UNIVERSIDAD CARLOS III DE MADRID

VoCS

Mejora del Sistema de almacenamiento voluntario en la nube. Sistema de Etiquetas

Autor: Héctor de Anta Pérez
Tutor: Luis Miguel Sánchez García

Agradecimientos

En primer lugar me gustaría dar las gracias a mi tutor Luis Miguel por permitirme realizar este proyecto bajo su tutelaje y por apoyarme y ayudarme ante mi falta de tiempo.

En segundo lugar, agradecer a mis amigos todo el apoyo que me han dado, sin eso la realización de este proyecto hubiera sido mucho más complejo.

Y por último, agradecer a mis padres todo lo que se han esforzado para mostrarme apoyo y afecto durante estos meses.

Contents

Índice de Tablas

Índice de Ilustraciones

1. Introducción

Este documento pretende describir el proyecto "VOCS: Ampliación de Funcionalidades", el cual consiste en:

- Análisis de un proyecto ya existente para proceder a la implantación de nuevas funcionalidades que lo mejoren, así como retocar el proyecto original para que sea más intuitivo y sencillo de utilizar.

A continuación se presentará la motivación que se ha tenido para proceder a realizar este proyecto así como los objetivos concretos del proyecto y los acrónimos que se van a encontrar a lo largo del documento.

1.1 Motivación

Actualmente hay un concepto en auge denominado Computación en la Nube. Este nuevo concepto surge de la gran globalización de Internet, un recurso al cual tiene acceso un gran número de particulares y la mayoría de las empresas en este momento.

Gracias a la Computación en la Nube, el usuario puede utilizar una serie de recursos y aplicaciones en Internet sin necesidad de tener un software instalado en su ordenador.

Dado que las empresas manejan un volumen de datos que va creciendo exponencialmente, el gasto en sistemas de almacenamiento era cada vez mayor, por lo que el concepto de almacenamiento en la Nube cada vez se presentaba más necesario.

Con este tipo de almacenamiento los usuarios podrán almacenar todos sus datos en Internet y disponer de ellos desde cualquier lugar en el cual posean conexión a la red. Esto permite una mayor facilidad a la hora de tratar con datos o ficheros y, por otro lado, un ahorro económico importante al no necesitar ningún elemento hardware nuevo para ello.

Como contrapartida, dependiendo del espacio necesitado, estas aplicaciones pueden necesitar de un pago para la ampliación de espacio, aunque siempre supondrá un ahorro.

La mayor desventaja que este tipo de aplicaciones posee es la de la seguridad, y este ha sido el motivo por el cual este tipo de aplicaciones, a pesar de llevar varios años desarrolladas, no ha empezado a tener éxito. Actualmente se están desarrollando aplicaciones con un elevado sistema de seguridad que asegura a los clientes la confidencialidad de sus archivos. Pero por mucha seguridad que ofrezca este tipo de aplicaciones, la confidencialidad siempre quedará supeditada a la malicia de la empresa responsable de la aplicación.

Por último, dado que este es un concepto muy reciente para la mayoría de la población, este tipo de aplicaciones deben centrarse en ser muy intuitivas ya que de no ser así, generará mayor desconfianza por parte de los usuarios.

Por lo tanto, la situación actual se centra en usuarios con necesidad de almacenaje de información de una forma segura y, por otro lado, la necesidad de una aplicación sencilla y atractiva a la vista para un manejo fácil de la misma.

1.2 Objetivos.

Como se ha comentado en el apartado anterior, se busca la seguridad y la simplicidad en este tipo de aplicaciones. Dado que el proyecto origen es una aplicación centrada en la seguridad, el objetivo principal de este proyecto es el de análisis de un proyecto ya existente de almacenamiento en la Nube para detectar deficiencias hacia el usuario y posteriormente un análisis, Diseño e Implementación de funcionalidades que permitan que el usuario utilice la aplicación de una forma simple e intuitiva.

A continuación se van a describir los objetivos principales que tenía el proyecto base:

- Uso de la filosofía open-source para el desarrollo del sistema final.
- Debe disponer de medidas de seguridad oportunas para garantizar la confidencialidad, integridad, autenticación, disponibilidad, accesibilidad y tolerancia a fallos en todo momento.
- Debe permitirse el acceso ubicuo y transparente a los datos almacenado en el sistema.
- Se debe tener un sistema con alta disponibilidad y que disponga de un rendimiento aceptable para el usuario.

Como se puede comprobar, esta aplicación se centraba en la parte de rendimiento y seguridad de la información, pero dejaba la parte de interacción con el usuario más descuidada.

Los objetivos de este proyecto son los siguientes:

- Integración de nuevas funcionalidades a la aplicación que proporcionen servicios nuevos a los usuarios.
- Integración de nuevos elementos en la aplicación que la hagan visualmente más atractiva.
- Realizar los cambios sin perder la estructura y apariencia del proyecto inicial.
- Realizar los cambios sin desmejorar los sistemas de seguridad que provee el proyecto base.

Para poder llevar a cabo estos objetivos, se han marcado los siguientes sub-objetivos:

- Incluir funcionalidades de utilización de etiquetas para los ficheros.
- Modificar la Base de Datos para permitir una clasificación de ficheros por etiquetas.
- Incluir un sistema de búsqueda de ficheros por etiquetas que funcione de una forma visual.
- Incluir un sistema de búsqueda de ficheros por etiquetas que permita hacer búsquedas múltiples.

- Incluir funcionalidades para que se incluyan etiquetas a todos los ficheros de forma automática.
- Incluir funcionalidades para permitir al usuario elegir qué etiquetas quiere agregar a cada fichero.

1.3 Acrónimos

AJAX: Asynchronous JavaScript and XML

API: Application Programming Interface (Interfaz de programación de aplicaciones)

ASP: Active Server Pages

BD: Base de Datos

CGI: Common Gateway Interface

CSS: Cascading Style Sheets (Hojas de estilo en cascada)

C#: C Sharp

DB: Data Base

DBM: DataBase Management.

DHTML: Dynamic HTML

DTD: Definición de Tipo de Documento

E/S: Entrada/Salida

HTML: HyperText Markup Language

HTTP: Hypertext Transfer Protocol

HTTPS: Hypertext Transfer Protocol Secure

IBM: International Business Machines

JSON: JavaScript Object Notation

JSP: JavaServer Pages

MVC: Modelo Vista Controlador

PC: Personal Computer

PHP: PHP Hypertext Pre-processor

SGBD: Sistema Gestor de Bases de Datos

SGML: Standard Generalized Markup Language

SQL: Structured Query Language

TFG: Trabajo de Fin de Grado.

UCD: Diseño centrado en el Usuario

URL: Uniform Resource Locator

W3C: World Wide Web Consortium

XML: eXtensible Markup Language

2. Estado de la Cuestión

2.1 Modelo Vista Controlador

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

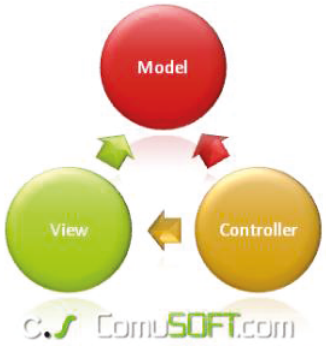

Ilustración 1. MVC

Este patrón fue descrito por primera vez en 1979 por Trygve Reenskaug, trabajador de Smalltalk, en unos laboratorios de gran investigación de Xerox.

Los MVC cumplen perfectamente el fin particular de cualquier framework.

El Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos (Modelo, Vista y Controlador). El Patrón MVC se ve frecuentemente en aplicaciones Web, donde la Vista es la página HTML y el código que provee de datos dinámicos a la página; el Modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio; el Controlador es el responsable de recibir los eventos de entrada desde la Vista.

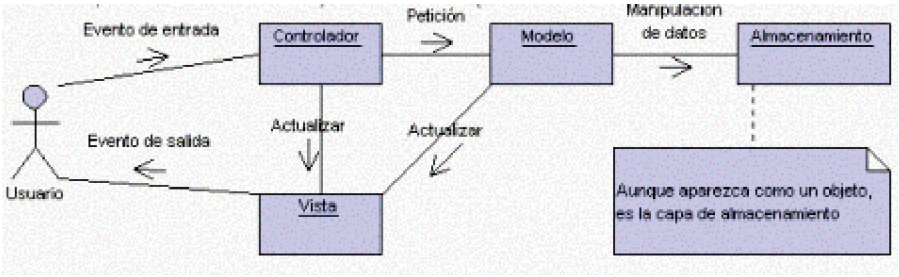

Ilustración 2. Funcionamiento MVC

La finalidad del modelo es mejorar la reusabilidad por medio del desacople entre la vista y el modelo.

La funcionalidad de cada uno de los componentes del patrón es la siguiente:

Modelo

Es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, o calculando los totales e impuestos del carrito de compra. Esto quiere decir que aquí se operan los datos y las reglas de negocio asociadas al sistema, incluyendo el análisis sintáctico y el procesamiento de los datos de entrada y de los datos de salida.

Por lo tanto, El Modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Definir las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: “Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor”.
- Llevar un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificar a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un fichero batch que actualiza los datos, un temporizador que desencadena una inserción, etc.). Un ejemplo de MVC con un modelo pasivo (aquel que no notifica cambios en los datos) es la navegación web, que responde a las entradas del usuario, pero no detecta los cambios en datos del servidor.

Vista

Este presenta el Modelo, usualmente la interfaz de usuario. La vista es la capa de la aplicación que ve el usuario en un formato adecuado para interactuar, en otras palabras, es nuestra interface gráfica.

Por lo tanto, la Vista:

- Recibe datos del modelo y mostrarlos al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de “Actualización()”, para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Controlador

El Controlador es la capa que controla todo lo que puede realizar nuestra aplicación. Responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Está compuesto por acciones que se representan con funciones en una clase. Por ejemplo, un controlador llamado "Clientes", y este controlador puede realizar las acciones "Crear","Editar","Listar" entre otras.

Por lo tanto, el Controlador:

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas. Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega(nueva_orden_de_venta)".

Flujo que sigue el control en una implementación general de un MVC

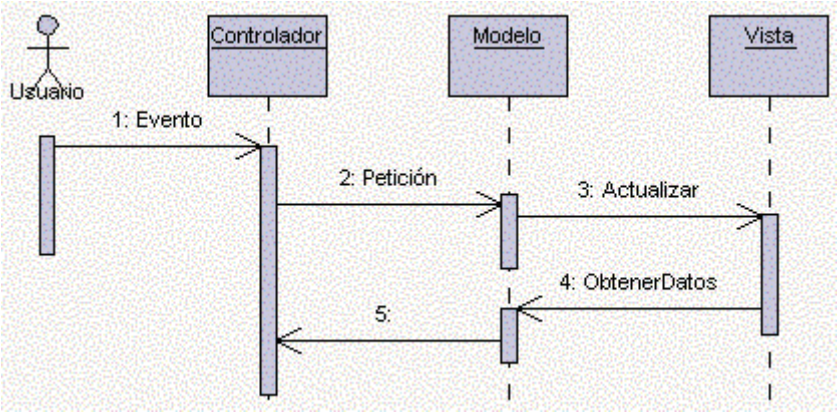

Ilustración 3. Flujo MVC

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo que sigue el control generalmente es el siguiente:

1. El usuario introduce el evento.
2. El Controlador recibe el evento y lo traduce en una petición al Modelo (aunque también puede llamar directamente a la vista).
3. El modelo (si es necesario) llama a la vista para su actualización.
4. Para cumplir con la actualización la Vista puede solicitar datos al Modelo.
5. El Controlador recibe el control.

A continuación se van a mostrar las principales ventajas que conlleva la utilización de un patrón MVC:

- Clara separación entre interfaz, lógica de negocio y de presentación, que además provoca parte de las ventajas siguientes.
- Sencillez para crear distintas representaciones de los mismos datos.
- Facilidad para la realización de pruebas unitarias de los componentes, así como de aplicar desarrollo guiado por pruebas (TDD).
- Reutilización de los componentes.
- Simplicidad en el mantenimiento de los sistemas.
- Facilidad para desarrollar prototipos rápidos.
- Los desarrollos suelen ser más escalables.
- Tener que ceñirse a una estructura predefinida, lo que a veces puede incrementar la complejidad del sistema. Hay problemas que son más difíciles de resolver respetando el patrón MVC.
- La curva de aprendizaje para los nuevos desarrolladores se estima mayor que la de modelos más simples como Webforms.
- La distribución de componentes obliga a crear y mantener un mayor número de ficheros.

2.2 Tecnologías Web

El proyecto a desarrollar será un sistema basado en web, por lo que es necesario realizar una comparación sobre las diferentes tecnologías web existentes.

En primer lugar se realizará un análisis a las diferentes tecnologías de desarrollo de aplicaciones web y en segundo lugar el análisis se centrará en los gestores de bases de datos relacionales.

2.2.1 Tecnologías de desarrollo de aplicaciones web

A continuación se muestra el análisis para las tecnologías de desarrollo de aplicaciones web.

PHP

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. PHP puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

Se caracteriza por la facilidad de aprendizaje por parte de los principiantes y la aportación de características avanzadas para programadores profesionales, como permitir programación orientada a objetos.

Ventajas de PHP:

- Es un lenguaje multiplataforma.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una Base de Datos.
- El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).

Desventajas de PHP:

Como es un lenguaje que se interpreta en ejecución para ciertos usos puede resultar un inconveniente que el código fuente no pueda ser ocultado. La ofuscación es una técnica que puede dificultar la lectura del código pero no la impide y, en ciertos casos, representa un costo en tiempos de ejecución

En cuanto a los desarrollos de PHP, se ha dado lugar a gran cantidad de *frameworks* que ofrecen una estructura conceptual y tecnológica de soporte, definido con módulos software. Gracias esto PHP puede ser fácilmente organizado y desarrollado.
Los *frameworks* más famosos son:
1- Zend Framework.

2- CakePHP.

CakePHP

CakePHP es un marco de desarrollo rápido para PHP, libre, de código abierto. Se trata de una estructura que sirve de base a los programadores para que éstos puedan crear aplicaciones Web.

Las ventajas de utilizar CakePHP son las siguientes:

- Comunidad activa
- Licencia flexible
- Compatible con PHP4 y PHP5
- CRUD integrado para la interacción con la base de datos
- Soporte de aplicación [scaffolding]
- Generación de código
- Arquitectura Modelo Vista Controlador (MVC)
- Despachador de peticiones [dispatcher], con URLs y rutas personalizadas y limpias
- Validación integrada
- Plantillas rápidas y flexibles (sintaxis de PHP, con ayudantes[helpers])
- Ayudantes para AJAX, Javascript, formularios HTML y más
- Componentes de Email, Cookie, Seguridad, Sesión y Manejo de solicitudes
- Listas de control de acceso flexibles
- Limpieza de datos
- Caché flexible
- Localización
- Funciona en cualquier subdirectorio del sitio web, con poca o ninguna configuración de Apache

Zend Framework

Zend Framework (ZF) es un framework de código abierto para desarrollar aplicaciones web y servicios web con PHP 5. ZF es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de ZF es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. A menudo se refiere a este tipo de diseño como "use-at-will" (uso a voluntad).

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de Zend Framework conforman un potente y extensible framework de aplicaciones web al combinarse. ZF ofrece un gran rendimiento y una robusta implementación MVC, una abstracción de base de datos fácil de usar, y un componente de formularios que implementa la prestación de formularios HTML, validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos. Otros componentes, como Zend_Auth y Zend_Acl, proveen autentificación de usuarios y autorización diferentes a las tiendas de certificados comunes. También existen componentes que implementan bibliotecas de cliente para acceder de forma sencilla a los web services más populares. Cualesquiera que sean las necesidades de su solicitud, usted tiene todas las posibilidades de encontrar un componente de Zend Framework que se pueda utilizar para reducir drásticamente el tiempo de desarrollo, con una base completamente sólida.

El principal patrocinador del proyecto Zend Framework es Zend Technologies, pero muchas empresas han contribuido con componentes o características importantes para el marco. Empresas como Google, Microsoft y StrikeIron se han asociado con Zend para proporcionar interfaces de servicios web y otras tecnologías que desean poner a disposición de los desarrolladores de Zend Framework.

JSP

JavaServer Pages (JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML entre otros tipos de documentos. JSP es similar a PHP pero usa el lenguaje de programación Java.

Para desplegar y correr JavaServer Pages, es requerido un servidor web compatible con contenedores servlet como Apache Tomcat o Jetty.

El rendimiento de una página JSP es el mismo que tendría el servlet equivalente, ya que el código es compilado como cualquier otra clase Java. A su vez, la máquina virtual compilará dinámicamente a código de máquina las partes de la aplicación que lo requieran. Esto hace que JSP tenga un buen desempeño y sea más eficiente que otras tecnologías web que ejecutan el código de una manera puramente interpretada.

La principal ventaja de JSP frente a otros lenguajes es que el lenguaje Java es un lenguaje de propósito general que excede el mundo web y que es apto para crear clases que manejen lógica de negocio y acceso a datos de una manera prolija. Esto permite separar en niveles las aplicaciones web, dejando la parte encargada de generar el documento HTML en el archivo JSP.

Otra ventaja es que JSP hereda la portabilidad de Java, y es posible ejecutar las aplicaciones en múltiples plataformas sin cambios. Es común incluso que los desarrolladores trabajen en una plataforma y que la aplicación termine siendo ejecutada en otra.

Los servlets y Java Server Pages (JSPs) son dos métodos de creación de páginas web dinámicas en servidor usando el lenguaje Java. En ese sentido son similares a otros métodos o lenguajes tales como el PHP, ASP o los CGIs, programas que generan páginas web en el servidor. Sin embargo, se diferencian de ellos en otras cosas.

Para empezar, los JSPs y servlets se ejecutan en una máquina virtual Java, lo cual permite que, en principio, se puedan usar en cualquier tipo de ordenador, siempre que exista una máquina virtual Java para él. Cada servlet (o JSP, a partir de ahora lo usaremos de forma indistinta) se ejecuta en su propio hilo, es decir, en su propio contexto; pero no se comienza a ejecutar cada vez que recibe una petición, sino que persiste de una petición a la siguiente, de forma que no se pierde tiempo en invocarlo (cargar programa + intérprete). Su persistencia le permite también hacer una serie de cosas de forma más eficiente: conexión a bases de datos y manejo de sesiones, por ejemplo.

ASP .NET

ASP.NET es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es utilizado para construir sitios web dinámicos, aplicaciones web y servicios web XML.

ASP.NET está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por .NET Framework (Python, Visual Basic, C#, etc.).

Los proyectos web creados mediante ASP .NET únicamente serán interpretados por servidores cuya plataforma sea Microsoft Windows.

Las ventajas que hacen destacar a ASP .NET sobre otros lenguajes son:

- Separación clara de dónde tiene que ir cada tipo de lógica (separation of concerns). Esto nos facilita el mantenimiento y la escalabilidad de la aplicación, además de facilitar el desarrollo en paralelo de las distintas partes o módulos.
- Soporte al desarrollo por medio de metodología TDD por defecto (Test Driven Development). Por tanto facilidad para crear mock tests y para poder correr los tests desde fuera del proceso de asp.net. Gran diferencia con respecto a cómo se tenían que hacer las pruebas anteriormente, o bien a mano, o bien con complejas macros para disparar eventos en la interfaz.

