un ensamblaje, que a su vez es utilizada en un plano.

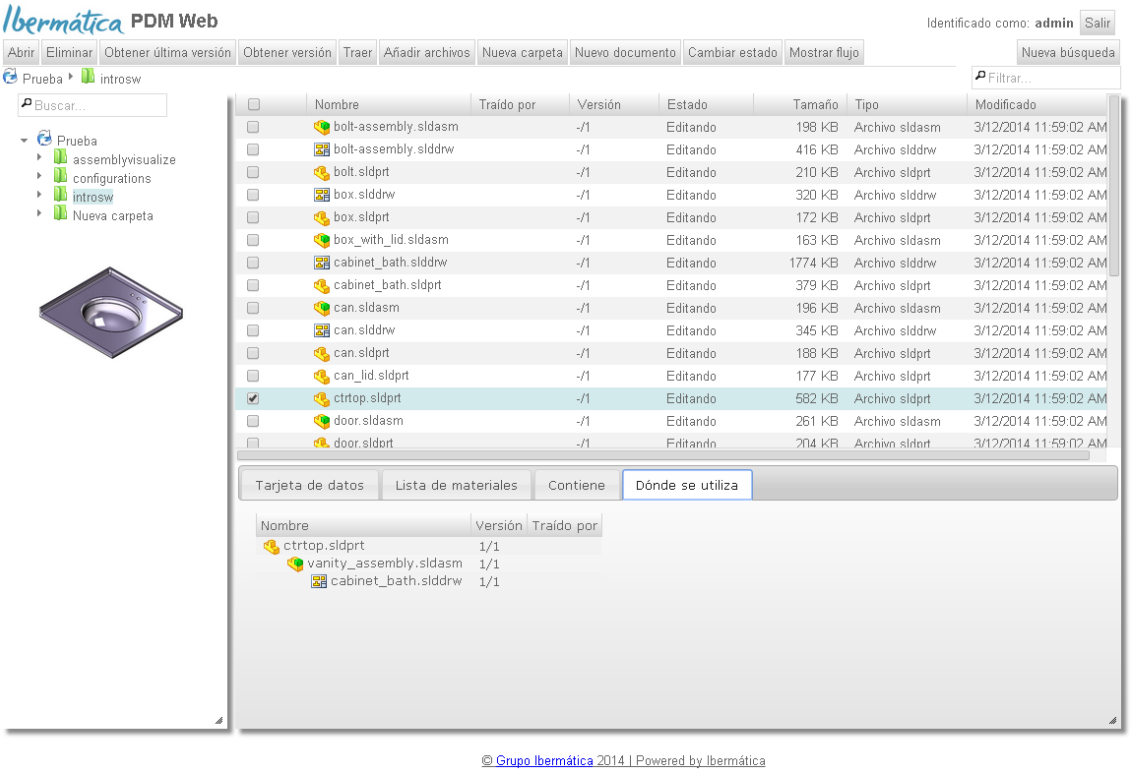

Ilustración 72: PdmWeb - Dónde se utiliza

PdmWeb soporta realizar operaciones sobre varios archivos seleccionados al mismo tiempo. Para realizar acciones como obtener, traer, deshacer o registrar, la aplicación lanza un recuadro que permite confirmar la acción sobre los archivos deseados. Además, si la acción se hace sobre un archivo con referencias, este recuadro construirá el árbol de las referencias para que también se pueda operar sobre ellos.

Los recuadros que se muestran encima de la aplicación pueden ser desplazados arrastrándolos con el ratón e incluso redimensionados. Para cancelar las acciones y cerrar el recuadro, basta con clicar el botón correspondiente o clicar fuera del recuadro.

En la siguiente *Ilustración 73*, por ejemplo, se puede observar el recuadro que muestra para la descarga de la selección de una pieza y un plano con referencias.

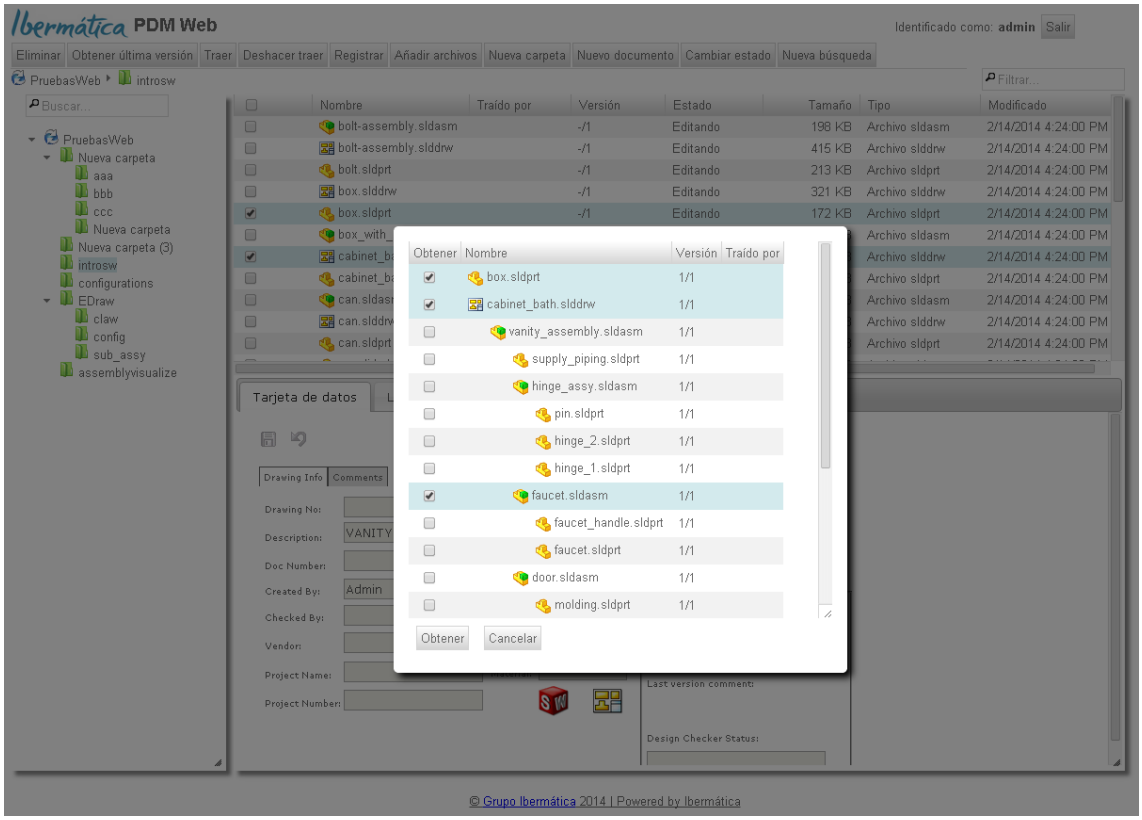

Ilustración 73: PdmWeb - Pantalla de descarga de archivos

Las demás operaciones ya mencionadas muestran una pantalla similar a la que se acaba de mostrar. Pero para la acción de cambiar el estado de un archivo, el recuadro es diferente si se ha seleccionado solo un archivo o más de uno.

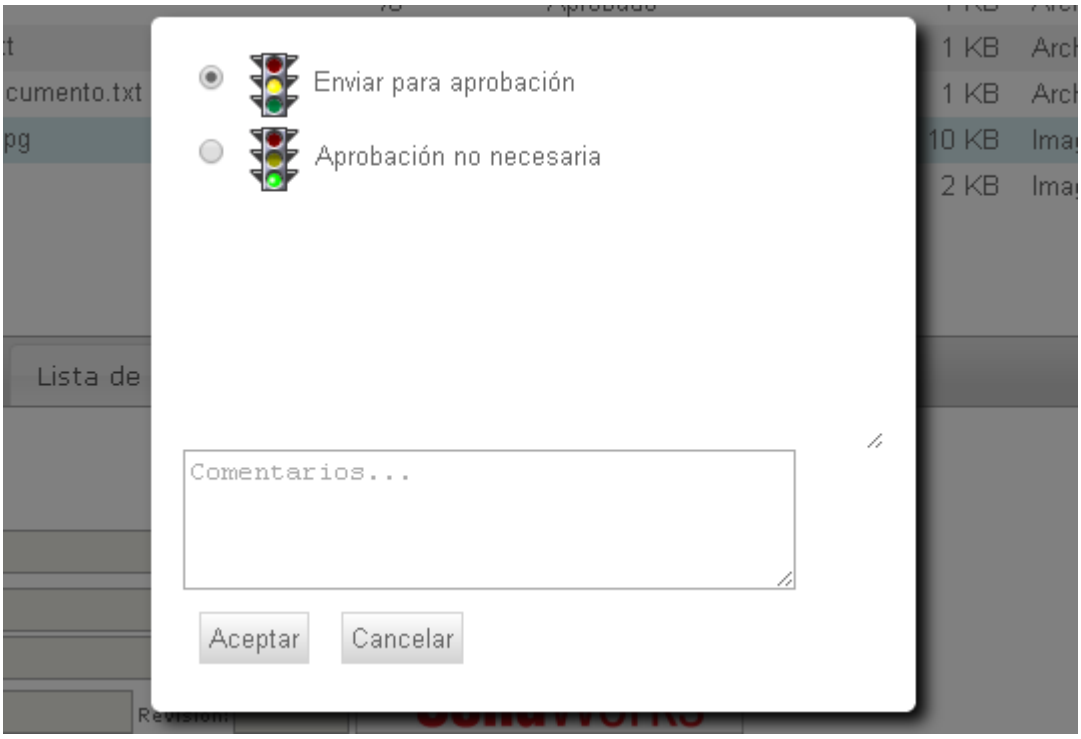

Ilustración 74: PdmWeb - Cambio de estado de un solo documento

La *Ilustración 74* muestra la interfaz del cambio de estado para un solo documento, mientras que la siguiente muestra el cambio de estados para una selección de archivos. Gracias a esto, es posible cambiar de estado más de un archivo con poco esfuerzo, ya que con los controles inferiores (lista y caja de texto), es posible realizar una misma transición para todos los archivos (siempre que su estado actual tenga dicha transición).

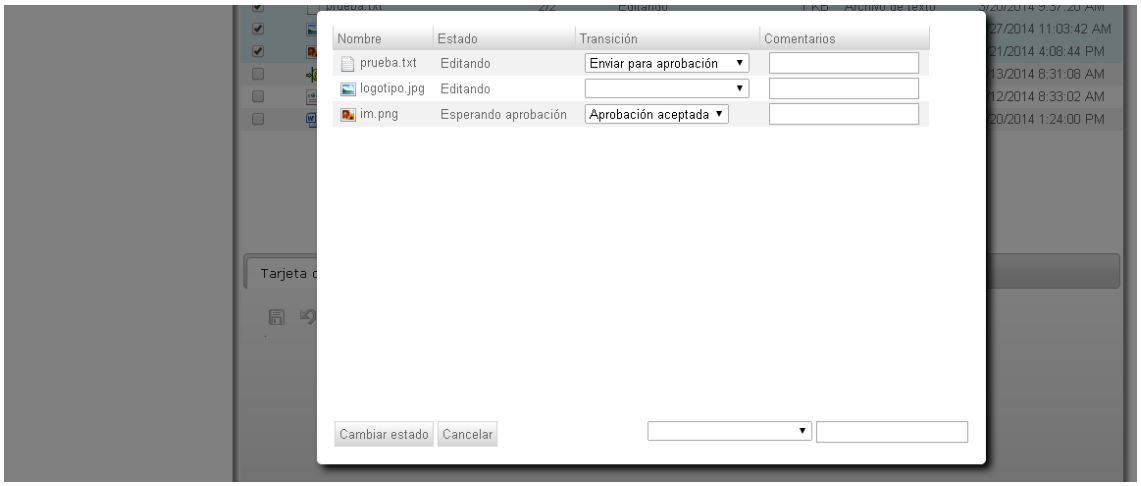

Ilustración 75: PdmWeb - Cambio de estado de varios documentos

Para finalizar, *PdmWeb* cuenta con una funcionalidad nueva que no ofrecen las versiones oficiales. Ésta nos permite poder visualizar el flujo de trabajo en el que se encuentra el archivo seleccionado, y así poder ver en qué estado del flujo se encuentra. El estado actual esta distinguido de color azul para que el usuario lo pueda identificar.

