UNIVERSIDAD CARLOS III DE MADRID

DEPARTAMENTO DE INFORMÁTICA

VoCS

Sistema de almacenamiento voluntario en la nube

Autor: Ignacio Nicolás Schiavón Raineri

Tutor: Luis Miguel Sánchez García

Agradecimientos

En primer lugar agradecer a mi tutor Luis Miguel permitirme realizar un proyecto bajo su tutelaje.

A mi compañero Nicolás por hacerme compañía durante la realización del proyecto y compartir su inteligencia y sabiduría conmigo.

A Analía por estar siempre presente y ayudarme a ser mejor cada día.

Y en especial, gracias a mi amigo y hermano Irkus, que siempre está conmigo.

Resumen

La computación en la nube responde a las necesidades del aumento de dispositivos conectados a Internet y el creciente volumen de datos manejados, ofreciendo acceso ubicuo y transparente a la información de forma segura. Esto ha tenido como consecuencia la apertura del mercado, ofreciendo muchas aplicaciones basadas en la nube como SkyDrive, Google Drive o Dropbox.

VoCS (Volunteer Cloud Storage) es un sistema de almacenamiento voluntario en la nube de código abierto y seguro, que pretende ofrecer alta disponibilidad y buen rendimiento. VoCS dispone de las siguientes características:

- Ofrece los servicios básicos y actuales en una aplicación dedicada al almacenamiento en la nube.
- Es de código abierto.
- Permite la investigación y desarrollo de nuevos modelos y servicios para el almacenamiento en la nube de manera sencilla, de forma que se pueda modificar el código para personalizarlo a las necesidades de los usuarios.
- Ofrece nuevos modelos de replicación de datos garantizando tolerancia a fallos e intentando conseguir un buen rendimiento.

Palabras clave: VoCS, sistema de almacenamiento, nube, replicación, voluntario.

Abstract

Cloud computing satisfies the needs of the increase of Internet-connected devices and the growing amount of data handled, offering ubiquitous and transparent access to the information in a secure way. This has had as a consequence the opening of the market, offering a lot of cloud based applications like SkyDrive, Google Drive or Dropbox.

VoCS (Volunteer Cloud Storage) is a cloud system of volunteer storage, open source and secure, that offers high availability and good performance. VoCS provides the next characteristics:

- It offers the basic services in an application dedicated to the cloud storage.
- It is open source
- It allows the investigation and development of new models and services for the cloud storage in a simple way, so that the code can be modified to personalize it with the user's needs.
- It offers new models of data replication, guaranteeing fault tolerance and good performance.

Key words: VoCS, storage system, cloud, replication.

Índice

Índice Figuras

Índice Tablas

1. Introducción

Este documento pretende describir el proyecto "VoCS: Diseño de un sistema de almacenamiento voluntario en la nube", el cual consiste en:

- Análisis, diseño, implementación e implantación de un sistema de almacenamiento en la nube de código abierto.
- Evaluación de diferentes modelos de replicación de ficheros.

A continuación se presentará la motivación para realizar el presente TFG, los objetivos del proyecto y los acrónimos y términos utilizados en el documento.

1.1 Motivación

Con los avances en infraestructuras en las tecnologías de la información y comunicación, los nuevos modos de programación y los nuevos modelos en su uso, ha llegado la denominada Computación en la Nube, donde los recursos y servicios informáticos son ofrecidos y consumidos como servicios a través de Internet, sin que los usuarios tengan ningún conocimiento de la infraestructura que hay detrás.

Cada vez es mayor el volumen de datos que se maneja en la industria de las tecnologías de la información, por esto, las medianas y grandes empresas cada vez más deciden utilizar la nube para reducir gastos de capital y de operaciones asociadas con el equipo material.

En la última década, el avance de nuevas tecnologías como Smartphones, tabletas, etc. han supuesto el incremento de dispositivos que las personas poseen. La nube ofrece servicios unificados para todos los dispositivos conectados a la red, como almacenamiento en la nube, sincronización de datos o compartición de información que dan a los usuarios la posibilidad de un acceso ubicuo y transparente a su información.

Por ello, se han creado numerosas aplicaciones que proporcionan servicios en la nube, tanto para empresas como para usuarios individuales. Muchas de estas aplicaciones están desarrolladas por las empresas más importantes del sector de la tecnología como: Google, Microsoft o Apple.

Las aplicaciones que ofrecen servicios de almacenamiento tienen un espacio inicial gratuito muy limitado, insuficiente para la mayoría de las empresas o personas que quieran trabajar con la aplicación. Se puede contratar espacio adicional que supone un coste económico. Por otro lado, la confidencialidad de los datos y de los archivos queda supeditada a la honestidad

de las empresas que ofrecen los servicios. Estas son desventajas importantes para empresas o personas que manejen gran cantidad de información y/o información confidencial.

A pesar de existir desde hace muchos años, es actualmente cuando se están llevando a cabo los mayores avances en la nube debido al crecimiento comercial. Uno de los temas más importantes en el campo, es la distribución de la información, en constante evolución, por lo que disponer de una plataforma desde la cual poder implantar nuevas soluciones es una necesidad para todo investigador o desarrollador.

Existen plataformas como *Owncloud*, de código abierto y sin costes adicionales por licencia, que permiten la implantación de sistemas de almacenamiento en la nube en servidores propios, no teniendo que confiar en terceros el almacenamiento de la información. Sin embargo, no están centradas en ofrecer facilidad de desarrollo de modelos o servicios propios para el almacenamiento en la nube.

Por esto, el presente Trabajo de Fin de Grado se centrará en desarrollar una plataforma que:

- Ofrezca los servicios básicos y actuales en una aplicación dedicada al almacenamiento en la nube.
- Permita la investigación y desarrollo de nuevos modelos y servicios para el almacenamiento en la nube de manera sencilla, de forma que se pueda modificar el código para personalizarlo a las necesidades de los usuarios.
- Ofrezca nuevos modelos de replicación de datos, con sus respectivos diseños, implementaciones y evaluaciones.

De esta manera VoCS puede ofrecer lo que otros sistemas no pueden:

- Será de código abierto y todas las tecnologías que se utilicen serán de código abierto, por lo que la implantación y uso no generarán costes.
- Será personalizable, por lo que será posible dar un espacio ilimitado a los usuarios de la aplicación con coste nulo.
- Permitirá la implantación de un sistema de almacenamiento en la nube propio, gestionado completamente por parte del usuario, por lo cual, no se confiará la información almacenada a terceros.
- Permitirá la investigación, diseño y desarrollo de nuevos modelos y servicios para el almacenamiento en la nube de manera sencilla.
- Ofrecerá a los usuarios funcionalidades avanzadas para el almacenamiento en la nube.

1.2 Objetivos

El principal objetivo que persigue el Trabajo de Fin de Grado (TFG) **es realizar el análisis, diseño e implantación de un sistema de almacenamiento voluntario en la nube** que disponga de las siguientes características:

- 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.

Para llevar a cabo el objetivo principal del TFG, de él se han marcado los siguientes sub-objetivos:

- Analizar y elegir las tecnologías Web actuales, que permitan el desarrollo de la aplicación VoCS y cumplan con sus requisitos y objetivos.
- Construir un entorno de trabajo necesario para un proyecto en la nube. Para ello, es necesario disponer de computadores conectados a la red, con un sistema operativo instalado con las librerías necesarias para el desarrollo.
- Desarrollar una aplicación Web para el acceso público, en la que sea posible hacer uso de las funcionalidades que se desarrollarán en el TFG.
- Diseñar e implementar un sistema de ficheros, en el que los usuarios de VoCS sean capaces de hacer uso de las funcionalidades comunes que aporta un sistema de ficheros convencional.
- Diseñar un sistema de compartición de ficheros.
- Diseñar, implementar y evaluar modelos de replicación de datos.
- Analizar, diseñar e implementar una base de datos para almacenar la información relevante del sistema y permita su gestión de forma rápida y sencilla.
- Diseñar interfaces de usuario intuitivas y usables.
- Desarrollar algoritmos que garanticen la seguridad de la información de los usuarios.
- Evaluar la subida y descarga de ficheros con los diferentes modelos de replicación.

1.3 Acrónimos

Esta sección contiene los acrónimos utilizados en el presente documento.

3D: 3 Dimensiones

3DES: 3 Data Encryption Standard

AES: Advanced Encryption Standard

AJAX: Asynchronous JavaScript and XML

ASD: Adaptive Software Development

ASCII: American Standard Code for Information Interchange (Código Estándar Estadounidense para el Intercambio de Información)

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

ASP: Active Server Pages

BD: Base de Datos

BSD: Berkeley Software Distribution (Licencia)

CGI: Common Gateway Interface

CRUD: Create, Read, Update, Delete

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

C#: C Sharp

DB: Data Base

DBM: DataBase Management.

DNS: Domain Name System

DOM: Document Object Model

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

IETF: Internet Engineering Task Force

JSON: JavaScript Object Notation

JSP: JavaServer Pages

LDAP: Lightweight Directory Access Protocol

MAC: Message Authentication Code

MD5: Message-Digest Algorithm 5

MIT: Massachusetts Institute of Technology

MVC: Modelo Vista Controlador

NIST: National Institute of Standards and Technology

PAM: Pluggable Authentication Module

PC: Personal Computer

PHP: PHP Hypertext Pre-processor

REST: Representational State Transfer

SCGI: Simple Common Gateway Interface

SGBD: Sistema Gestor de Bases de Datos

SGML: Standard Generalized Markup Language

SNI: Server Name Indication

SHA: Secure Hash Algorithm

SQL: Structured Query Language

SSL: Secure Sockets Layer

TCP: Transmission Control Protocol

TFG: Trabajo de Fin de Grado.

TLS: Transport Layer Security

UCD: Diseño centrado en el Usuario

URL: Uniform Resource Locator

VRML: Virtual Reality Modeling Language

W3C: World Wide Web Consortium

XML: eXtensible Markup Language

1.4 Glosario

DELETE: Método de petición de HTTP. Borra el recurso especificado.

GET: Método de petición de HTTP. El método GET requiere la devolución de información al cliente identificada por la URI.

POST: Método de petición de HTTP. Es utilizado cuando el cliente necesita enviar datos al servidor como parte de una petición.

PUT: Método de petición de HTTP. El método PUT permite guardar el contenido de la petición en el servidor bajo la URI de la petición.

Licencia MIT ó X11: Licencia de código abierto permisiva. Se puede crear una obra derivada sin que ésta tenga obligación de protección alguna.

Licencia BSD: Licencia de código abierto permisiva. Se puede crear una obra derivada sin que ésta tenga obligación de protección alguna.

Licencia GNU/GPL: Licencia de software de código abierto robusta fuerte. Contienen una cláusula que obliga a que las obras derivadas o modificaciones que se realicen al software original se deban licenciar bajo los mismos términos y condiciones de la licencia original.

Salt: Comprende bits aleatorios que son usados como una de las entradas en una función derivadora de claves

Array: Es una zona de almacenamiento continuo.

Webservice: es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

Layout: En programación Web, layout se refiere a la estructura de las páginas Web.

2. Estado de la Cuestión

En este punto se van a sentar los precedentes del proyecto, el contexto actual, las tecnologías que involucra y otros proyectos desarrollados. Este punto nos va a ayudar a vislumbrar las novedades que aporta VoCS sobre otros desarrollos, la variedad de tecnología y posibilidades relacionadas a un proyecto de estas características.

2.1 Evolución de las tecnologías Web

En 1993, el *National Centre for Supercomputing* (Centro Nacional de Supercomputación) desarrolló el primer navegador libre llamado "Mosaic". Este navegador era capaz de mostrar texto y gráficos, pero tenía muchas limitaciones. No obstante, era suficiente para las páginas Web de aquellos años, las cuales eran muy lineales y funcionales para los científicos que querían compartir información. Las primeras páginas web, escritas en HTML, eran tan simples como un documento de texto o un simple folleto en el que el desarrollo más elaborado consistía en usar efectos en las fuentes y colores.

Con el paso del tiempo, el desarrollo del comercio electrónico y las necesidades comunicativas, las páginas web comenzaron a tener estructuras más complejas y otros lenguajes de programación y metodologías entraron en juego, como JavaScript.

En 1994 fue creado el consorcio W3C para establecer estándares y objetivos en el desarrollo del lenguaje HTML. Desde entonces, HTML ha tenido varias versiones hasta la actual HTML 5.0.

Por aquel entonces, surgió una nueva generación de páginas web cuyas principales características de diseño eran los iconos que sustituían a palabras, fondos de pantallas que se formaban con mosaicos, botones con bisel y relieve, banners que sustituían a las cabeceras y menús desplegables. Surgió la estructura de tablas pensada para introducir datos estadísticos, pero fue usada también para el diseño.

La tecnología avanzó y las páginas web se centraron más en la interactividad con el usuario: el lenguaje DHTML y Flash. También se avanzo en materia de diseño, se introdujo el lenguaje CSS.

Una vez que se fue estandarizando la comunicación entre plataformas (PC, Mainframe, Mac, etc.), también empezaron a proliferar lenguajes de programación avanzados que permitían prestaciones de las que hasta entonces no disponían las páginas Web. Se trata de PHP, C# y la tecnología ASP entre otros. Esto permitía acciones como conectar la página con una base de

datos. El diseño comenzó a centrarse en la integración de contenido multimedia como sonidos, videos, animaciones, los mundos 3D y el VRML (Virtual Reality Modeling Language).

A principios de siglo XXI, gracias al desarrolló de ciertas herramientas comenzaron a aparecer sitios Web donde el usuario podía participar como: blogs, redes sociales y sistemas de mensajería sofisticados. Nació así la Web 2.0., cuyos patrones de diseño se caracterizan porque el usuario puede manejar los aspectos visuales y las metáforas de presentación con tecnologías como AJAX y un conjunto de lenguajes y técnicas de programación, que no eran novedosos pero que, al ser combinados, facilitan el desarrollo de interfaces de alta usabilidad. El diseño de las páginas se vuelve elástico y se adopta el diseño centrado en el usuario (UCD).

2.2 Patrones de arquitectura: MVC

Es un patrón que define la organización independiente del Modelo (Objetos de Negocio), la Vista (interfaz con el usuario u otro sistema) y el Controlador (controlador del flujo de la aplicación).