- Arquitectura "pluggable" y extensible, en la que los componentes pueden ser aumentados o extendidos, algo que ya se podía hacer en las versiones anteriores de asp.net, pero que ahora va más allá. Ahora se pueden usar inyectores de dependencia como Castle Windsor, Spring.net oUnity.
- Motor de routing (asociación de una URL concreta con el controller correcto) para tener url más legibles, al estilo REST.
- No hay ViewState ni ciclo de vida de las páginas. Menos peso, menos complejidad.
- Control total del HTML generado. No tenemos controles que añaden su propio markup. Esto nos permite además integrar más fácilmente librerías como JQuery o MooTools, y olvidarnos de todo el código que asp.net inyecta para el mantenimiento del estado y el ViewState.

ColdFusion

Coldfusion (Adobe ColdFusion) es un servidor de aplicaciones y un lenguaje de programación usado para desarrollar aplicaciones de Internet, generalmente sitios web generados dinámicamente. En este aspecto, es un producto similar a ASP, JSP o PHP.

ColdFusion es una herramienta que corre en forma concurrente con la mayoría de los servidores web de Windows, Mac OS X, Linux y Solaris (también en servidores web personales en Windows 98y puede ser usado para intranets). El servidor de aplicaciones web de ColdFusion trabaja con el servidor HTTP para procesar peticiones de páginas web. Cada vez que se solicita una página de ColdFusion, el servidor de aplicaciones ColdFusion ejecuta el guion o programa contenido en la página.

No es un lenguaje de bases de datos, pero interacciona de manera simple con bases de datos (Sybase, Oracle, MySQL, SQL Server, o Access). Usando SQL estándar, las páginas y aplicaciones web pueden fácilmente recuperar, guardar, formatear y presentar información dinámicamente.

Coldfusion es la elección más popular de tecnología en servidores de aplicaciones para muchos desarrolladores y potentes sitios web, tales como Lloyds TSB, DHL y Microsoft's bCentral.

Coldfusion es actualmente, el principal servidor de aplicaciones web cross-platform existente. Con intuitivas herramientas visuales y probada tecnología de servidor.

Coldfusion le brinda la manera más rápida para construir y desarrollar soluciones escalables que integran, navegadores, servidores y tecnologías de bases de datos.

Django

Django es un framework de desarrollo web de código abierto, escrito en Python, que cumple en cierta medida el paradigma del Modelo Vista Controlador. Fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la World Company de Lawrence, Kansas, y fue liberada al público bajo una licencia BSD en julio de 2005; el framework fue nombrado en alusión al guitarrista de jazz gitano Django Reinhardt.

En junio del 2008 fue anunciado que la recién formada Django Software Foundation se haría cargo de Django en el futuro.

La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio No te repitas (DRY, del inglés Don't Repeat Yourself). Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos.

En junio de 2008 fue anunciado que la recién formada Django Software Foundation se haría cargo de Django en el futuro.

Las ventajas que ofrece Django son las siguientes:

- Proporciona una serie de características que facilitan el desarrollo rápido de páginas orientadas a contenidos.
- Proporciona una aplicación incorporada para administrar los contenidos, que puede incluirse como parte de cualquier página desarrollada con Django e incluso administrar varias páginas desarrolladas con Django a partir de una misma instalación.
- Proporciona un sistema incorporado de "vistas genéricas". Así el desarrollador no está obligado a escribir la lógica de ciertas tareas comunes.
- Cuenta con un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas.

Ruby on Rails

Ruby on Rails, también conocido como RoR o Rails, es un framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC). Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración. El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible. Rails se distribuye a través de RubyGems, que es el formato oficial de paquete y canal de distribución de bibliotecas y aplicaciones Ruby.

Las ventajas de Ruby frente a otros lenguajes son las siguientes:

- Ruby es un lenguaje orientado a objetos: todos los tipos de datos son un objeto, incluidas las clases y tipos que otros lenguajes definen como primitivas.
- Dispone de cuatro niveles de ámbito de variable: global, clase, instancia y local.
- Proporciona un recolector de basura automático.
- Las aplicaciones desarrolladas mediante Ruby son totalmente portables.
- Incluye carga dinámica de bibliotecas compartidas en la mayoría de las plataformas.

C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes, tales como ROOT.

Una particularidad del C++ es la posibilidad de redefinir los operadores, y de poder crear nuevos tipos que se comporten como tipos fundamentales.

El nombre C++ fue propuesto por Rick Mascitti en el año 1983, cuando el lenguaje fue utilizado por primera vez fuera de un laboratorio científico. Antes se había usado el nombre "C con clases". En C++, la expresión "C++" significa "incremento de C" y se refiere a que C++ es una extensión de C.

Las ventajas de C++ frente a otros lenguajes son las siguientes:

- Lenguaje de programación orientado a objetos y multiparadigma.
- Lenguaje muy potente y robusto destinado a la creación de sistemas complejos.
- Es un lenguaje muy empleado, por lo que hay disponible un gran número de fuentes de documentación para solucionar cualquier problema que pueda surgir durante el desarrollo.

2.2.2 Comparativa

Una vez descritas las alternativas se realiza una valoración de las mismas, considerando el impacto en el desarrollo del proyecto actual, tanto desde el punto de vista tecnológico como desde el punto de vista operativo.

Para cada tecnología web propuesta se valora el impacto en el proyecto y se establece su viabilidad económica. Para ello comprobará la existencia de varios aspectos tecnológicos y la posibilidad de los aspectos económicos.

A la hora de puntuar los aspectos tecnológicos se tendrán en cuenta las facilidades de desarrollo que ofrece la tecnología web así como los requisitos necesarios para lograr un sistema funcional, óptimo y de calidad pensando en futuros cambios.

El análisis comparativo se muestra en la siguiente tabla:

TECNOLOGÍA	Soporte AJAX	Soporte base de datos relacional	Soporte ORM	Independencia entorno de desarrollo	Multi-plataforma	Soporte de frameworks de migración de datos	Licencia	Coste
JSP	Sí	Sí	Sí	No	Sí	Mediante complementos	Propietaria	Bajo
PHP	Sí	Sí	Sí	Sí	Sí	Sí	Libre	Bajo
ASP .NET	Sí	Sí	Sí	No	No	Mediante complementos	Propietaria	Alto
ColdFusion	Sí	Sí	Sí	No	No	Mediante complementos	Propietaria	Medio
Django	Mediante complementos	Sí	Sí	No	Sí	Mediante complementos	Libre	Bajo
Ruby on Rails	Mediante complementos	Sí	Sí	No	Sí	Sí	Libre	Bajo
C++	Sí	Sí	Sí	Sí	Sí	No	Depende del compilador	Depende del compilador

Tabla 1. Comparación Tecnologías

El lenguaje seleccionado es PHP. La razón de su puntuación y por tanto de su elección reside en los siguientes puntos:

- PHP como se ha comentado es multiplataforma, no requiere ningún entorno de desarrollo específico. La mayoría de las tecnologías evaluadas sí requieren de un entorno en concreto o tiene más restricciones. Por otro lado, hay algunas tecnologías que no requieren ese entorno específico, pero son herramientas mucho menos potentes que PHP, por lo que esta sería la opción más adecuada.

- PHP es un lenguaje libre, sin propietario. Esto hace que sea uno de los lenguajes más utilizados por los desarrolladores, por lo que existen numerosas fuentes de documentación y ayuda para prácticamente cualquier problema que pueda surgir durante la fase de implantación.

- Como ya se ha comentado, es un lenguaje libre. Debido a esto, no generaría costes extras en el proyecto. El único coste será la máquina servidora de la aplicación web. Dicho coste será más económico que si se hubiera escogido ASP .NET, ya que el servidor puede tener un sistema operativo libre, como Linux.

- PHP admite la utilización de MySQL de una forma rápida y sencilla. Esto será de gran utilidad para llevar a cabo la implementación de la base de datos.

Se escoge por tanto PHP, ya que es una tecnología orientada a proyectos medianos que permite obtener resultados óptimos en un tiempo moderado.

2.2.3 Sistemas gestores de bases de datos

A continuación se muestra el análisis para los sistemas gestores de bases de datos MySQL, NoSQL, Oracle y PostgreSQL.

MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.1 MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

Las ventajas de MySQL frente al resto de sistemas gestores de bases de datos son las siguientes:

- Amplio subconjunto del lenguaje SQL.
- Disponibilidad en gran cantidad de plataformas y sistemas.
- Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferente velocidad de operación, soporte físico, capacidad, distribución geográfica, transacciones, etc.
- Soporte para transacciones y claves ajenas.
- Proporciona una conectividad segura.
- Sistema muy rápido. Hasta 80 veces más que Oracle.
- Admite búsqueda e indexación de campos de texto.
- Permite escoger entre múltiples motores de almacenamiento para cada tabla.
- Soporte óptimo para la herramienta phpMyAdmin escrita en PHP. Facilidad de manejo con los datos almacenados gracias a la herramienta.

NoSQL

NoSQL hace referencia a una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales. El aspecto más destacado es que no usan SQL como el principal lenguaje de consultas.

La mayoría de sistemas NoSQL emplean una arquitectura distribuida, manteniendo los datos de forma redundante en varios servidores, usando frecuentemente una tabla hash distribuida. De esta forma, el sistema puede realmente escalar añadiendo más servidores y el fallo en un servidor puede ser tolerado.

MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico (MongoDB llama ese formato BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

El desarrollo de MongoDB empezó en octubre de 2007 por la compañía de software 10gen. Ahora MongoDB es una base de datos lista para la producción de uso y con muchas características. Esta base de datos es altamente utilizada en las industrias. MTV Network, Craiglist y Foursquare son algunas de las empresas que utilizan esta base de datos.

Las ventajas de este tipo de sistemas son las siguientes:

- Pueden manejar gran cantidad de datos sin verse alterado su rendimiento.
- La gestión de datos no genera cuellos de botella.
- Los sistemas NoSQL son de código abierto.
- Disponen de una gran escalabilidad en el caso de ampliación de los sistemas.

Oracle

Oracle es un sistema de gestión de base de datos objeto-relacional desarrollado por Oracle Corporation. Recientemente, Oracle adquirió a Sun Microsystems y con ella la empresa encargada comercial de MySQL.

Oracle está considerado como uno de los sistemas de bases de datos más completos por su gran estabilidad y escalabilidad. La única edición gratuita es la Express Edition, que es compatible con las demás ediciones de Oracle Database.

Las ventajas de Oracle frente a otros sistemas son las siguientes:

- Oracle es el motor de base de datos relacional más usado a nivel mundial.
- Dispone de un lenguaje de diseño de bases de datos muy completo que permite implementar diseños activos, con triggers y procedimientos almacenados, con una integridad bastante potente.
- Permite el uso de particiones para mejorar la eficiencia de replicación e incluso ciertas versiones admiten la administración de bases de datos distribuidas.
- El software del servidor puede ejecutarse en multitud de sistemas operativos.
- Oracle está orientado a sistemas de datos de gran tamaño.

PostgreSQL

PostgreSQL es un SGBD relacional orientado a objetos y libre, publicado bajo la licencia BSD.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyados por organizaciones comerciales. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group)

Las características de PostgreSQL más relevantes son las siguientes:

- Alta concurrencia. Mediante un sistema denominado MVCC (Acceso concurrente multiversión) PostgreSQL permite que mientras un proceso escriba en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.

- Amplia variedad de tipos nativos como textos con longitud ilimitada, figuras geométricas, números de precisión arbitraria o incluso tipos creados por el usuario.

- Dispone de otras características más comunes como triggers, claves ajenas, herencia de tablas y operaciones geométricas.

2.2.4 Comparativa

Al igual que con las tecnologías de desarrollo de aplicaciones web, se va a realizar una comparación entre los sistemas gestores de bases de datos analizados anteriormente.

Se tendrán en cuenta los aspectos técnicos, la viabilidad económica y la adecuación de cada uno de ellos al proyecto que se va a desarrollar con el lenguaje PHP.

En lo que se refiera a aspectos técnicos, se valorará la compatibilidad de cada sistema con los sistemas operativos más utilizados actualmente.

SISTEMA GESTOR	Windows	Linux	MacOS	Compatible con PHP	iOS	Android	Symbian	Viabilidad económica
MySQL	Sí	Sí	Sí	Sí	Mediante complementos	No	Sí	Alta
NoSQL (MongoDB)	Sí	Sí	Sí	Sí	No	No	No	Alta
Oracle	Sí	Sí	Sí	Mediante complementos	No	No	No	Baja
PostgreSQL	Sí	Sí	Sí	Sí	Sí	No	No	Alta

Tabla 2. Comparativa BBDD

El sistema seleccionado es MySQL. La razón de su puntuación y por tanto de su elección reside en los siguientes puntos:

- MySQL es uno de los gestores de bases de datos con mejor rendimiento gracias a la velocidad en que realiza las operaciones.

- MySQL es compatible con la mayoría de sistemas operativos existentes.

- Es totalmente gratuito.

- Dado que es un sistema con un consumo de recursos muy bajo, una aplicación con bases de datos realizadas en MySQL tendría la ventaja de poder ser ejecutada en casi cualquier máquina

- Unidos los dos puntos anteriores, se puede confirmar que MySQL es muy viable desde el punto de vista económico.

Se puede concluir que PHP unido a MySQL forman un sistema de desarrollo web muy ágil, de calidad y orientado a proyectos medianos.

Si a PHP y MySQL se le unen Linux como sistema operativo y Apache como servidor de aplicaciones, obtenemos el conocido modelo LAMP (Linux, Apache, MySQL y PHP).

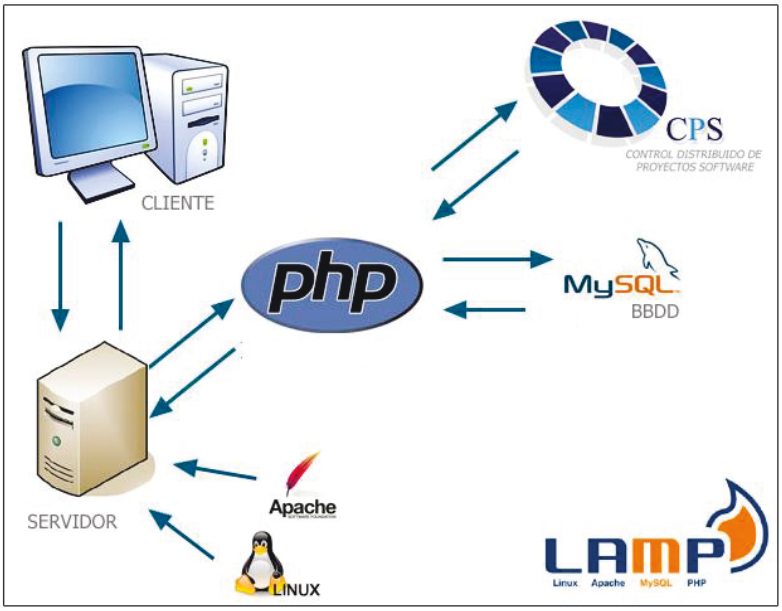

Ilustración 4. Modelo LAMP

Este modelo proporciona una funcionalidad completa para un proyecto web como el que se desea desarrollar. Este es, por tanto, el modelo que se utilizará en el momento del desarrollo y la implementación del sistema (incluyendo el framework ZEND para php).

A continuación se van a explicar varias tecnologías que también afectan en el desarrollo de este proyecto: HMTL, HTML5, JavaScript, CSS, Jquery y ZEND framework.

HTML

HTML, siglas de HyperText Markup Language («lenguaje de marcado hipertextual»), hace referencia al lenguaje de marcado predominante para la elaboración de páginas web que se utiliza para describir y traducir la estructura y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes. El HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo, JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.1

HTML también sirve para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (como HTML 4.01 y anteriores).

Ventajas e Inconvenientes del HMTL

- Es el lenguaje de formateo para los navegadores web.
- Es fácil de entender y utilizar
- Su uso es muy extendido
- No tiene semántica. Uso de etiquetas con nombres diferentes.
- El contenido no puede ser reconocido ni procesado por programas
- Tiene un costoso mantenimiento de las páginas
- No tiene estándares comunes.
- Solo tiene hiperenlaces simples (XML puede tener de 1 a n enlaces).

En resumen, el html es un lenguaje muy fácil de comprender y muy utilizado para la presentación de la información, pero esta no se puede procesar ni almacenar, ya que no permite su manipulación por un programa debido a su anarquía.

HTML5

HTML5 (HyperText Markup Language, versión 5) es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. HTML5 especifica dos variantes de sintaxis para HTML: un «clásico» HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser servida como XML (XHTML) (application/xhtml+xml).1 2 Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo.

Todavía se encuentra en modo experimental, lo cual indica la misma W3C; aunque ya es usado por múltiples desarrolladores web por sus avances, mejoras y ventajas.

Algunas de las novedades que HTML5 presenta son las siguientes:

- HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos. Algunos de ellos son técnicamente similares a las etiquetas y , pero tienen un significado semántico.
- También establece mejoras en el elemento <canvas>, capaz de renderizar en los navegadores más importantes (Mozilla, Chrome, Opera, Safari e IE) elementos 3D.
- Incorpora etiquetas (canvas 2D y 3D, audio, video) con codecs para mostrar los contenidos multimedia. Actualmente hay una lucha entre imponer codecs libres (WebM + VP8) o privados (H.264/MPEG-4 AVC).
- Incluye Etiquetas para manejar grandes conjuntos de datos: Datagrid, Details, Menu y Command. Permiten generar tablas dinámicas que pueden filtrar, ordenar y ocultar contenido en cliente.
- Mejoras en los formularios. Nuevos tipos de datos (eMail, number, url, datetime …) y facilidades para validar el contenido sin Javascript.

- Visores: MathML (fórmulas matemáticas) y SVG (gráficos vectoriales). En general se deja abierto a poder interpretar otros lenguajes XML.
- Drag & Drop. Nueva funcionalidad para arrastrar objetos como imágenes.
- Añade etiquetas para manejar la Web Semántica (Web 3.0): header, footer, article, nav, time (fecha del contenido), link rel=‘’ (tipo de contenido que se enlaza).
- Estas etiquetas permiten describir cual es el significado del contenido. Por ejemplo su importancia, su finalidad y las relaciones que existen. No tienen especial impacto en la visualización, se orientan a buscadores.
- Los buscadores podrán indexar e interpretar esta meta información para no buscar simplemente apariciones de palabras en el texto de la página.
- Permite incorporar a las páginas ficheros RDF / OWL (con meta información) para describir relaciones entre los términos utilizados.
- API para hacer Drag & Drop. Mediante eventos.
- API para trabajar Off-Line. Permite descargar todos los contenidos necesarios y trabajar en local.
- API de Geoposicionamiento para dispositivos que lo soporten.
- API Storage. Facilidad de almacenamiento persistente en local, con bases de datos (basadas en SQLite) o con almacenamiento de objetos por aplicación o por dominio Web (Local Storage y Global Storage). Se dispone de una Base de datos con la posibilidad de hacer consultas SQL.
- WebSockets. API de comunicación bidireccional entre páginas. Similar a los Sockets de C.
- WebWorkers. Hilos de ejecución en paralelo.

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas.

Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java.

Las ventajas de utilizar JavaScript:

JavaScript es una excelente solución para poner en práctica la validación de datos de un formulario en el lado del cliente. Si un usuario omite escribir su nombre en un formulario, una función de validación en JavaScript puede desplegar en pantalla un mensaje popup para hacerle saber al usuario acerca de la omisión. Este tipo de funcionalidades son más ventajosas que tener una rutina de validación del lado del servidor para controlar el error, dado que el servidor en éste caso no tiene que hacer ningún tipo procesamiento de información adicional. Una rutina de ASP o PHP podría ser escrita para lograr la misma tarea pero un formulario desarrollado en JavaScript no permitiría que la información se enviase a menos que se complete correctamente el formulario.