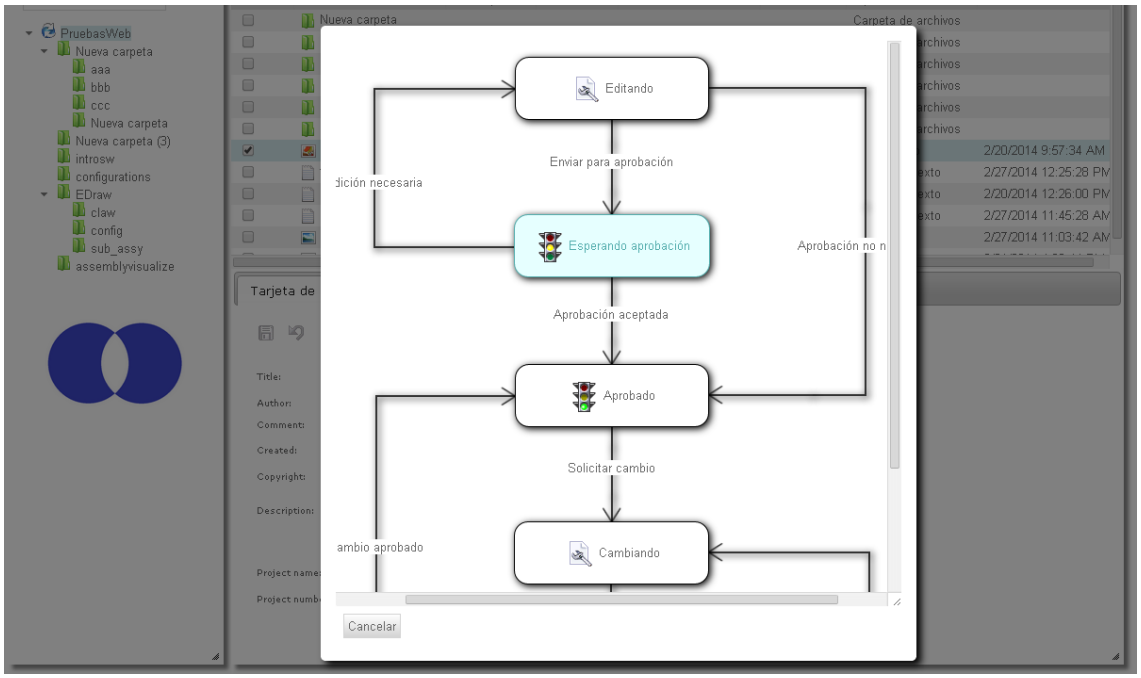

Ilustración 76: PdmWeb - Flujo de trabajo

5.1.2 INTERFAZ MÓVIL

A diferencia de la interfaz web, la interfaz móvil ha tenido desde el principio el mismo aspecto. Gracias a *JQuery Mobile* se ha conseguido lograr una interfaz adaptada a los dispositivos táctiles de hoy en día. Además, al haber implementado las funcionalidades básicas en servicios WCF, solo se ha tenido que realizar llamadas a los servicios y mostrar la información de forma adecuada. Esto ha permitido desarrollar la aplicación móvil en tan solo unos días.

Para empezar, el usuario accederá a la pantalla de identificación, en el que, al igual que la interfaz web, deberá proporcionar el almacén, el usuario y la contraseña para seguir adelante. La cabecera que contiene el nombre de la aplicación será visible en todas las pantallas, con el que se podrá interactuar mediante sus botones para volver atrás, mostrar la información de la aplicación etc.

Ilustración 77: Pantalla de identificación móvil

Una vez identificado, el usuario accede a la pantalla principal donde se muestra el contenido del almacén. En la *Ilustración 78* se puede ver cómo la pantalla nos ofrece controles para poder navegar por los directorios del almacén. Además, permite filtrar el resultado mediante el cuadro de texto ubicado bajo estos controles de navegación.

También cabe destacar, que al igual que la interfaz web, ésta también obtiene y muestra el contenido a partir del idioma configurado para el usuario de *Enterprise PDM*.

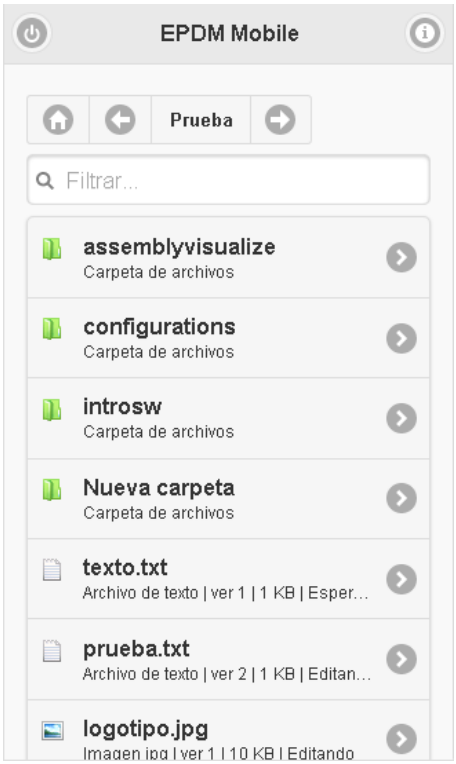

Ilustración 78: Navegación móvil

Para poder visualizar el contenido de un directorio basta con hacer clic sobre una carpeta. Por otro lado, seleccionar un archivo redirigirá al usuario a una nueva pantalla que mostrará los detalles del archivo y una vista previa del mismo.

Además de ofrecer información adicional, esta pantalla contiene dos botones que permiten al usuario descargar el archivo y cambiar de estado respectivamente. El primer botón descarga la última versión del archivo en el directorio de descargas por defecto del dispositivo.

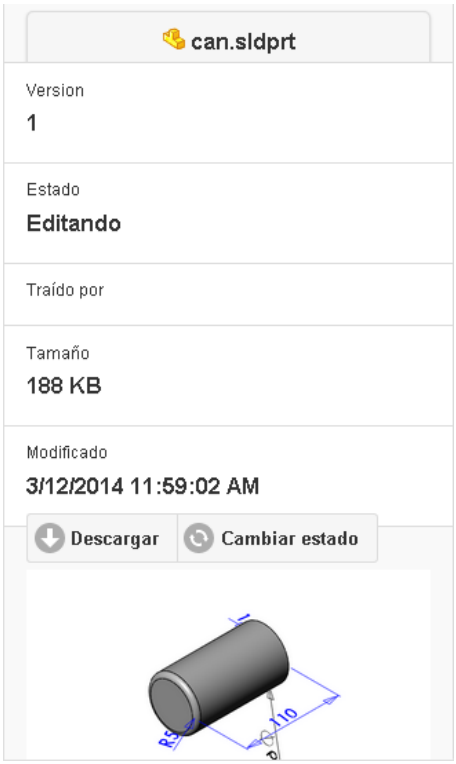

Ilustración 79: Visualización de documento en móvil

El segundo botón, ofrece al usuario una nueva pantalla en el que podrá cambiar de estado el archivo seleccionado. En la Ilustración 80 se puede ver que esta pantalla permite seleccionar la transición deseada y registrar el cambio con comentarios.

Ilustración 80: Pantalla móvil de cambio de estado

5.2 LÓGICA DE NEGOCIO

La parte lógica se ha desarrollado sobre el entorno *Visual Studio 2010* utilizando el lenguaje *C#* y la plataforma *.NET*. Como ya se ha visto en el diagrama de clases, la lógica parte de un servicio principal, una clase llamada **PdmService** que ofrecerá a la interfaz web todos los métodos necesarios para operar sobre los almacenes de *EPDM*.

Para organizar mejor la estructura del código, los métodos se han agrupado en distintas clases con funcionalidad similar, como son: **FileService**, **CardService** y **LoginService**. De este modo, estas clases implementan la lógica de cada método, y la clase **PdmService** toma el papel de un proxy realizando las llamadas a dichas clases. Ya que la clase **PdmService** es la que interactúa directamente con la interfaz web, ésta se encarga de serializar los datos transmitidos en formato *JSON*.

```
public class PdmService : IPdmService
{
    private IEdmVault5 vault = new EdmVault5();

    private ILoginService loginService = new LoginService();
    private IFileService fileService = new FileService();
    private ICardService cardService = new CardService();
    private IProjectService projectService = new ProjectService();

    private string connectionString = Config.Settings.ConnectionString;
    /// <summary>
    /// Identifica un usuario
    /// </summary>
    /// <param name="user">Nombre del usuario</param>
    /// <param name="password">Contraseña del usuario</param>
    /// <param name="vaultName">Nombre del vault</param>
    /// <returns>Devuelve true si se ha identificado con éxito, pero devuelve false si ha habido algún error</returns>
    public string login(string user, string password, string vaultName)
    {
        return (new JavaScriptSerializer().Serialize(loginService.login(user, password, vaultName, vault)));
    }
```

Ilustración 81: Implementación de PdmService

Como se puede ver en la anterior imagen, la clase **PdmService** actúa como proxy haciendo la llamada a otra clase. En este caso, la función *login* realiza la llamada al método *login* de la instancia de la clase **LoginService** y devuelve el resultado serializándolo mediante la clase *JavaScriptSerializer*. Por otro lado, tenemos la implementación de este método.

```
public class LoginService : ILoginService
{
    public bool login(string user, string password, string vaultName, IEdmVault5 vault)
    {
        try
        {
            vault.Login(user, password, vaultName);
        }
        catch (Exception e)
        {

            return false;
        }

        if (vault.IsLoggedIn)
        {
            return true;
        }

        return false;
    }
```

Ilustración 82: Implementación de LoginService

En la *Ilustración 82* se puede ver la implementación del método *login* de **LoginService**. Este método hace uso de la *API* de *PDM* para poder identificar al usuario mediante los parámetros dados. En caso de haber algún error o no identificarse correctamente, devolverá el valor "false" y si todo va bien, devolverá "true".

Esto es un resumen de cómo están implementados los servicios web. Todas las funciones ofrecidas comparten la misma estructura y hace que sea más fácil extender los servicios con nuevas funcionalidades.

5.3 CAPA DE DATOS

En la capa de datos no se han implementado concretamente las clases del dominio, ya que como se ha comentado anteriormente, el Diagrama de Clases del Dominio solo muestra una aproximación de lo que internamente contiene *SolidWorks Enterprise PDM*. Los objetos y métodos que ofrece la *API* realizan consultas a la base de datos.

Esta base de datos se crea automáticamente en el proceso de instalación del servidor *EPDM*, por lo que no ha sido necesario crear ninguna tabla. De todas formas, la instalación requiere tener instalado un Sistema de Getsión de Bases de Datos en la máquina, en este caso *SQL Server*.

Para la capa de datos se ha instalado *SQL Server* en su versión 2008 R2. Se desconoce el modo en el que la *API* realiza las consultas a la base de datos, pero de todas formas en algún caso se ha tenido que hacer consultas directas a la base de datos sin pasar por las llamadas a la *API*, ya que era necesario obtener información que la *API* no ofrece. Estas consultas se han podido realizar gracias a la tecnología *OLE DB* desarrollada por *Microsoft*.

```
connectionString += ";Initial Catalog=" + vault.Name;
OleDbConnection connection = new OleDbConnection(connectionString);
connection.Open();

var sql = "select ButtonCmd from dbo.CardControls where ControlId=" + controlId;
var cmd = new OleDbCommand(sql, connection);
string dr = (string)cmd.ExecuteScalar();

connection.Close();
```

Ilustración 83: Conexión OLE DB

Como podemos ver en la imagen, *OLE DB* nos permite realizar las consultas a la base de datos. La conexión se realiza mediante una cadena de texto que contiene todos los parámetros. Este código, especifica el nombre de la base de datos (nombre del almacén) a la que se quiere realizar la conexión y realiza la consulta *SQL* deseada.

Los parámetros guardados en la variable *connectionString* inicialmente se cargan desde el archivo de configuración de los servicios. Gracias a él, el administrador del sistema podrá configurar los datos necesarios para dar acceso a la base de datos.