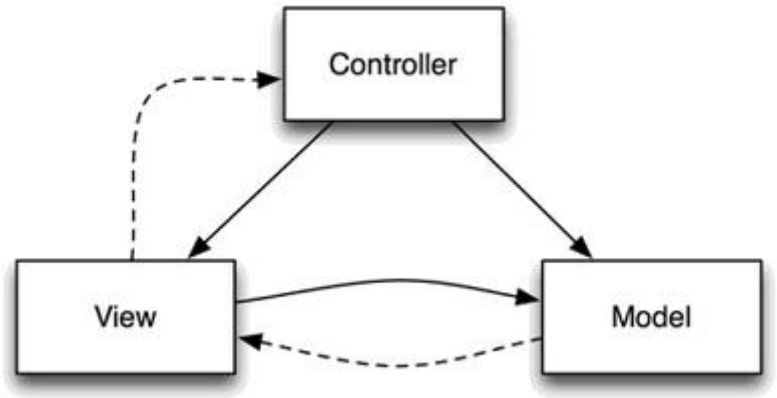


Figura 2.1: Arquitectura MVC

En la Figura 2.1 se muestran los tres elementos del patrón MVC. Los elementos MVC están representados en cajas y las líneas que los conectan representan las relaciones (concretas o abstractas). Las líneas solidas representan una referencia concreta y las líneas discontinuas representan relaciones abstractas. Estas relaciones pueden ser expresadas de la siguiente manera:

- **Controlador:** Los controladores manejan todas las entradas y manipulan las instancias específicas de los modelos y vistas. Son los encargados de unir los elementos que componen la arquitectura MVC. Manipulan los modelos, transmiten los datos que necesitan o dan el control a otro controlador y deciden que se va a mostrar en base a la solicitud del usuario y otros factores.
- **Modelo:** Los modelos representan los datos utilizados por la aplicación y no tienen conocimiento de los controladores. Tienen acceso a una interfaz abstracta de la vista. Es la parte de la aplicación que define la funcionalidad básica de un conjunto de abstracciones. Rutinas de acceso de datos y parte de la lógica de negocio.
- **Vista:** Las vistas representan datos de instancias específicas de modelos, al usuario. Tienen un acceso abstracto al controlador que los creó. Las vistas definen exactamente lo que se presenta al usuario. Por lo general, los controladores pasan los datos a cada vista para que los representen en algún formato. También, las vistas recopilan datos del usuario.

Uno de los principales problemas para implementar MVC es que el elemento de la Vista se ejecuta en la parte del cliente (navegadores) y el modelo y el controlador se ejecutan en el lado del servidor. Por lo que muchos *frameworks* han optado por añadir un cuarto elemento a la arquitectura: el *Front Controller*.

El *Front Controller* es el encargado de reunir las peticiones HTTP de forma que pueda ser asignada a la arquitectura MVC.

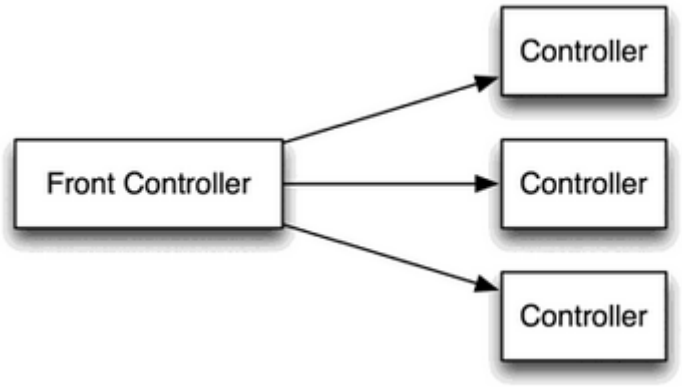


Figura 2.2: *Front Controller*

2.3 Tecnologías Web

Una aplicación Web es un sistema compuesto por: servidores Web, redes, HTTP o HTTPS y un navegador, en el que la interacción del usuario afecta al estado del negocio. Las aplicaciones Web también pueden incluir estructuras de almacenamiento para persistir la información.

Las aplicaciones Web utilizan la arquitectura cliente-servidor, donde la comunicación del cliente con el servidor utiliza una red, Internet o Intranet, con el protocolo HTTP. La comunicación básica entre un cliente y un servidor consiste en:

1- Petición HTTP de un contenido al servidor, por parte del cliente.
2- El servidor procesa la petición (PHP, Java, etc.) y envía la respuesta (HTML Y CSS) al cliente.
3- El cliente interpreta los datos y los muestra al usuario.

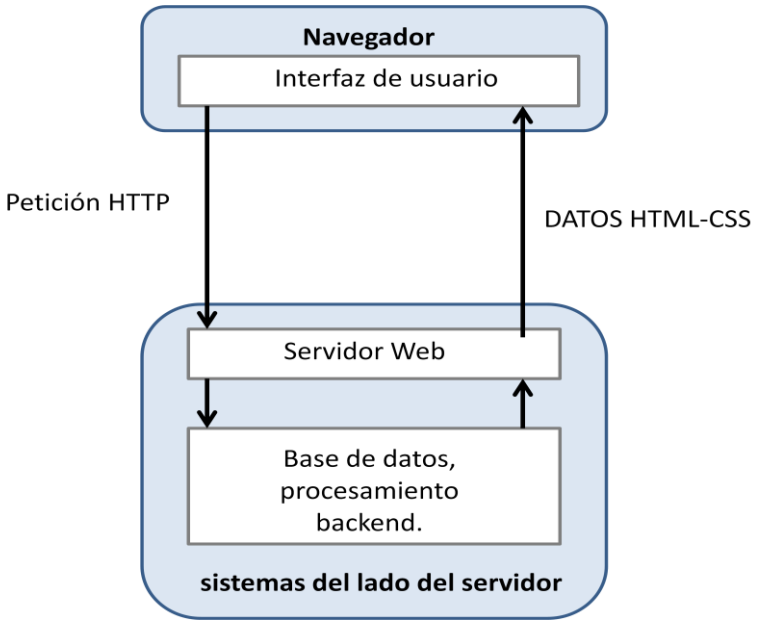


Figura 2.3: Esquema aplicación Web

El cliente es el encargado de gestionar las peticiones del usuario, gestionar la recepción de las páginas provenientes del servidor e interpretar los documentos HTML y sus recursos. Las tecnologías más utilizadas en el lado del cliente son:

1- HTML
2- CSS
3- JavaScript
4- ActiveX
5- Applets en Java.

El servidor gestiona las peticiones del cliente. La gestión consiste en recibir una petición, procesarla y responderla con contenido HTML y CSS. Las peticiones pueden ser de páginas estáticas o dinámicas:

1- Las páginas estáticas están almacenadas en el servidor. Las únicas tareas realizadas por el servidor es localizarlas y enviarlas al cliente.
2- Las páginas dinámicas son generadas en el momento en el que el servidor recibe la petición. Para ello, dispone de programas que el ejecutarse dan lugar a páginas generadas.

En ocasiones, el servidor necesita utilizar sistemas de almacenamiento para persistir la información. Los encargados de gestionar el almacenamiento de la información es el software (Sistemas Gestores de Bases de Datos, sistemas de ficheros, etc.) instalados en el servidor.

Las tecnologías más utilizadas del lado del servidor son:

3- CGI
4- ASP.NET de Microsoft.
5- JSP y servlets.
6- PHP

2.3.1 Tecnologías del lado del servidor

En este punto se detallarán las tecnologías que se ejecutan en el servidor.

2.3.1.1 JSP

Java Server Pages (JSP) o Páginas de servidor Java, es una tecnología Web, del lado del servidor, orientada a crear páginas web dinámicas con programación Java. JSP es un producto de la compañía Sun Microsystems.

Una de las principales ventajas de JSP, es que permite al programador integrar los scripts con clases de Java (*servlets*), lo que permite tener por separado los módulos que se encargan de hacer los procesos de datos, de los que se encargan de presentar visualmente dichos datos.

Las características principales de la tecnología JSP son:

- Multiplataforma y portable: Es posible ejecutar un código, sin necesidad de modificarlo, en cualquier sistema operativo que disponga de una Máquina Virtual Java.
- La Maquina Virtual de Java presenta un recolector de basura que libera la memoria que no se esté usando.

- Presenta gran número de *frameworks* que pueden ser utilizados como: Struts, JSF o Tiles. Estos *frameworks* presentan características avanzadas que permiten un desarrollo más ágil y sencillo de las aplicaciones.
- Soporte. Java continúa en desarrollo, por lo que las actualizaciones están al día y presenta una gran comunidad de soporte.
- Orientado a objetos. Java es un lenguaje orientado a objetos que facilita el uso de patrones de diseño y la reusabilidad de código. También permite la encapsulación, herencia y polimorfismo que facilitan el mantenimiento, reutilización y modularidad del producto final.
- Necesidad de un servidor que permita la ejecución de JSP, como Tomcat.

Los JSPs son en realidad *servlets*, el proceso que sigue la tecnología JSP cuando se invoca a un fichero JSP es:

- El JSP se compila y da lugar a un programa en Java la primera vez que se invoca.
- Del programa en Java se crea una clase que se empieza a ejecutar en el servidor como *servlet*.

La principal diferencia entre los *servlets* y los JSPs es el enfoque de la programación: un JSP es una página Web con etiquetas especiales y código Java incrustado, mientras que un *servlet* es un programa que recibe peticiones y genera a partir de ellas una página Web.

2.3.1.2 ASP.NET

ASP.NET es un framework desarrollado y comercializado por Microsoft para la creación de aplicaciones web, donde se puede programar en cualquiera de los lenguajes de .NET. Apareció en el año 2002, y es la tecnología sucesora de Active Server Pages (ASP) que existe desde 1996. Entre sus principales características se encuentran:

- ASP.NET se integra totalmente con .NET y sus páginas se pueden programar en cualquier de los lenguajes de .NET, haciendo uso de la programación orientada a eventos.
- ASP.NET es un lenguaje compilado. Esto ofrece un rendimiento mucho mejor, y una depuración mucho más potente.
- ASP.NET garantiza la compatibilidad de los controles de la aplicación Web con distintos navegadores, ya que las etiquetas de la página HTML, que incluyen controles en la

interfaz de usuario, son independientes del HTML que después se genera para construir la interfaz de usuario.

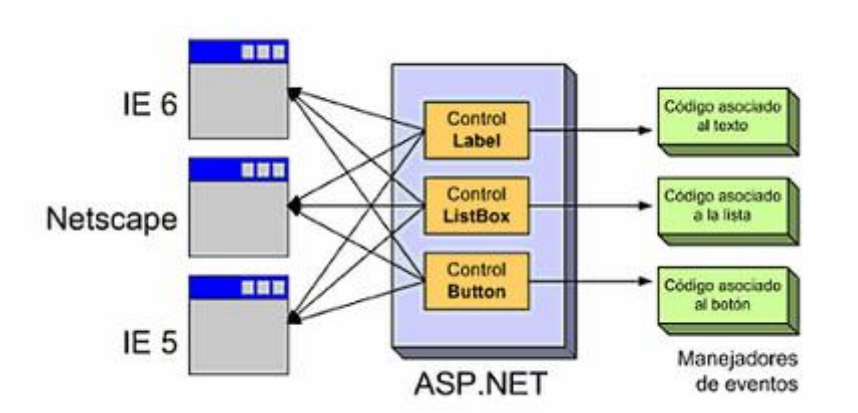


Figura 2.4: ASP.NET se encarga de mostrar los formularios dependiendo del navegador

Las paginas ASP.NET se denominan formularios Web y son archivos con extensión ".aspx". Que permiten dos formas de programarlos:

- Incluyendo tanto los controles como el código en un único fichero .aspx.
- Manteniendo los controles de la interfaz en un fichero .aspx y dejar todo el código en un lugar aparte.

2.3.1.3 PHP

Hypertext Preporcessor (PHP), es un lenguaje Open Source interpretado a alto nivel. Es un lenguaje de scripting del lado del servidor, de plataforma cruzada e incrustado en HTML. El objetivo principal que persigue es permitir escribir páginas Web dinámicas de manera rápida y fácil. [11] [12].

PHP es de plataforma cruzada porque es soportado por muchos sistemas operativos como Linux, muchas versiones de Unix, Windows y Mac. Además soporta la mayoría de los servidores Web, incluyendo Apache. También soporta gran cantidad de bases de datos.

Se dice que PHP es un lenguaje de scripting, en oposición a un lenguaje de programación. Esto significa que PHP está diseñado para ejecutar después de que se produce un evento.

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.

PHP se centra en la programación Web, pero también ofrece otras muchas posibilidades, de las que se diferencian tres campos:

1- Scripts del lado del servidor. Este campo es el más usado y dónde se concentran más trabajos. Para poder realizar scripts del lado del servidor es necesario contar un intérprete PHP (módulo CGI), un servidor Web y un navegador.
2- Scripts en la línea de comandos. PHP da la posibilidad de crear scripts PHP y ejecutarlos sin necesidad de un servidor web o un navegador. Solo es necesario contar con el interprete PHP y es comúnmente usado para ejecución de tareas programadas.
3- Escribir aplicaciones de interfaz gráfica. PHP no está centrado en este tipo de programación pero ofrece la posibilidad con PHP-GTK de escribir programas de interfaz gráfica. Esta extensión no viene incluida con la distribución principal de PHP.

El funcionamiento de PHP con su entorno es el siguiente:

1- Usuario realiza una petición al servidor.
2- Ejecuta los script en el lado del servidor generando código HTML
3- El resultado lo envía al cliente.
4- El cliente interpreta el resultado y lo muestra al usuario. El cliente no puede saber cuál es el código del script ejecutado ya que solo recibe el resultado HTML.

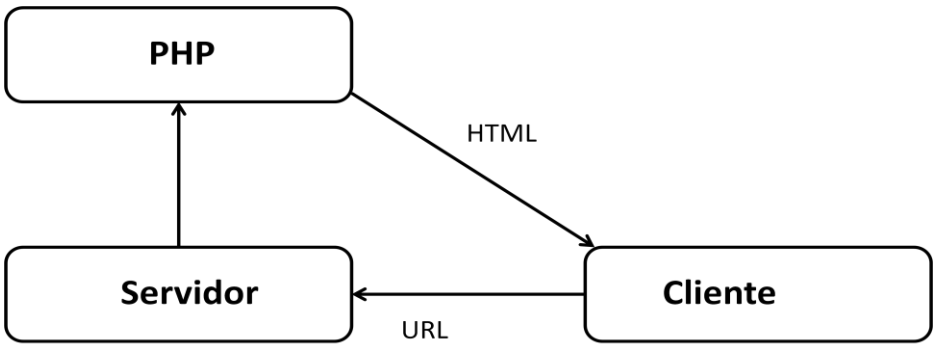


Figura 2.5: Proceso entre un cliente, el servidor y el módulo PHP

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.
3- Symfony.