Una de las áreas en la que sobresale radicalmente JavaScript es en la creación de efectos dinámicos tales como imágenes dinámicas y presentaciones de diapositivas, donde su uso se ha convertido algo común hoy en día. Debido a que JavaScript se ejecuta dentro del navegador de los clientes, se puede utilizar para cambiar el aspecto de la pantalla en el dispositivo de los usuarios después que la página ha sido enviada por el servidor. Esto le permite al desarrollador web crear efectos dinámicos muy impresionantes mejorando así la experiencia que recibe un usuario momento de entrar a un sitio web.

Las desventajas de utilizar JavaScript:

La seguridad sigue siendo el talón de Aquiles de Javascript. Los fragmentos de código de JavaScript una vez añadidos a las páginas web en los servidores, estos son descargados y ejecutados en el navegador del cliente permitiendo así que cierto código malicioso pueda ser ejecutado en la máquina del cliente con el objetivo de explotar alguna vulnerabilidad de seguridad conocida en una de las aplicaciones, navegadores o el mismo sistema operativo. Es verdad que hoy día existen estándares de seguridad que restringen la ejecución de código por parte de los navegadores, pero aun así, se puede ejecutar código que dañe, robe o destruya información del lado del cliente.

Otra desventaja de JavaScript es que este tiende a introducir una cantidad enorme de fragmentos de código en nuestros sitios web. Por suerte, el problema de grandes

fragmentos de código JavaScript se resuelve fácilmente mediante el almacenamiento del código JavaScript dentro de archivos separados del código HTML con la extensión. .Js, dejando una página web mucho más limpia y legible de cara al desarrollador.

Debido a la tendencia de JavaScript de acrecentar el código de las páginas web, se hace necesario organizar el código JavaScript en archivos separados al código HTML para que los motores de búsqueda (Google) puedan descifrar fácilmente la calidad del contenido de la página web y esta pueda ser indexada correctamente en los resultados de las búsquedas

CSS

Las hojas de estilo en cascada (Cascading Style Sheets, o sus siglas CSS) hacen referencia a un lenguaje de hojas de estilos usado para describir la presentación semántica (el aspecto y formato) de un documento escrito en lenguaje de marcas. Su aplicación más común es dar estilo a páginas webs escritas en lenguaje HTML y XHTML, pero también puede ser aplicado a cualquier tipo de documentos XML, incluyendo SVG y XUL.

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

CSS se ha creado en varios niveles y perfiles. Cada nivel de CSS se construye sobre el anterior, generalmente añadiendo funciones al previo.
Los perfiles son, generalmente, parte de uno o varios niveles de CSS definidos para un dispositivo o interfaz particular. Actualmente, pueden usarse perfiles para dispositivos móviles, impresoras o televisiones.

Algunas limitaciones que se encuentran en el uso del CSS hasta la versión CSS2.1, vigente, pueden ser:

- Los selectores no pueden usarse en orden ascendente según la jerarquía del DOM (hacia padres u otros ancestros) como se hace mediante XPath
  La razón que se ha usado para justificar esta carencia por parte de la W3C, es para proteger el rendimiento del navegador, que de otra manera, podría verse comprometido. XSLT soporta en la actualidad un mayor número de sistemas operativos. Así mismo, también es mejor para trabajar con la mayoría de buscadores de Internet.
- Dificultad para el alineamiento vertical; así como el centrado horizontal se hace de manera evidente en CSS2.1, el centrado vertical requiere de diferentes reglas en combinaciones no evidentes, o no estándares.
- Las *pseudo-clases* dinámicas (como :hover) no se pueden controlar o deshabilitar desde el navegador, lo que las hace susceptibles de abuso por parte de los diseñadores en banners, o ventana emergentes.

Algunas ventajas de utilizar CSS son:

- Control centralizado de la presentación de un sitio web completo con lo que se agiliza de forma considerable la actualización del mismo.
- Separación del contenido de la presentación, lo que facilita al creador, diseñador, usuario o dispositivo electrónico que muestre la página, la modificación de la visualización del documento sin alterar el contenido del mismo, sólo modificando algunos parámetros del CSS.
- Optimización del ancho de banda de la conexión, pues pueden definirse los mismos estilos para muchos elementos con un sólo selector; o porque un mismo archivo CSS puede servir para una multitud de documentos.
- Mejora en la accesibilidad del documento, pues con el uso del CSS se evitan antiguas prácticas necesarias para el control del diseño (como las tablas), y que iban en perjuicio de ciertos usos de los documentos, por parte de navegadores orientados a personas con algunas limitaciones sensoriales.

JQuery

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Fue presentada el 14 de enero de 2006 en el BarCamp NYC.

jQuery es software libre y de código abierto, posee un doble licenciamiento bajo la Licencia MIT y la Licencia Pública General de GNU v2, permitiendo su uso en proyectos libres yprivativos.1 jQuery, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

2.3 Sistemas de almacenamiento comerciales

Existen numerosos proyectos dedicados al almacenamiento en la nube. A continuación se exponen las características de los proyectos más importantes en el almacenamiento en la nube: Dropbox, SkyDrive y Google Drive.

2.3.1 Dropbox

Dropbox es un servicio de alojamiento de archivos multiplataforma en la nube, operado por la compañía Dropbox. El servicio permite a los usuarios almacenar y sincronizar archivos en línea y entre ordenadores y compartir archivos y carpetas con otros.1 Existen versiones gratuitas y de pago, cada una de las cuales con opciones variadas. Está disponible para Android, Blackberry e IOS (Apple). Dropbox es un software que enlaza todas las computadoras mediante una sola carpeta, lo cual constituye una manera fácil de respaldar y sincronizar los archivos.

Dropbox proporciona 2 GB de espacio de almacenamiento de forma gratuita, previa inscripción. Los usuarios puedes obtener espacio de almacenamiento extra de hasta 10 GB al referirse a nuevos usuarios Dropbox.

Las características más destacadas que ofrece Dropbox:

- Es muy buena para compartir una carpeta a la vez.
- Tiene una aplicación inteligente para subir archivos.
- Galería de fotos online.
- Interfaz limpia.
- Versiones de archivos.
- Tiene un disco duro virtual en el cual se puede trabajar libremente.
- Integración excelente en el sistema operativo.
- Comparte archivos con otra persona sin necesidad que se agregue al grupo.
- Las carpetas compartidas solo puede ver las personas que invites.

Desventajas

- Solo se puede sincronizar lo que se encuentra en la carpeta.
- Cuando se comparte archivos no se puede trabajar en el mismo archivo en el mismo tiempo ya que genera conflictos.
- No tiene cliente para dispositivos excepto el iPhone.

2.3.2 SkyDrive

SKyDrive ofrece un almacenamiento y compartición de archivos en línea en un servicio gratuito. SkyDrive permite editar tus documentos en Microsoft Office en línea, ofrece una versión lite en tres de los software más importantes del grupo Word, PowerPoint y Excel.
El espacio inicial que provee SkyDrive es de 7 GB, que permite por un corto periodo de tiempo aumentarlo a 25 GB de forma gratuita.

Las características más destacadas que ofrece SkyDrive:

- Comparte archivos sin importar tu conexión.
- Fácil de usar.
- Tiene contraseña para tener control de los datos.
- Tiene una capacidad de 25Gb.
- Es gratuito.

Desventajas:

- Ausencia de API

2.3.3 Google Drive

Google Drive es un sistema de almacenamiento en la nube donde se puede cargar, acceder y compartir archivos. Ofrece una capacidad de almacenamiento inicial de 5GB gratuitos. Google Drive puede sincronizar automáticamente los archivos, o es posible subirlos de forma manual.
Google Drive está integrado con Google Docs., esto permite que se pueda crear y colaborar en tiempo real a través de Google Docs.

Las características más destacadas que ofrece Google Drive:

- transfiere los archivos a través de ssl (que es seguro).
- es una remodelación de google docs.
- permite subir archivos de gran tamaño (hasta de 1 giga byte por archivo).
- cuenta con varios servicios como: google+ e incluso con google docs.
- tiene plataformas con mayor compatibilidad con android.

Desventajas

- que solo tiene 5 gigabites de almacenamiento gratis.
- que al usar google drive en dispositivos en algunos casos inicia sesión automáticamente.
- pero la ventaja tres también tiene límite que es de 300 mg.
- es un poco confuso

3. Análisis y Diseño

Para el desarrollo del proyecto es necesario hacer una evaluación inicial de los elementos técnicos necesarios y del tiempo estimado para su realización

Para hacer un buen análisis de estos campos, es necesario conocer cuál será la complejidad del proyecto en términos generales. En este caso se trata de un proyecto sencillo compuesto por un grupo pequeño de desarrolladores. El proyecto se compondrá de las siguientes etapas, en las cuales se profundizará en los apartados posteriores:

Análisis y diseño: Se describe y comprende correctamente el problema y las necesidades que requiere, y se busca la solución a estas necesidades. Mediante la toma de requisitos y los casos de uso se analiza el problema, y posteriormente con el diagrama de funciones se haya una solución. Esta es la labor de un analista de sistemas.

- Implementación: Siguiendo las instrucciones del analista, un programador debe escribir el código que genera la aplicación.

En cuanto a la metodología, se ha escogido utilizar una metodología ágil mezclado con metodología RUP. Aunque existen muchas metodologías que podrían cubrir todas las necesidades de este proyecto, la mayoría de ellas resultan metodologías pensadas para proyectos grandes que harían que la realización del nuestro proyecto se hiciera de una forma mucho más lenta. Unos ejemplos de esto serían RUP o Métricav3.

Las principales ventajas que se tendrán escogiendo una metodología ágil son las siguientes:

- Desarrollo dirigido por casos de uso.
- En vez de ser un proyecto centrado en la documentación, se trata de un proyecto centrado en la funcionalidad del software.
- Permite un desarrollo iterativo e incremental.
- Aunque se ha hecho una valoración de posibles trabajos futuros, el proyecto desarrolla únicamente lo necesario en ese momento.
- Permite ver una versión temprana y funcional de la aplicación.
- Es ideal para proyectos y equipos pequeños y medianos, por el rápido desarrollo de software.

A continuación se muestra la planificación que se llevará a cabo mediante un diagrama de Gantt.

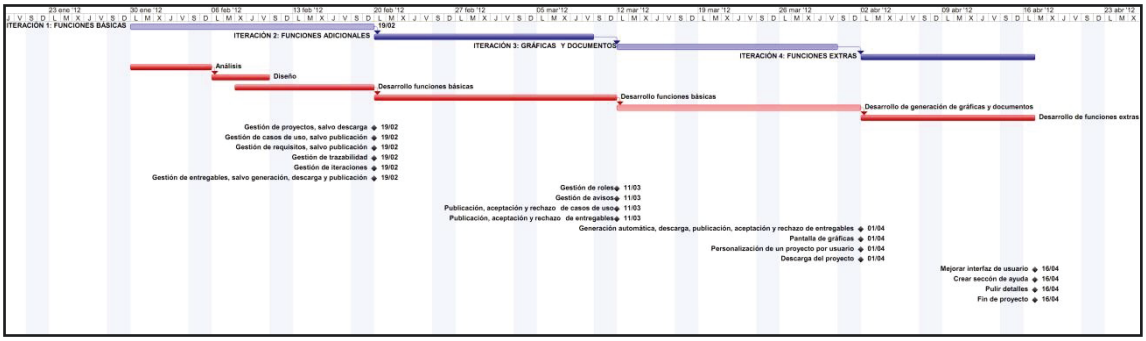

Ilustración 5. Diagrama de Gantt

3.1 Plan de Iteraciones

Como se ha comentado en la metodología, el plan de trabajo será iterativo e incremental. Para que esto pueda ser así, es necesario establecer un plan de iteraciones que trazará la ruta que seguirá el desarrollo del proyecto.

En este caso, el trabajo se va a dividir en tres iteraciones diferentes. Estas iteraciones se componen de las siguientes tareas:

Iteraciones			
Nombre	**Descripción**	**Fecha inicio**	**Fecha fin**
Funciones extras	Mejorar interfaz, pulir detalles. Generación de la ayuda al usuario.	01/04/2013	15/06/2013
Funciones adicionales	Completar condiciones, acciones, interacciones y otras funciones.	01/03/2013	31/03/2013
Funciones básicas	Funcionalidades básicas que permiten hacer uso de la aplicación.	01/01/2013	28/02/2013

Tabla 3. Plan de iteraciones

Funciones básicas			
Descripción:	Funcionalidades básicas que permiten hacer uso de la aplicación: • Adaptación de la base de Datos y de la aplicación existente para el uso de etiquetas. • Inclusión de Etiquetas en la Base de Datos. • Inclusión de extensión como etiqueta por defecto. • Inclusión de uno de los métodos buscadores		
Fecha inicio:	01/01/2013 09:00:00 AM Europa/Madrid	**Fecha fin:**	28/02/2013 11:59:59 PM Europa/Madrid

Tabla 4. Iteración 1: Funciones básicas.

Funciones adicionales			
Descripción:	Completar condiciones, acciones, interacciones y otras funciones: • Inclusión de la nube de etiquetas. • Modificación de buscador existente para permitir el uso de la nube. • Modificación de buscador de texto por desplegable. • Gestión de etiquetas y ficheros eliminados.		
Fecha inicio:	01/03/2013 09:00:00 AM Europa/Madrid	**Fecha fin:**	31/03/2013 11:59:59 PM Europa/Madrid

Tabla 5 Iteración 2: Funciones adicionales.

Funciones extras			
Descripción:	Añadir otras funcionalidades: • Mejorar interfaz de usuario. • Crear un buscador múltiple. • Mejorar tolerancia a fallos. • Pulir detalles.		
Fecha inicio:	01/04/2013 09:00:00 AM Europa/Madrid	**Fecha fin:**	15/06/2013 11:59:59 PM Europa/Madrid

Tabla 6 Iteración 3: Funciones extra.

3.2 Casos de uso

A continuación se muestran las funcionalidades que tendrá la aplicación.

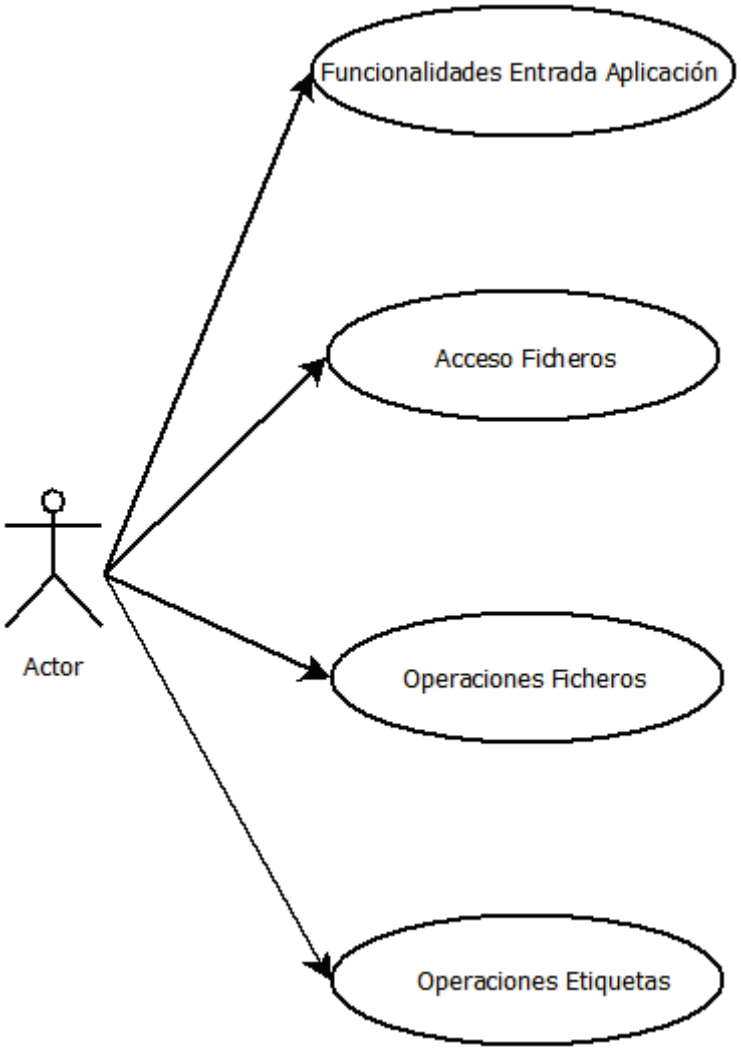

Ilustración 6.Caso de uso Total

Se ha dividido la Aplicación en 4 bloques de Funcionalidades que a continuación se explicarán detalladamente.

3.2.1 Funcionalidades Entrada Aplicación

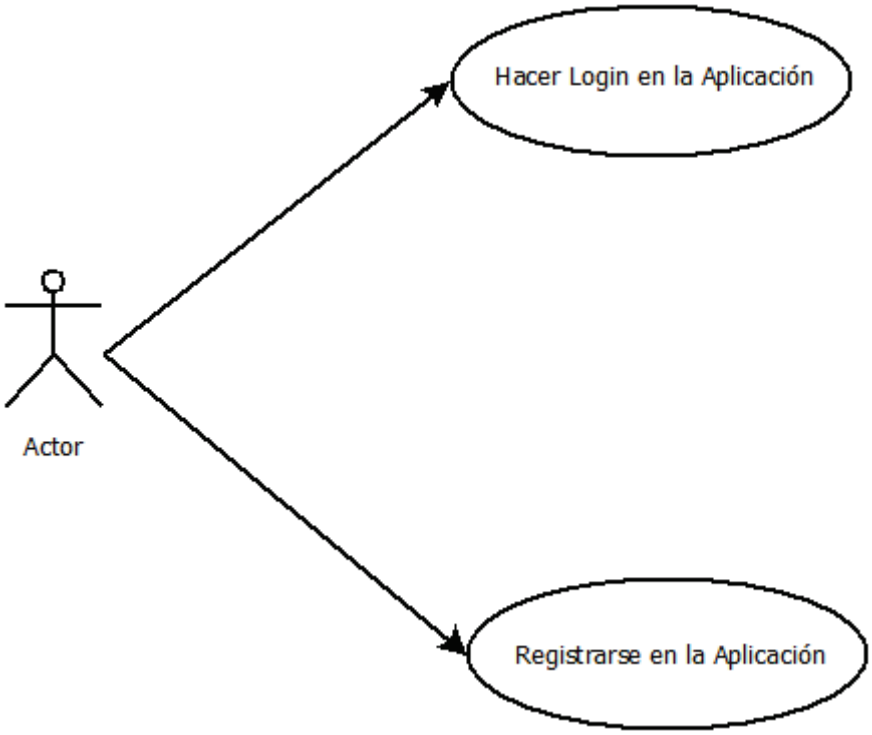

Ilustración 7. Caso de Uso Entrada Aplicación

En una primera instancia, el usuario tendrá que acceder a la aplicación. Para ello deberá registrarse introduciendo sus datos personales, así como una dirección de correo electrónico válida, pues deberá acceder a ella para confirmar el registro.

Una vez el usuario está registrado en la aplicación, deberá introducir el usuario y contraseña elegidos para poder hacer una operación de Login.