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <section name="dbConfig" type="Config"/>
  </configSections>
  <dbConfig
    connectionString="Data Source=localhost;Persist Security Info=True;User ID=uruario;Password=***;Provider=sqloledb"
    log="C:\pdmweb-error.log"
    />
  <system.web>
    <compilation debug="true" targetFramework="4.0"/>
  </system.web>
```

Ilustración 84: Configuración OLE DB

5.4 PROBLEMAS TÉCNICOS Y SOLUCIONES

A lo largo del desarrollo de este Proyecto de Fin de Carrera han ido surgiendo nuevos problemas que se han tenido que analizar a fondo para poder proponer soluciones efectivas. La inexperiencia del autor en ciertas áreas y tecnologías privativas, como son las de *Microsoft*, ha obligado a invertir un esfuerzo extra en la búsqueda de las soluciones de dichos problemas. Algunas de las soluciones propuestas pueden ser en un futuro indispensables para la implantación de este sistema en otras máquinas. Para verlo de forma más clara, los problemas se clasificarán en distintas categorías:

5.4.1 SISTEMA DE ARCHIVOS

En esta categoría cabe destacar las decisiones tomadas acerca del sistema de archivos, que recoge desde la replicación hasta la forma en el que se transferirán los archivos por la red. Un requisito fundamental en el proyecto es conseguir replicar el almacén en el sistema de archivos local. Desde el principio se analizaron y se hizo una presentación de las diferentes tecnologías que se podían usar para conseguir este fin:

- **El nuevo estándar de HTML5** implementa un *framework* que permite acceder al sistema de archivos local. Desgraciadamente este *framework* solo esta implementado en el navegador Chrome.
- **Active X** es la tecnología propietaria de Microsoft y actualmente usada por la interfaz web oficial de EPDM. Sin embargo, es solamente funcional en *Internet Explorer* y esta es una de las razones por las que se quiere evitar esta tecnología.
- **Los Applets de Java** ofrecen la posibilidad de acceder a los recursos de la maquina local siempre y cuando se les dé los permisos adecuados de ejecución. El hecho de ser multiplataforma y la facilidad que ofrece para integrarse con HTML/JavaScript hace que sea el candidato perfecto para lograr acceder al sistema de archivos.

Por lo tanto, se ha desarrollado un *Applet* que pone a disposición del usuario funciones capaces de acceder al sistema de archivos y descargar contenido de EPDM para poder replicar el almacén en el sistema local.

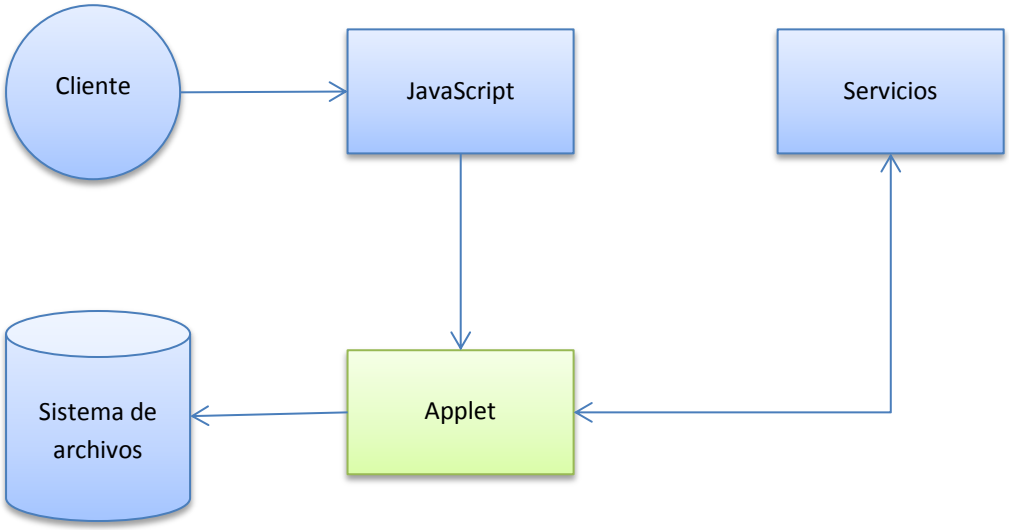

Ilustración 85: Solución Java

La comunicación entre *Java* y *JavaScript* se hace a nivel de código, realizando una llamada desde *JavaScript* al objeto *DOM (Document Object Model)* que representa el *Applet* de *Java*.

5.4.2 ARQUITECTURA DEL SISTEMA

La arquitectura del sistema ha sido un tema importante a lo largo del desarrollo del proyecto. A pesar de especificar una arquitectura inicial, por problemas que han ido surgiendo se han tenido que plantear otras soluciones más adecuadas.

En un principio se implementó la arquitectura pensando en que se tendría cliente y un servidor. Un servidor que alojaría las páginas HTML, los servicios web (adaptador) y el servidor PDM. Ya que los servicios web hacen uso del API, es necesario que se alojen en la misma máquina que el servidor PDM. Más adelante se vio la necesidad de separar el servidor web del resto de los componentes.

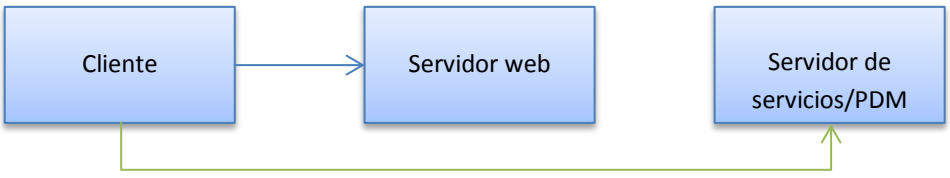

Ilustración 86: Arquitectura web

Pero esta solución trajo otros problemas relacionados con la seguridad. El mayor problema reside en el uso de JavaScript y está relacionado con el acceso del navegador del cliente al servidor de servicios. Técnicamente, el cliente descarga las páginas HTML del servidor web, y es el navegador quien ejecuta el código JavaScript y realiza la llamada a los servicios. Estas llamadas se hacen utilizando la técnica *Ajax* (*Asynchronous JavaScript And XML*), que por cuestiones de seguridad no permite las llamadas *Cross-domain*, es decir, acceder al contenido de un servidor desde las páginas de otro servido en diferente dominio.

Para solucionar este problema, finalmente se ha planteado crear una capa intermedia (un *proxy*) alojada en la misma máquina que el servidor web. Así, el cliente realizará las peticiones solamente al servidor web, y será éste el encargado de redirigir y responder a las peticiones de forma adecuada.

Ilustración 87: Arquitectura de proxy

5.4.3 SERVICIOS WEB

Uno de los problemas surgidos a la hora de implementar los servicios, ha sido un tema de permisos en la parte del servidor. Siempre que EPDM accede al almacén del servidor de archivado, lo hace con los permisos del usuario identificado. Sin embargo, siempre que los servicios implementados acceden a los archivos del almacén, lo hacen con el usuario especificado en la configuración del servidor IIS.

Por esta razón, los archivos creados por un usuario mediante el cliente pesado de EPDM no son accesibles para los usuarios web. La solución a esto es dar los permisos adecuados a la carpeta donde residen los archivos del almacén, para que todos los elementos existentes y los que vayan a ser creados, se creen con los permisos adecuados.

```
cacls c:\PruebasWeb /e /g "IIS APPPOOL\pdmservice":F
```

Con este comando, daremos control total al usuario "pdmservice" del IIS para todos los archivos usando la opción ":F".

Además de este problema de permisos, el más problemático surgió a la hora de implementar la funcionalidad de registrar los archivos. Por alguna razón, la ejecución de esta función lanzaba una excepción de tipo RPC_E_SERVERFAULT, que se refiere a un error en los objetos COM. Un error general que es difícil solucionar si no se tiene acceso al código fuente de la API de EPDM. De todas formas, gracias al visor de eventos de *Windows* se pudo solucionar el problema dando los permisos adecuados:

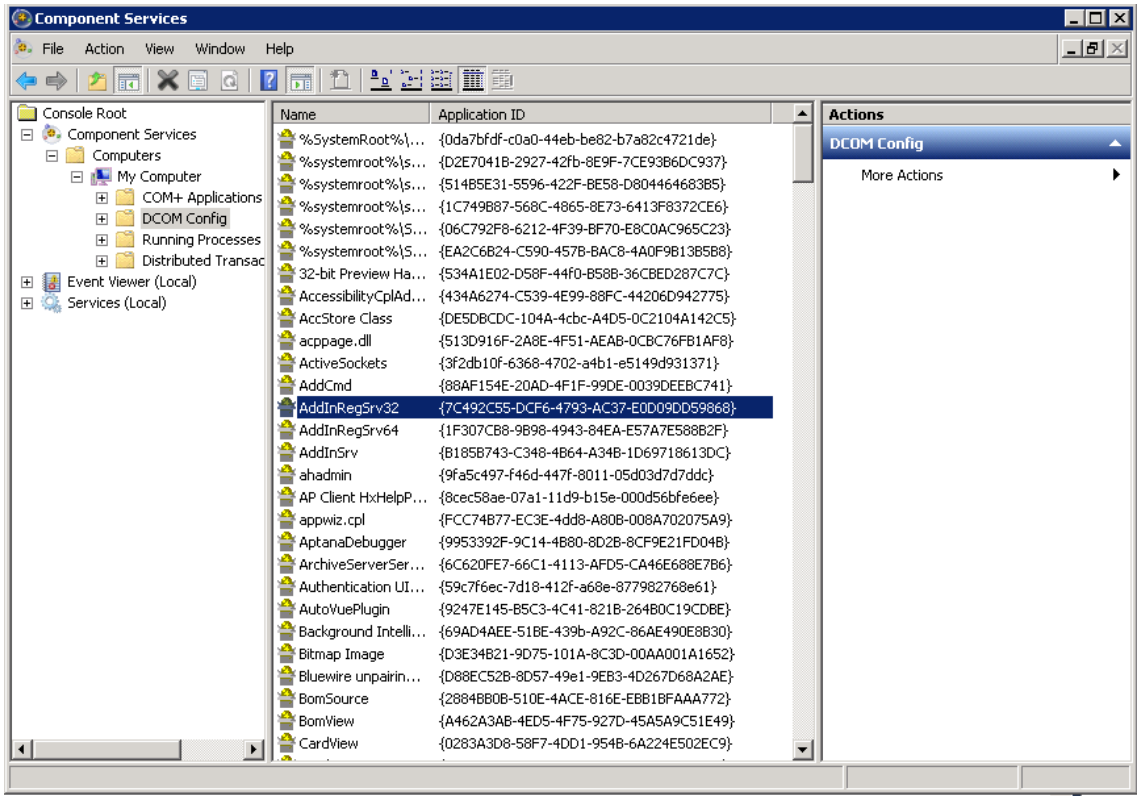

Ilustración 88: Configuración DCOM

1. Ejecutar → dcomcnfg → Component Services → Computar → My Computer → DCOM Config
2. Poner en vista de detalles y buscar las aplicaciones que han dado error a partir del identificador obtenido en el visor de eventos. En este caso, las aplicaciones son *AddInRegSrv32* y *AddInRegSrv64*.
3. Clic derecho en la aplicación → Propiedades → Security
4. Editar la opción *Launch & Activation Permission* → Añadir usuario "IIS APPPOOL\pdmwervice"
5. Darle permisos de *Local Launch* y *Local Activation*
6. Reiniciar

5.4.4 TRANSMISIÓN Y SERIALIZACIÓN DE DATOS

En esta categoría se habla de los problemas que han surgido en la administración de los datos que se transmiten entre los distintos elementos de la arquitectura, generalmente datos de gran tamaño como son los archivos.

A la hora de enviar información sobre la estructura de carpetas del almacén, las tarjetas etc., no es necesario preocuparse de la cantidad de información enviada a través de la red. Sin embargo, esto no es así cuando entra en juego la transmisión de archivos superiores a 1MB. El primer problema a tener en cuenta es la limitación que tienen los servicios para enviar y recibir información por la red, hay que configurar los servicios y el proxy de forma que sean capaces de transmitir gran cantidad de datos. La configuración de los servicios se hace mediante un archivo *XML* y es complicado su no se tiene experiencia en ello. Se deben configurar los "endpoint" de tal manera que aumenten el límite de datos transferibles.