CakePHP

CakePHP es un *framework* (marco de desarrollo) rápido para PHP y de código abierto. Se trata de una estructura que sirve de base a los programadores para que éstos puedan crear aplicaciones Web en PHP. El principal objetivo de CakePHP es permitir trabajar de manera estructurada y rápida, sin pérdida de flexibilidad. [13].

Cake PHP tiene un equipo de desarrolladores y una comunidad activos, los que permite que el núcleo de las aplicaciones se mejoren constantemente y esté bien probado.

Unas de las características que hacen de CakePHP uno de los *frameworks* más utilizados son:

- Licencia flexible. Es distribuido bajo la licencia X11, más conocida como MIT License.
- Comunidad activa con gran soporte.
- Compatible con PHP4 y PHP5.
- CRUD (Create, Read, Update, Delete o Crear, Leer, Actualizar, Eliminar) integrado para la interacción con base de datos.
- Estructura de aplicaciones (*Scaffolding*): permite al programador hacer uso de un conjunto de convenciones aplicables a la estructura de la base de datos de la aplicación. El *framework* se encarga de generar el código para la interacción a lo largo de todas las capas de la aplicación.
- Arquitectura MVC.
- Validación incorporada.
- Incorporación de asistentes de construcción de vistas: para la automatización de la generación de código en AJAX (Asynchronous JavaScript and XML), JavaScript y formularios HTML, entre otros.
- Almacenamiento en caché de las vistas para acelerar la descarga de las páginas web.
- Despachador de peticiones (*dispatcher*), permite acceder a la aplicación a través de URLs personalizadas y amigables.
- Plantillas rápidas y flexibles (sintaxis de PHP, con ayudantes).
- Componentes de email, *cookies*, seguridad, sesión y manejo de solicitudes.
- Listas de control de acceso flexibles: para gestionar el ingreso de usuarios a la aplicación construida con el *framework*.

CakePHP es compatible y permite trabajar con los principales sistemas gestores de bases de datos como: MySQL, PostgreSQL y SQLite.

CakePHP incluye las clases Controlador (*Controller*), Modelo (*Model*), Vista (*View*) y otras clases y objetos que hacen que el desarrollo en MVC sea más rápido. Los Componentes (*Components*), Comportamientos (*Behaviors*), y Ayudantes (*Helpers*) son clases que proporcionan extensibilidad y reusabilidad; agregan rápidamente funcionalidad a las clases base MVC de las aplicaciones.

El comportamiento del *framework* ante la petición de un usuario es la siguiente:

- El usuario realiza una petición.
- El enrutador analiza la URL y extrae los parámetros de la petición.
- Se mapea la petición a una acción de controlador.
- El controlador puede usar modelos para el acceso a la base de datos.
- El controlador entrega la información preparada a la Vista.

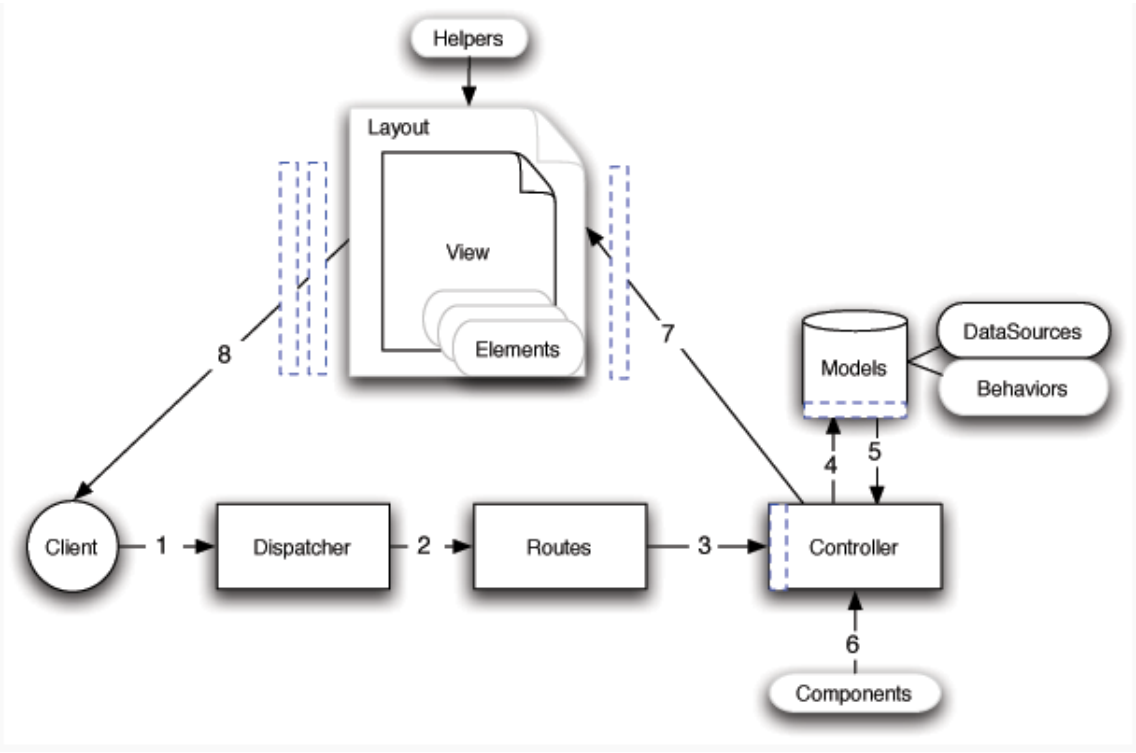


Figura 2.6: Comportamiento CakePHP

Zend Framework

Zend Framework es un proyecto de código abierto con licencia BSD, permite el desarrollo de aplicaciones y servicios Web utilizando PHP5. Es un *framework* que utiliza una programación orientada a objetos, estructurando sus componentes

de manera que contienen muy pocas dependencias entre sí. Esta arquitectura de acoplamiento flexible permite a los desarrolladores usar componentes de manera individual. [14] [15].

Zend Framework proporciona una implementación del patrón MVC. También incluye una interfaz sencilla de bases de datos SQL. Proporciona adaptadores para conectarse a gran variedad de Sistemas Gestores de Bases de Datos:

1- IBM DB2.
2- MariaDB.
3- MySQL.
4- Microsoft SQL server.
5- Oracle.
6- PostgreSQL.
7- SQLite.
8- Firebird.

La interfaz SQL es independiente del Sistema Gestor de Bases de Datos, permitiendo que la aplicación sea escrita de una única manera y se pueda desplegar con distintos SGBD.

Zend Framework también incluye una estructura de elementos que permite el desarrollo ágil y estructurado de las aplicaciones Web. A continuación se detalla la estructura de elementos de Zend.

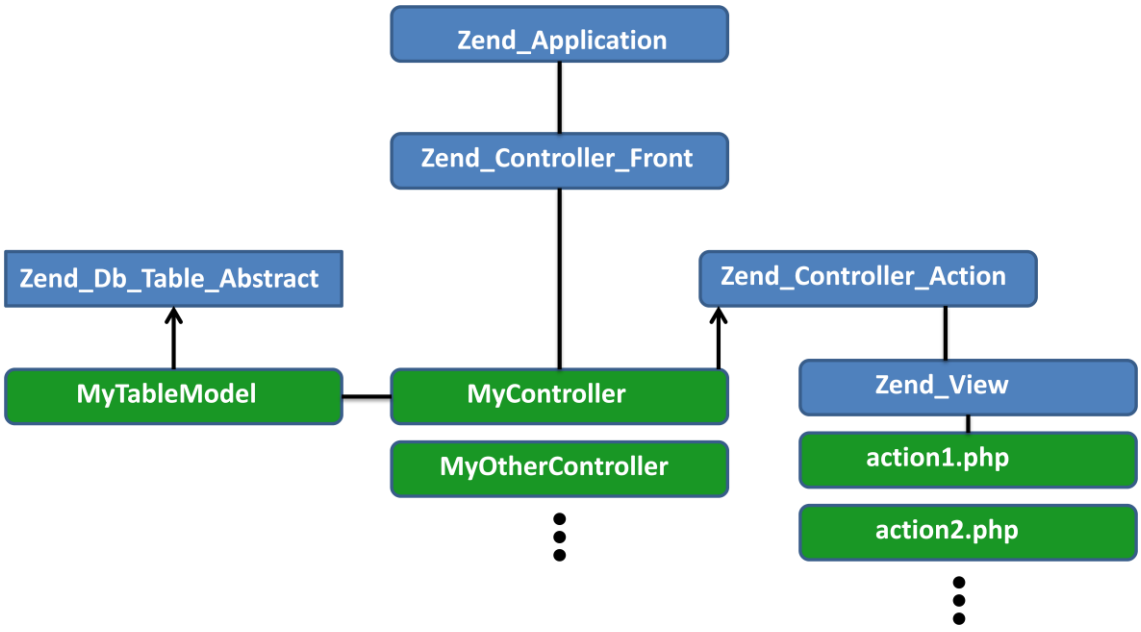


Figura 2.7: Estructura Zend Framework

1- Zend_Application.

Carga el entorno PHP, incluyendo include_paths y autocargados. Crea una instancia de la clase Bootstrap solicitada.

La clase Bootstrap proporciona una interfaz estándar para la carga de módulos y recursos de forma predeterminada. Gracias a ella se pueden implementar las funciones necesarias para la carga de la aplicación y que ésta funcione correctamente.

2- Zend_Controller_Front implementa el patrón *Front Controller*, en el que todas las peticiones son interceptadas y distribuidas a las acciones individuales de cada controlador en función de la URL. El flujo de trabajo del *Front Controller* consiste en un proceso secuencial de un conjunto de elementos:

- El objeto petición, Zend_Controller_Request_Abstract. Representa una petición no procesada y es responsable de evaluar la petición del usuario y proveer de la información necesaria de dicha petición al resto de procesos.
- El enrutador, Zend_Controller_Router_Interface. Inspecciona el objeto petición y determina qué controlador y qué acción deben ser procesadas.
- El dispatcher, Zend_Controller_Dispatcher_Interface. Toma la información que el enrutador provee, instanciando la acción correcta y ejecutándola.
- La respuesta, Zend_Controller_Response_Abstract. El objeto respuesta es responsable de reunir y devolver la respuesta desde las acciones del controlador.

2- MyController. Controladores.

Estos gestionan el flujo de trabajo de la aplicación, mapean las peticiones a los modelos y vistas apropiados. Un controlador incluirá uno o más métodos, llamados acciones (Zend_Controller_Action), que podrán ser solicitados a través de la Web. Estas peticiones Web por defecto en Zend siguen el siguiente esquema "/controlador/acción/parámetros".

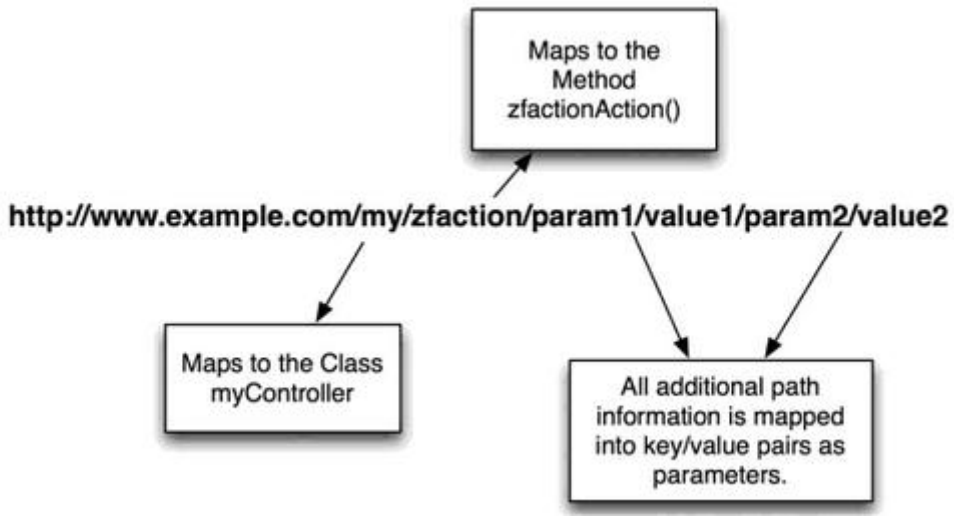


Figura 2.8: URL Zend Framework

Por defecto, Zend Framework proporciona dos controladores:

- IndexController. Controlador inicial, suele servir a la página inicial de la aplicación Web.
- ErrorController. Controlador que utiliza Zend para mostrar los errores producidos en la aplicación.

2- Zend_View. Vistas.

Encargadas de crear la presentación correspondiente a cada acción de cada controlador. Por defecto, en el proyecto Zend se incluyen las vistas para los controladores IndexController y ErrorController.

3- Zend_Db. Modelos.

Encargadas del acceso de datos. Implementarán las funciones necesarias para que el encargado de la lógica del negocio pueda acceder y manipular la base de datos. Zend_Db provee de una interfaz de bases de datos SQL. Genera una capa abstracta intermedia, por lo cual, el programador puede usar unas herramientas comunes independientes del tipo de base de datos que esté utilizando.

En la Figura 2.9 se muestra el comportamiento interno de Zend Framework al recibir una petición de un usuario.