3.2.2Acceso Ficheros

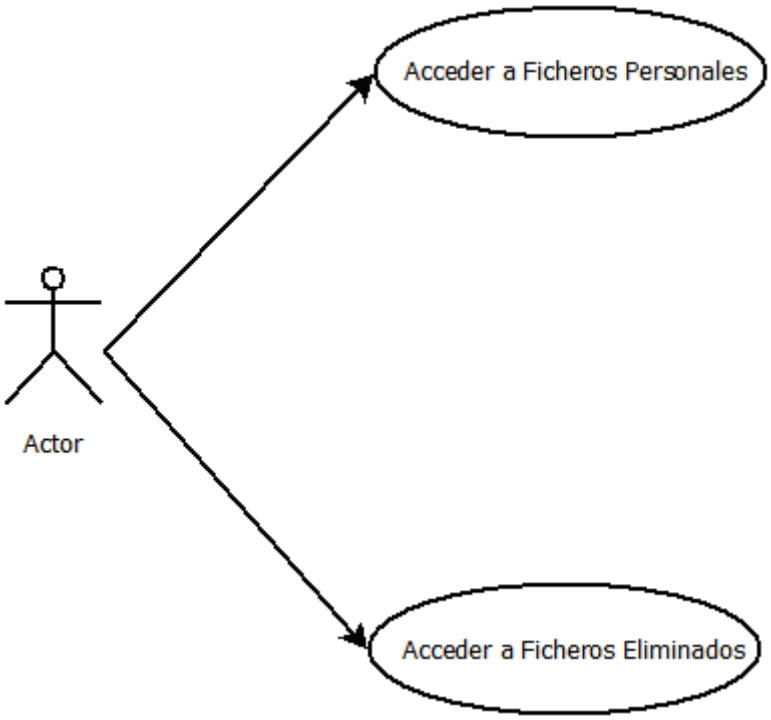

Ilustración 8.Caso de Uso Acceso Ficheros

El usuario tendrá acceso a dos tipos de Ficheros dependiendo del estado de los mismos.

Accediendo al apartado Mis Ficheros, se desplegará una tabla con todos los ficheros que el usuario tiene disponibles, así como información sobre cada uno de ellos, sobre sus etiquetas, tamaño, posibilidad de descarga y un acceso al buscador.

Por otro lado, el usuario podrá acceder a la parte de ficheros eliminados. Estos no aparecerán en el despegable de Mis Ficheros comentado anteriormente. Aquí dispondrá de la misma información que anteriormente pero no podrá descargarlo salvo que realice una recuperación del mismo.

3.2.3 Operaciones Ficheros

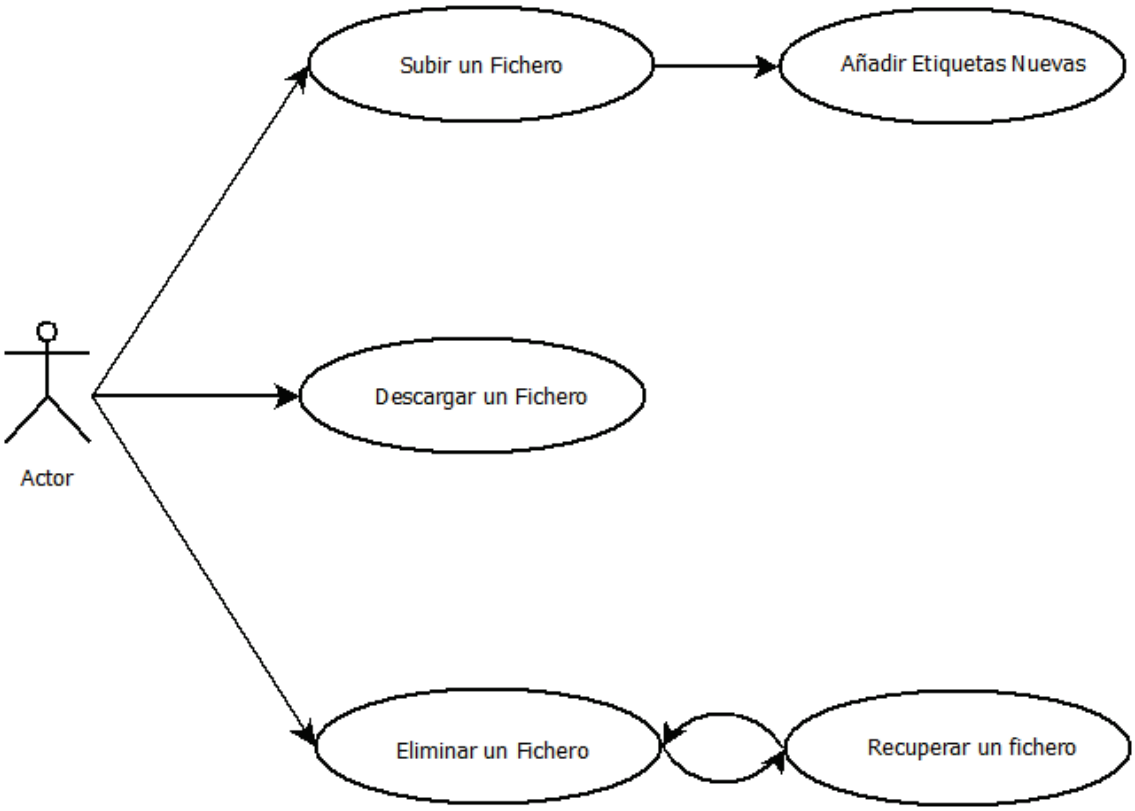

Ilustración 9.Caso de Uso Operaciones Ficheros

El usuario podrá realizar varias operaciones en lo referente a los ficheros.

Podrá realizar subidas de ficheros para que se le muestren en el apartado de Mis Ficheros. Al subir un fichero, el usuario tendrá la opción de añadirle nuevas etiquetas a dicho fichero para su clasificación, siempre y cuando lo considere oportuno.

Por otro lado, el usuario podrá descargar Ficheros que previamente ha subido a la aplicación.

Por último, cualquier fichero disponible puede ser borrado, pasando a pertenecer al conjunto de ficheros eliminados. Este puede ser recuperado posteriormente para que pertenezca al conjunto de ficheros activos.

3.2.4 Operaciones Etiquetas

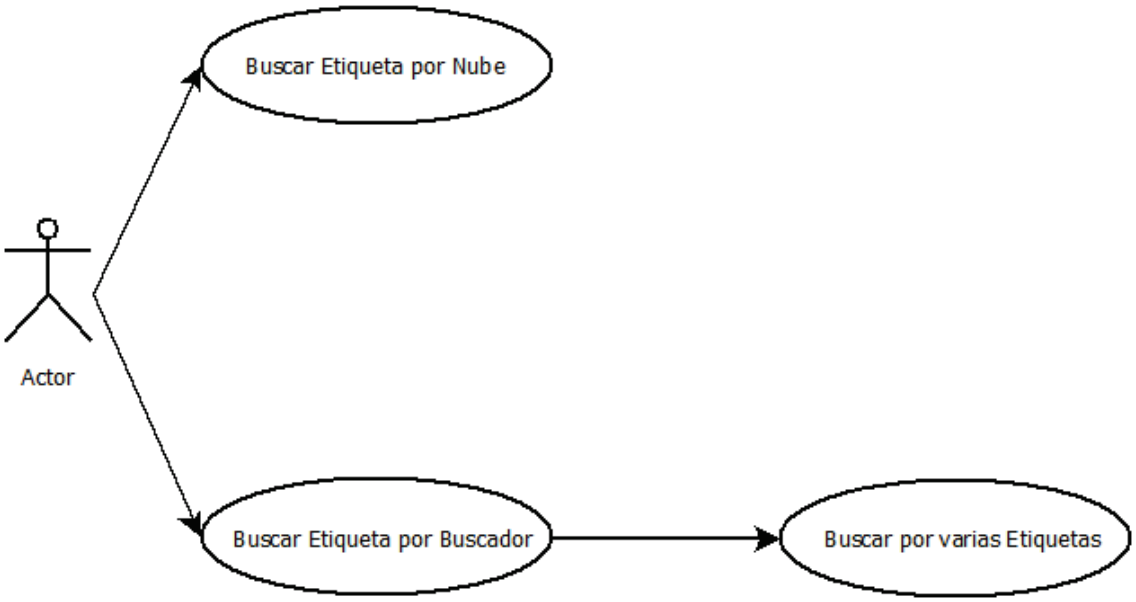

Ilustración 10. Caso de Uso Etiquetas

A la hora de buscar un fichero mediante las etiquetas asociadas, el usuario dispone de dos opciones diferenciadas.

Al hacer click sobre una de las etiquetas de la nube situada en la esquina inferior izquierda, la aplicación mostrará todos los ficheros que tienen esa etiqueta asociada.

Por otro lado, el usuario dispone de un buscador en la parte superior en el cual puede escoger una etiqueta de todas las existentes en la aplicación mediante una lista desplegable. Al escoger una etiqueta, la aplicación mostrará todos los ficheros que tienen esa etiqueta asociada. De forma adicional, si el usuario lo desea, puede escoger nuevas etiquetas de la lista desplegable para realizar una búsqueda por el conjunto de etiquetas escogido.

3.3 Requisitos

Una vez realizado el análisis de casos de uso se obtienen los requisitos del sistema a diseñar. Esta sección recoge los Requisitos de Usuario y los Requisitos de Software.

Los requisitos incluidos en esta sección no incluyen los requisitos utilizados para la aplicación de la cuál parte este proyecto, sólo aquellos que ha sido necesario modificar debido a las especificaciones del proyecto y aquellos nuevos que han surgido para la realización del proyecto.

3.3.1 Requisitos de Usuario

Los requisitos de Usuario definen lo que el usuario será capaz de realizar en el sistema. Para su correcta definición, cada requisito dispondrá de los siguientes campos:

- Nombre: compuesto por las iniciales 'RU' seguidas del número de requisito, y un título breve del mismo.
- Descripción: contiene una explicación más amplia del requisito.
- Tipo:
  - Capacidad: lo que el usuario puede hacer
  - Restricción: lo que el usuario no puede hacer.
- Necesidad: describe si el requisito es esencial u opcional. Valores posibles: Baja, Media y Alta
- Prioridad: describe la prioridad de implementación del requisito. Valores posibles: Baja, Media y Alta
- Verificabilidad: describe en qué medida el requisito es verificable. Valores posibles: Baja, Media y Alta

RU001: Acceso			
Descripción:	El usuario deberá poder acceder a la aplicación vía Web.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 7. RU001

RU002: Inicio de Sesión			
Descripción:	El usuario podrá autenticarse ante el sistema y acceder a la zona de la aplicación para usuarios autenticados.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 8. RU002

RU003: Registrar usuario			
Descripción:	El usuario deberá registrarse en la Aplicación para su posterior acceso		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 9. RU003

RU004:Autenticación			
Descripción:	El usuario no podrá acceder a la Aplicación sin haberse registrado previamente.		
Tipo:	Restricción.		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 10. RU004

RU005: Acceso a Ficheros			
Descripción:	El usuario deberá poder a los ficheros que haya subido previamente.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 11. RU005

RU006: Acceso a Eliminados			
Descripción:	El usuario deberá poder acceder a los ficheros que haya eliminado previamente.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 12. RU006

RU007: Subir Fichero			
Descripción:	El usuario deberá poder subir un Fichero a la Aplicación.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 13. RU007

RU008: Introducir Etiquetas al subir Fichero			
Descripción:	El usuario deberá poder Introducir manualmente a mano las etiquetas que él elija para la descripción y clasificación del Fichero.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 14. RU008

RU009: Eliminar Fichero			
Descripción:	El usuario deberá poder eliminar los ficheros que escoja siempre y cuando los haya subido previamente.		
Tipo:	Capacidad		
Necesidad:	Media	**Prioridad:**	Media
Verificabilidad:	Alta		

Tabla 15. RU009

RU010: Descargar Fichero			
Descripción:	El usuario deberá poder descargarse los ficheros que haya subido previamente.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 16. RU010

RU011: Recuperar Eliminados			
Descripción:	El usuario deberá poder recuperar los ficheros que haya eliminado anteriormente.		
Tipo:	Capacidad		
Necesidad:	Baja	**Prioridad:**	Baja
Verificabilidad:	Alta		

Tabla 17. RU011

RU012: Mostrar Etiquetas			
Descripción:	El usuario deberá poder observar todas las etiquetas que pertenecen a un fichero.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 18. RU012

RU013: Buscar por Nube			
Descripción:	El usuario deberá poder seleccionar una etiqueta de la nube de etiquetas para realizar una búsqueda		
Tipo:	Capacidad		
Necesidad:	Media	**Prioridad:**	Media
Verificabilidad:	Alta		

Tabla 19. RU013

RU014: Buscar por Etiqueta			
Descripción:	El usuario deberá poder escoger una etiqueta del desplegable de etiquetas para realizar una búsqueda		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 20. RU014

RU015: Buscar por etiquetas múltiples			
Descripción:	El usuario deberá poder realizar una búsqueda de un fichero por medio de varias etiquetas simultáneamente.		
Tipo:	Capacidad		
Necesidad:	Baja	**Prioridad:**	Baja
Verificabilidad:	Alta		

Tabla 21. RU015

3.3.2 Matriz de Trazabilidad

	CU4: Operaciones Etiquetas	CU3:Operaciones Ficheros	CU2: Acceso Ficheros	CU1: Funcionalidades Entrada Aplicación
RU001: Acceso				✓
RU002: Inicio de Sesión				✓
RU003: Registrar usuario				✓
RU004:Autenticación				✓
RU005: Acceso a Ficheros			✓	✓
RU006: Acceso a Eliminados			✓	✓
RU007: Subir Fichero		✓		✓
RU008: Introducir Etiquetas al subir Fichero		✓		✓
RU009: Eliminar Fichero		✓		✓
RU010: Descargar Fichero		✓		✓
RU011: Recuperar Eliminados		✓		✓
RU012: Mostrar Etiquetas	✓	✓		✓
RU013: Buscar por Nube	✓			✓
RU014: Buscar por Etiqueta	✓			✓
RU015: Buscar por etiquetas múltiples	✓			✓

Tabla 22. Trazabilidad

3.3.3 Requisitos Software

Los requisitos de software describen la funcionalidad que deberá tener el sistema. Para su correcta definición, los requisitos se componen de los siguientes campos:

- Nombre: compuesto por las iniciales 'RS' seguidas del número de requisito.
- Descripción: contiene una explicación del requisito.
- Tipo: indica el tipo del requisito de software.
  - Funcional: define el comportamiento del software.
  - No funcional: impone restricciones en el diseño o la implementación.

- Necesidad: describe si el requisito es esencial u opcional.
- Prioridad: describe la prioridad de implementación del requisito. Valores posibles: Baja, Media y Alta
- Verificabilidad: describe en qué medida el requisito es verificable. Valores posibles: Baja, Media y Alta

RS01			
Descripción:	El sistema deberá permitir al usuario acceder a la aplicación vía Web, desde cualquier navegador.		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 23. RS01

RS02			
Descripción:	La interfaz del software será Web. .		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 24. RS02

RS03			
Descripción:	El sistema deberá permitir al usuario autenticarse y acceder a la zona de usuarios autenticados.		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 25. RS03

RS04			
Descripción:	El sistema deberá proveer un formulario de autenticación de dos campos: usuario (texto) y contraseña (texto).		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 26. RS04

RS05			
Descripción:	El sistema permitirá a los usuarios registrarse en la aplicación a través de un formulario vía Web.		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 27. RS05

RS06			
Descripción:	La información para registrarse en la aplicación está restringida a los siguientes campos: - Nombre de usuario - Contraseña - Nombre - Apellidos - Correo electrónico - DNI - La superación de un CAPTCHA		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 28. RS06

RS07			
Descripción:	El sistema no permitirá autenticarse a usuarios sin estar registrados		
Tipo:	No Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 29. RS07

RS08			
Descripción:	El sistema permitirá a los usuarios subir ficheros al sistema de almacenamiento en la nube que este proporciona.		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 30. RS08

RS09			
Descripción:	El sistema proporcionará una interfaz gráfica para la subida de ficheros, que permitirá: Seleccionar ficheros para subir Subir ficheros Cancelar seleccionados Borrar ficheros subidos Introducir Etiquetas Manualmente.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 31. RS09

RS10			
Descripción:	El sistema proporcionará una interfaz gráfica para la subida de varios ficheros simultánamente.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 32. RS10

RS11			
Descripción:	El sistema deberá almacenar automáticamente la extensión del fichero como primera etiqueta.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 33. RS11

RS12			
Descripción:	El sistema deberá mantener la relación de los ficheros almacenados con los usuarios y el directorio al que pertenecen.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 34. RS12

RS13			
Descripción:	El sistema deberá mantener la relación de los ficheros almacenados con la extensión del fichero.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 35. RS13

RS14			
Descripción:	El sistema deberá mantener la relación de los ficheros almacenados con las etiquetas introducidas por el usuario.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 36. RS14

RS15			
Descripción:	El sistema se encargará de recopilar la información necesaria para mostrar el estado de la subida de ficheros.		
Tipo:	Capacidad		
Necesidad:	Baja	**Prioridad:**	Baja
Verificabilidad:	Alta		

Tabla 37. RS15

RS16			
Descripción:	El sistema permitirá a los usuarios descargar ficheros asociados a su cuenta.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 38. RS16

RS17			
Descripción:	El sistema permitirá a los usuarios eliminar ficheros que tengan asociados a su cuenta.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 39. RS17

RS18			
Descripción:	El sistema no mostrará los ficheros que no hayan sido subidos por el usuario previamente.		
Tipo:	No Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 40. RS18

RS19			
Descripción:	El sistema deberá marcar el fichero como borrado por el usuario, sin eliminarlo del sistema.		
Tipo:	Capacidad		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 41. RS19

RS20			
Descripción:	El sistema deberá poder listar los ficheros eliminados de un usuario en una interfaz gráfica.		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 42. RS20

RS21			
Descripción:	El sistema no mostrará ningún fichero eliminado que no esté asociado a la cuenta del usuario.		
Tipo:	No Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 43. RS21

RS22			
Descripción:	El sistema deberá permitir a los usuarios restaurar sus ficheros eliminados.		
Tipo:	Funcional		
Necesidad:	Media	**Prioridad:**	Media
Verificabilidad:	Alta		

Tabla 44. RS22

RS23			
Descripción:	El sistema deberá mostrar la ruta en la que se encuentra el usuario.		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 45. RS23

RS24			
Descripción:	El sistema mostrará una nube de elementos que contienen cada una de las etiquetas existentes en la aplicación		
Tipo:	Funcional		
Necesidad:	Media	**Prioridad:**	Media
Verificabilidad:	Alta		

Tabla 46. RS24

RS25			
Descripción:	El sistema permitirá una búsqueda por etiquetas al escoger una etiqueta de la nube		
Tipo:	Funcional		
Necesidad:	Baja	**Prioridad:**	Baja
Verificabilidad:	Alta		

Tabla 47. RS25

RS26			
Descripción:	El sistema sólo mostrará los ficheros pertenecientes al usuario al realizar una búsqueda		
Tipo:	No Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 48. RS26

RS27			
Descripción:	El sistema mostrará un desplegable que incluya todas las etiquetas pertenecientes a archivos del usuario.		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 49. RS27

RS28			
Descripción:	El sistema realizará una búsqueda al seleccionar una etiqueta del desplegable.		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 50. RS28

RS29			
Descripción:	El sistema permitirá la selección de más etiquetas una vez seleccionada la primera para la búsqueda.		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 51. RS29

RS30			
Descripción:	El sistema realizará una búsqueda de los ficheros que incluyan alguna de las etiquetas seleccionadas por el usuario.		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 52. RS30

RS31			
Descripción:	El sistema indicará en todo momento las etiquetas por las cuáles se está realizando la búsqueda.		
Tipo:	Funcional		
Necesidad:	Media	**Prioridad:**	Media
Verificabilidad:	Alta		

Tabla 53. RS31

RS32			
Descripción:	El sistema permitirá el reseteo de etiquetas para la búsqueda.		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 54. RS32

RS33			
Descripción:	El sistema mostrará todas las etiquetas pertenecientes a cada fichero al visualizarlos.		
Tipo:	Funcional		
Necesidad:	Alta	**Prioridad:**	Alta
Verificabilidad:	Alta		

Tabla 55. RS33

Matriz de Trazabilidad. Requisitos de Usuario – Requisitos Software

	RU015: Buscar por etiquetas múltiples	RU014: Buscar por Etiqueta	RU013: Buscar por Nube	RU012: Mostrar Etiquetas	RU011: Recuperar Eliminados	RU010: Descargar Fichero	RU009: Eliminar Fichero	RU008: Introducir Etiquetas al subir Fichero	RU007: Subir Fichero	RU006: Acceso a Eliminados	RU005: Acceso a Ficheros	RU004:Autenticación	RU003: Registrar usuario	RU002: Inicio de Sesión	RU001: Acceso
RS01												✓	✓	✓	✓
RS02												✓	✓	✓	✓
RS03												✓		✓	✓
RS04												✓	✓	✓	✓
RS05													✓		✓
RS06													✓		✓
RS07												✓		✓	✓
RS08									✓						
RS09								✓	✓						
RS10								✓							
RS11								✓							
RS12									✓						
RS13								✓							
RS14								✓							
RS15									✓						
RS16						✓									
RS17							✓								
RS18							✓								

RS19							✓								
RS20							✓								
RS21							✓								
RS22					✓										
RS23										✓	✓				
RS24			✓												
RS25			✓												
RS26	✓	✓	✓												
RS27		✓													
RS28		✓													
RS29	✓														
RS30	✓														
RS31	✓	✓													
RS32	✓	✓	✓												
RS33						✓				✓	✓				

Tabla 56. Trazabilidad RU-RS

3.4 Base de Datos

Para un correcto tratamiento de los datos que se van a almacenar, es necesaria la creación de una base de datos. El proyecto desde el cuál parte este ya dispone de una base de datos que gestionaba los ficheros correspondientes a cada usuario para su posible descarga.