```
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="BasicHttpBinding_IPdmService" closeTimeout="00:01:00"
        openTimeout="00:01:00" receiveTimeout="00:10:00" sendTimeout="00:01:00"
        allowCookies="false" bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
        maxBufferSize="2147483647" maxBufferPoolSize="2147483647" maxReceivedMessageSize="2147483647"
        messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
        useDefaultWebProxy="true">
        <readerQuotas maxDepth="2147483647" maxStringContentLength="2147483647" maxArrayLength="2147483647"
          maxBytesPerRead="2147483647" maxNameTableCharCount="2147483647" />
      </binding>
    </basicHttpBinding>
  </bindings>
```

Ilustración 89: Configuración XML del servicio

Para ello se deben cambiar los parámetros llamados *maxReceivedMessageSize*, *maxBufferSize*, *maxJsonLength* y *maxContentStringLength* entre otros. Estos parámetros son los que indican el límite y se han establecido al valor máximo (2147483647 bytes, alrededor de 2 GB). De todas formas, más adelante por cuestiones de seguridad se ha pensado en limitar este valor para evitar ataques *DoS* (*Denial of Service*).

Dejando de lado los límites que nos pone la tecnología usada, hay que prestar especial atención a la cantidad de información que se puede transmitir. En un principio se transmitían los datos en un vector de bytes serializado en *JSON*, es decir:

```
[116,101,120,116,111]
```

De esta forma, cuando el cliente recibe el objeto, lee todos los bytes y construye un archivo a partir de esa información. Se puede ver claramente que ésta no es la forma más adecuada de enviar la información. De hecho, en el ejemplo anterior queremos enviar un archivo de 5 bytes, cuando realmente estamos enviando 20 bytes, uno por cada carácter de la cadena. Hay que tener en cuenta que estamos representando cada byte con 3 caracteres y también estamos transmitiendo las comas.

Por esta razón, se ha decidido codificar el contenido del archivo en Base64, que representa la información solamente en caracteres imprimibles (*ASCII*). Al codificar la información del ejemplo anterior, obtendríamos este resultado:

```
dGV4dG8=
```

De ésta forma vamos a transmitir 8 bytes, una cifra muy inferior a los 20 bytes transmitidos antes. Además cuanto mayor sea el tamaño del archivo, mayor será la diferencia. Esta técnica es muy usada en la Web, y reduce considerablemente el tiempo de descarga de la información.

5.4.5 EPDM API

A pesar de que la *API* de *EPDM* se mantenga en constante desarrollo, las antiguas versiones pueden tener errores sin arreglar. De hecho, a la hora de implementar ciertas funcionalidades se han encontrado errores o limitaciones que se han tenido que sortear mediante el uso de otras técnicas de programación.

Entre ellas cabe destacar el problema a la hora de dibujar las tarjetas de los archivos del almacén. La API ofrece la posibilidad de obtener la información de todos los controles (*input*) que forman la tarjeta. Dicha información contiene una propiedad que determina si una caja de texto (*textbox*) es multilínea o no, pero desgraciadamente contiene un error que devuelve siempre el valor “false”, es decir, que una caja de texto nunca es multilínea. De todas formas, se ha implementado manualmente esta

funcionalidad de forma que si la altura de la caja de texto excede del tamaño estándar, lo transforme en una caja multilínea.

En otro caso, se ha observado que hay una limitación a la hora de tratar con las tarjetas de búsqueda. La API ofrece la posibilidad de obtener la tarjeta de búsqueda a partir de su identificador o nombre. El problema está en que no ofrece ninguna función que obtenga las tarjetas existentes, algo que puede ser embarazoso para el usuario ya que le obliga a conocer de antemano las tarjetas guardadas en la base de datos.

Para solucionar este problema se ha tenido implementar un servicio que accede a la base de datos directamente, algo peligroso ya que no existe documentación alguna sobre la implementación de *EPDM*. Por esta razón, se ha realizado un pequeño estudio de las tablas de la base de datos hasta dar con la solución. El hecho de ser una función de solamente obtener información, evita cualquier peligro de alterar la base de datos, ya que en ningún momento se modificarán ni insertarán datos.

Gracias a la siguiente consulta SELECT se han conseguido las tarjetas de búsqueda alojadas en la base de datos.

```
select * from dbo.Cards where CardType=3
```

La tabla “Cards” aloja la información de todas los tipos de tarjetas y mediante la propiedad “CardType” se filtran solamente las de tipo de búsqueda (tipo 3).

Relacionado con las tarjetas de datos, también existe un problema la funcionalidad que ofrecen los botones en las tarjetas de PDM. La API no devuelve correctamente el comando asociado al botón de la tarjeta, por lo que, por la misma razón, se ha tenido que consultar directamente la base de datos e interpretar el resultado para implementar esta funcionalidad.

Entre las acciones que puede ejecutar el clic de estos botones, se encuentra la posibilidad de seleccionar una carpeta del almacén donde se quiera realizar la búsqueda. Ésta es la consulta que se ha utilizado para conseguir el comando asociado a un botón concreto:

```
select ButtonCmd from dbo.CardControls where ControlId=ident
```

Donde “ident” es el número identificador del botón cuyo comando queremos obtener. Gracias a esto, es posible detectar la acción que debe ejecutar cada botón.

5.4.6 NAVEGADORES

En el momento de probar las funcionalidades implementadas mediante el uso de distintos navegadores, se ha detectado que es necesario activar el *plugin* de Java para Firefox y evitar problemas a la hora de acceder a la web. Para ello, simplemente se debe acceder a la lista de complementos del navegador, y seleccionar la opción “Activar siempre” sobre el *plugin* de Java.

6. PRUEBAS

En el presente capítulo se exponen las pruebas que se han realizado para corroborar el buen funcionamiento de la nueva herramienta. Para ello, se han diseñado y ejecutado pruebas en cada plataforma para las que se ha desarrollado el producto.

En la ejecución de las pruebas se muestra una tabla de resultados para cada plataforma. Tras obtener estos resultados, se dio paso a arreglar las funcionalidades que daban errores.

6.1 DISEÑO DE PRUEBAS

La prueba exhaustiva del software es impracticable, por lo que se han realizado pruebas sobre las funcionalidades básicas de *PdmWeb*:

- **Login:** Funcionalidad que realiza la identificación en el sistema. Requiere el almacén, el usuario y la contraseña como parámetro. En caso de proporcionar datos válidos, debería acceder a la página principal. De lo contrario, debe saltar un cuadro de texto indicando que no es posible realizar la identificación.
- **Preview:** Permite visualizar el archivo en una vista previa antes de descargarlo. Solo se ha implementado para los archivos de *SolidWorks* e imágenes, por lo que los demás archivos no deberían visualizarse. Para hacerlo, se seleccionará un único archivo, ya que en caso de seleccionar más de uno no se mostrará la vista previa.
- **Abrir:** Permite abrir el documento alojado en la máquina local del usuario con el programa asociado a ese tipo de archivos. Ésta acción depende del sistema operativo, por lo que en caso de no encontrar un programa asociado, no debería realizar ninguna acción y no mostrará mensaje de error.
- **Eliminar:** Elimina el archivo o carpeta seleccionado. En caso de la carpeta, elimina todo su contenido de forma recursiva. Debe mostrar un cuadro de confirmación antes. Además de eliminarlo del almacén, también deberá eliminar los elementos seleccionados del sistema local.
- **Obtener:** Descarga la última versión del documento seleccionado. Sobrescribe cualquier cambio realizado sobre el archivo local, esté traído o no.
- **Obtener versión:** Descarga la versión seleccionada del documento, para ello, muestra un cuadro con todas sus versiones guardadas. Si selecciona una versión, el programa sobrescribe el archivo local con la descargada.
- **Comparación de versión:** Esta funcionalidad es la que permite detectar la versión del archivo guardado en la máquina local a partir de su fecha de modificación. Cada vez que se cargue el contenido de las carpetas, se debería mostrar la versión local de cada documento.
- **Deshacer traer:** Esta acción solo se puede realizar sobre documentos traídos, por lo que para los demás documentos estará desactivado. Desbloqueará el archivo traído.
- **Traer:** Obtiene la última versión del archivo seleccionado y a continuación lo bloquea para que nadie más pueda registrar nuevos cambios. Solo se podrá realizar sobre archivos que no estén traídos por otros usuarios, de lo contrario no se mostrará la opción.
- **Registrar:** El documento seleccionado se sube al servidor y se registra una nueva versión quedando desbloqueado. En caso de haber un error en la subida, debe abortar el registro.
- **Nueva carpeta:** Creará una nueva carpeta con el nombre introducido por el usuario gracias a una caja te de texto que se muestra. Si la carpeta ya existe, lanzará un error informando de lo sucedido al usuario.

- **Cambio de estado:** Cambia de estado los documentos seleccionados, no realizará el cambio si no es posible. No permitirá el cambio de estado si el archivo está traído.
- **Ver variables:** Visualiza el valor de las variables asociadas al documento en la tarjeta de datos.
- **Guardar variables:** Guarda los cambios realizados en las variables del archivo. Para poder editar los valores, el archivo debe estar traído.
- **Búsqueda:** Se muestra una lista de las tarjetas de búsquedas disponibles. El usuario selecciona y rellena la deseada para proceder a realizar la búsqueda con los parámetros proporcionados. Si no se proporcionan valores adecuados, la búsqueda no mostrará archivos encontrados.
- **Filtro:** Filtra el contenido de la carpeta a partir de la cadena de texto introducida. Si ningún archivo coincide con el patrón, no se mostrará nada. Si se usa una cadena de texto vacía como patrón, se mostrará todo el contenido tal y como estaba.
- **Tarjeta:** Muestra la tarjeta de datos mediante formularios web. Si hay algún error y no es posible mostrar la tarjeta por estar defectuosa, avisará con un mensaje de error al usuario y se mostrará la parte de la tarjeta que es posible dibujar. Al igual que la vista previa, deberá seleccionarse un único archivo para poder ver la tarjeta, de lo contrario no se mostrará nada.
- **Lista de materiales:** Muestra la lista de materiales de la pieza seleccionada. Si se trata de otro documento, la lista de materiales mostrará el propio archivo.
- **Contiene:** Si el archivo contiene referencias, mostrará el árbol de referencias de forma recursiva, siendo el mismo archivo la raíz del árbol. Si no contiene referencias, la lista solo mostrará el propio archivo.
- **Dónde se utiliza:** Al igual que el anterior, si el archivo es referenciado por otro, mostrará una estructura de árbol que indica qué archivos lo referencian. Si no contiene referencias, la lista solo mostrará el propio archivo.
- **Ayuda:** Muestra un cuadro de información sobre la aplicación.
- **Mensaje Java no disponible:** En caso de no tener Java instalado en el equipo, deberá mostrar un mensaje diciendo que es necesario el uso de Java para realizar algunas de las acciones.

6.2 EJECUCIÓN DE PRUEBAS

		Internet Explorer	Firefox	Chrome	IOS	Android
Login		✓	✓	✓	✓	✓
Preview	docx	✗	✗	✗	✗	✗
	xlsx	✗	✗	✗	✗	✗
	slddrw	✓	✓	✓	✓	✓
	sldprt	✓	✓	✓	✓	✓
	sldasm	✓	✓	✓	✓	✓
	edrw	✗	✗	✗	✗	✗
	jpg	✓	✓	✓	✓	✓
	bmp	✓	✓	✓	✓	✓
	txt	✗	✗	✗	✗	✗
	msg	✗	✗	✗	✗	✗
Abrir		✓	✓	✓ (menos los bmp)	(no disponible)	(no disponible)
Eliminar		✓ (falla a veces con carpetas)	✓ (falla a veces con carpetas)	✓ (falla a veces con carpetas)	(no disponible)	(no disponible)
Obtener		✓	✓	✓	✓ (no lo guarda con un nombre adecuado)	✓ (no permite abrir directamente)
Obtener versión		✓	✓	✓	(no disponible)	(no disponible)
Comparación de versión		✗	✓	✓	(no disponible)	(no disponible)
Deshacer traer		✗	✓ (presenta más documentos de los seleccionados)	✓	(no disponible)	(no disponible)
Traer		✗	✓ (falta la selección por defecto)	✓ (falta la selección por defecto)	(no disponible)	(no disponible)
Registrar		✓	✓	✓	(no disponible)	(no disponible)
Añadir	docx	✗	✗	✓	(no disponible)	(no disponible)
	xlsx	✗	✗	✓	(no disponible)	(no disponible)
	slddrw	✗	✓	✓	(no disponible)	(no disponible)
	sldprt	✗	✓	✓	(no disponible)	(no disponible)
	sldasm	✗	✓	✓	(no disponible)	(no disponible)
	edrw	✗	✓	✓	(no disponible)	(no disponible)
	jpg	✗	✓	✓	(no disponible)	(no disponible)

	bmp	✘	✓	✓	(no disponible)	(no disponible)
	txt	✘	✓	✓	(no disponible)	(no disponible)
	msg	✘	✓	✓	(no disponible)	(no disponible)
Nueva carpeta		✓	✓	✓	(no disponible)	(no disponible)
Cambio de estado		✓	✓	✓		
Ver variables		✓	✓	✓	(no disponible)	(no disponible)
Guardar variables		✓	✓	✓	(no disponible)	(no disponible)
Búsqueda		✓ (faltan valores por defecto)	✓ (faltan valores por defecto)	✓ (faltan valores por defecto)	(no disponible)	(no disponible)
Filtro		✓	✓	✓	✓	✓
Tarjeta	Imágenes	✓ (no siempre)	✓ (no siempre)	✓ (no siempre)	(no disponible)	(no disponible)
	Textbox	✓	✓	✓	(no disponible)	(no disponible)
	Textbox multilínea	✓	✓	✓	(no disponible)	(no disponible)
Lista de materiales		✓	✓	✓	(no disponible)	(no disponible)
Contiene		✓	✓	✓	(no disponible)	(no disponible)
Dónde se utiliza		✓	✓	✓	(no disponible)	(no disponible)
Ayuda		✘	✘	✘	✓	✓
Mensaje java no disponible		✘	✘	✘	(no disponible)	(no disponible)

Tabla 25: Pruebas

7. CONCLUSIONES Y LÍNEAS FUTURAS

En este capítulo se analizan todas las tareas que han sido necesarias para completar los objetivos presentados en el capítulo 2. Se muestra una comparativa entre las horas estimadas y las horas reales invertidas que terminan con conclusiones generales y personales acerca de este proyecto.

Se ha conseguido una primera versión de este nuevo programa, pero su desarrollo no termina con este Proyecto de Fin de Carrera. Por esta razón se incluyen sugerencias sobre las mejoras y nuevas funcionalidades que posiblemente se implementarán en versiones posteriores de la aplicación.

7.1 OBJETIVOS ALCANZADOS

A lo largo de la memoria se ha mostrado el análisis y el diseño de los casos de uso necesarios para cumplir con los requisitos establecidos por el cliente, que en este caso ha sido la misma empresa. Además, se han documentado los problemas que se han tenido a la hora de desarrollar la aplicación y las decisiones que se han tenido que tomar para solucionar dichos problemas. Gracias al seguimiento continuo y a las pruebas realizadas por el grupo de *Ibermática*, se han logrado arreglar numerosos errores de *PdmWeb*.

Así, se ha logrado una nueva versión de *SolidWorks Enterprise EPDM* personalizada que es capaz de ofrecer la mayoría de sus funcionalidades e incluso de extender su potencial. A continuación se analizan los objetivos del proyecto uno a uno:

- **Objetivo 1: Investigación de las nuevas tecnologías y estándares.**
  Se han investigado diversas tecnologías que podrían ser adecuadas para desarrollar aplicaciones sobre la plataforma web. Desde un principio, la idea principal era usar la tecnología *ASP.NET* para crear las páginas web y acceder a la *API* de *EPDM* utilizando el lenguaje *C#*. Pero tras analizar otras tecnologías y estándares más recientes, se decidió hacer uso de HTML5 y JavaScript para programar páginas dinámicas en la parte del cliente. Cabe destacar el uso de *Java*, ya que se vio la necesidad de tener que acceder al sistema de archivos local y esta tecnología ofrece acceder a los recursos de la máquina local tras obtener el consentimiento del usuario.
- **Objetivo 2: Implementación de un servicio web que ofrezca las funcionalidades de EPDM.**
  Para ofrecer las funcionalidades de *EPDM*, ha sido necesario usar la API oficial. Para poder trabajar sobre los almacenes del servidor, ha sido imprescindible implementar servicios web en la parte del servidor mediante el lenguaje *C#.* Se han configurado los servicios de forma que cualquier interfaz que soporte el protocolo *HTTP* pueda acceder a ellos. De esta forma, la interfaz web es capaz de realizar peticiones mediante la técnica *AJAX* y ofrecer las mismas funcionalidades que la *API*.
  Además, la empresa ha remarcado la importancia de la seguridad en la nueva herramienta, y por esta razón, se incluyó un intermediario que actúa de proxy y redirige las peticiones al servidor principal. De esta forma, se logra una arquitectura de varios niveles y se restringe el acceso directo a los datos desde fuera.
- **Objetivo 3: Desarrollo de una interfaz web que actúe sobre los servicios web.**
  Gracias a la tecnología web, concretamente a HTML5, JavaScript y JQuery, se ha logrado establecer la comunicación con los servicios y plasmar el contenido de forma que se pueda ver desde los navegadores web. La compatibilidad entre los navegadores como Internet Explorer y Mozilla Firefox entre otros, ha sido un inconveniente a la hora de desarrollar la

aplicación web, ya que no todos ofrecen las mismas capacidades. En este aspecto, es Google Chrome sin duda el que mejor se adapta a los nuevos estándares web.

- **Objetivo 4: Desarrollo de una aplicación ligera para dispositivos móviles en HTML5 que explote los servicios.**
  Tras desarrollar la aplicación web principal, se ha desarrollado una aplicación específica para dispositivos táctiles y ofrecer un cliente más ligero a los usuarios de estas plataformas. La ventaja está en que a pesar de ser otra aplicación, consume los mismos servicios que el cliente web pesado, de modo que se ha ahorrado cantidad de trabajo. También cabe destacar que para crear esta interfaz, se ha utilizado el *JQueryMobile*, un framework para crear aplicaciones web para dispositivos móviles sin preocuparse demasiado por el aspecto.

7.2 COMPARATIVAS ENTRE LA ESTIMACIÓN Y LAS HORAS INVERTIDAS

En esta sección se muestra una comparativa entre las horas totales estimadas en la planificación inicial y las horas reales invertidas en el proyecto. En la primera tabla se puede apreciar exactamente la diferencia entre las horas estimadas y las horas invertidas. Cabe destacar que en algunas de las tareas se han invertido más horas de las estimadas, de hecho, el efecto del cambio continuo de los requisitos a lo largo del proyecto se ha visto reflejado en la inversión de horas extra en dichas tareas.

Fases	Tareas	Horas estimadas	Horas invertidas	Desvío
Gestión	Reuniones	15:00	9:30	36,67%
	Planificación	6:00	4:45	20,83%
	Seguimiento	25:00	6:45	73,00%
	Copias de seguridad	10:00	5:30	45,00%
Formación	Visual Studio	40:00	67:55	69,79%
	SolidWorks Enterprise PDM	20:00	30:30	52,50%
	Asp.net	16:00	4:30	71,88%
Instalación	Entorno de desarrollo	8:00	7:15	9,37%
Captura de requisitos	Casos de uso	8:00	6:30	18,75%
	Prototipo de la interfaz	8:00	11:00	37,50%
Diseño	Diseño de la lógica de negocio	24:00	31:15	30,21%
	Diagrama de clases	16:00	4:30	71,88%
Implementación	Definir la arquitectura	4:00	5:45	43,75%
	Implementación de la interfaz	80:00	87:00	8,75%
	Implementación de la lógica de negocio	160:00	200:00	25,00%
	Integración	8:00	13:30	68,75%
Pruebas	Diseño de pruebas	8:00	1:00	87,50%
	Ejecución de pruebas	24:00	15:00	37,50%