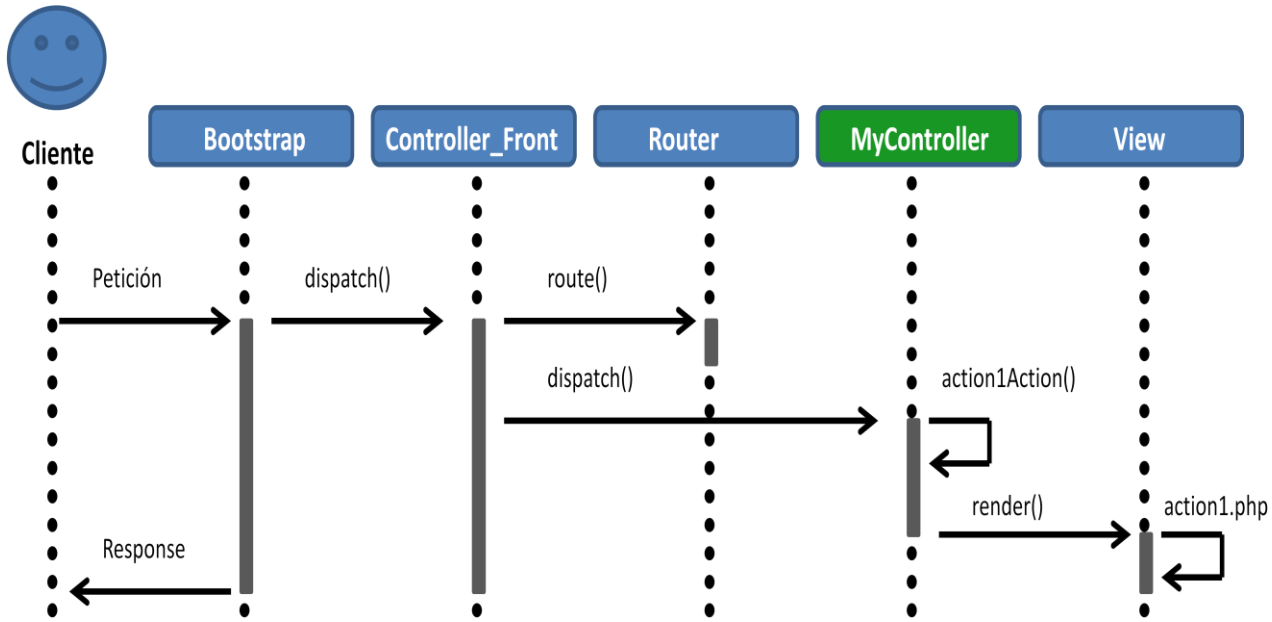


Figura 2.9: Comportamiento Zend

Comparativa Frameworks de PHP

En este punto se va a realizar una comparativa de los frameworks de PHP comentados teniendo en cuenta el impacto de éstos en el desarrollo del proyecto VoCS y su viabilidad económica.

Framework	MVC	AJAX	ORM	Migración de base de datos	Seguridad Framework	Caché	*Scaffolding*	Validación de formularios	Licencia
Zend	SI	SI	SI	SI	ACL	SI	NO	SI	BSD
CakePHP	SI	SI	SI	SI	ACL	Memcache, XCache, APC, File.	SI	SI	X11

Tabla 2.1: Comparativa frameworks PHP

2.3.1.4 Comparativa tecnologías del lado del servidor

En este punto se va a realizar una comparativa de las tecnologías Web de lado del servidor teniendo en cuenta el impacto de estas en el desarrollo del proyecto VoCS y su viabilidad económica.

Tecnología	Soporte para SGBD relacionales	Independiente del entorno de desarrollo	Licencia	AJAX	ORM	*Frameworks* de seguridad	*Frameworks* de migración de datos
JSP	SI	NO	GNU GPL	SI	SI	Con complementos	Con complementos
ASP.NET	SI	NO	Propietaria	SI	SI	Con complementos	Con complementos
PHP	SI	SI	PHP	SI	SI	Con complementos	SI

Tabla 2.2: Comparativa tecnologías lado del servidor

2.3.2 Tecnologías del lado del cliente

En esta sección se detallarán las tecnologías que se ejecutan en el cliente.

2.3.2.1 HTML

Hypertext Markup Language (HTML) es el lenguaje estándar utilizado en la Web para representar la información intercambiada por sus usuarios en forma de documentos de hipertexto. Se puede definir este lenguaje como un documento ASCII con una serie de etiquetas, que indican al navegador cómo interpretar y dar formato al texto que las acompaña. HTML es un estándar y esto quiere decir que cualquier navegador que use las normas estándar de visualización de documentos Web podrá leer e interpretar perfectamente HTML. [9] [10].

El lenguaje HTML ofrece una variedad de posibilidades de uso, que son:

1- Publicar documentos en línea con encabezados, textos, tablas, listas, fotos, etc.
2- Obtener información en línea a través de enlaces de hipertexto.
3- Diseño de formularios para realizar transacciones con servicios remotos.
4- Incluir hojas de cálculo, videoclips, clips de sonido y otras aplicaciones directamente en sus documentos.

Las páginas HTML se basan en el uso de marcas o etiquetas. Estas etiquetas comienzan por el símbolo de mayor (“<”) y terminan con el símbolo de menor (“>”). Por ejemplo: <HTML> o <BODY>. La palabra que encierra indica qué acción debe realizar el navegador.

Para delimitar el ámbito en el que debe aplicarse una etiqueta, se debe indicar hasta donde actúa. Para ello, se utiliza la misma etiqueta pero con el símbolo "/" justo delante de la palabra. Por ejemplo: </HTML> o </BODY>.

Toda página HTML comienza y termina con las etiquetas <HTML> </HTML> que delimita el documento. El documento HTML se divide en dos partes bien diferenciadas: cabecera y cuerpo.

1- Cabecera. Se emplea para proporcionar información acerca del documento.
2- Cuerpo. Contiene la información que se va a presentar al usuario.

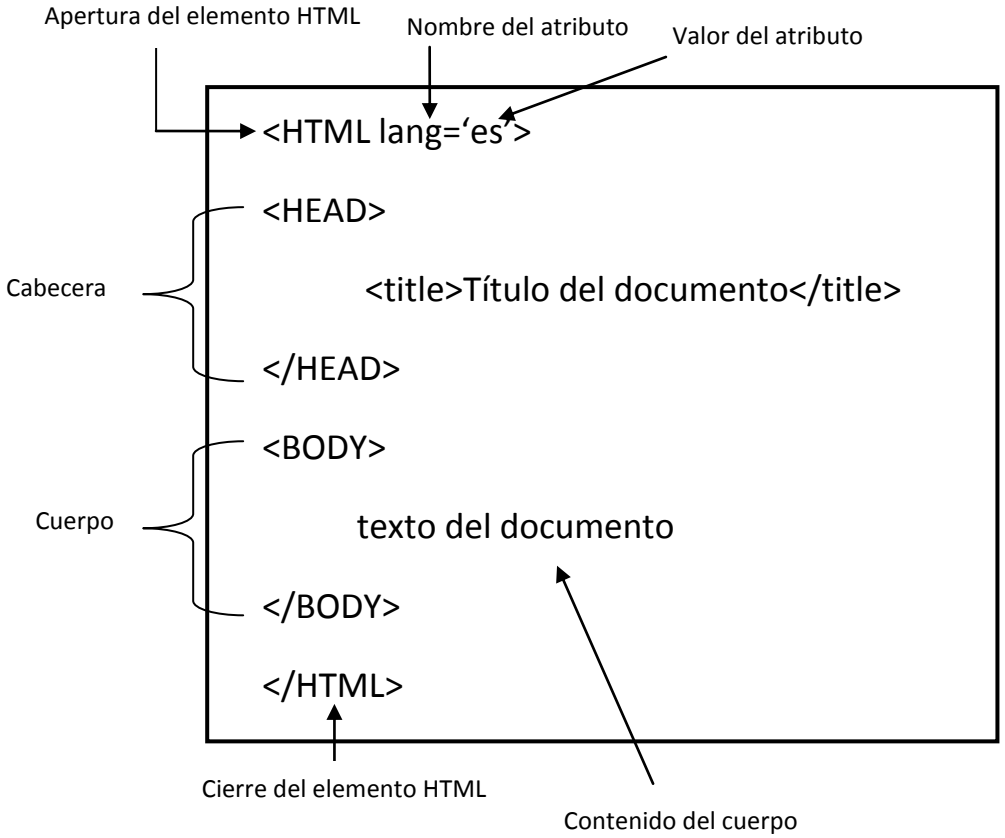


Figura. Estructura de un documento HTML.

2.3.2.2 JavaScript

HTML ofrece una atractiva visualización de la información pero no es capaz por sí solo de ofrecer interactividad a los usuarios. Por esto, fue necesaria la creación de un lenguaje sencillo que permitiera que los usuarios interactúen con la aplicación web. [16].

El objetivo de este lenguaje es crear páginas más atractivas y amigables para el usuario, sin utilizar programación del lado del servidor. Para ello, era necesario que el navegador fuera

capaz de interpretar el código JavaScript. Netscape Communications creó el lenguaje LiveScript. Poco después Sun Microsystems se unió para conseguir que el nuevo lenguaje fuera un estándar de Internet para la escritura de órdenes acerca de la Web. Entonces, LiveScript se renombró a JavaScript.

JavaScript se define como un lenguaje dinámico de scripts, que soporta la creación de prototipos orientado a objetos, es ejecutado en el lado del cliente (navegador) y permite ofrecer una interacción con el usuario que HTML no puede ofrecer.

JavaScript soporta programación de procedimientos y orientada a objetos. Los objetos son creados mediante programación en JavaScript, por la unión de métodos y propiedades. De manera contraria, los objetos vacios se crean en tiempo de ejecución en oposición a las clases en lenguajes compilados. Una vez que el objeto ha sido construido, puede ser usado como prototipo para crear objetos similares. Por lo tanto, JavaScript es un lenguaje interpretado.

Algunas de las posibilidades más avanzadas que ofrece JavaScript son:

1- Uso de la tecnología XML.
2- Uso de la tecnología AJAX.
3- Validaciones de formularios.
4- Captura de cualquier evento del usuario con la aplicación.
5- Creación de interfaces sencillas y atractivas.
6- Creación y manejo de animaciones.
7- Creación de gráficas.
8- Control absoluto de los elementos de la vista.

La programación en JavaScript ofrece dos variedades:

1- JavaScript interno. HTML ofrece una etiqueta”<script></script>” donde se puede incluir el código JavaScript que el navegador ejecutará. Cualquier código JavaScript que esté fuera de estas etiquetas no se considerará como tal.
2- JavaScript externo. El código JavaScript está escrito en un fichero externo, con extensión “.js”. Este fichero es incluido en el documento HTML con la etiqueta “<script></script>”, indicando con el atributo “src” (origen) la ruta del fichero externo. Por ejemplo:

```
<script src=”ruta/myjs.js”></script>
```

Ninguna opción es mejor que otra, solo es una opción de programación.

En cuanto a los desarrollos de JavaScript, se han creado de numerosos *frameworks* como jQuery, Mootols y Dojo. Estos *frameworks* ofrecen una estructura conceptual y tecnológica de soporte, definido con módulos software. Gracias esto, el proyecto puede ser fácilmente organizado y desarrollado.

jQuery

jQuery es una librería JavaScript creada para ayudar a los diseñadores y desarrolladores Web a escribir y extender las interacciones JavaScript rápidamente y de forma concisa, utilizando un conjunto de métodos que envuelven las funciones nativas de JavaScript. jQuery no ofrece ninguna funcionalidad nueva, pero toma las APIs existentes de JavaScript que son difíciles de entender y de escribir, haciéndolas disponibles a una audiencia más amplia a través de una sintaxis fácil de entender y de escribir.

Una de las características que hace de JQuery uno de los frameworks más usados es que no necesita una programación muy difícil para poder manipular el DOM. También tiene áreas avanzadas donde se requiere conocimientos de JavaScript, como trabajar con métodos AJAX, crear propios *plug-ins* jQuery y trabajar con sitios Web móviles.

jQuery fue lanzado bajo la licencia MIT o GNU/GPL versión 2.

Algunas de las posibilidades más avanzadas que ofrece jQuery son:

1- Eventos. Incluyendo ratón, teclado, formularios y interacciones del usuario.
2- Efectos. Incluyendo mostrar/ocultar, deslizamiento, desvanecimiento y propias animaciones.
3- Animaciones, que permiten la utilización de CSS y los efectos nativos.
4- Métodos AJAX, para interactuar con el servidor utilizando XML o JSON.
5- Posibilidad de crear *plug-ins* propios que extienden del núcleo de JQuery.
6- Manipulación DOM.
7- Manipulación CSS.
8- Utilidades de detección de navegador e interfaces más fáciles que las comunes de JavaScript.

El uso de frameworks beneficia el desarrollo de aplicaciones aportando módulos, funcionalidades y abstracciones que sin ellos serían tediosas de realizar. En particular jQuery ofrece estas ventajas:

1- Facilidad.

Escribir JQuery es más fácil que escribir código JavaScript porque JQuery ofrece un API donde muchas de las funcionalidades están basadas en la interactividad con HTML y CSS.

2- Open Source.

Las librerías JavaScript son soportadas por la comunidad de código abierto, que mantienen las librerías probadas y actualizadas.

3- Documentación.

La documentación es completa y bien estructurada. En su página oficial se encuentra el API con numerosos ejemplos.

4- Escribir JavaScript con poco código.

JQuery es JavaScript, todo lo que se puede hacer en JavaScript se puede hacer en JQuery. JQuery ofrece una base sobre la que construir, pero no está limitado a lo que te ofrece. Cuando se usa el framework existen tres opciones para programar:

1- Usar el API de JQuery.
2- Crear un plug-in JQuery.
3- Codificar con JavaScript común.

Otra característica reseñable es el poco código necesario para realizar una operación. Por ejemplo para cambiar el fondo de una elemento del DOM:

1- Usando JavaScript común.

```
document.getElementById('mydiv').style.backgroundColor = 'red';
```

2- Usando JQuery

```
$('#mydiv').css('background-color','red');
```

Como se puede observar, el código es más corto y entendible utilizando JQuery que JavaScript común.

5- Encadenamiento.

JQuery permite encadenar las funciones una tras otra. Esto simplifica el código y lo hace más legible, realizando la acción que se quiere realizar en una sola línea.

6- Compatibilidad con los navegadores.

Compatible con todos los navegadores populares como Internet Explorer 6.0 +, Mozilla Firefox 2 +, Safari 3.0 +, Opera 9.0 +, y Google Chrome.

7- Compatible con CSS3.

JQuery soporta las últimas tecnologías de CSS3 y es por tanto compatible con CSS3.