En este apartado se van a comentar las nuevas tablas y funciones que se han tenido que incluir para el correcto funcionamiento del proyecto, así como los cambios que se han producido en las tablas existentes anteriormente para asemejarlas a las nuevas especificaciones.

3.4.1 Creación de tablas

Anteriormente, el proyecto únicamente manejaba ficheros, nombres de ficheros y usuarios. Por lo tanto, para una gestión de etiquetas es necesaria la inclusión de varias tablas.

Etiquetas.

La primera tabla contendría datos de todas las etiquetas pertenecientes a los ficheros que se han subido a la aplicación en la nube.

En un primer lugar, contempló la posibilidad de que cada fichero que se subiese con sus etiquetas, guardara un registro por cada etiqueta que tuviera. De esta forma la relación de las etiquetas con los ficheros se haría de una forma muy simple.

Al implementar esta opción, se observó como la tabla de etiquetas era algo grande y costosa de mantener para el volumen de datos que se tenía. Ya con poco datos se convertía en una tabla muy densa, por lo que se descartó esa opción.

Finalmente, se llegó a la conclusión de que una tabla con un identificador único por cada etiqueta y controlando que las etiquetas no pudiesen repertirse sería lo más adecuado. Esto complicaría la gestión de las etiquetas con los ficheros pero de esta forma la base de datos sería más consistente.

En la versión final de la tabla se guardan los nombres de cada una de las etiquetas así como un identificador que servirá para poder referenciar a estas etiquetas desde otras tablas.

Además, este identificador, al ser único, permitirá que no haya duplicidad de etiquetas ya que, si varios ficheros tienen la misma, podrán hacer referencia a ella sin que esto cause ninguna duplicidad o error en la información del fichero.

Por lo tanto, de esta forma se consigue una base de datos sólida, estable y sobretodo, eficiente.

Ficheros-Etiquetas

Para conseguir una relación entre los ficheros y las etiquetas, inicialmente se pensó en incluir un campo extra en la tabla de los ficheros. Este campo contendría una cadena que incluiría todas las etiquetas pertenecientes a cada fichero en concreto.

Esta opción fue implementada en un inicio pero, al realizar pruebas manejando un número de ficheros elevado, se observó que era un gasto de la capacidad de la base de datos evitable. Con esta opción, todas las etiquetas pertenecientes a un fichero aparecerían al lado de cada fichero en la base de datos, independientemente de si esa etiqueta ya existía o no.

Además, si se quería eliminar una etiqueta de un fichero en concreto, era necesario buscar el fichero en la tabla, seleccionar toda su columna de etiquetas

Como ya se ha comentado, una etiqueta puede pertenecer a varios ficheros. Al haber incluido un identificador único a cada etiqueta, la relación de las etiquetas con los ficheros se podría hacer con una tabla intermedia aprovechando los identificadores de los ficheros (cuyos cambios se comentan en el apartado siguiente).

La tabla final consiste en una dupla de identificadores, una por parte de la etiqueta y otra por parte del fichero al que está relacionado. En el caso de que varios ficheros tengan la misma etiqueta, la columna de la etiqueta hará referencia al mismo identificador pero el fichero será distinto. No podrá haber ninguna dupla repetida.

Cambios

En la versión original del proyecto, el módulo de ficheros no permitía la inclusión de ficheros con el mismo nombre para el mismo usuario.

Se hizo una valoración dadas las nuevas especificaciones y se decidió que un fichero con el mismo nombre podía contener alguna diferencia que llevara a una asociación de etiquetas distinta, por lo que se cambió la base de datos para que esto pudiese ser posible.

3.5 Diseño

En este apartado se va a proceder a la explicación de los nuevos módulos incluidos en la aplicación.

En la aplicación original había módulos de seguridad, de directorios y de ficheros.

Esta memoria se va a centrar en el módulo de etiquetas que es la funcionalidad nueva incluida.

Módulo Etiquetas.

Este módulo provee al sistema de las funcionalidades necesarias para el control y la manipulación de etiquetas.

Las funcionalidades que permite este módulo según lo explicado en los casos de uso y en los requisitos son las siguientes:

- Permitir la asignación de etiquetas a un fichero subido.
- Asignar la extensión del fichero automáticamente como etiqueta.
- Mostrar las etiquetas de un fichero.
- Buscar un fichero mediante sus etiquetas a través de una nube.
- Buscar un fichero mediante sus etiquetas a través de una lista desplegable.
- Buscar un fichero mediante la concatenación de varias etiquetas al mismo tiempo.

Dadas estas funcionalidades, este módulo se dividirá en sub-módulos que permitirán la realización de cada una de estas tareas.

- Módulo de asignación
- Módulo mostrar
- Módulo buscar

Módulo de asignación.

En este módulo mostrará una nueva interfaz al subir un fichero que permitirá al usuario la inclusión de las etiquetas que él elija en el fichero.

Las funciones que llevará a cabo son:

- Mostrar la interfaz de inserción de etiquetas.
- Recoger las etiquetas introducidas por el usuario.
- Asignar la extensión del fichero como etiqueta por defecto.
- Asignar las etiquetas recogidas al fichero.

El siguiente diagrama de flujo muestra el proceso que se lleva a cabo para asignar etiquetas al fichero.

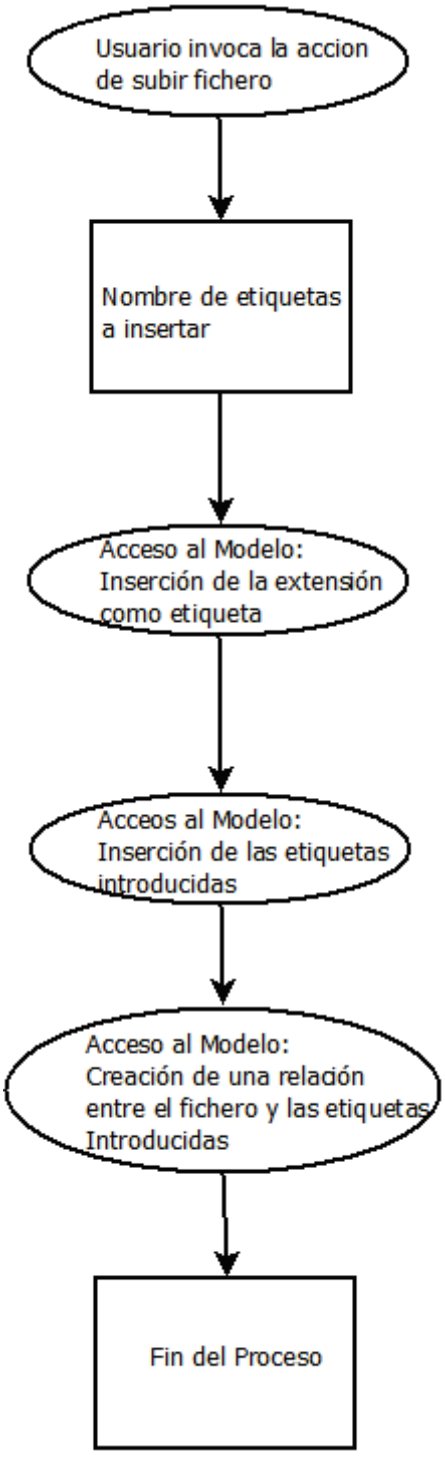

Ilustración 11.Diagrama de Flujo Asignar

Módulo Mostrar

En este módulo se cambiará la interfaz existente para que muestre las etiquetas de los ficheros además de los datos que contenía anteriormente.

Las funciones que llevará a cabo son:

- Mostrar las etiquetas asociadas a cada fichero.

El siguiente diagrama de flujo muestra el proceso para mostrar las etiquetas de un fichero.

Ilustración 12. Diagrama de Flujo Mostrar

Módulo Buscar

En este módulo se cambiará la interfaz existente para la inclusión de dos nuevos componentes: una nube de etiquetas y un desplegable de etiquetas.

Las funciones que llevará a cabo son:

- Mostrar la nube de etiquetas.
- Mostrar el desplegable de etiquetas.
- Mostrar la ruta de la búsqueda actual.
- Buscar por etiqueta de la nube
- Buscar por etiqueta del desplegable.
- Buscar por más de una etiqueta del desplegable.

El siguiente diagrama de flujo muestra el proceso a seguir para realizar la búsqueda de un fichero por etiqueta.

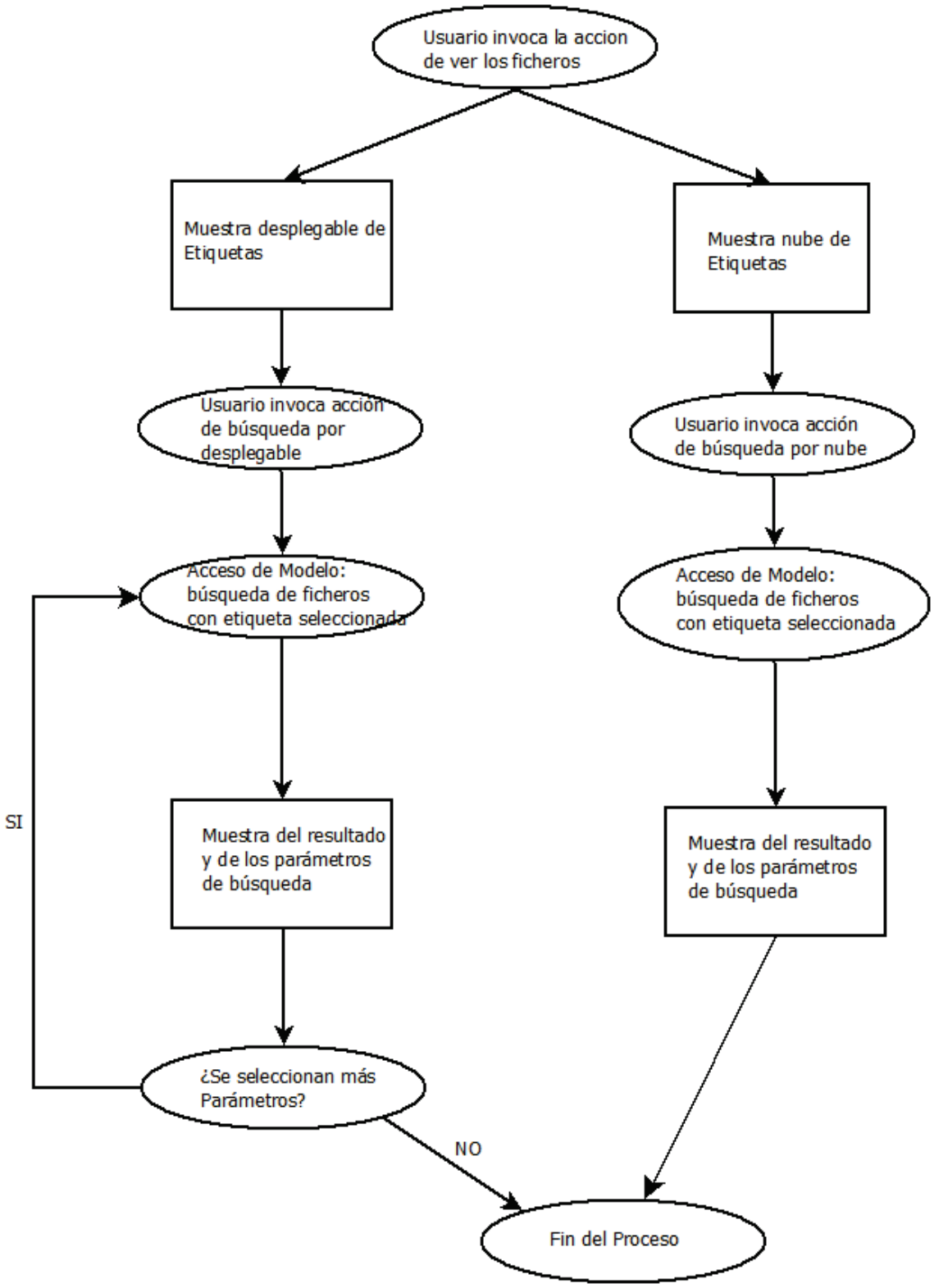

Ilustración 13. Diagrama de Flujo Buscar

3.6 Interfaces.

En este apartado se muestra la relación entre las diferentes interfaces a las que podrá acceder el usuario.

En la siguiente figura se muestra una relación entre las diferentes interfaces relacionadas en función de la navegación del usuario.

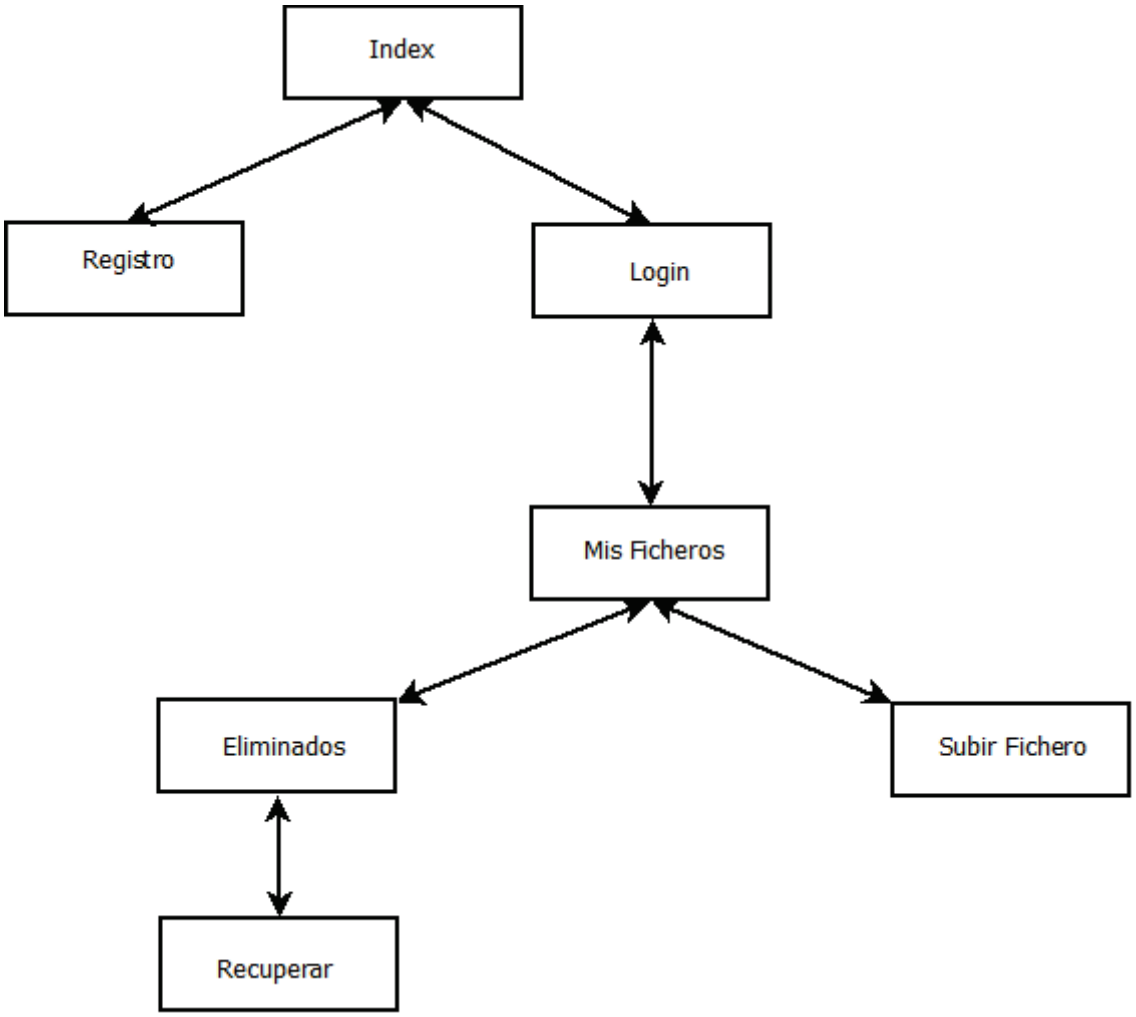

Ilustración 14. Diagrama Relación de Interfaces

3.7 Modelo relacional de la base de datos.

En este apartado se va a explicar la relación entre las nuevas tablas de la base de datos.