Documentación	Desarrollo de la memoria	32:00	60:45	89,84%
	Manual de usuario	24:00	0:00	100,00%
	Elaboración de la presentación	8:00	0:00	100,00%
TOTAL		544:00	572:55	5,32%

Tabla 26: Horas estimadas vs horas invertidas

Se puede ver que la diferencia total entre las horas estimadas y las horas invertidas no es tan grande. Pero si se analiza cada tarea independientemente, el desvío llega a ser significante. El desvío de color verde significa que se ha ahorrado tiempo, por otra parte, el color rojo significa que se ha tenido que invertir más tiempo en dicha tarea.

El haber estimado demasiado tiempo en algunas tareas, ha permitido invertir más horas en las tareas que lo requerían, de modo que esta compensación ha sido posible gracias a una planificación flexible como ya se especifica en el documento de objetivos.

En la formación de Visual Studio y SolidWorks EPDM se han invertido más horas que en la de ASP.NET. De hecho, finalmente se decidió no usar esta tecnología, por lo que no se siguió realizando la formación.

Otra de las diferencias que más destaca es la de la fase de implementación, que ha superado con creces toda estimación planificada. El hecho del cambio continuo de los requisitos y el tener que añadir nuevas funcionalidades, ha repercutido directamente en la implementación, sobre todo, de la lógica de negocio. Estos cambios además, también han tenido que ver en el diseño y su representación en la memoria, por lo que también han requerido esfuerzo extra para llevarlos a cabo.

Cabe destacar el ahorro del tiempo para crear en manual de usuario. Al tratarse de una versión temprana, se ha decidido no desarrollar el manual y dejarlo fuera del Proyecto de Fin de Carrera. Por otra parte, la elaboración de la presentación se pretende hacer tras la finalización de la memoria, por lo que es imposible incluir las horas invertidas.

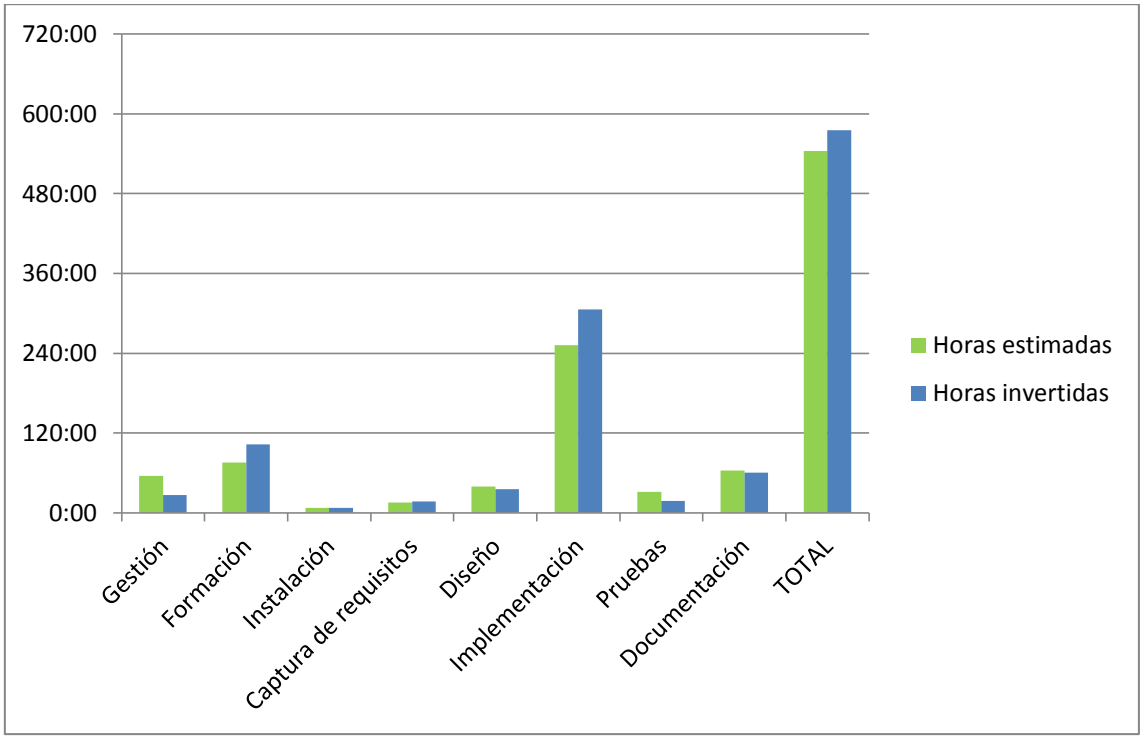

Ilustración 90: Gráfico de horas invertidas por fases

Al igual que en la tabla anterior se podía ver el desvío en porcentajes, en el gráfico de barras se aprecia la diferencia de cada fase en horas. Aunque en porcentajes estas diferencias sean mayores, la diferencia de horas es cercana a la media en la mayoría de las tareas. Este detalle es el que ha permitido compensar las horas estimadas e invertidas entre las tareas, y así, minimizar la diferencia total.

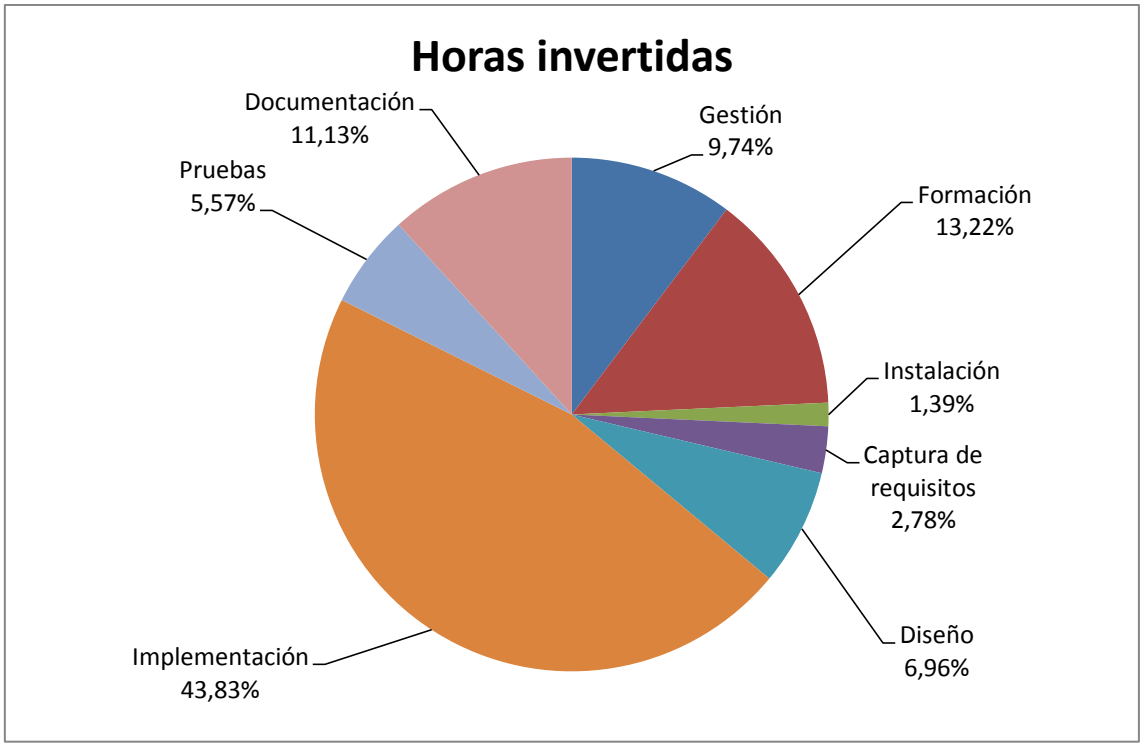

Ilustración 91: Peso de las horas invertidas de cada fase

Por último, en el gráfico circular se puede ver el peso que ha tenido cada fase a lo largo del Proyecto de Fin de Carrera. Los resultados concuerdan totalmente con lo que supone actualmente trabajar en una empresa, es decir, son la formación y la implementación las fases más importantes y con más carga de trabajo. De todas formas, la necesidad de documentar un proyecto de tal envergadura para el fin de carrera, ha requerido invertir horas para detallar todo lo que se ha realizado.

7.2.1 GANTT REAL

PDM Web
Ibermática

Líder de proyecto: Julen
Fecha de inicio: 14/10/2013 lunes

ID	Tareas	Líder	Inicio	Fin	Duración (Días)	% Completado	Días de trabajo	Días completado	Días restantes
1	**Gestión**	Julen	23/10/2013	14/04/2014	174	100%	124	174	0
1.1	Reuniones		30/10/2013	01/04/2014	154	100%	110	154	0
1.2	Planificación		28/10/2013	20/12/2013	54	100%	40	54	0
1.3	Seguimiento		23/10/2013	14/04/2014	174	100%	124	174	0
1.4	Copias de seguridad		12/11/2013	14/04/2014	154	100%	110	154	0
2	**Formación**	Julen	14/10/2013	14/11/2013	32	100%	24	32	0
2.1	Visual Studio		14/10/2013	30/10/2013	17	100%	13	17	0
2.2	SolidWorks EPDM		14/10/2013	04/11/2013	22	100%	16	22	0
2.3	Asp.net		24/10/2013	14/11/2013	22	100%	16	22	0
3	**Instalación**	Julen	23/10/2013	21/11/2013	30	100%	22	30	0
3.1	Entorno de desarrollo		23/10/2013	21/11/2013	30	100%	22	30	0
4	**Captura requisitos**	Julen	06/11/2013	03/02/2014	90	100%	64	90	0
4.1	Casos de uso		07/11/2013	31/12/2013	55	100%	39	55	0
4.2	Prototipo de interfaz		06/11/2013	03/02/2014	90	100%	64	90	0
5	**Diseño**	Julen	15/11/2013	07/03/2014	113	100%	81	113	0
5.1	Lógica de negocio		15/11/2013	21/02/2014	99	100%	71	99	0
5.2	Diagrama de clases		15/11/2013	07/03/2014	113	100%	81	113	0
6	**Implementación**	Julen	05/11/2013	17/03/2014	133	100%	95	133	0
6.1	Definir arquitectura		05/11/2013	12/12/2013	38	100%	28	38	0
6.2	Interfaz		07/11/2013	04/03/2014	118	100%	84	118	0
6.3	Lógica de negocio		11/11/2013	17/03/2014	127	100%	91	127	0
6.4	Integración		11/11/2013	10/03/2014	120	100%	86	120	0
7	**Pruebas**	Julen	14/03/2014	14/04/2014	32	100%	22	32	0
7.1	Diseño		14/03/2014	31/03/2014	18	100%	0	18	0
7.2	Ejecución		14/03/2014	14/04/2014	32	100%	22	32	0
8	**Documentación**	Julen	23/10/2013	14/04/2014	174	100%	124	174	0
8.1	Memoria		23/10/2013	14/04/2014	174	100%	124	174	0
8.2	Manual de usuario				0		0	0	0
8.3	Presentación				0		0	0	0

Ilustración 92: Diagrama Gantt real

7.3 CONCLUSIÓNES

Se presentan las conclusiones generales y las personales, dejando constancia de lo que ha supuesto el desarrollo del proyecto.

7.3.1 GENERALES

Como ya se ha visto, tras la finalización del proyecto puede concluirse que se han cumplido todos los objetivos del proyecto. Se ha logrado dos interfaces web que son ofrecen la funcionalidad básica sobre almacenes *EPDM*.

Con esto, se ha dado un primer paso en lo que va a ser un proyecto de desarrollo continuo en la empresa. A pesar de las limitaciones que ofrecen las tecnologías usadas, se ha desarrollado una arquitectura que daba solución a numerosos problemas.

El cambio continuo de los requisitos del proyecto ha hecho que sea difícil mantener un producto estable durante su desarrollo, pero gracias ello, se ha conseguido una aplicación adaptada a los futuros clientes de *Ibermática* que ofrece muchas funcionalidades. Estos cambios también ha dificultado el mantenimiento de la documentación, que se ha tratado de llevar al día a lo largo del ciclo de vida del proyecto.

Finalmente, cabe destacar que esta nueva herramienta es una alternativa a las versiones oficiales de *EPDM* y ofrece otra solución más a la gestión documental. A pesar de contar con recursos limitados en el uso de las funciones de *EPDM*, hay que decir que el desarrollo ha sido un éxito y que el proyecto cuenta con un futuro prometedor. La empresa está satisfecha con el trabajo realizado y pretende continuar con el proyecto para ofrecer a sus clientes nuevas soluciones a la hora de gestionar sus documentos y proyectos mediante la nueva herramienta.

7.3.2 PERSONALES

Este proyecto ha sido una gran experiencia a nivel personal y profesional. El hecho de trabajar junto a un equipo de programadores ha superado mis expectativas de lo que es trabajar en una empresa. Realizar reuniones, proponer soluciones y tomar decisiones con el apoyo de otros miembros de la empresa ha sido de gran ayuda para ser uno más en el grupo y verse involucrado totalmente en el desarrollo del proyecto.

Además, gracias a la experiencia que he obtenido estos últimos años con la programación en C++, Java y la programación orientada a objetos en general, no he tenido problemas para adaptarme a la programación en .NET y al uso de una API que ofrece sus métodos en lenguaje C#. La experiencia en el desarrollo web me ha permitido investigar con mucha rapidez nuevas tecnologías y tomar decisiones adecuadas para cada problema al que me he enfrentado.

Por otra parte, el mantenimiento de la documentación ha sido una tarea llevada a cabo a lo largo del proyecto. El hecho de trabajar sin plazos intermedios de entrega ha ayudado a realizar una planificación más flexible. Pero los numerosos cambios en los requisitos han acarreado cambios importantes en el diseño. Por esta razón, fue inevitable aplazar el desarrollo de diversas partes de la memoria.

Por último, tengo que decir que gracias a este proyecto he dado un paso grande y he adquirido experiencia profesional que me será imprescindible en un futuro próximo.

7.4 LÍNEAS FUTURAS

Con este Proyecto de Fin de Carrera, se ha conseguido una primera versión de una aplicación que posteriormente ofrecerá nuevas funcionalidades y mejoras. De modo que para un futuro, se propone lo siguiente:

7.4.1 TRATAMIENTO DE ERRORES EN LA INTERFAZ

En la fase final del proyecto no se han podido realizar las pruebas por falta de tiempo y por la urgencia de querer obtener una versión temprana por parte de la empresa. Por esta razón, la interfaz no avisa al usuario de los errores que se han podido dar en las acciones realizadas.

Trabajar sobre un servidor que corría en modo de depuración, ha sido de gran ayuda para tener constancia de los errores que pueden arruinar el funcionamiento de la aplicación. Pero a la hora de publicarla, es necesario no dar información con detalles al usuario sobre los problemas surgidos, por seguridad.

Esto no quiere decir que deban ignorarse los errores, ya que actualmente ante algunos errores, el usuario no tiene información suficiente de lo que ha ocurrido. Por ello se propone el desarrollo de un sistema de avisos para que el usuario sea consciente de cuándo una acción se ha realizado con éxito o no.

7.4.2 TRATAMIENTO DE ERRORES EN LOS SERVICIOS

Para lograr un tratamiento de errores efectivo en la interfaz, es necesario detectar dichos errores en la parte del servidor. Por eso, es necesario que los servicios ofrecidos por el servidor sean capaces de devolver un valor que identifique el problema, para que posteriormente la interfaz sea capaz de interpretar el resultado.

Además, se recomienda revisar el código escrito en C# para captar posibles excepciones lanzadas por la *API* de *EPDM* y hacer uso de sentencias “try-catch” para evitar errores tan frecuentes como las violaciones de acceso en la memoria.

Estos errores, provocan la ejecución de un sistema de depuración llamado *Jus-In-Time Debugger* del Visual Studio que permite depurar el error a tiempo real. Esto puede traer serios problemas a la hora de publicar la herramienta, ya que en el momento de la depuración, el servidor queda totalmente bloqueado dejando al usuario sin obtener ninguna respuesta.

7.4.3 OPTIMIZAR EL TRÁFICO DE RED

El uso de los servicios hace que constantemente se esté enviando información a través de la red, en ambas direcciones desde el cliente al servidor. Teniendo en cuenta que en un futuro podría haber varios usuarios trabajando sobre el sistema, es imprescindible analizar el impacto que podría suponer el exceso de tráfico.

Por esta razón, para reducir el tiempo de espera se propone optimizar las peticiones que se hacen al servidor. De modo que solo se envíe la información necesaria en cada momento o no se realicen peticiones innecesarias al servidor si la información está guardada localmente. Para esto, sería eficiente administrar una especie de caché que guarde los datos obtenidos en peticiones anteriores y solo los cambie al realizar ciertas acciones.

7.4.4 SEGURIDAD

Los servicios web programados no son persistentes, es decir, los datos que se guardan se crean y se destruyen con cada llamada a sus funciones. Esto requiere la necesidad de enviar los datos del usuario en cada petición y así poder identificar al usuario antes de realizar cualquier acción.

Esto puede suponer un problema para clientes que requieren cierta seguridad, y se propone, por ello, la implantación del sistema mediante el protocolo HTTPS para que toda la información enviada por la red quede fuera del alcance de posibles atacantes.

Además, en este tipo de gestores documentales suele ser necesario llevar un control de todos los cambios y acciones que realizan los usuarios. Para esto, se propone crear un sistema simple de monitorización que registre toda esta información en un documento o en una base de datos. Por ejemplo, para cada acción realizada se podría registrar; el almacén, el usuario, la dirección IP, nombre de la acción etc.

De esta forma, se podrían monitorizar los errores generados por la aplicación y detectar otro tipo de acciones sospechosas.

7.4.5 TARJETAS DE DATOS PERSONALIZADAS

Las tarjetas de datos actuales de EPDM presentan ciertas limitaciones de acuerdo a las necesidades de los clientes frente a esta herramienta. Para saciar estas necesidades la empresa cuenta con un proyecto para crear tarjetas personalizadas que interactúan mejor con el usuario y presentan más información.

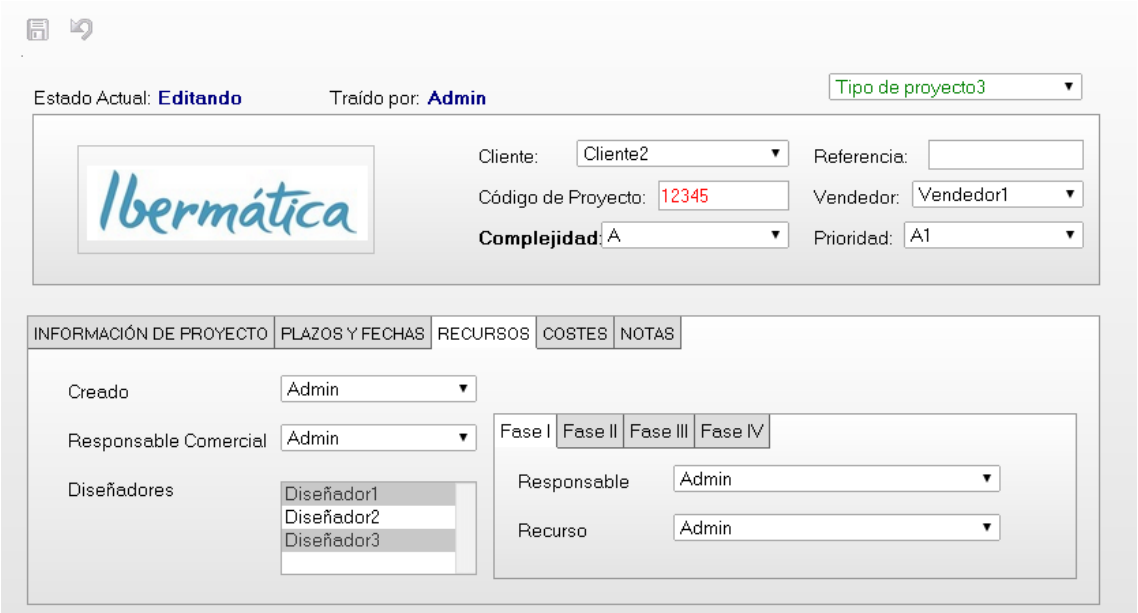

Ilustración 93: Tarjeta personalizada

La idea es plasmar estas tarjetas personalizadas en la web, y para ello será necesario hacer algún cambio en la arquitectura para que *PdmWeb* pueda interactuar con las mismas librerías que usa el editor de tarjetas de la empresa. De todas formas, en un futuro lejano, se pretende incluso migrar el editor de tarjetas a la plataforma web.

7.4.6 VISTA PREVIA DE DOCUMENTOS OFFICE

La implementación de la vista previa de documentos ha sido algo que se ha ido investigando a lo largo del proyecto con poco éxito. Aun así, se ha logrado visualizar los documentos más importantes desde el punto de vista de los requisitos del cliente: las piezas, los ensamblajes y planos de *SolidWorks*.

De todos modos, se considera imprescindible ofrecer una vista previa de los documentos Office y documentos PDF entre otros. Por ello, se propone indagar más en el tema y tratar de averiguar una forma eficiente de lograr este objetivo.

Sería interesante analizar la posibilidad de mantener una base de datos que aloje una vista previa en formato de imagen para cada documento. Cada registro de un nuevo archivo o archivo existente, crearía una nueva imagen y la guardaría como imagen de una nueva versión. Esta idea podría complicar más la arquitectura del sistema, por eso se propone analizar la eficiencia de esta solución.

7.4.7 CONFIGURACIÓN DEL CLIENTE

La empresa ofrece los servicios a un amplio abanico de clientes, y teniendo en cuenta que sus necesidades difieren, es imprescindible hacer la aplicación configurable para la parte del cliente. De modo que cada uno de ellos pueda personalizarlo como más le convenga.