2.3.2.3 Hojas de Estilo en Cascada

Las Hojas de Estilo en Cascada más conocidas como Cascading Style Sheets (CSS) permiten aplicar formatos a los documentos HTML de una forma eficiente, sencilla, limpia y funcional. Por lo que proporcionan a HTML una forma mucho más flexible de representar la información de las páginas Web. [17].

Con el fin de suplir las limitaciones lingüísticas de HTML, con respecto al control de la presentación, han surgido cantidad de herramientas y técnicas como:

1. Utilización de etiquetas no estándares, inventadas por los creadores de algunos navegadores como Netscape o Explorer.
2. Conversión del texto en imagen.
3. Uso de imágenes transparentes para crear espacios en blanco.
4. Uso de tablas para forzar determinadas presentaciones.
5. Utilización de programas o lenguajes ajenos a HTML para conseguir determinados fines.

La aparición de estas técnicas y herramientas supusieron un gran avance en materia de presentaciones, pero en general han tenido efectos secundarios, aumentando la complejidad de las páginas creadas, limitando la flexibilidad de actualización y modificación de las mismas y, lo más importante de todo, haciendo incompatibles las presentaciones con gran número de navegadores que basan su funcionamiento en estándares, limitando de esta forma el uso de las mismas a un grupo de usuarios muy reducido. Las hojas de estilo sobrepasan la limitada gama de mecanismos de presentación que se han añadido a HTML.

Las hojas de estilo son complementos de código añadido al HTML que se encargan de la apariencia del documento.

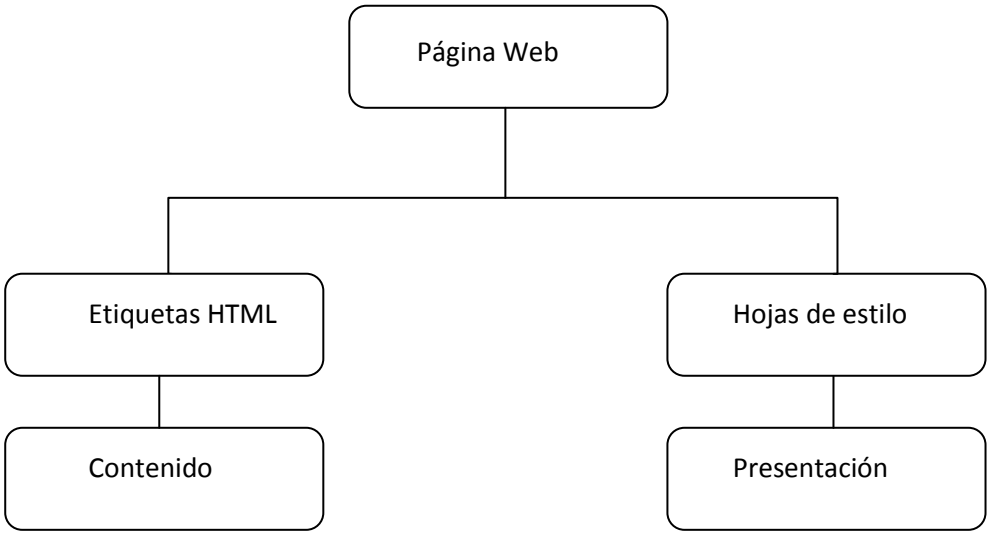


Figura 2.10: HTML con hojas de estilo

interlineados, etc.) hasta la presentación para la impresión de un documento o incluso para su audición a través de interfaces vocales.

El concepto de hojas de estilo reside en la separación entre el contenido y el formato en la elaboración de documentos HTML. Así, un mismo contenido podrá utilizarse, en función de la hoja de estilo adoptada, para la visualización en soportes tan diversos como una pantalla tradicional, la pantalla de un móvil o Smartphone, hojas de papel, un teclado braille, etc. Las hojas de estilo pueden así gestionar todo lo que concierne a la apariencia, dejando a HTML la función de estructura y de codificar la información bruta.

Como se ha visto se han desarrollado técnicas para las presentaciones de documentos HTML, entre ellas el lenguaje CSS. A continuación se explican las ventajas que ofrece CSS respecto a otras técnicas:

1- Aportación de posibilidades de presentación ausentes en el código HTML, algunas de ellas son:
    a. Tamaño de letra ilimitado.
    b. Nuevas formas de bordes.
    c. Control del interlineado y del espaciado.
    d. Sangrado de los textos.
    e. Colocación precisa de imágenes.
    f. Modificación del cursor.
2- Código más simple.

Las hojas de estilo nos permiten aligerar el código haciéndolo más legible y accesible, gracias a la separación del formato y el contenido.

3- Mantenimiento más fácil.

La dimensión de los sitios web ha aumentado considerablemente, existiendo sitios web con cientos de páginas. Una actualización gráfica puede constituir un trabajo gigantesco.

Las hojas de estilo nos permiten con una sola modificación de un único archivo cambiar la presentación gráfica del conjunto de un sitio, asegurando una coherencia gráfica.

4- Accesibilidad.

Las hojas de estilo nos permiten ofrecer recursos más accesibles a personas con trastornos de visión.

2.3.2.4 AJAX

AJAX es una técnica de desarrollo para crear aplicaciones Web basada varias tecnologías ya existentes.

Las tecnologías que forman AJAX son las siguientes:

1- Las páginas Web que son parte de la aplicación Web.

2- El DOM, que permite el acceso a los elementos de una página web, mediante JavaScript y posibilita tratar los elementos como objetos, que poseen propiedades y métodos. También es necesario el objeto XMLHttpRequest definido por JavaScript para el intercambio de asíncrono de información.

3- La transmisión de la información entre las páginas (parte del cliente) y el servidor se hará de manera asíncrona mediante XML. AJAX no solo permite el intercambio de información mediante XML, ya que también permite la utilización de ficheros de texto plano, textos pre-formateados, HTML, JSON y XSLT.

4- JavaScript para unir el resto de tecnologías.

En las aplicaciones Web tradicionales, las acciones del usuario en la página dan lugar a llamadas al servidor. El servidor procesa la petición del usuario y responde con una nueva página HTML al navegador del usuario.

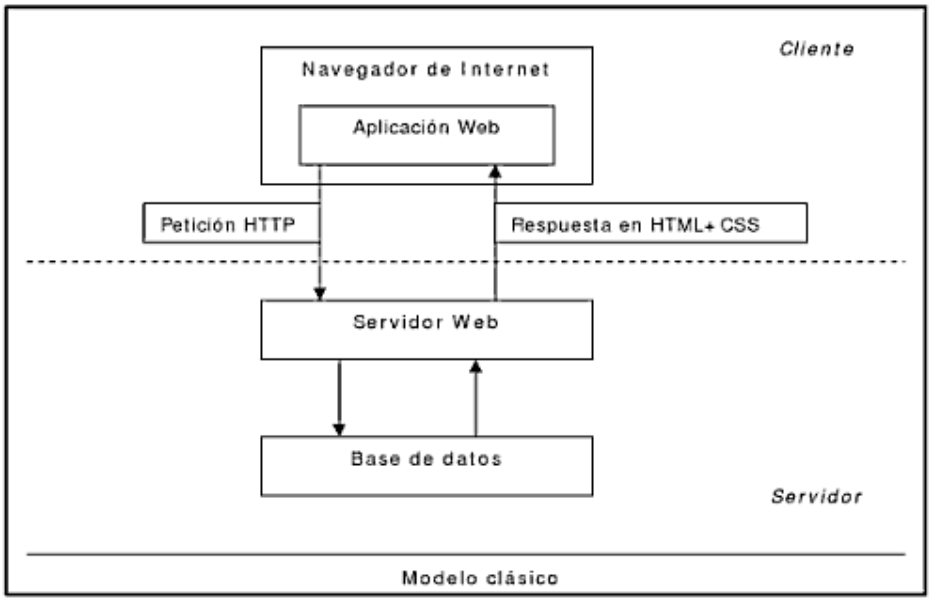


Figura 2.11: Modelo clásico de una aplicación Web

Si una aplicación Web realiza peticiones continuas al servidor puede producir largas esperas, por parte del usuario, a que se recargue la página con los cambios solicitados.

AJAX permite mejorar la interacción con el usuario, evitando recargas constantes realizando la comunicación con el servidor en un segundo plano.

La aplicaciones que utilizan AJAX eliminan la recarga constante de páginas creando una capa intermedia que se sitúa entre el usuario y el servidor. Es esta capa la que hace de intermediario entre las comunicaciones cliente-servidor, y así, el usuario nunca tendrá la ventana del navegador vacía esperando la respuesta del servidor.

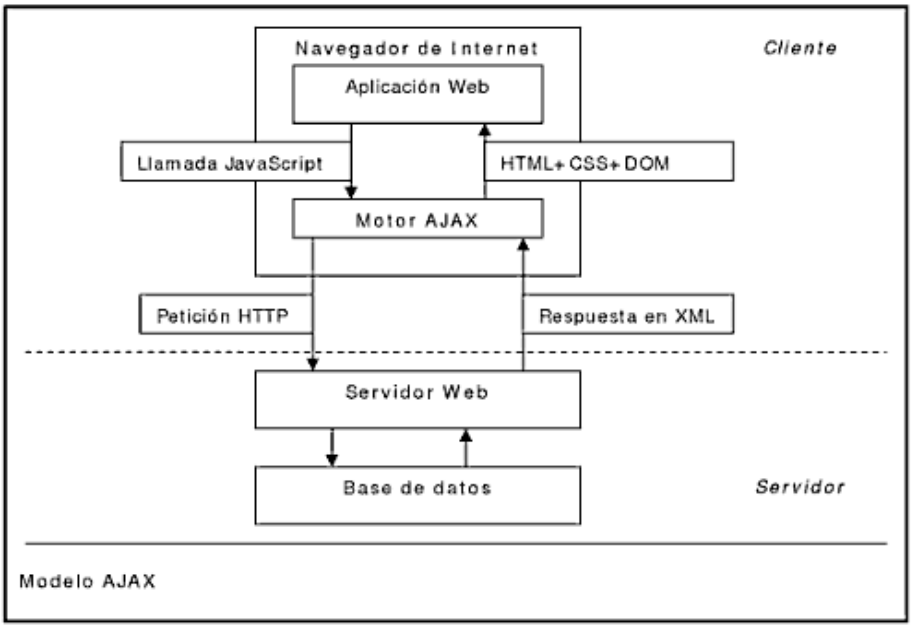


Figura 2.12: Modelo AJAX

Como se explico anteriormente AJAX se basa en comunicaciones asíncronas y el modelo tradicional en comunicaciones síncronas. Las siguientes imágenes muestran como son dichas interacciones.

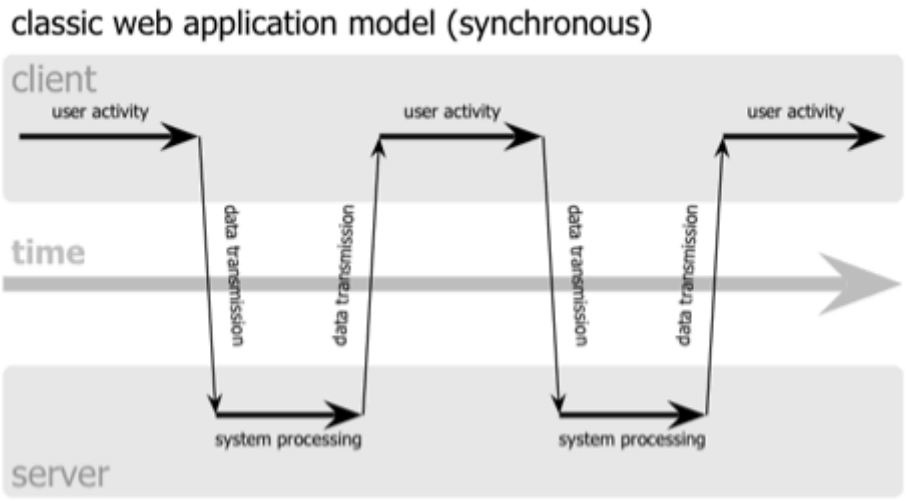


Figura 2.13: Modelo clásico de una aplicación web

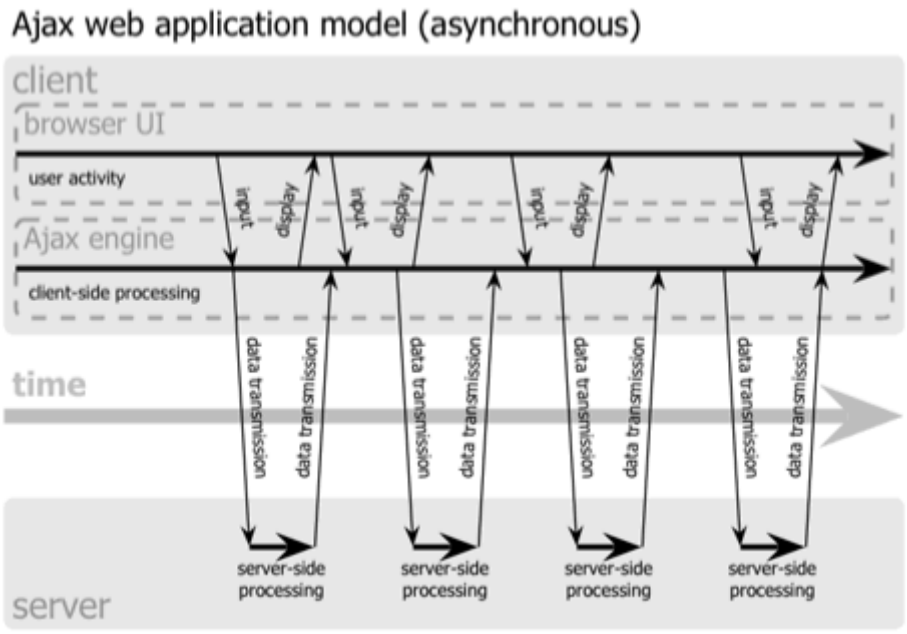


Figura 2.14: Modelo AJAX de una aplicación Web

Para la correcta comprensión de AJAX, a continuación se detallarán los elementos que la forman.

Las tecnologías HTML y CSS proporcionan el medio para presentar los contenidos mediante páginas Web. XML nos proporciona un medio estandarizado para transmitir la información. JavaScript considerado el motor de AJAX, es la tecnología que permite crear comunicaciones asíncronas entre cliente-servidor.