El modelo relacional de la base de datos es el siguiente:

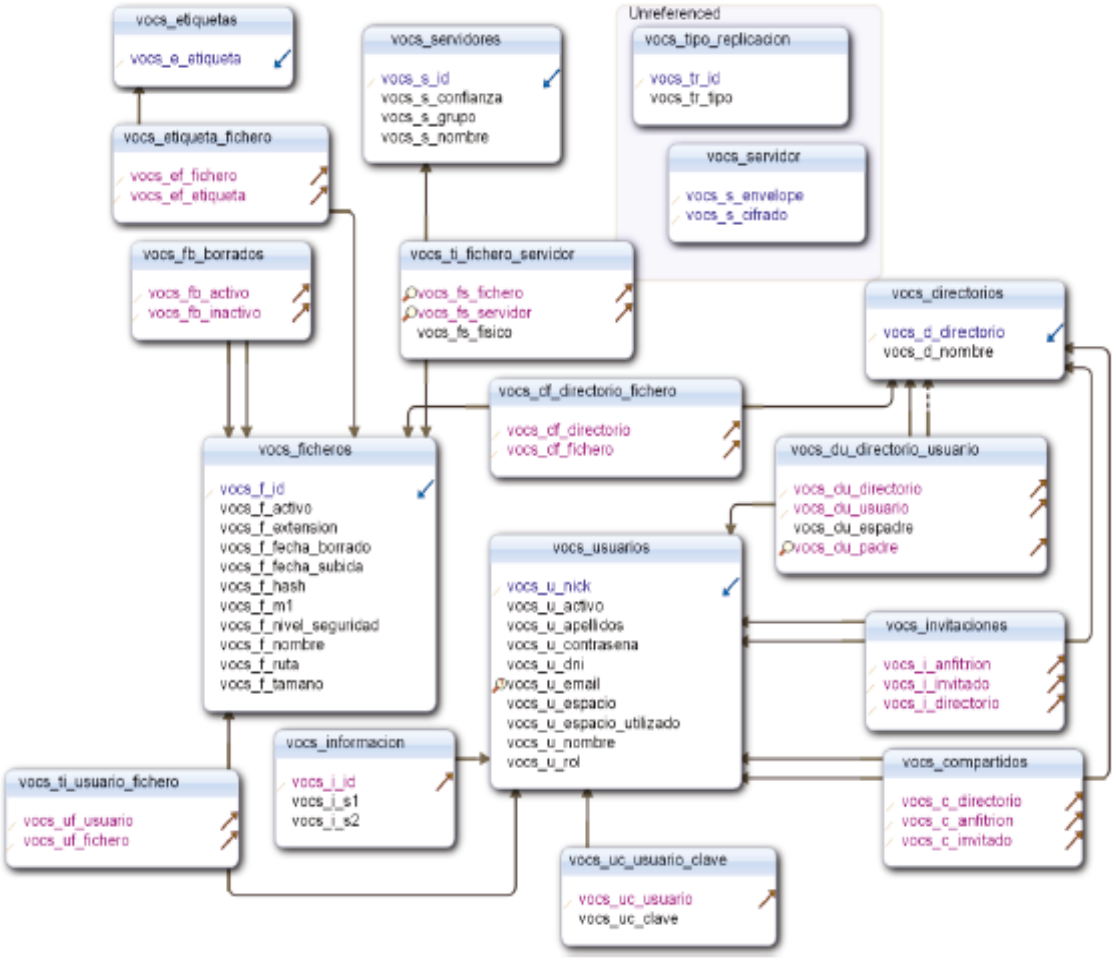

Ilustración 15. Modelo Relacional BBDD

A continuación se van a describir cada una de las tablas de la base de datos:

VOCS_COMPARTIDOS		
vocs_c_directorio	Identificador del directorio.	Entero sin signo.
vocs_c_anfitrion	Identificador del usuario anfitrión.	Cadena de caracteres longitud máxima 255.
vocs_c_invitado	Identificador del usuario invitado.	Cadena de caracteres longitud máxima 255.
Claves		
Clave primaria	(vocs_c_directorio, vocs_c_anfitrion, vocs_c_invitado)	
Claves ajenas		
anfitrion	(vocs_c_anfitrion) referencia vocs_usuarios (vocs_u_nick)	
directorio	(vocs_c_directorio) referencia vocs_directorios(vocs_d_directorio)	
invitado	(vocs_c_invitado) referencia vocs_usuarios (vocs_u_nick)	

Tabla 57. Tabla Compartidos

VOCS_DF_DIRECTORIO_FICHERO		
vocs_df_directorio	Identificador del directorio que contiene un fichero.	Entero sin signo.
vocs_df_fichero	Identificador del fichero que está contenido en un directorio.	Entero sin signo.
Claves		
Clave primaria	(vocs_df_directorio, vocs_df_fichero)	
Claves ajenas		
directorio	(vocs_df_directorio) referencia vocs_directorios (vocs_d_directorio)	
fichero	(vocs_df_fichero) referencia vocs_ficheros (vocs_f_id)	

Tabla 58. Tabla Directorio Fichero

VOCS_DIRECTORIOS		
vocs_d_directorio	Identificador de la entidad	Entero sin signo autoincremental.
vocs_d_nombre	Nombre del directorio.	Cadena de caracteres longitud máxima 255.
Claves		
Clave primaria	(vocs_d_directorio)	

Tabla 59. Tabla Directorios

VOCS_DU_DIRECTORIO_USUARIO		
vocs_du_directorio	Identificador del directorio que posee el usuario.	Entero sin signo.
vocs_du_usuario	Identificador del usuario que posee un directorio.	Cadena de caracteres longitud máxima 255.
vocs_du_espadre	BIT que indica si el directorio que posee el usuario es padre o no. 0=> no padre, 1=> padre.	Booleano.
vocs_du_padre	Identificador del directorio padre.	Entero sin signo.
Claves		
Clave primaria	(vocs_du_directorio, vocs_du_usuario)	
Claves ajenas		
directorio	(vocs_du_directorio) referencia vocs_directorios (vocs_d_directorio)	
du_padre	(vocs_du_padre) referencia vocs_directorios (vocs_d_directorio)	
du_usuario	(vocs_du_usuario) referencia vocs_usuarios (vocs_u_nick)	

Tabla 60. Tabla Directorio Usuario

VOCS_ETIQUETA_FICHERO		
vocs_ef_fichero	Identificador del fichero que tiene una etiqueta.	Entero sin signo.
vocs_ef_etiqueta	Identificador de la etiqueta del fichero.	Cadena de caracteres longitud máxima 255.
Claves		
Clave primaria	(vocs_ef_fichero, vocs_ef_etiqueta)	
Claves ajenas		
etiqueta	(vocs_ef_etiqueta) referencia vocs_etiquetas (vocs_e_etiqueta)	
fichero	(vocs_ef_fichero) referencia vocs_ficheros (vocs_f_id)	

Tabla 61. Tabla Etiqueta Fichero

VOCS_ETIQUETAS		
Vocs_e_id	Identificador de la entidad	Entero sin signo
vocs_e_etiqueta	Descripción de la entidad	Cadena de caracteres longitud máxima 255.
Claves		
Clave primaria	(vocs_e_id)	

Tabla 62. Tabla Etiquetas

VOCS_FB_BORRADOS		
vocs_fb_activo	Identificador del fichero que es primera versión.	Entero sin signo.
vocs_fb_inactivo	Identificador del fichero que es versión antigua.	Entero sin signo.
Claves		
Clave primaria	(vocs_fb_activo, vocs_fb_inactivo)	
Claves ajenas		
fb_activo	(vocs_fb_activo) referencia vocs_ficheros (vocs_f_id)	
fb_inactivo	(vocs_fb_inactivo) referencia vocs_ficheros (vocs_f_id)	

Tabla 63. Tabla Borrados

VOCS_FICHEROS		
vocs_f_id	Identificador del fichero que es primera versión.	Entero sin signo autoincremental.
vocs_f_activo	Indica si el fichero ha sido borrado o no. 0 indica borrado, 1 indica activo.	Booleano.
vocs_f_extension	Extensión del fichero	Cadena de caracteres longitud máxima 9.
vocs_f_fecha_borrado	Fecha de borrado del fichero. NULL si nunca fue borrado.	Fecha
vocs_f_fecha_subida	Fecha de subida.	Fecha
vocs_f_hash	Hash del contenido del fichero.	Cadena de caracteres longitud máxima 255.
vocs_f_nivel_seguridad	Nivel de seguridad aplicado al fichero.	Entero
vocs_f_nombre	Nombre del fichero.	Cadena de caracteres longitud máxima 255.
vocs_f_ruta	Ruta física del fichero.	Cadena de caracteres

		longitud máxima 512.
vocs_f_tamano	Tamaño en Bytes del fichero.	Número en coma flotante.
vocs_f_m1	Cifrado de la primera mitad de la clave de cifrado del usuario	Cadena de caracteres longitud máxima 255.
Claves		
Clave primaria	(vocs_f_id)	

Tabla 64. Tabla Ficheros

VOCS_INFORMACION		
vocs_i_id	Identificador del usuario.	Cadena de caracteres longitud máxima 255.
vocs_i_s1	Salt para la generación de la cadena de autenticación (comparación).	Cadena de caracteres longitud máxima 255.
vocs_i_s2	Salt para la generación de segunda mitad de la clave de cifrado	Cadena de caracteres longitud máxima 255.
Claves		
Clave primaria	(vocs_i_id)	
Claves ajenas		
usuario	(vocs_i_id) referencia vocs_usuarios (vocs_u_nick)	

Tabla 65. Tabla Información

VOCS_INVITACIONES		
vocs_i_anfitrion	Identificador del usuario anfitrión de la invitación.	Cadena de caracteres longitud máxima 255.
vocs_i_invitado	Identificador del usuario invitado.	Cadena de caracteres longitud máxima 255.
vocs_i_directorio	Identificador del directorio al que se invita.	Entero sin signo.
Claves		
Clave primaria	(vocs_i_anfitrion, vocs_i_invitado, vocs_i_directorio)	
Claves ajenas		
anfitrion	(vocs_i_anfitrion) referencia vocs_usuarios (vocs_u_nick)	
directorio	(vocs_i_directorio) referencia vocs_directorios (vocs_d_directorio)	
invitado	(vocs_i_invitado) referencia vocs_usuarios (vocs_u_nick)	

Tabla 66. Tabla Invitaciones

VOCS_SERVIDOR		
vocs_s_envelope	Claves de envoltura para el cifrado y descifrado de la cadena de entrada utilizada en la generación de la segunda mitad de la clave de cifrado.	Cadena de caracteres longitud máxima 255.
vocs_s_cifrado	Cifrado de la cadena de entrada utilizada en la generación de la segunda mitad de la clave de cifrado	Cadena de caracteres longitud máxima 255.
Claves		
Clave primaria	(vocs_s_envelope, vocs_s_cifrado)	

Tabla 67. Tabla Servidor

VOCS_SERVIDORES		
vocs_s_id	Identificador del servidor.	Entero sin signo autoincremental.
vocs_s_confianza	Columna que indica si el servidor es de confianza.	Booleano
vocs_s_grupo	Grupo de servidores al que pertenece.	Entero
vocs_s_nombre	Nombre ó IP del servidor.	Cadena de caracteres longitud máxima 255.
Claves		
Clave primaria	(vocs_s_id)	

Tabla 68. Tabla Servidores

VOCS_TI_FICHERO_SERVIDOR		
vocs_fs_fichero	Identificador del fichero.	Entero sin signo.
vocs_fs_servidor	Identificador del servidor donde está alojado el fichero.	Entero sin signo
vocs_fs_fisico	Nombre físico del fichero.	Cadena de caracteres longitud máxima 512.
Claves		
Clave primaria	(vocs_fs_fichero, vocs_fs_servidor)	
Claves ajenas		
fichero	(vocs_fs_fichero) referencia vocs_ficheros (vocs_f_id)	
servidor	(vocs_fs_servidor) referencia vocs_servidores (vocs_s_id)	

Tabla 69. Tabla Fichero Servidor

VOCS_TI_USUARIO_FICHERO		
vocs_uf_usuario	Identificador de usuario.	Cadena de caracteres longitud máxima 255.
vocs_uf_fichero	Identificador del fichero perteneciente al usuario.	Entero sin signo.
Claves		
Clave primaria	(vocs_uf_usuario, vocs_uf_fichero)	
Claves ajenas		
fichero	(vocs_uf_fichero) referencia vocs_ficheros (vocs_f_id)	
usuario	(vocs_uf_usuario) referencia vocs_usuarios (vocs_u_nick)	

Tabla 70. Tabla Usuario Fichero

VOCS_TIPO_REPLICACION		
vocs_tr_id	Identificador de replicación.	Entero sin signo.
vocs_tr_tipo	Indica el tipo de replicación del sistema. 0=> Replicacion VoCS. 1=> Anillo	Entero sin signo.
Claves		
Clave primaria	(vocs_tr_id)	

Tabla 71. Tabla Tipo Replicación

VOCS_UC_USUARIO_CLAVE		
vocs_uc_usuario	Identificador del usuario.	Cadena de caracteres longitud máxima 255.
vocs_uc_clave	Clave de confirmación de registro de usuario.	Cadena de caracteres longitud máxima 255.
Claves		
Clave primaria	(vocs_uc_usuario)	
Claves ajenas		
usuario	(vocs_uc_usuario) referencia vocs_usuarios (vocs_u_nick)	

Tabla 72. Tabla Usuario Clave

VOCS_USUARIOS		
vocs_u_nick	Identificador del usuario	Cadena de caracteres longitud máxima 255.
vocs_u_activo	Marca si el usuario está activo o inactivo. 0=inactivo, 1=activo	Entero 1 bit.
vocs_u_apellidos	Apellidos del usuario	Cadena de caracteres longitud máxima 50.
vocs_u_contrasena	Contraseña del usuario	Cadena de caracteres longitud máxima 255.
vocs_u_dni	DNI del usuario	Cadena de caracteres longitud máxima 9.
vocs_u_email	Correo electrónico del usuario	Cadena de caracteres longitud máxima 255.
vocs_u_espacio	Espacio Total del que dispone el usuario en MB	Número en coma flotante.
vocs_u_espacio_utilizado	Espacio utilizado en MB	Número en coma flotante.
vocs_u_nombre	Nombre	Cadena de caracteres longitud máxima 45
vocs_u_rol	Rol del usuario.	Entero
Claves		
Clave primaria	(vocs_u_nick)	

Tabla 73. Tabla Usuarios

3.8 Opciones de implementación

En este apartado se justificará la elección de las tecnologías utilizadas.

3.8.1 Lenguaje de lado del servidor

Para la elección del lenguaje de programación se tendrá en cuenta el impacto en el proyecto y el coste económico.

El objetivo de este proyecto es ampliar las funcionalidades de una aplicación sin que esta deje de ser una aplicación de código abierto. Por otro lado se ha escogido el patrón MVC dado que es la arquitectura del proyecto de origen. Teniendo esto en cuenta, y con lo explicado sobre tecnologías en el punto 2.2, se ha escogido PHP por los siguientes motivos:

- PHP no requiere entornos de desarrollo específicos.
- Es un lenguaje con licencia PHP, de uso libre y con una comunidad muy activa. Por lo que, el aprendizaje es rápido, el soporte es bueno y el coste es nulo.
- Hay frameworks desarrollados para el lenguaje PHP de licencia libre, que implementan el patrón MVC, tal y como es la arquitectura del sistema.
- PHP permite el manejo de bases de datos relacionales de forma rápida y sencilla.
- El equipo de desarrollo cuenta con experiencia en proyectos similares con el lenguaje.

Dentro de PHP y con la inclusión de frameworks, se ha elegido continuar con ZEND Framework como framework de PHP. Este framework era el utilizado en la proyecto de origen por lo que continuar con él tendría un gran impacto en el tiempo de adaptación del código.

3.8.2 Lenguajes del lado del cliente

El lenguaje del lado del cliente escogido es JavaScript que es el lenguaje más común, y sobre el que más información y soporte hay. Dentro de JavaScript, el framework JQuery es de los más populares y ofrece soporte y facilidad a la hora de interactuar con las opciones del cliente.

Para la parte gráfica se ha decidido utiliza CSS , HTML4 y HTML5 como componentes fundamentales de las interfaces.

3.8.3 Base de datos

Antes de escoger la base de datos, hay que tener en cuenta que ya se ha escogido PHP como lenguaje de programación.

MySQL es un gestor de base de datos muy completo y que tiene una gran complicidad con PHP, por lo que fue la primera opción a contemplar en el desarrollo de este nuevo proyecto.

Por otro lado, MySQL también era el Sistema Gestor de Base de Datos Utilizado por el proyecto origen, por lo que escogerlo supondría un ahorro en el tiempo de creación de una nueva base de datos desde el inicio.

3.8.4 Servidor

El servidor escogido ha sido determinado por el anterior proyecto. Un cambio de servidor implicaría un cambio integral, por lo que este proyecto pasaría de ser una ampliación de funcionalidades de un proyecto existente a ser la creación de un proyecto completo desde el principio.

3.8.5 LAMP

Teniendo en cuenta que las tecnologías utilizadas principalmente son PHP, Apache, MySQL y Linux como Sistema Operativo, es obligatorio hablar de LAMP. LAMP es una plataforma de desarrollo para aplicaciones WEB que consta de estos 4 elementos ya citados. Por lo cual será de gran ayuda y se utilizará como plataforma para la continuación del desarrollo del proyecto.

4. Desarrollo

En este apartado se va a proceder a la explicación de los mecanismos utilizados para lograr el correcto funcionamiento de la aplicación.

Anteriormente en el bloque de Diagrama de Flujos, se explicó que el único bloque nuevo de la aplicación era el módulo de etiquetas. En esta sección se va a explicar detalladamente dicho bloque pero, además de eso, se explicarán las diferentes modificaciones que se han debido realizar en otros bloques distintos pertenecientes al proyecto de origen para que la aplicación sea un éxito.

4.1 Cambios Producidos

En primer lugar, hubo que realizar cambios en el módulo de Login. Este módulo es el responsable de la conexión entre el usuario y la aplicación. Este módulo funcionaba correctamente, pero con algunos cambios en la aplicación, provocaba algunos problemas de conexión, dando error al usuario a la hora de logearse en la aplicación.

En la aplicación se cambió la página de inicio que ve el usuario al logearse. Ahora la página de inicio se llama “index/index”. Debido a esto, se cambió la redirección de la función de login para que apuntase a dicha página.

```
/* Initialize action controller here */
        $auth = Zend_Auth::getInstance();
           if (! $auth->hasIdentity()) {
                  return $this->_redirect('index/index');
           }
```

Submódulo Asignación

La primera tarea que debía realizarse era la de asignar etiquetas a un fichero.

Dentro de esta funcionalidad, el primer paso fue asignar por defecto la extensión de un fichero como primera etiqueta. De este modo ningún fichero se quedaría sin etiquetas aunque el usuario no quisiera asignarle ninguna personalmente.

Para ello, se habría de modificar la clase UploadController.php.

UploadController.php

Esta clase es la encargada de permitir a un usuario elegir un fichero de su ordenador y subirlo a la nube.

En esta clase se recogen los datos del fichero para posteriormente insertarlos en la base de datos.

Aprovechando dicha recogida de datos, cuando se obtiene el nombre completo del fichero, se invoca una llamada a un método de php que puede separar cadenas de texto.

Gracias a este método se separa la cadena de texto en nombre de fichero y extensión del mismo.

```
//separar nombre de la extension
                $snom = explode(".",$info['name']);
                if(count($snom)<2){
                        $nom = $snom[0];
                        $extension = null;
                }else{
                        //obtener extension
                        $extension = $snom[count($snom)-1];
                        //obtener el nombre sin la extension
                        $nom = "";
                        for ($j=0;$j<count($snom)-1;$j++){
                                $nom .= $snom[$j];
                        }
                }
```

Como se puede observar en el código insertado, la obtención de la extensión está controlada para evitar errores. En el caso de que un fichero se suba sin extensión, la aplicación no realizará la gestión de las etiquetas para esa subida, evitando así que la aplicación deje de funcionar.

Este tipo de controles se han incluido a lo largo de toda la aplicación para conseguir una aplicación con una tolerancia a fallos muy alta. A lo largo de la descripción del código se irán explicando cuales son estos controles introducidos.

Una vez que se ha obtenido la extensión es necesario guardarla en una base de datos.

El primer paso para ello es crear dos tablas nuevas en la base de datos que se encarguen de la gestión de las etiquetas: vocs_e_etiquetas y vocs_ef_etiqueta_fichero. Los detalles de estas tablas están recogidos en el apartado de Modelo Relacional de Base de Datos.

Una vez las tablas han sido creadas, se realiza una comprobación de la tabla de etiquetas de la base de datos. Si la etiqueta que se está intentando insertar ya existe en dicha tabla, se realiza una búsqueda de la etiqueta para obtener su identificador. En el caso de que la etiqueta no exista, se inserta la etiqueta en la tabla vocs_e_etiquetas asignándole automáticamente un identificador único.

De esta forma, las etiquetas nunca se podrán duplicar consiguiendo así el objetivo de una aplicación con un rendimiento alto.