Entre las opciones se podrían implementar las siguientes: la configuración de la ruta del almacén local, posibilidad de cambiar los logotipos de la empresa, cambiar los colores y el tema de la interfaz, configuración de las columnas, configuración de la monitorización del sistema etc.

8. REFERENCIAS

CanIUse. (s.f.). *Can I use...* Obtenido de http://caniuse.com/

CodeProject. (1 de 11 de 2010). *A Simple Sample WCF Service.* Obtenido de http://www.codeproject.com/Articles/123067/A-Simple-Sample-WCF-Service

CodeProject. (25 de 11 de 2010). *Consuming WCF REST Services Using jQuery AJAX Calls.* Obtenido de http://www.codeproject.com/Articles/128478/Consuming-WCF-REST-Services-Using-jQuery-AJAX-Call

Fundación Wikimedia Inc. (s.f.). *Wikipedia.* Obtenido de http://es.wikipedia.org/

GeeksWithBlogs. (11 de 12 de 2007). *WCF maxStringContentLength.* Obtenido de http://geekswithblogs.net/niemguy/archive/2007/12/11/wcf-maxstringcontentlength-maxbuffersize-and-maxreceivedmessagesize.aspx

Ibermática. (s.f.). *Ibermática*. Obtenido de Ibermática: http://www.ibermatica.com/

Ibermática. (s.f.). *SolidWorks.* Obtenido de http://soluciones.ibermatica.com/vdoc/resource/filecenter/document/042-000023-000/0.1%20SolidWorks%202012

Microsoft. (s.f.). *Microsoft Developer Network*. Obtenido de http://msdn.microsoft.com/

Oracle. (s.f.). *Java*. Obtenido de http://www.java.com/es/

SolidWorks Corp. (s.f.). *SolidWorks API Help.* Obtenido de http://help.solidworks.com/2014/English/api/SWHelp_List.html?id=9f5eb504b928475e99c82f369d98fe45#Pg0

SolidWorks Corp. (s.f.). *SolidWorks Enterprise PDM*. Obtenido de http://www.solidworks.es/sw/products/product-data-management/solidworks-enterprise-pdm.htm

Stack Exchange Inc. (s.f.). *Stackoverflow*. Obtenido de http://stackoverflow.com/

The JQuery Foundation. (s.f.). *JQuery*. Obtenido de http://jquery.com/

W3C. (s.f.). *W3C*. Obtenido de http://www.w3.org/

ANEXOS

ANEXO I - ACTAS DE REUNIÓN

A continuación se muestran las actas recogidas para cada una de las reuniones que se han realizado con la directora del proyecto. También se incluyen las actas de las reuniones llevadas a cabo en la empresa con la codirectora del proyecto y un equipo reducido de Ibermática; generalmente formado por informáticos, mecánicos y/o un comercial.

Por mantener el anonimato de los asistentes a la reunión, se han omitido sus nombres en las actas de las reuniones realizadas en la empresa.

1. Acta de reunión

Fecha: 30/10/2013
Hora de inicio: 9:30
Hora fin: 10:00
Lugar: Ibermática (Zuatzu)

Asistentes a la reunión:

Asistente	Rol	Asistencia	Razón
Julen Salgado	Programador	Si	-

Orden del día

- Definir el alcance del proyecto

Decisiones tomadas

- Se implementará una versión web de EPDM usando tecnologías como HTML5, ASP, Web services…

Tareas para la próxima reunión

- Investigar e implementar una aproximación de la identificación de varios usuarios simultáneos de EPDM con ASP.
- Pensar en una solución y una posible arquitectura sobre la que empezar el desarrollo.

Próxima reunión

-

2. Acta de reunión

Fecha: 12/11/2013
Hora de inicio: 17:00
Hora fin: 17:30
Lugar: Facultad de informática

Asistentes a la reunión:

Asistente	Rol	Asistencia	Razón
Julen Salgado	Programador	Si	-
Ana Sánchez	Directora	Si	-

Orden del día

- Definir la estructura de la memoria del proyecto

Decisiones tomadas

- Se implementará una versión web de EPDM usando tecnologías como HTML5, ASP, Web

services…

Tareas para la próxima reunión

- Seguir avanzando la memoria con los requisitos definidos hasta el momento.

Próxima reunión

Martes 17 de diciembre del 2013, a las 17:00

3. Acta de reunión

Fecha: 02/12/2013
Hora de inicio: 9:00
Hora fin: 10:30
Lugar: Ibermática (Zuatzu)

Asistentes a la reunión:

Asistente	Rol	Asistencia	Razón
Julen Salgado	Programador	Si	-

Orden del día

- Presentar soluciones a la transferencia (descarga) de archivos y proponer tecnologías para lograrlo.

Decisiones tomadas

- Se creará una estructura de archivos local para las descargas, lo cual requiere permisos de usuario para acceder al sistema de archivos. Por esta razón se hará uso de Java que implementará funciones que serán accesibles desde la página HTML.

Tareas para la próxima reunión

- Implementar la solución propuesta mediante el uso de Java.

Próxima reunión

-

4. Acta de reunión

Fecha: 17/12/2013
Hora de inicio: 17:00
Hora fin: 17:30
Lugar: Facultad de informática

Asistentes a la reunión:

Asistente	Rol	Asistencia	Razón
Julen Salgado	Programador	Si	-
Ana Sánchez	Directora	Si	-

Orden del día

- Revisar los la captura de requisitos y los casos de uso del proyecto.

Decisiones tomadas

- Los casos de uso no se plasmarán todavía en la memoria debido al continuo cambio de decisiones en los requisitos.

Tareas para la próxima reunión

- Ir avanzando en la memoria completando la introducción y los antecedentes del proyecto.

Próxima reunión
-

5. Acta de reunión

Fecha: 31/12/2013
Hora de inicio: 10:00
Hora fin: 11:15
Lugar: Ibermatica (Miramón)

Asistentes a la reunión:

Asistente	Rol	Asistencia	Razón
Julen Salgado	Programador	Si	-

Orden del día

- Analizar los requisitos básicos de la aplicación con la ayuda de un comercial para ajustarlos a las necesidades del mercado.

Decisiones tomadas

- Se ve la necesidad de mostrar una vista previa para que el usuario tenga una idea del archivo que va a visualizar.
- Se exige que se pueda obtener la lista de materiales.
- La subida de los archivos debería incluir la operación de registrarlo también, así hacerlo en un solo paso.

Tareas para la próxima reunión

- Continuar desarrollando las funcionalidades básicas hasta la próxima reunión.
- Empezar a investigar las tecnologías o librerías con las que se podría generar la vista previa de los archivos.

Próxima reunión
-

6. Acta de reunión

Fecha: 07/01/2014
Hora de inicio: 9:00
Hora fin: 10:00
Lugar: Ibermatica (Miramón)

Asistentes a la reunión:

Asistente	Rol	Asistencia	Razón
Julen Salgado	Programador	Si	-

Orden del día

- Analizar el estado actual del proyecto.
- Redefinir nuevos requisitos a partir de las limitaciones surgidas hasta el momento.
- Decidir cómo se implementará la versión para dispositivos móviles.

Decisiones tomadas

- La información sobre la versión de los ficheros se obtendrá leyendo el sistema de archivos local y utilizando las fechas de modificación.

- Se implementará la vista previa de los archivos con imágenes.
- Se desarrollará otra aplicación web paralela con funciones limitadas y con una resolución reducida que será para acceder desde los dispositivos móviles.

Tareas para la próxima reunión

- Corregir algunas funciones ya implementadas.
- Implementar el versionado con archivos locales.
- Mostrar dependencias de los ensamblajes y la lista de materiales.
- Desarrollar la funcionalidad que permite realizar búsquedas.
- Implementar la vista previa.
- Mostrar el flujo de trabajo del archivo que se ha seleccionado.
- Hacer la aplicación configurable añadiendo distintos idiomas, la posibilidad de cambiar el logotipo y cambiar el tamaño de las ventanas.
- Crear un prototipo de la versión móvil sin funcionalidad.

Próxima reunión

-

7. Acta de reunión

Fecha: 29/01/2014
Hora de inicio: 9:15
Hora fin: 10:00
Lugar: Ibermatica (Miramón)

Asistentes a la reunión:

Asistente	Rol	Asistencia	Razón
Julen Salgado	Programador	Si	-

Orden del día

- Analizar el aspecto actual de la aplicación y comprobar el funcionamiento de las funcionalidades ya implementadas.

Decisiones tomadas

- El aspecto se tiene que mejorar y se hará otra reunión para comentar los cambios que hay que hacer.
- Existen problemas a la hora la obtener y registrar archivos.
- Al mostrar el árbol de dependencias de los ensamblajes, se debe dar la opción de obtener y/o traer el archivo.
- Se ofrecerá la vista previa únicamente de los archivos especificados. Dicha especificación se irá completando según avanza el proyecto y se descubran nuevas librerías para hacerlo.

Tareas para la próxima reunión

- Corregir los fallos detectados en la reunión.
- Comenzar a desarrollar las funcionalidades básicas de navegación para la aplicación móvil.

Próxima reunión

-

8. Acta de reunión

Fecha: 04/02/2014
Hora de inicio: 17:00
Hora fin: 17:30
Lugar: Facultad de informática

Asistentes a la reunión:

Asistente	Rol	Asistencia	Razón
Julen Salgado	Programador	Si	-
Ana Sánchez	Directora	Si	-

Orden del día

- Realizar el seguimiento de la documentación del proyecto.

Decisiones tomadas

- Los requisitos del proyecto se consideran fijos y no habrá cambios, por lo que se plasmará el análisis y el diseño en la memoria.

Tareas para la próxima reunión

- Avanzar en el desarrollo de la memoria ahora que los requisitos no se van a cambiar.

Próxima reunión

Martes 4 de marzo del 2014, a las 17:00

9. Acta de reunión

Fecha: 10/02/2014
Hora de inicio: 10:30
Hora fin: 12:00
Lugar: Ibermatica (Miramón)

Asistentes a la reunión:

Asistente	Rol	Asistencia	Razón
Julen Salgado	Programador	Si	-

Orden del día

- Decidir qué hacer con las búsquedas.
- Analizar el impacto de las operaciones con archivos múltiples.
- Debatir sobre el aspecto de la aplicación.

Decisiones tomadas

- En las búsquedas se implementarán las funcionalidades básicas que filtra los archivos por el usuario que lo ha traído, el estado del flujo y la ubicación de la carpeta.
- Se extenderá la funcionalidad de las acciones para que acepten múltiples archivos. Además, faltan las acciones de eliminar y añadir archivos.
- El aspecto será más claro con tonos grisáceos, botones con bordes sin redondear y degradados. Se ajustará el contenido para aprovechar mejor el espacio. Cada línea de la tabla de archivos será de un color diferente.

Tareas para la próxima reunión

- Implementar todas las decisiones tomadas sobre las búsquedas, las operaciones sobre múltiples archivos y el aspecto.

Próxima reunión

-

10. Acta de reunión

Fecha: 03/03/2014
Hora de inicio: 11:00
Hora fin: 12:00
Lugar: Ibermatica (Miramón)

Asistentes a la reunión:

Asistente	Rol	Asistencia	Razón
Julen Salgado	Programador	Si	-

Orden del día

- Analizar el estado y funcionamiento de la aplicación para proponer mejoras.
- Decidir qué hacer con la funcionalidad de "Buscar en" de las tarjetas de búsqueda.

Decisiones tomadas

- Mostrar la lista de materiales en formato árbol.
- Que los botones de acción se agrupen en desplegables para ahorrar espacio.
- Añadir pestañas en la vista de tarjeta para mostrar también la lista de materiales, la funcionalidad "contiene" y la "dónde se utiliza".
- Si hace falta acceder directamente a la base de datos para completar la funcionalidad de las tarjetas, hacerlo.

Tareas para la próxima reunión

- Implementar todas las decisiones tomadas.
- Analizar de qué archivos podemos obtener las vistas previas.
- Analizar favoritos de búsqueda.
- Investigar la configuración de columnas.
- Idear la monitorización de las acciones de los usuarios y por IP. (Descargas…)
- Implementar la descarga de archivos en dispositivos móviles mediante ASP.

Próxima reunión

-

11. Acta de reunión

Fecha: 05/03/2014
Hora de inicio: 17:00
Hora fin: 17:30
Lugar: Facultad de informática

Asistentes a la reunión:

Asistente	Rol	Asistencia	Razón
Julen Salgado	Programador	Si	-
Ana Sánchez	Directora	Si	-

Orden del día

- Comentar las correcciones de la memoria.

Decisiones tomadas

- Rediseñar el diagrama de casos de uso.
- Ampliar la información sobre los productos de *SolidWorks*.

- La introducción y la estructura del documento deben aportar más información sobre el tema del proyecto.

Tareas para la próxima reunión

- Corregir y seguir completando la memoria.

Próxima reunión

Aproximadamente dentro de 2-4 semanas

ANEXO II – FIRMA DE UN APPLET

¿POR QUÉ ES NECESARIO?

Como ya se ha mencionado en este documento, la aplicación web hace uso de un *Applet* de *Java* en determinados momentos para poder ofrecer al usuario funcionalidades como la de poder acceder al sistema de archivos local. Para que el *Applet* tenga permisos de ser ejecutado y leer el sistema, es necesario que esté firmado. La explicación de por qué es necesario este paso es simple: por seguridad. Un *Applet* por defecto no puede tener acceso a los recursos del ordenador que lo ejecuta puesto que si fuera de otra manera podríamos encontrarnos ante *Applets* que, simplemente por ser ejecutados, borran discos duros, acceden a ficheros personales, etc.

Pero es evidente que en casos como el que nos ocupa es necesario que el *Applet* tenga acceso a ciertos recursos del equipo, puesto que queremos que el usuario pueda, por ejemplo, crear la estructura del almacén EPDM en su sistema local. Así pues existe un método para que el *Applet* pueda acceder a los recursos del equipo: firmar el *Applet* digitalmente. De esta forma, cuando se vaya a ejecutar el *Applet* firmado, se preguntará al usuario si confía en la persona o entidad que ha firmado el mismo y, en caso positivo, se le dará los permisos necesarios para ejecutarse.

PROCESO PARA REALIZAR LA FIRMA

En primer lugar deberemos hacer los siguientes preparativos:

- Para firmar digitalmente un *Applet* es necesario tener instalado un kit de desarrollo de *Java* (*Java Development Kip, JDK*), no siendo suficiente disponer de un entorno de ejecución de *Java* (*Java Runtime Environment, JRE*). Podremos descargar el JDK de la página oficial de Java.
- Debemos conocer dónde se encuentra el *JDK* instalado. Por defecto suele encontrarse en "Archivos de Programa\Java". Anotaremos la ruta de dicha carpeta (por ejemplo, "C:\Program Files\Java\jdk1.7.0_51") a la que de ahora en adelante nos referiremos como RUTA_JAVA.
- Una vez se tiene el *Applet* en un archivo *.jar*, es necesario abrir la consola de comandos con permiso de administrador. Para ello, seleccionamos Inicio>Programas>Accesorios, hacemos clic derecho sobre "Símbolo del sistema" y seleccionamos "Ejecutar como administrador".

En primer lugar deberemos crear la firma (este paso no es necesario repetirlo cada vez que queramos firmar un *Applet*):

- Escribimos en la consola el siguiente texto y pulsamos “intro”:

```
RUTA_JAVA\bin\keytool -genkey -alias nombreClave -validity
120 –v
```

  donde "nombreClave" es un alias para esta clave y "120" será el número de días de validez que tendrá la clave.
- Se nos pedirán los siguientes datos personales:
  - Contraseña del almacén de claves (si no se ha creado ninguna firma anteriormente, habrá que elegir una contraseña que deberemos recordar).
  - Nombre y Apellido.

    - Nombre de la unidad de organización (departamento).
    - Nombre de la organización.
    - Nombre de la ciudad o localidad.
    - Nombre de la provincia.
    - Código del país de dos letras (ES, de España).
- Una vez rellenados los datos se nos mostrarán de nuevo en pantalla y se nos pedirá que confirmemos que son correctos.
- Por último se nos pedirá una contraseña para la clave. Una vez elegida, se creará la firma digital.

A continuación firmaremos el *Applet* deseado de la siguiente forma:

- En la consola de comandos escribimos el siguiente texto y pulsamos "intro":

```
RUTA_JAVA\jarsigner.exe RUTA_APPLET\miApplet.jar nombreClave
-verbose
```

  donde:
    - RUTA_JAVA: la ruta del JDK (como ya se ha mencionado antes)
    - RUTA_APPLET: la ruta de la carpeta en la que se encuentra el *Applet* (cuidado, que no sea de sólo lectura).
    - miApplet.jar: el nombre del *Applet* que queramos firmar.
    - nombreClave: el alias de la firma que queramos usar.
- Se nos pedirá que introduzcamos la clave de la firma.
- Se muestra un aviso de que la validez de la firma será de X días (120 en el ejemplo).
- El *Applet* está firmado y listo para usarse.

ANEXO III – TECNOLOGÍA USADA

INTERNET INFORMATION SERVICES

Ilustración 94: Logo IIS7

Internet Information Services es un servidor web para el sistema operativo Microsoft Windows. En este proyecto se utilizará para montar un servidor web que aloje páginas *HTML* y *ASP.net*. También se creará un sitio web que permitirá a los usuarios consumir los servicios web que harán uso de la API de *Enterprise PDM*.

WEB SERVICE (WCF)

Ilustración 95: Logo WCF

Windows Communication Foundation es un conjunto de librerías que provee Microsoft en el *Framework .NET* para la construcción de aplicaciones orientadas a servicios. Usa una variedad de protocolos amplia comparado con antiguas tecnologías de *web service* de Microsoft, los cuales permiten el acceso a los servicios hospedados en servidores mediante *HTTP*.

WFC usa la clase *DataContractSerializer* para serializar los objetos y optimizar el tamaño de las peticiones que se generan. En caso de programar aplicaciones web mediante la plataforma .NET, ofrece un acceso intuitivo a los objetos, y por lo tanto a los servicios, alojados en el servidor realizando llamadas a clases de *C#* o *Visual Basic*.

En este caso, también se usará la técnica *AJAX (Asynchronous JavaScript And XML)* en *JavaScript* para realizar peticiones *HTTP* al servidor, en cuyo caso, los datos se serializarán en formato *JSON (JavaScript Object Notation)*.

SOLIDWORKS ENTERPRISE PDM API

Ilustración 96: Logo EPDM

La *API* de *SolidWorks Enterprise PDM* es la interfaz de programación que permite trabajar y crear aplicaciones que extiendan sus funcionalidades. Gracias a esta interfaz, es posible crear un cliente completo y personalizado en lenguajes como *C#* y *Visual Basic*.

Éste es, de hecho, el objetivo del proyecto: crear un cliente ligero de visualización que permita acceder a las funcionalidades básicas del Enterprise a través de dispositivos móviles y navegadores convencionales.

PLATAFORMA JAVA

Ilustración 97: Logo Java

La plataforma Java es un entorno desarrollado por *Sun MicroSystems* que es capaz de ejecutar aplicaciones programadas en el lenguaje de programación Java. Funciona como una máquina virtual que es capaz de ejecutar las aplicaciones. Ofrece un amplio abanico de tecnologías que ofrecen distintas soluciones.

Por ejemplo, es posible incrustar un programa en las páginas web mediante la tecnología *Applet*, siempre y cuando el usuario tenga instalado el entorno de ejecución de Java en su máquina. Parte de las funcionalidades de este proyecto se han desarrollado utilizando esta tecnología, ya que era necesario acceder al sistema de archivos local, algo que por temas de seguridad está prohibido por la web.

El *Applet* es capaz de ejecutarse en una página y obtener el consentimiento del usuario para acceder a sus recursos. Por si esto fuera poco, es posible ejecutar la aplicación en un segundo plano y hacer llamadas a sus funciones desde la página web usando *JavaScript*, como si de una librería se tratase. Esto ha permitido implementar funcionalidades que hubieran sido imposibles solamente con *JavaScript* y *HTML*.

Ilustración 98: Logo JQuery

JQuery es una biblioteca JavaScript que ofrece una forma más fácil y simplificada de manipular los documentos *HTML*. Así es capaz de acceder a los objetos *DOM* del documento, manejar eventos, desarrollar animaciones e incluso hacer uso de la técnica *AJAX* de forma transparente.

Además, ofrece extensiones que amplían sus funciones. *JQuery-UI* ofrecen la posibilidad de crear interfaces complejas multiplataforma sin necesidad de tener que preocuparse de programar la lógica. Por otra parte, también existe *JQuery Mobile*, que ofrece al usuario abstenerse de crear diseños complejos para dispositivos móviles y así preocuparse por el adaptar el contenido.