JavaScript permite acceder a los elementos de la página usando DOM y nos posibilita la comunicación utilizando el objeto XMLHttpRequest, implementado en la mayoría de los navegadores.

El objeto XMLHttpRequest es un API implementado en el navegador Web, desarrollado para crear un canal de comunicación independiente entre el servidor y la página Web que se está mostrando en el navegador.

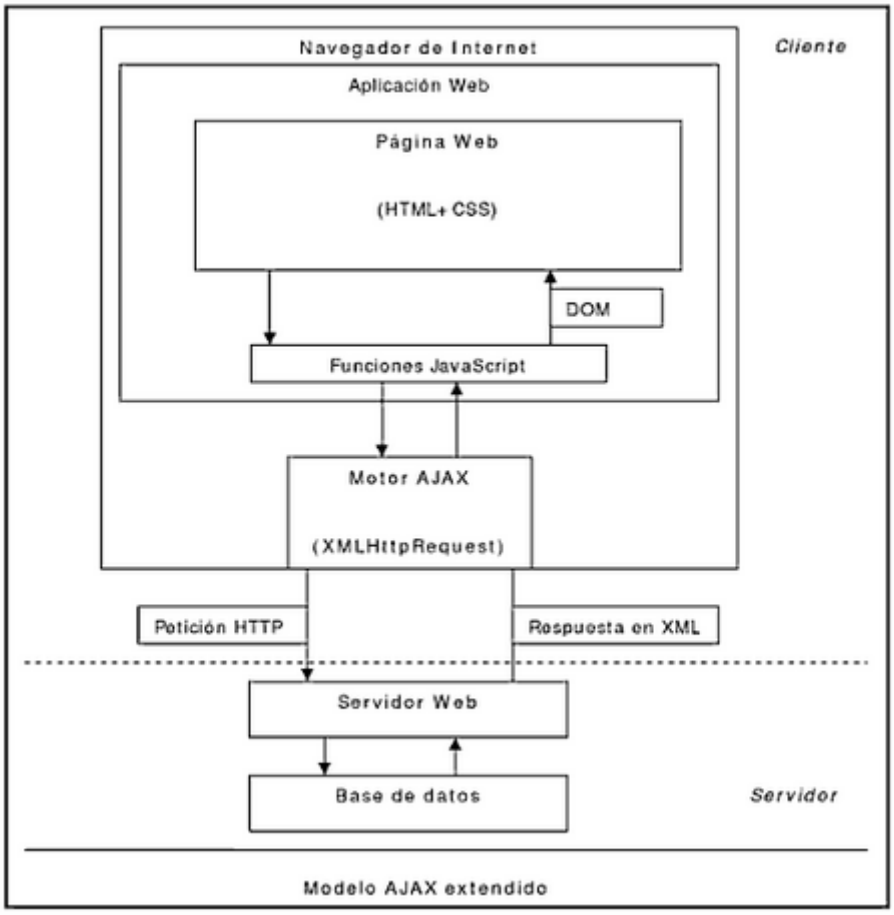


Figura 2.15: Modelo AJAX extendido

En definitiva, las ventajas que ofrece AJAX frente al modelo convencional son:

1- Transparencia.

   La comunicación con el servidor se realiza en segundo plano y de manera asíncrona. Por lo que se puede decir que esta comunicación se realiza de forma transparente al usuario.

2- No recarga.

   Permite crear aplicaciones Web interactivas que puedan cambiar y actualizarse sin necesidad de que dicha página tenga que ser recargada al completo.

3- Rapidez.

   Al no realizar peticiones de toda la página, las peticiones son pequeñas, de ahí la rapidez de respuesta.

4- Portabilidad.

   AJAX se basa en estándares de cada tecnología que usa, por lo que garantiza que las aplicaciones funcionarán en todos los navegadores que implementen dichas tecnologías.

AJAX nos permite crear interfaces de usuario más amenas, interactivas y completas. Estas interfaces proporcionan al usuario información del estado de su solicitud y que puedan acceder de manera más rápida y efectiva a la información.

XML

El primer borrador publicado en noviembre de 1996 por W3C define XML como un dialecto de SGML, diseñado para que su implementación sea simple y pueda inter-operar tanto con SGML como con HTML. La función principal de XML es representar la estructura de la información contenida en el documento, sin datos de proceso, formato o cualquier otro detalle.

La versión 1.0 del lenguaje de XML se publicó como recomendación del W3C el 10 de febrero de 1998, versión que se ha revisado hasta llegar a la tercera edición publicada el 4 de febrero de 2004.

Es en lenguaje extensible que permite crear documentos con marcas a medida para cada necesidad y puede hacer uso de elementos como las DTD para describir los elementos que pueden ser utilizados en dichos documentos.

Ejemplo de documento XML:

```
<?xml versión = "3.0"?>
<LIBROS>
       <LIBRO ISBN ="84-415-14567-1">
              <TITULO>Manual de Word</TITULO>
              <AUTOR> Francisco del Valle</AUTOR>
       </LIBRO>
       <LIBRO ISBN ="84-415-14567-1">
              <TITULO>Manual de Word</TITULO>
              <AUTOR> Francisco del Valle</AUTOR>
       </LIBRO>
</LIBROS>
```

Los elementos están delimitados entre los símbolos "<" y ">", los atributos se introducen en las etiquetas de aperturas y el contenido puede ser una secuencia de caracteres u otros elementos.

Los documentos XML pueden ser utilizados en muchos y diferentes escenarios.

- Comunicación de datos. Por ejemplo en *Web Services*.
- XML como base de datos.
- Ahorro de recursos en consultas de bases de datos.
- Almacenamiento de gráficos vectoriales.
- Formulas matemáticas con XML (*Mathematical Markup Language*)

- Estructuras moleculares e información científica y química (*Chemical Markup Language*).

2.3.3 Servidores Web

Un servidor Web es un computador donde se ejecuta un programa servidor HTTP, por lo que puede denominarse servidor HTTP. Puede utilizarse para publicar un sitio web en Internet, Intranet o Extranet.

Es un computador que ejerce la función de servidor donde se puede instalar más de un tipo de software servidor. Por ejemplo, es frecuente instalar software servidor HTTP y software servidor FTP.

2.3.3.1 Apache

El nombre de Apache viene de A PATCHY Server, es decir, se basa en un código y un conjunto de ficheros parches. El servidor Apache es un servidor Web potente y flexible, compatible con HTTP/1.1. Implementa los protocolos más recientes, incluyendo HTTP/1.1 (RFC2616). Es un servidor modular altamente configurable y extensible con módulos de terceros, permitiendo también la escritura de módulos propios, proporcionando un API de Apache. [20].

Apache Server es de código abierto por lo que proporciona el código fuente completo y viene con una licencia sin restricciones. Es multiplataforma pudiéndose ejecutar en Windows 2000, Netware 5.x o superior, OS/2, y las mayoría de las versiones Unix, así como otros sistemas operativos. El grupo apache está actualmente activo desarrollando constantemente Apache Server.

Implementa muchas funciones de uso frecuente, tales como:

1- Bases de datos DBM, así como bases de datos relacionales y LDAP para la autenticación.
2- Permite configurar fácilmente las páginas protegidas con contraseña con un enorme número de usuarios autorizados, sin atascarse el servidor.
3- Respuestas personalizadas a los errores y problemas.
4- Le permite configurar los archivos o scripts CGI que son devueltos por el servidor en respuesta a los errores y problemas.
5- Varias directivas DirectoryIndex.
6- Reescritura y aliasing de URL flexible e ilimitado.
7- Máquinas virtuales.

8- Permite al servidor distinguir entre las solicitudes realizadas a diferentes direcciones IP o nombres. Apache también ofrece configuración dinámica de virtual hosting.
9- Es posible configurar Apache para generar registros en el formato que desee. Además, en la mayoría de las arquitecturas Unix, Apache puede enviar archivos de registro a una tubería, lo que permite la rotación del registro, filtrado en tiempo real, la división de virtual hosts múltiples en registros separados, y la resolución de DNS asíncronas sobre la marcha.

Apache server está compuesto por un núcleo y un conjunto de módulos, la unión de todo da la funcionalidad de Apache Server. Algunos de los módulos son:

1. Core. Características del núcleo que siempre están disponibles.
2. Mod_rewrite. Proporciona un motor basado en reglas de reescritura para reescribir la URL solicitada sobre la marcha.
3. Mod_ssl. Criptografía fuerte con el Secure Sockets Layer (SSL) y Transport Layer Security (TLS).
4. Mod_dbd. Administra las conexiones de base de datos SQL.

2.3.3.2 Cherokee

En el año 2001 comenzó el proyecto Cherokee, liderado por un desarrollador de software libre llamado Álvaro López. Su motivación fue desarrollar un servidor web nuevo, no tan grande y pesado como los ya existentes. Actualmente es desarrollado y mantenido por una comunidad abierta de desarrolladores. [19].

Cherokee es un servidor web ligero, de alto rendimiento y multi-plataforma, que ofrece un rendimiento nativo para Unix, Linux y Windows. Es muy rápido, flexible y fácil de configurar. Ofrece soporte para las tecnologías difundidas hoy en día:

1. FastCGI, SCGI, PHP, CGI, SSL, TLS y conexiones cifradas SSL.
2. Hosts virtuales.
3. Autenticación.
4. Codificación en caliente.
5. Equilibrio de carga.
6. Los archivos de registro son compatibles con Apache.
7. Balanceador de la base de datos.
8. Actualizaciones sin parada del servicio.
9. Proxy HTTP Inverso.

Se publica bajo la Licencia Pública General de GNU, por lo cual todos y cada uno puede tener acceso al código para su estudio y retoque, lo que le permite personalizar, modificar o ampliar todo lo posible para adaptarse a las necesidades específicas. Actualmente se sigue desarrollando y mejorando Cherokee.

Fue diseñado para ofrecer un servidor Web con unas características de las que Apache carece por diseño inicial, como la rapidez y la capacidad de ser empotrado. Es muy eficiente, muy ligero y proporciona una estabilidad sólida.

Entre sus muchas características, hay una que merece especial crédito: una interfaz de usuario amigable llamada cherokee-admin que se proporciona para una configuración sin problemas de todas las características individuales del servidor. Esta interfaz de administración le permite configurar el servidor web sin tener que preocuparse acerca de cómo modificar un archivo de texto escrito con una sintaxis determinada.

Dispone de Cherokee-market, con el que se puede instalar ciertos servicios de forma muy cómoda y con poca configuración, como instalar un servicio de Drupal, Wordpress, phpBB, phpMyAdmin y algunos otros. También ofrece la posibilidad de Host Virtuales con SSL, ya que Cherokee hace uso del método Server Name Indication (SNI), por el que en la negociación de la sesión TLS se envía el nombre del host que se quiere acceder.

En cuanto al rendimiento de Cherokee, se ha demostrado que está entre los más rápidos del mercado, incluyendo mejoras como la inclusión de un servidor de caché interno.

Cherokee está compuesto por un núcleo y un conjunto de módulos cargables. Hay tres grandes grupos de módulos cargables: handlers (manejadores), encoders (codificadores) y validators (validadores).

1. Handlers, son manejadores de peticiones. Cuando el servidor procesa una petición, decide qué clase de manejador debe de utilizar para responder a dicha petición. Dependiendo del módulo, la respuesta será una u otra. Cherokee incorpora el concepto de asociación de manejadores a directorios, de forma que el usuario puede definir que manejador desea utilizar en cada uno de los directorios servidos por web.
2. Encoders, son módulos que implementan una funcionalidad de conversión de la información que se va a enviar a los clientes. Actualmente, el encoder más útil es el de GZip. Este módulo comprime la información que se sirve antes de enviarla a los clientes, ahorrando ancho de banda y acelerando la transmisión.

3. Los validadores, son los módulos que implementan posibles formas de validar a los usuarios. Cherokee implementa módulos para validar con LDAP, PAM y htpasswd.

2.4 Bases de Datos

Una base de datos es un conjunto de datos pertenecientes al mismo contexto, almacenados de forma organizada y estructurada. Los datos solo pueden ser removidos de la base de datos por una solicitud explicita al Sistema Gestor de Base de Datos, por lo que se puede definir una base de datos como un conjunto de datos persistentes que es utilizado por los sistemas de aplicación. El principal objetivo de los SGDB es la organización de las base de datos, y por lo tanto, rápidas escrituras y lecturas de datos. [3][5]

Los SGBD es un sistema de software que facilita los procesos de definición, construcción, manipulación y compartición de base de datos entre varios usuarios y aplicaciones. Otros objetivos de las SGBD son la protección del sistema contra el defectuoso funcionamiento del hardware o software y la protección del sistema contra accesos no autorizados.

Se define Sistema de Base de Datos a la combinación de base de datos y software SGBD.