Para que una etiqueta pueda ser insertada en una nueva tabla, es necesaria la creación de un nuevo modelo dentro de nuestro código. Este modelo es Etiquetas.php

Etiquetas.php

Este modelo es el encargado de manejar las etiquetas dentro de la base de datos, permitiendo el acceso a ellas desde la aplicación.

Este modelo tendrá los métodos de insertar una nueva etiqueta, de buscar las etiquetas ya existentes tanto por nombre como por id y borrar una etiqueta existente.

Para la funcionalidad que se está llevando a cabo en este momento, las funciones que se utilizarán del módulo son la de buscar las etiquetas existentes por el nombre (que es el de la extensión del fichero) recogiendo el id en el caso de que exista y la función de insertar una nueva etiqueta en la tabla en caso necesario.

En el caso de que alguno de los métodos no devuelva el resultado esperado, se han insertado controles dependiendo del resultado de vuelta de las funciones del modelo.

En el caso de que alguno de los métodos devuelva un valor "-1" el controlador "UploadController" lo interpretará como un error y realizará las funciones siguientes con normalidad pero evitando el control de estas etiquetas. De este modo si el fichero no tuviese extensión como ya se ha comentado, el fichero se subiría igualmente pero sin asignarle ninguna etiqueta.

Una vez que la etiqueta ha sido insertada o se capturado su identificador, es necesario relacionarla con el fichero.

Para poder hacer esa relación es necesaria la nueva tabla creada anteriormente vocs_ef_etiqueta_fichero y la creación de un nuevo módulo que permita incluir dicha relación.

Etiqueta_Fichero

Este módulo es similar al anterior creado pero con la diferencia de que la tabla únicamente contendrá datos de dos identificadores: vocs_ef_etiqueta y vocs_ef_fichero.

Debido a esto, los métodos que contendrá serán la búsqueda de identificadores tanto de etiqueta como de fichero, comprobar la existencia de alguno de ellos, obtener los identificadores relacionados con alguno de ellos, borrar una relación y la inserción de una nueva relación indicándole los dos identificadores.

Dado que el fichero que se vaya a insertar nunca puede existir en la base de datos previamente, el identificador siempre será único por lo que no es necesaria ninguna comprobación previa de si se puede insertar un duplicado.

Que un fichero no pueda repetirse es una de las modificaciones que se ha optado por realizar. Anteriormente un fichero podía subirse nuevamente y se producía un reemplazo. Después de analizarlo, se decidió que un fichero podría subirse nuevamente con el mismo nombre y que no contuviese la misma información, por lo que el usuario podría asignarle nuevas etiquetas diferentes a la versión anterior.

Cuando el fichero ya se ha insertado, se conoce su identificador único y, tras la búsqueda o inserción de la etiqueta en la base de datos, también se conoce su identificador.

Por lo tanto, el siguiente paso sería realizar una inserción en la base de datos de la nueva relación indicando los dos identificadores obtenidos.

Una vez todos los ficheros tienen sus extensiones como etiquetas, el siguiente paso es permitir al usuario introducir a cada fichero las etiquetas que él desee.

Para ello se debe de modificar la interfaz de subida de ficheros incluyendo una caja de texto en la que el usuario podrá introducir, separadas por comas, las etiquetas que el desee.

Para efectuar este cambio se ha modificado el layout de subida para incluir dicha modificación. Este layout está en index.phtml.

```
<td class="Etiqueta01">
                                    <input  type ="text" name="etiquetasSubir" id="etiquetasSubir"
style="margin-left:0px; width: 125px;">

                                    </input></td>
```

Una vez el usuario ha introducido las etiquetas que desea incluir, la aplicación deberá obtener dichas etiquetas para poder incluirlas.

El proceso a seguir es similar al que se produce al introducir la extensión pero con una serie de diferencias que van a ser explicadas ahora.

En primer lugar, es necesario modificar el método POST creado con anterioridad que se utilizaba para el paso del fichero y que se aprovechó para la extensión.

Es necesario crear otro método nuevo que incluya estas etiquetas para que posteriormente puedan ser tratadas de forma correcta.

```
$fichid = new Application_Model_Ficheros();

                    //Gestion de etiquetas adicionales
                    $etis = $_POST['etiquetasSubir'];

                    $plainMsg = "Etiqueta: ".$_POST['etiquetasSubir'];
                    $logger = Zend_Registry::getInstance()->logger;
                    $logger->log($plainMsg, Zend_Log::INFO);
                    if($_POST['etiquetasSubir'] != NULL){
$array_etiquetas = explode(",", $_POST['etiquetasSubir']);
```

Una vez que se tienen las etiquetas, como se puede observar en el código se realiza una separación de las mismas utilizando las comas que tienen entre medias.

Cuando las etiquetas se han separado se procede de la misma forma que anteriormente. Si la etiqueta existe en la base de datos se obtiene su identificador único y después se inserta la relación entre el fichero y la etiqueta. En el caso de que no exista, se crea una nueva etiqueta con su identificador y se inserta la relación.

Para evitar fallos por parte del Usuario y hacer la aplicación tolerante a fallos, se han incluido varias comprobaciones a la hora de evaluar la cadena de etiquetas.

Dado que es una cadena que viene separada por comas, se pueden dar casos en los que el usuario deje espacios entre coma y coma sin insertar etiquetas, que haya diferencias entre mayúsculas y minúsculas etc. Por lo que antes de insertar una etiqueta se realizan comprobaciones que evitan este tipo de errores.

El sistema no hace diferencia entre mayúsculas y minúsculas a la hora de insertar y además, se realizan compresiones de la cadena para quitar los espacios. De esta forma, si el usuario introduce alguna etiqueta vacía, el sistema la obviaría y pasaría a la siguiente.

```
if($trim  != NULL){

$buscareti = $etiquet->existeEtiqueta($trim);
```

Además, y como medida de seguridad, se ha decidido que las etiquetas únicamente pueden estar compuestas por caracteres alfanuméricos. Así se evitan algunos errores relacionados con la base de datos al introducir caracteres no alfanuméricos.

```
if(ctype_alpha($trim){
$buscareti = $etiquet->existeEtiqueta($trim);
```

Con todo esto, la subida de nuevas etiquetas está completamente protegida frente a una inserción de etiquetas maliciosa o errónea por parte del usuario.

Una vez que todos los ficheros tienen todas las etiquetas que el usuario desea, es necesario poder buscar los ficheros por las etiquetas que tienen relacionadas.

Para esto se han incluido dos métodos que el usuario puede utilizar de una forma sencilla e intuitiva.

En primer lugar se ha introducido una nube giratoria de etiquetas en la aplicación. Esta nube contiene todas y cada una de las etiquetas del sistema y, al hacer click en una de ellas, el sistema realiza una búsqueda de todos los ficheros que contienen esa etiqueta.

Para poder realizar esto han sido necesarias la inclusión de varias modificaciones.

La primera modificación es la inserción de la propia nube. Para ello se han creado las clases Canvas.

Canvas

Estas clases están compuestas por varios javaScript que almacenan los elementos que se les indica y los muestran de una forma esférica.

Por otro lado, para que la aplicación web pueda mostrar este tipo de elementos móviles, ha sido necesaria la utilización de HMTL5.

HMTL5 provee a las aplicaciones web la inclusión de elementos de vídeo o en movimiento en el tiempo como la nube que está incluida en la aplicación.

```
if( ! $('#myCanvas').tagcanvas({

    textColour : '#000000',
    outlineThickness : 1,
    maxSpeed : 0.10,
    depth : 0.75,
    shape : 'sphere',
    dragControl : 'true',
    textHeight: 20
    //lock : 'x'
  },'')){
              // TagCanvas failed to load
              $('#myCanvasContainer').hide();
            }
```

Una vez se tiene la nube en funcionamiento, es necesario indicarle qué elementos tiene que mostrar en la nube.

Para ello se ha de modificar el layout nuevamente.

Layout

En este caso, la clase que va a contener el método que permita la inclusión de elementos va a ser la propia interfaz. Esto es debido a que cuando la interfaz vaya a mostrar la nube de elementos, es necesario que ya tenga todos los datos necesarios para hacerlo de una forma completa.

Lo que se desea mostrar en la nube como ya se ha comentado es la totalidad de las etiquetas para que el usuario pueda visualizarlas de una forma ágil y seleccionar una.

Se realiza una llamada al modelo de las etiquetas creado al principio de la aplicación utilizando el método de búsqueda de todas las etiquetas de la aplicación por usuario.

```
<div id="myCanvasContainer" leftpadding:'50px' style="color: blue"><canvas width="350"
        height="300" id="myCanvas">
<p>Anything in here will be replaced on browsers that support the canvas
element</p>
<ul>
<?php
$etiquet = new Application_Model_Etiquetas();
$buscareti = $etiquet->getAll();
foreach($buscareti as $row){
        $enlace =
"http://vocs2.local/ficheros/misficheros?texto_buscar=".$row['vocs_e_etiqueta'];
        echo "<li><a href='".$enlace."'>".$row['vocs_e_etiqueta']."</a></li>";
}

?>
```

Con esto ya se tendrían todas las etiquetas existentes en la nube y en funcionamiento.

Una vez esto se ha realizado, el usuario puede seleccionar una de ellas para realizar una búsqueda.

Para realizar la búsqueda es necesario referenciar desde la clase MisFicheros a la nube.

MisFicheros

Esta clase es la encargada de mostrar todos los ficheros del usuario. Dado que es aquí donde el usuario ve todos sus ficheros, es lo más lógico que desde aquí se pueda realizar la búsqueda de etiquetas y se muestre la nube.

Desde aquí por lo tanto se referencia a la nube creada en index.phtml y se hace una llamada al método onClick de php.

Con este método, se pueden incluir funciones que se realicen únicamente cuando el usuario pulse en uno de los elementos seleccionables.

Por lo tanto, el método que se ha de invocar cuando el usuario pulse en una de las etiquetas, es el método de buscar perteneciente al modelo de etiquetas que ya se ha utilizado anteriormente.

Con esto se enviaría el nombre del elemento pulsado al método de buscar, se obtendría el identificador y el modelo buscaría en la base de datos todos los elementos con ese identificador.

Con esto se redigiría la aplicación a la página de MisFicheros nuevamente y se mostrarían únicamente los ficheros con la etiqueta seleccionada.

Como ya se ha comentado, el usuario tendrá la posibilidad de buscar etiquetas mediante dos métodos distintos. El primero es la nube ya explicada y el segundo es mediante un desplegable de etiquetas.

Esta modificación se realiza directamente en la clase MisFicheros ya que es la que muestra todos los ficheros disponibles.

En primer lugar es necesaria la creación del desplegable que contenga todas las etiquetas.

Previamente a este desplegable se introdujo una zona de texto en la que el usuario podía poner la etiqueta que quería buscar. Se hicieron controles de errores y era correcto pero, al querer realizar una búsqueda por varias etiquetas al mismo tiempo, se hacía bastante menos intuitivo que si se realiza sobre un desplegable, por lo que este fue el motivo del cambio.

Además, al ser un desplegable que contiene todas las etiquetas que existen, es mucho más resistente a fallos que una cadena de texto en la que habría que realizar muchas comprobaciones y siempre puede darse algún error.

Una vez el desplegable se ha incluido en MisFicheros, hay que incluirle todas las etiquetas que el usuario puede seleccionar. Para ello se hace una búsqueda de todas las etiquetas y se obtienen todos los nombres asociados a esas etiquetas para su posterior muestra.

```
<?php
                    $etiqueti = new Application_Model_Etiquetas();
                    $fila_etiqueta = $etiqueti->getAll();
                echo "<script type='text/javascript'>alert('".$etiqueti->getAll()."');</script>";
                foreach($fila_etiqueta as $fetiqueta){
                //      echo "<script
type='text/javascript'>alert('".$fetiqueta['vocs_e_etiqueta']."');</script>";
echo "<option value='".$fetiqueta['vocs_e_id']."'>".$fetiqueta['vocs_e_etiqueta']."</option>";
                                                                                    }
```

Con esto se tendrían todas las etiquetas del usuario incluidas en el desplegable.

El siguiente paso es que si el usuario realiza click sobre una de las etiquetas, la aplicación busque los ficheros relacionados con ella. Como punto extra, al utilizar este método el usuario podrá buscar un fichero por varias etiquetas a la vez, por lo que el método buscar anterior y el phtml de MisFicheros no servirían para este método y habría que realizar uno nuevo que se coordine con los existentes para que esto sea posible.

MisEtiquetas

Esta clase phtml se encargará, conjuntamente a MisFicheros, del desarrollo de las búsquedas por múltiples etiquetas.

En primer lugar, y como ya se ha explicado, el usuario hará click en alguna de las etiquetas del desplegable y el sistema deberá realizar una búsqueda.

Para ello se utiliza la función onClick de php para que en el momento que el usuario seleccione alguno de los elementos, se invoque la función pedida.

En este caso la función será la búsqueda normal que se ha utilizado hasta ahora. La aplicación recogerá el nombre de la etiqueta, buscará su identificador y buscará todos los ficheros que tengan ese identificador de etiqueta asociado en la tabla vocs_ef_etiqueta.

En ese momento se realizará la búsqueda y la aplicación mostrará aquellos ficheros que se han seleccionado.

Cuando se muestran los ficheros, la aplicación también mostrará en la parte superior de la pantalla la etiqueta por la cual se está buscando en este momento. De esta forma el usuario siempre tendrá conciencia de cuáles son los parámetros de búsqueda que está introduciendo.

Una vez que se han mostrado los resultados de la búsqueda, el usuario podrá volver a seleccionar otra etiqueta del desplegable y, de este modo, realizar una búsqueda de ficheros que contengan cualquiera de las dos etiquetas. Con esto el usuario tendrá más fácil encontrar un fichero del cual no recuerda sus etiquetas exactas.

Para esto primero se tiene en cuenta la longitud de la cadena de búsqueda. Cuando el usuario selecciona una segunda etiqueta, se pasa por parámetro POST una cadena que contiene el conjunto de etiquetas anteriores concatenado con una coma ‘,’, y seguido de la nueva etiqueta que se desea buscar.

Antes de realizar una búsqueda se hace una separación de la cadena recogida por parámetro en función de las comas, se guardan todos los elementos en un array y se realiza una búsqueda.

El primer paso de esta búsqueda es una búsqueda normal pero, si el array guarda algún elemento en la posición i+1, se crea una cadena que se transformará en la select de la base de datos.

Ejemplo: Si la select para una búsqueda normal es select * from tabla where etiqueta = ‘xxx’, al haber más elementos se realiza una concatenación a ese select con la nueva etiqueta quedando de la siguiente forma: sel * from tabla where etiqueta = ‘xxx’ or etiqueta = ‘yyy’. Con esto se conseguirían buscar todos los ficheros que cumplan alguna de las condiciones requeridas.

Además la cadena de las etiquetas separada por comas se pasaría como ruta de búsqueda a la aplicación, que la colocaría en la parte superior de la pantalla para que el usuario supiese el conjunto de etiquetas por las cuáles está buscando.

Con toda la gestión de las etiquetas hecha falta por tratar el caso del borrado de ficheros y por tanto de etiquetas.

Cuando un fichero se borra, deben borrarse también las etiquetas asociadas a él. En el caso de que las etiquetas pudiesen estar duplicadas sería algo tan sencillo como eso, pero dado que una etiqueta no puede duplicarse por cuestiones de optimización, una etiqueta puede pertenecer a varios ficheros al mismo tiempo.

Cuando un fichero es borrado por tanto, puede que sus etiquetas pertenezcan a algún otro fichero y no deberían ser borradas.

Para esto se ha creado una nueva clase de utils llamada Etiquetas.php

Utils Etiquetas

Esta clase se invoca desde la clase de Ficheros.php cuando un fichero se elimina de la lista del usuario.

Cuando se invoca la clase, se llama al método borrarEtiquetaGestion con el identificador que se ha cogido de la relación de etiqueta_fichero al borrarla (esta relación siempre debe borrarse).

Con este identificador se buscan todos los ficheros en la tabla vocs_ef_etiqueta_fichero que contienen esa etiqueta. En el caso de que el resultado sea distinto de 1 (es de 1 en vez de 0 dado que esta acción se realiza justo antes y en un momento en paralelo con el borrado de la relación de la tabla, por lo que el fichero que se quiere borrar seguirá existiendo en este momento) la etiqueta no se borrará y se borrará la relación entre fichero y etiqueta.

Si el resultado de la operación es 1 se borrará la relación así como la etiqueta en cuestión ya que sería una etiqueta sin más relaciones con ficheros y mantenerla en la base de datos sería un gasto de recursos evitable.

4.2 Interfaces

En este apartado se mostrarán capturas de pantalla de las interfaces creadas.

Interfaz Login

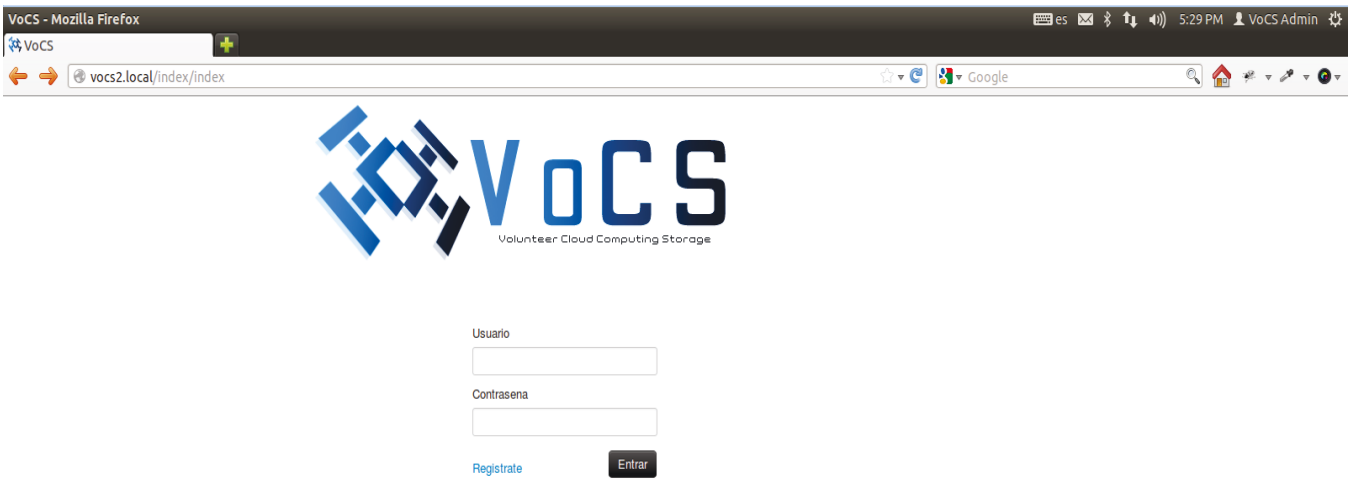

Ilustración 16. Interfaz Login

En ella se puede ver la pantalla inicial de la aplicación. El usuario introducirá sus datos de login para poder acceder a la misma.