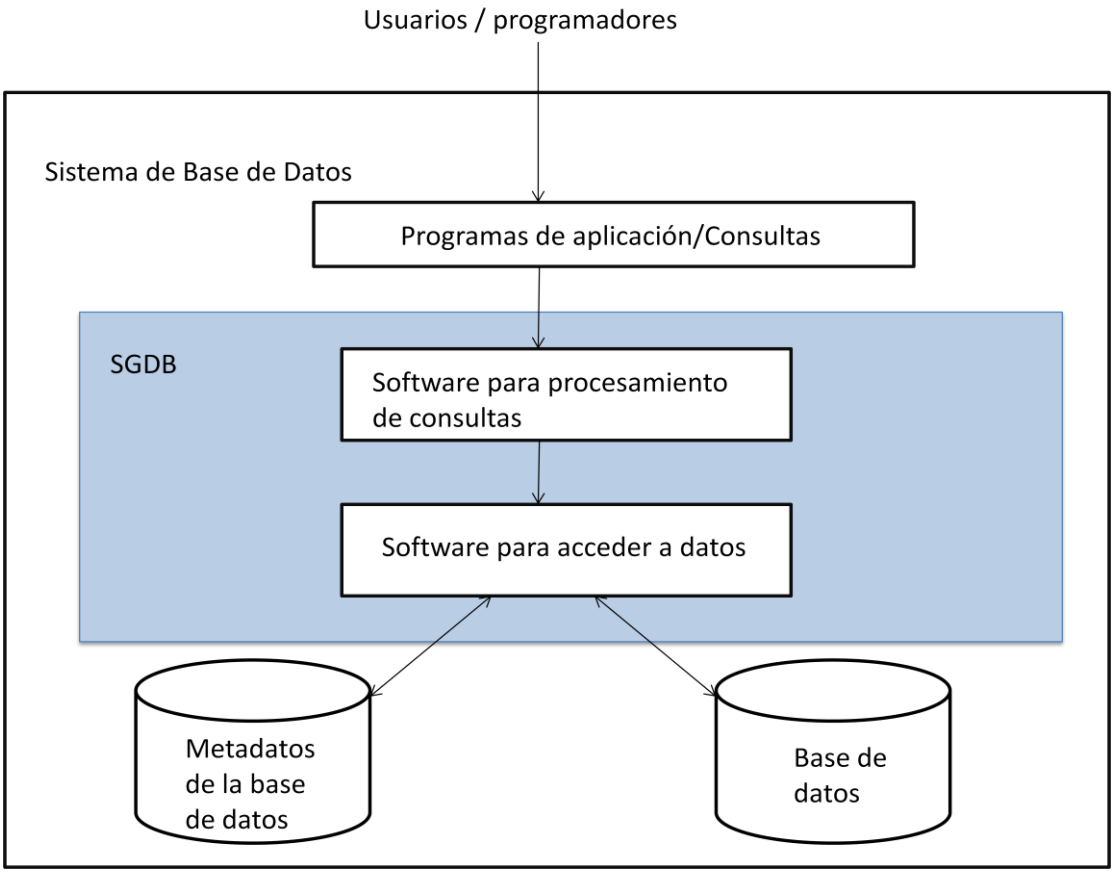


Figura 2.16: Sistema de base de datos

2.4.1 Sistemas de Gestión de Bases de Datos relacionales

Se define un modelo de datos como: un conjunto de conceptos, reglas y convenciones bien definidas que nos permiten aplicar una serie de abstracciones a fin de describir y manipular los datos de un cierto mundo real que deseamos almacenar en la base de datos.

Los modelos se clasifican en:

1- Modelos conceptuales, facilitan la descripción global del conjunto de información al nivel más próximo al usuario, los conceptos son cercanos al mundo real.
2- Modelos convencionales o lógicos, se encuentran instrumentados en la SGBD y orientados a describir los datos a nivel lógico para el SGBD.

Los SGBD relacionales que implementan el modelo relacional son los más vendidos en la actualidad. Las características más importantes que incorpora el modelo relacional son:

1- Sencillez y uniformidad. La estructura fundamental de la base de datos es la tabla, los usuarios ven la base de datos como un conjunto de ellas que unido a lenguajes orientados al usuario final, da como resultado sencillez y uniformidad de los sistemas relacionales.
2- Sólida fundamentación teórica. El modelo está definido con rigor matemático, por lo que es posible realizar el diseño y la evaluación del mismo por métodos sistemáticos basados en abstracciones.
3- Independiente de la interfaz de usuario. Los lenguajes relacionales proporcionan una gran independencia respecto a la forma en la que los datos están almacenados.

Los tres conceptos más importantes para entender el modelo relacional son: dominio, relaciones y atributos.

1- Dominio. Se especifica lógicamente mediante un nombre y un formato. Tiene independencia de las relaciones.
2- Atributo. Es la interpretación de un determinado dominio en una relación.
3- Relación. Es un conjunto de elementos (tuplas) que definen relaciones <atributo, dominio>. Se representa utilizando una tabla donde las columnas son los atributos que expresan las propiedades de la relación (grado de relación). Cada fila (tupla) es un elemento del conjunto que es la relación (cardinalidad).

Los SGBD más importantes son: MySQL, Oracle, PostgreSQL y Sysbase. Todos ellos utilizan SQL para manipular la información de la base de datos.

Figura 2.17: Tablas en modelo relacional

2.4.1.1 MySQL

Es un sistema de administración de base de datos relacionales rápido, robusto y fácil de usar. Se adapta bien a la administración de datos en un entorno de red, especialmente en arquitecturas cliente-servidor. [6]

MySQL es propiedad de Oracle Corporation, empresa multinacional de gran prestigio. Oracle Corporation ofrece MySQL con licencia pública que permite la utilización del programa, consulta y modificación del código fuente del mismo. Puede ser descargado y usado bajo la licencia GNU GPL, aunque para los usuarios que deseen utilizarlo en productos más privativos pueden contratar una licencia que se lo permita.

Su velocidad y fiabilidad han convertido en una popular alternativa a los sistemas de bases de datos propietarias. MySQL es uno de los gestores de bases de datos más importantes de la actualidad, siendo utilizado por pequeñas empresas y también por grandes empresas como: la NASA, Google, CISCO.

2.4.1.2 Oracle Database

Es un sistema de administración de base de datos relacionales propiedad de Oracle Corporation. Existen varias versiones del producto:

1. Enterprise Edition a un precio de $47,500 por procesador.
2. Standard Edition a un precio de $17,500 por procesador.
3. Standart Edition One a un precio de $5,800 por procesador.

Oracle Database proporciona una única plataforma integrada ofreciendo seguridad, alto rendimiento y escalabilidad de las tecnologías más populares utilizadas por los desarrolladores de aplicaciones. Oracle Database añade nuevas capacidades a todos los ambientes principales de desarrollo de aplicaciones, lo que le permite aumentar la productividad del desarrollador y acortar el tiempo de comercialización.

Oracle es el referente en sistemas de gestores de base de datos.

2.4.1.3 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es uno de los sistemas de gestión de bases de datos de código abierto más potentes del mercado.

PostgreSQL, basado en POSTGRES fue desarrollado en la Universidad de California en Berkeley Departamento de Informática. POSTGRES fue pionero en muchos conceptos que sólo estuvo disponible en algunos sistemas de bases de datos comerciales mucho más tarde.

Soporta gran parte del estándar SQL: 2003 y ofrece muchas otras características modernas:

1. Consultas complejas
2. Claves externas
3. Triggers
4. Vistas.
5. Integridad de las transacciones
6. Control de concurrencia multi-versión.

También se pueden añadir otros módulos no oficiales gracias al grupo de usuarios y productos comerciales.

2.4.1.4 Comparativa

En este apartado se va a realizar una comparativa de los tres gestores de bases de datos mencionados. [7][8]

	MySQL	PostgreSQL	Oracle
Datos almacenados Modelos de almacenamiento Confiabilidad Escalabilidad	 MyISAM, InnoDB, Berkeley DB, full-text. Alta/Muy alta Larga/Muy larga	 Postgres. Alta Larga/Muy Larga	 Bitmapped, B-tree, IOT, function-based. Alta/Muy alta Larga/Muy larga
Índices Individual y múltiple columna, clave primaria, y full text.	SI	SI	SI
Integridad de datos Requerimientos ACID, bloqueo a nivel de fila, backup en caliente, y rollback parcial.	SI	SI	SI
Replicación Máster individual Múltiple Máster Clustering	 SI SI SI	 SI SI SI	 SI SI SI
Interfaces ODBC/JDBC, C/C++, y Java	SI	SI	SI
Características avanzadas Stored procedures, vistas, triggers, secuencias, y cursores.	SI	SI	SI
Integridad con otras plataformas o lenguajes. Sistemas operativos	AIX, Amiga, BD, FreeBSD, HP-UX,Kurisu OS, GNU/Linux, Mac OS x, NetBSD, OpenBSD, Solaris, SunOS, Digital Unix, NetBSD, NetWare, OpenBSD, Warp, SCO OpenServer, SCO UnixWare, SCO OpenServer, SGI Irix, Solaris, SunOS, Tru64 Unix, Windows.	Windows, Linux y Unix.	Windows, Linux, Unix.

Tabla 2.3: Comparativa Sistemas Bases de Datos relacionales

2.5 Seguridad

En este apartado se van a mencionar las tecnologías relevantes para la seguridad en aplicaciones Web.

2.5.1 SSL

El protocolo de transporte Secure Sockets Layer fue desarrollado por Netscape Communications, a principios de los años 90. La primera versión de este protocolo ampliamente difundida e implementada fue la 2.0. Poco después Netscape publicó la versión 3.0, con muchos cambios respecto a la anterior, que hoy ya casi no se utiliza. [20] [21].

La especificación Transport Layer Security fue elaborada por la IETF. La versión 1.0 del protocolo TLS está publicada en el documento RFC 2246. Es prácticamente equivalente a SSL 3.0 con algunas pequeñas diferencias, por lo que en ciertos contextos se considera el TLS 1.0 como si fuera el protocolo "SSL 3.1".

El objetivo inicial del diseño del protocolo SSL fue proteger las conexiones entre clientes y servidores web con el protocolo HTTP. Esta protección debía permitir al cliente asegurarse que se había conectado al servidor auténtico, y enviarle datos confidenciales.

Las funciones de seguridad no se implementaron directamente en el protocolo de aplicación HTTP, si no que se optó por introducirlas a nivel de transporte. De este modo podría haber muchas más aplicaciones que hicieran uso de esta funcionalidad.

Con este fin se desarrolló una interfaz de acceso a los servicios del nivel de transporte basada en la interfaz estándar de los sockets. En esta nueva interfaz, funciones como connect, accept, send o recv fueron sustituidas por otras equivalentes pero que utilizaban un protocolo de transporte seguro: SSL_connect, SSL_accept, SSL_send, SSL_recv, etc. El diseño se realizó de tal modo que cualquier aplicación que utilizara TCP a través de las llamadas de los sockets podía hacer uso del protocolo SSL solamente cambiando estas llamadas. De aquí proviene el nombre del protocolo.

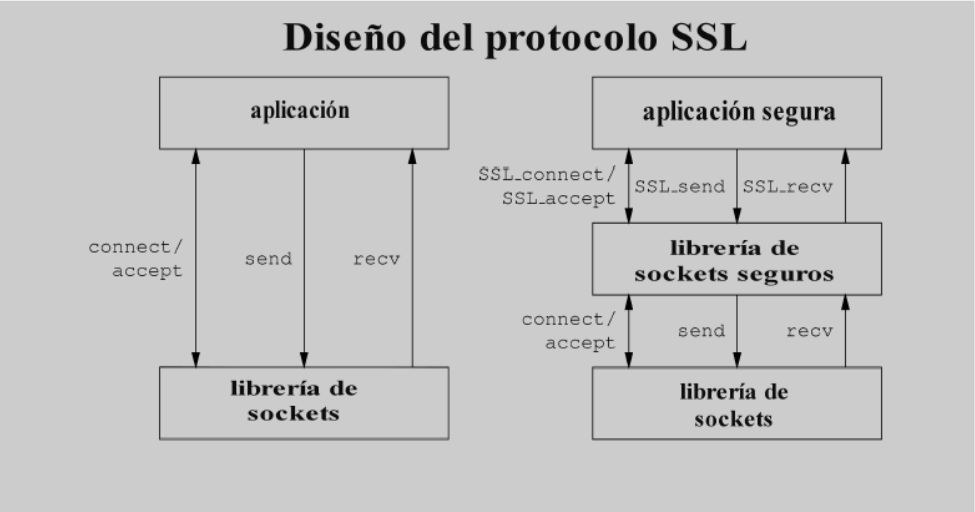


Figura 2.18: Diseño del protocolo SSL

Los servicios de seguridad que proporcionan los protocolos SSL/TLS son:

- Confidencialidad. El flujo normal de información en una conexión SSL/TLS consiste en intercambiar paquetes con datos cifrados mediante claves simétricas. Al inicio de cada sesión, cliente y servidor se ponen de acuerdo en que claves utilizarán para cifrar los datos. Siempre se utilizan dos claves distintas: una para los paquetes enviados del cliente al servidor, y la otra para los paquetes enviados en sentido contrario.
- Para evitar que un intruso que esté escuchando el diálogo inicial pueda saber cuáles son las claves acordadas, se sigue un mecanismo seguro de intercambio de claves, basado en criptografía de clave pública. El algoritmo concreto para este intercambio también se negocia durante el establecimiento de la conexión.
- Autenticación de entidad. Con un protocolo de reto-respuesta basado en firmas digitales el cliente pude confirmar la identidad del servidor al cual se ha conectado. Para validar las firmas el cliente necesita conocer la clave pública del servidor, y esto normalmente se realiza a través de certificados digitales.
- SSL/TLS también provee la autenticación del cliente frente al servidor. Esta posibilidad no se usa tan a menudo porque muchas veces, en lugar de autenticar automáticamente el cliente a nivel de transporte, las mismas aplicaciones utilizan su propio método de autenticación.
- Autenticación de mensaje. Cada paquete enviado en una conexión SSL/TLS, además de ir cifrado, puede incorporar un código MAC para que el destinatario compruebe que nadie ha modificado el paquete. Las claves secretas para el cálculo de los códigos MAC (una para cada sentido) también se acuerdan de forma segura en el diálogo inicial.

Los protocolos SSL/TLS también están diseñados para ofrecer las siguientes características:

- Eficiencia. Dos de las características de SSL/TLS, la definición de sesiones y la compresión de los datos, permiten mejorar la eficiencia de la comunicación.
    - Si el cliente pide dos o más conexiones simultáneas o muy seguidas, en lugar de repetir la autenticación y el intercambio de claves (operaciones computacionalmente costosas porque intervienen algoritmos de clave pública), hay la opción de reutilizar los parámetros previamente acordados. Si se hace uso de esta opción, se considera que la nueva conexión pertenece a la misma sesión que la anterior. En el establecimiento de cada conexión se especifica un identificador de sesión, que permite saber si la conexión empieza una sesión nueva o es continuación de otra.
    - SSL/TLS prevé la negociación de algoritmos de compresión para los datos intercambiados, para compensar el tráfico adicional que introduce la seguridad. Pero ni SSL 3.0 ni TLS 1.0 especifican ningún algoritmo concreto de compresión.
- Extensibilidad. Al inicio de cada sesión, cliente y servidor negocian los algoritmos que utilizarán para el intercambio de claves, la autenticación, el cifrado y la compresión. Las especificaciones de los protocolos incluyen unas combinaciones predefinidas de algoritmos criptográficos, pero dejan abierta la posibilidad de añadir nuevos algoritmos si se descubren otros que sean más eficientes o más seguros.

A continuación se va a explicar el funcionamiento de SSL para garantizar una comunicación segura.