Interfaz Inicial Aplicación

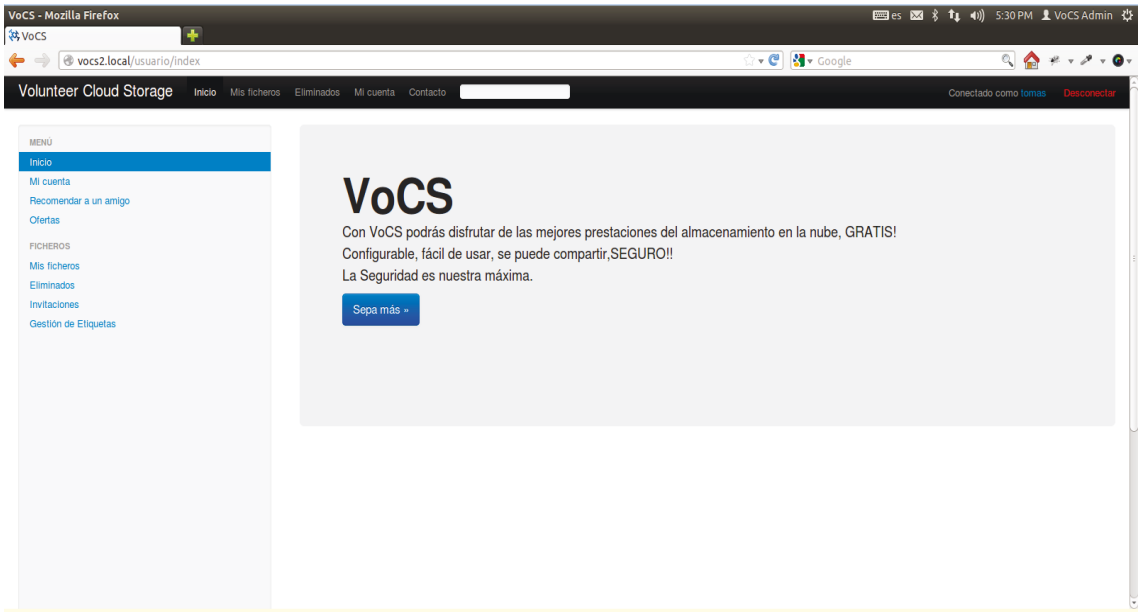

Ilustración 17. Interfaz Inicial

En esta Ventana se puede observar lo que el usuario ve nada más logearse a la aplicación. En la pantalla de bienvenida.

Interfaz Mis Ficheros

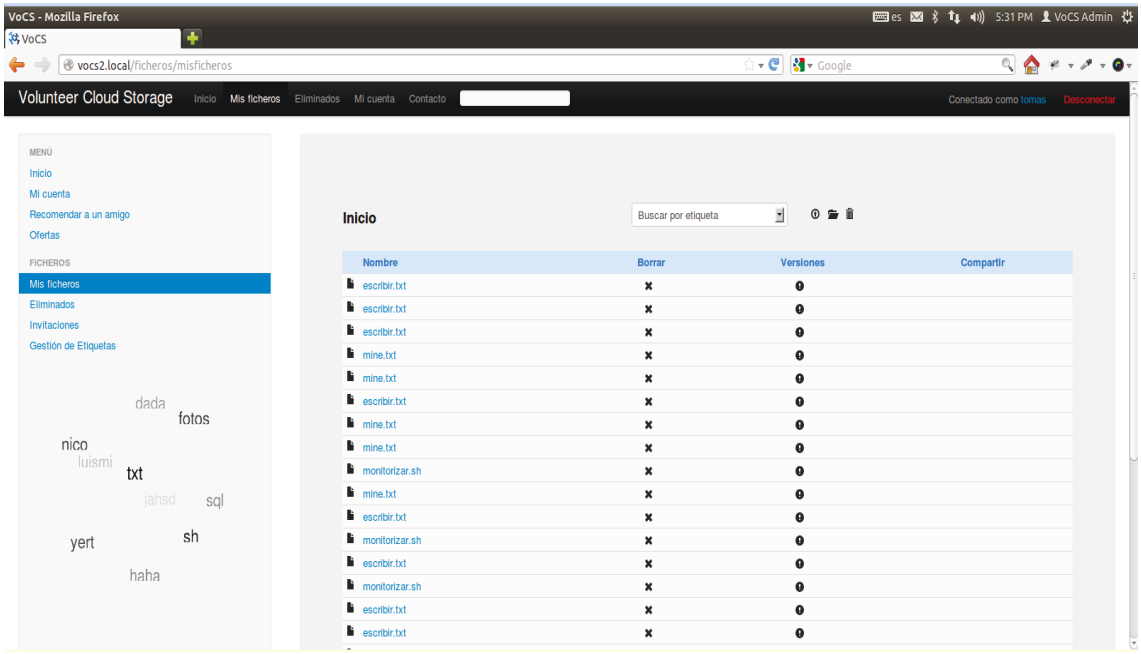

Ilustración 18. Interfaz MisFicheros

En esta pantalla el usuario dispone de todos los Ficheros que ha subido con anterioridad. En la parte inferior izquierda de la pantalla puede verse la nube de etiquetas. En la parte superior central puede verse el desplegable de etiquetas.

A continuación se mostrarán dos pantallas más que contienen la nube y el desplegable más de cerca.

Interfaz Nube de Etiquetas

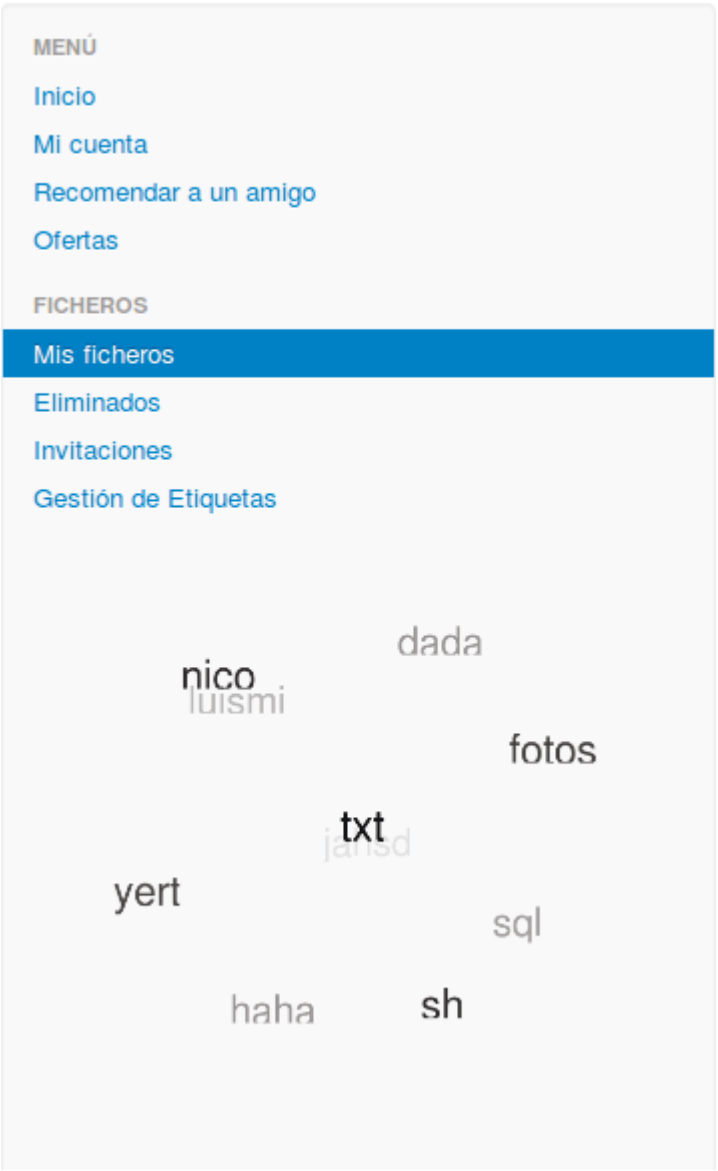

Ilustración 19. Interfaz Nube

Esta nube de etiquetas contiene todas las etiquetas del usuario y al clicar sobre ella puede rotarse para acceder a las de la parte trasera.

Interfaz Desplegable

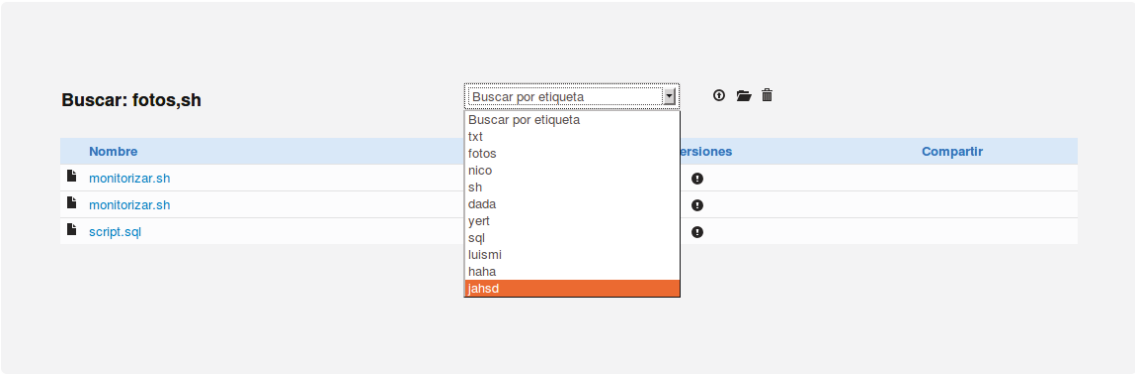

Ilustración 20. Interfaz Desplegable

En esta interfaz pueden verse todas las etiquetas del usuario listadas para que las seleccione.

Interfaz Upload

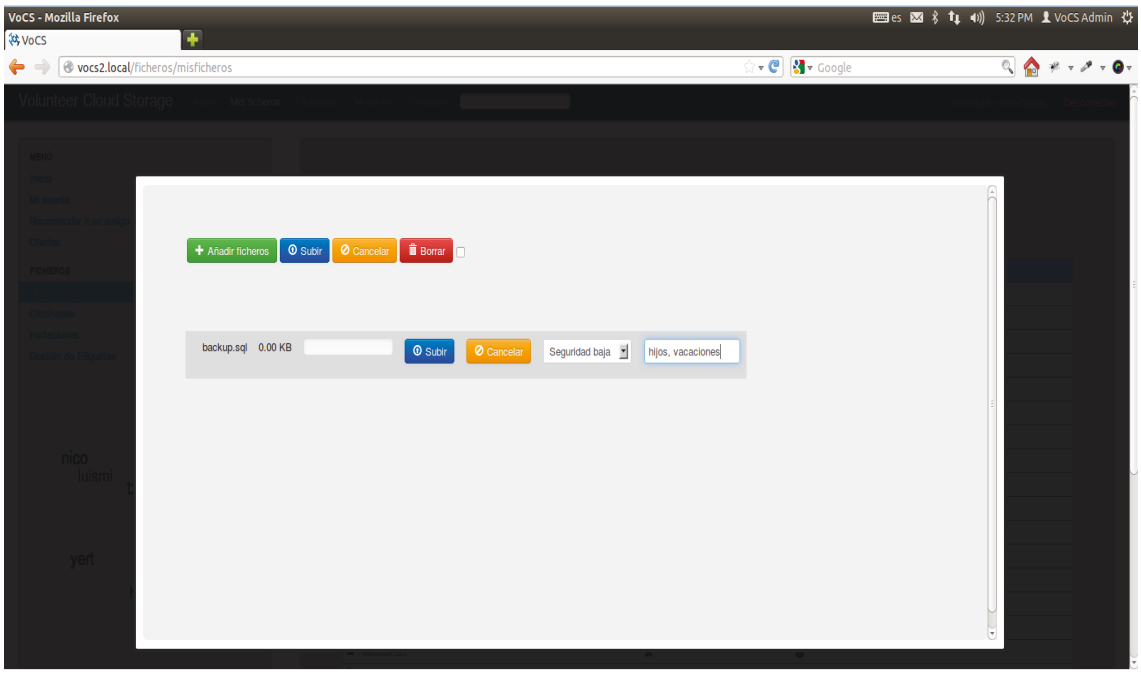

Ilustración 21. Interfaz Upload

Esta interfaz es la utilizada cuando se va a subir un fichero a la aplicación. En la parte derecha se puede observar el cuadro de texto en el cual se insertan las etiquetas que el usuario introduce.

Ilustración 22. Interfaz Upload2

Interfaz Búsqueda Múltiple

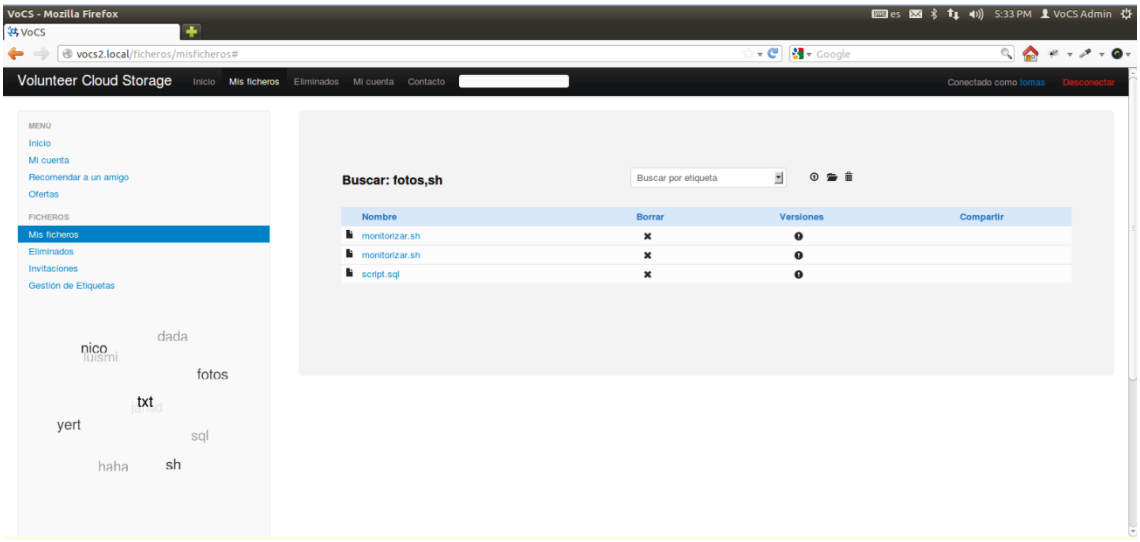

Ilustración 23. Interfaz Búsqueda

En esta imagen se puede ver el resultado cuando un usuario realiza una búsqueda por varias etiquetas a la vez. El resultado de la búsqueda son ficheros que solo contienen las etiquetas escogidas. Dichas etiquetas se muestran en la parte superior de la pantalla en negrita.

Ilustración 24. Interfaz Búsqueda2

Como se puede ver en esta última imagen, el usuario puede comprobar en todo momento los parámetros de búsqueda (en este caso txt y sql). También se puede observar como solo se muestran los resultados con esas etiquetas.

5. Conclusiones

Durante la realización del proyecto se han extraído una serie de conclusiones y de trabajos futuros que se van a desarrollar en este apartado.

5.1 Conclusiones Generales

El objetivo de este proyecto, tal y como está expuesto en el apartado de objetivos, es la modificación de un proyecto existente para incluirle las funcionalidades necesarias para la gestión de etiquetas. . A lo largo del documento se ha podido seguir el desarrollo y la consecución de este objetivo con la elaboración completa de dicha aplicación.

Se han logrado cumplir todos los objetivos primarios y secundarios que se propusieron en un comienzo para considerar que el proyecto estaba terminado:

- Inclusión de las extensiones como etiquetas de forma automática.
- Inclusión por parte del usuario de las etiquetas que él deseara para cada fichero.
- Un sistema de búsqueda visual que hiciera a la aplicación más atractiva a la vista al usuario.
- Un sistema de búsqueda por varias etiquetas conjuntamente.
- Modificación de la aplicación para el soporte de las etiquetas en la función de eliminar un fichero.
- Modificar la Base de Datos existente para permitir la inclusión de las etiquetas.
- Mantener un rendimiento de la aplicación alto.
- Crear mejoras que fueran totalmente tolerantes a fallos humanos.
- Conseguir crear mejoras que permitan ser reusables y que pueda ser extensible a nuevas mejoras.
- Crear una aplicación Online que funcione en varios navegadores (IE Explorer, Opera, Google Chrome, Safari y Firefox)
- Conocer todos los aspectos que ofrece HTML, HTML5, PHP y el nuevo frameWork ZEND.

- Poner en práctica los conocimientos adquiridos en la especialidad “Ingeniería de computadores” en cuanto a plataformas distribuidas y colaborativas.

5.2 Trabajos Futuros

Debido a que la aplicación se ha mejorado de una forma modular con el modelo Vista Controlador, a lo largo del desarrollo de la aplicación se han observado una serie de mejoras y módulos distintos que podrían integrarse en la aplicación sin causar impacto en lo ya existente y añadiéndole funcionalidades interesantes para el usuario.

En primer lugar se podrían realizar inserciones de etiquetas automáticas por más criterios aparte de la extensión. Uno de estos ejemplos podría ser por el propio tamaño del fichero (una etiqueta indicaría si es pequeño, mediano o grande en función de un criterio que habría que establecer). La idea más interesante que ha surgido mientras se desarrollaba el proyecto es la de incluir un algoritmo de semánticas de analizado de textos. Gracias a él se podría escanear el texto incluido en un fichero y sacar las palabras clave del mismo. De esta forma se tendrían etiquetas por defecto que otorgarían mucha información extra al usuario.

Por otro lado se podría realizar una aplicación que integrase aún más elementos visualmente atractivos para el usuario aprovechando las ventajas que ofrece HTML5.

Otra opción interesante sería la de adaptar la aplicación a Smartphones y Tablets. Esto supondría u proyecto bastante grande pero daría una capacidad de atracción a la aplicación muy grande dado el gran número de personas que se decantan por una aplicación en función de esto.

Esta última opción supondría cambiar la estructura de las interfaces utilizando Responsive Design. Esta técnica de diseño se basa en el uso de estructuras e imágenes fluidas y css para cambiar la interfaz web en función del entorno del usuario. Si un usuario se conecta desde el PC tendrá una estructura pero, en el caso de hacerlo desde un Smartphone la interfaz debería ser distinta y adaptarse a las capacidades de ese entorno.

Anexo 1. Bibliografía

http://www.einicio.com/paginas/tecnologia-web.html

http://www.tecnoweb2.com/tecnologias-webhttp://www.zend.com/en/

http://framework.zend.com/

http://php.net/

http://www.w3schools.com/

http://www.desarrolloweb.com/

http://www.lamphowto.com/

http://www.apache.org/

http://www.mysql.com/

http://www.mysqlya.com.ar/

http://cakephp.org/

Anexo 2. Presupuesto

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1- Autor
Héctor de Anta Pérez
2-Departamento
Área de Arquitectura y Tecnología de Computadores
3- Descripción del Proyecto
- Titulo: Diseño de Mejoras de un proyecto de Almacenamiento en la Nube
- Duración: 4 Meses
Tasa de Costes Indirectos: 0,2
4- Presupuesto Total del Proyecto
15705,7
5- Desglose Presupuestario (Costes Directos)

PERSONAL

Apellidos y Nombre	N.I.F. (Solo a modo Informativo)	Categoría	Dedicación (Hombres/mes)	Coste Hombre/mes	Coste (Euros)
De Anta Pérez, Héctor	-	Analista	1,5	4158,23	6237,35
De Anta Pérez, Héctor	-	Diseñador	1,5	3565,6	5348,4
De Anta Pérez, Héctor	-	Programador	1	2400,98	2400,98
				Total	13986,7

EQUIPOS

Descripción	Coste	%Uso Dedicado al Proyecto	Dedicación (meses)	Periodo Depreciación	Coste Imputable
Ordenador Intel Core i5 inside	620	100	3	60	31
				Total	31

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e)]

Descripción	Empresa	Costes imputable
Internet	Jazztel	90,00
Luz	Iberdrola	100,00
Microsoft Office 2010 estudiantes	Microsoft	140,00
	Total	330,00

[e)] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,...

6- Resumen de Costes

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	13986,73
Amortización	31
Subcontratación de Tareas	0
Costes de Funcionamiento	330
Costes Indirectos	1358
Total	15705,73