La capa de transporte seguro que proporciona SSL/TLS se puede considerar dividida en dos subcapas.

- La subcapa superior se encarga básicamente de negociar los parámetros de seguridad y de transferir los datos de la aplicación. Tanto los datos de negociación como los de aplicación se intercambian en mensajes.
- En la subcapa inferior, estos mensajes son estructurados en registros a los cuales se les aplica, según corresponda, la compresión, la autenticación y el cifrado.

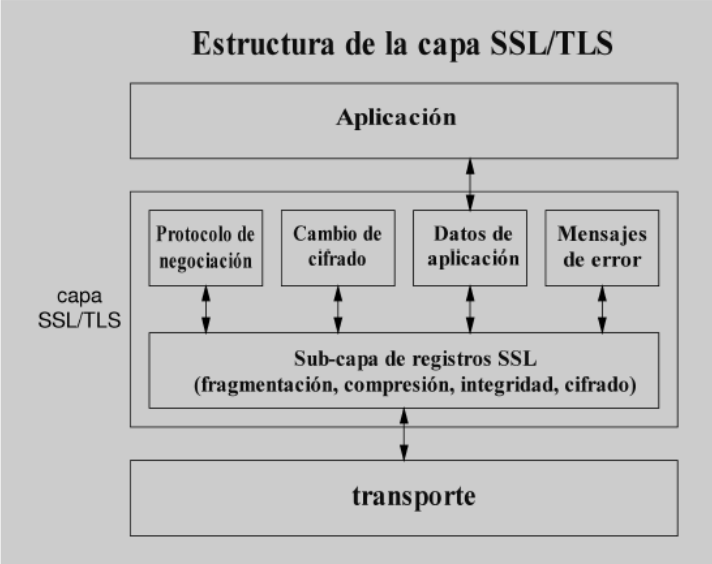


Figura 2.19: Estructura de la capa SSL/TLS

El protocolo de registros SSL/TLS es el que permite que los datos protegidos sean convenientemente codificados por el emisor y interpretados por el receptor. Los parámetros necesarios para la protección, como pueden ser los algoritmos y las claves, se establecen de forma segura al inicio de la conexión mediante el protocolo de negociación SSL/TLS.

- Protocolo de registros. La información que se intercambian cliente y servidor en una conexión SSL/TLS se empaqueta en registros, que tienen el formato mostrado en la Figura 2.20.

Figura 2.20: Formato de los registros SSL/TLS

  - El primer campo indica cual es el tipo de contenido de los datos, que puede ser: un mensaje del protocolo de negociación, notificación de cambio de cifrado, un mensaje de error o datos de aplicación.
  - El segundo campo son dos bytes que indican la versión del protocolo.
  - El tercer campo indica el tamaño del resto del registro.

- El cuarto campo son los datos, comprimidos si se ha acordado algún algoritmo de compresión.
- El quinto campo es el código de autenticación MAC. En el cálculo de este MAC intervienen la clave MAC, un número de secuencia implícito de 64 bits y el contenido del registro.

Si se ha acordado utilizar un algoritmo en bloque para cifrar los datos, es preciso añadir bytes adicionales (*padding*) a cada registro para tener un número total que sea múltiple de la longitud del bloque. La técnica que se usa para saber cuántos bytes adicionales hay es poner al menos uno, y el valor del último byte siempre indica cuantos otros bytes de *padding* hay antes (este valor puede ser 0 si sólo faltaba un byte para tener un bloque entero).

El protocolo de registros SSL/TLS se encarga de formar cada registro con sus campos correspondientes, calcular el MAC, y cifrar los datos.

- Protocolo de negociación. El protocolo de negociación SSL/TLS, también llamado *Handshake Protocol*, tiene por finalidad autenticar el cliente y/o el servidor, y acordar los algoritmos y claves que se utilizaran de forma segura, es decir, garantizando la confidencialidad y la integridad de la negociación.

  Los mensajes que se envían cliente y servidor en el protocolo de negociación son:

  1- Petición de saludo.

  Cuando se establece una conexión, el servidor normalmente espera que el cliente inicie la negociación. Alternativamente, puede optar por enviar un mensaje *Hello Request* para indicar al cliente que está preparado para empezar. Si durante la sesión el servidor quiere iniciar una renegociación, también lo puede indicar al cliente enviando un mensaje de este tipo.

  2- Saludo del cliente. El cliente envía un mensaje *Client Hello* al inicio de la conexión o como respuesta a un *Hello Request*. Este mensaje contiene:

  - La versión del protocolo que el cliente quiere utilizar.
  - Una cadena de 32 bytes aleatorios.
  - Opcionalmente, el identificador de una sesión anterior, si el cliente desea volver a utilizar los parámetros que se han acordado.

- La lista de las combinaciones de algoritmos criptográficos que el cliente ofrece utilizar, por orden de preferencia. Cada combinación incluye el algoritmo de cifrado, el algoritmo de MAC y el método de intercambio de claves.
- La lista de los algoritmos de compresión ofrecidos, por orden de preferencia (como mínimo debe haber uno, aunque sea el algoritmo nulo).

3- Saludo del servidor.

Como respuesta, el servidor envía un mensaje *Server Hello*, que contiene esta información:

- La versión del protocolo que se usará en la conexión. La versión será igual a la que envió el cliente, o inferior si esta no es soportada por el servidor.
- Otra cadena de 32 bytes aleatorios.
- El identificador de la sesión actual.
- La combinación de algoritmos criptográficos escogida por el servidor de entre la lista de las enviadas por el cliente.
- El algoritmo de compresión escogido por el servidor

4- Certificado del servidor.

Si el servidor puede autenticarse frente al cliente, que es el caso más habitual, envía el mensaje *Certificate*. Este mensaje normalmente contendrá el certificado X.509 del servidor, o una cadena de certificados. Si el servidor no tiene certificado, o se ha acordado un método de intercambio de claves que no precisa de él, debe mandar un mensaje *Server Key Exchange*, que contiene los parámetros necesarios para el método a seguir.

5- Petición de certificado.

En caso que se deba realizar también la autenticación del cliente, el servidor le envía un mensaje *Certificate Request*. Este mensaje contiene una lista de los posibles tipos de certificado que el servidor puede admitir, por orden de preferencia, y una lista de los DN de las autoridades de certificación que el servidor reconoce.

6- Fin de saludo del servidor.

Para terminar esta primera fase del diálogo, el servidor envía un mensaje Server Hello Done.

7- Certificado del cliente.

Una vez el servidor ha mandado sus mensajes iniciales, el cliente ya sabe como continuar el protocolo de negociación. En primer lugar, si el servidor le ha pedido un certificado y el cliente tiene alguno de las características solicitadas, lo envía en un mensaje *Certificate*.

8- Intercambio de claves del cliente.

El cliente envía un mensaje Client Key Exchange, el contenido del cual depende del método de intercambio de claves acordado. En caso de seguir el método RSA, en este mensaje hay una cadena de 48 bytes que se usará como secreto pre-maestro, cifrada con la clave pública del servidor.

Entonces, cliente y servidor calculan el **secreto maestro**, que es otra cadena de 48 bytes. Para realizar esta cálculo, se aplican funciones *hash* al secreto pre-maestro y a las cadenas aleatorias que se enviaron en los mensajes de saludo.

A partir del secreto maestro y las cadenas aleatorias, se obtienen:

- Las dos claves para el cifrado simétrico de los datos (una para cada sentido: de cliente a servidor y de servidor a cliente).
- Las dos claves MAC (también una para cada sentido).
- Los dos vectores de inicialización para el cifrado, si se utiliza un algoritmo en bloque.

9- Verificación del certificado.

Si el cliente ha mandado un certificado en respuesta a un mensaje Certificate Request, ya puede autenticarse demostrando que posee la clave privada correspondiente mediante un mensaje Certificate Verify. Este mensaje contiene una firma, generada con la clave privada del cliente, de una cadena de bytes obtenida a partir de la concatenación de todos los mensajes de negociación intercambiados hasta el momento, desde el Client Hello hasta el Client Key Exchange.

10- Finalización.

A partir de este punto ya se pueden utilizar los algoritmos criptográficos negociados. Cada parte manda a la otra una notificación de cambio de

cifrado seguida de un mensaje *Finished*. La notificación de cambio de cifrado sirve para indicar que el siguiente mensaje será el primer enviado con los nuevos algoritmos y claves.

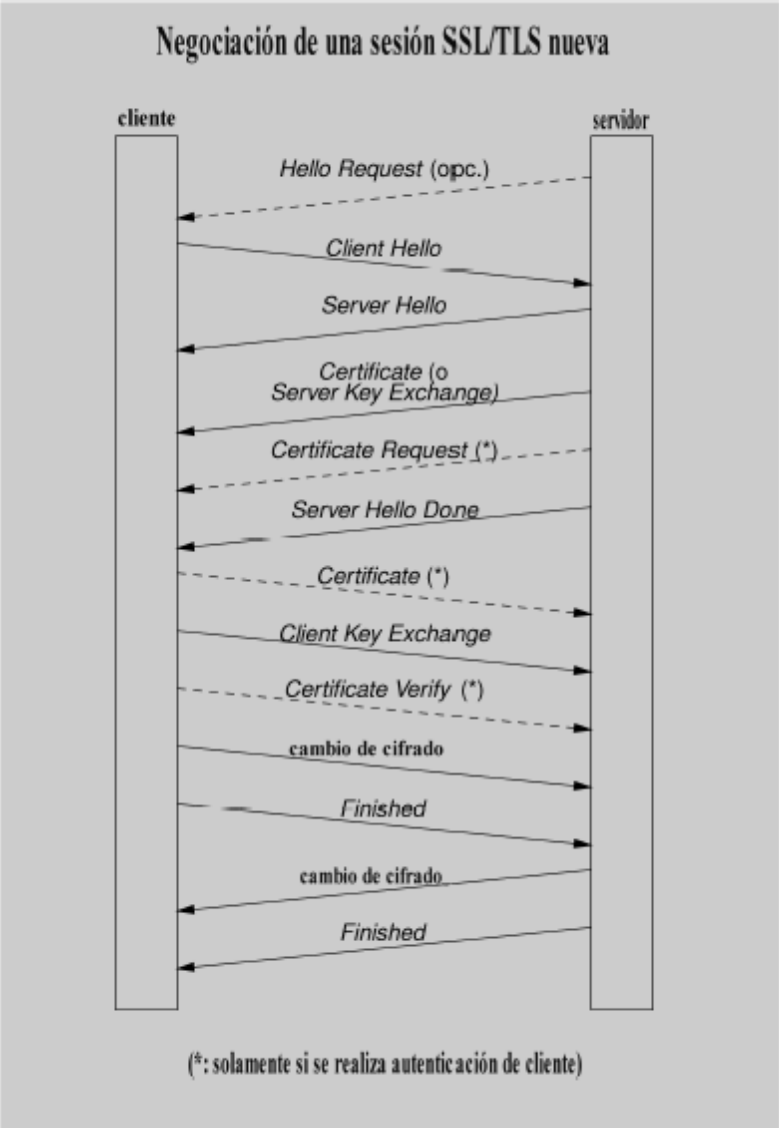


Figura 2.21: Negociación SSL/TLS

2.5.2 Cifrado

En la criptografía moderna hay que tener en cuenta a Claude Shannon, autor de dos artículos fundamentales "Teoría matemática de la comunicación" y "Teoría de las comunicaciones secretas". A partir de estos dos artículos la criptografía queda establecida como una ciencia de base matemática e íntimamente ligada con la informática. Posteriormente la popularización del ordenador personal y de Internet ha obligado al desarrollo de una criptografía civil, que durante unos años ha seguido el mismo camino que la militar, pero con un retraso.

Cifrar es transcribir en guarismos, letras o símbolos, de acuerdo con una clave, un mensaje cuyo contenido se quiere ocultar.

Un esquema de cifrado es computacionalmente seguro si el texto cifrado generado cumple uno o los dos criterios siguientes:

- El coste de "romper" el cifrado excede el valor de la información cifrada.
- El tiempo necesario para "romper" el cifrado excede el tiempo de vida útil de la información.

2.5.2.1 AES

Advanced Encryption Standard o estándar de cifrado avanzado. El NIST en 1997 realizó un concurso de propuestas para el desarrollo de un nuevo estándar de cifrado avanzado, que debía ser tan robusto o más que 3DES y que mejorara significativamente la eficiencia. Además de estos requisitos generales, el NIST especifico que el AES debía ser un cifrador simétrico de bloque, con una longitud de bloque de 128 bits y permitiera longitudes de clave de 128, 192 y 256 bits. Los criterios de evaluación incluyen la seguridad, la eficiencia computacional, los requisitos de memoria, la idoneidad para hardware y software y la flexibilidad. [22] [23].

En la primera etapa de evaluación, se aceptaron quince de los algoritmos propuestos. La segunda etapa los redujo a cinco. El NIST completo el proceso de evaluación y publico el estándar final (FIPS PUB 197) en Noviembre de 2001. El algoritmo seleccionado por el NIST fue el Rijndael , desarrollado y presentado por dos criptógrafos belgas: Dr. Joan Daemen y Dr. Vincent Rijmen.

El estándar de cifrado avanzado (AES) especifica un algoritmo criptográfico que se puede utilizar para proteger los datos electrónicos. El algoritmo AES es un cifrado de bloques simétrico que puede cifrar y descifrar la información.

El algoritmo AES es capaz de utilizar las claves de cifrado de 128, 192 y 256 bits para cifrar y descifrar los datos en bloques de 128 bits. El tamaño de clave da lugar al los llamados: AES-128, AES-192 y AES-256.

AES, como cifrador de bloque, consiste en someter a los bloques de texto en claro una serie de operaciones que se repiten cierto número de rondas. El proceso de cifrado de cada bloque de 128 bits de texto en claro sigue los siguientes tres pasos:

1- Se efectúa una suma XOR entre bloque de texto en claro y los 128 primeros bits de la clave. El resultado se parte en 16 bytes y se forma con ellos una matriz 4x4.

2- Esta matriz de bytes se somete a una serie de rondas de idénticas operaciones. El número de rondas depende del tamaño de clave: 9 si la calve es de 128 bits, 11 si es de 192 y 13 si es de 256. Las operaciones son las siguiente:

1. Cada byte se transforma en otro por la acción de una <<caja>>. Más adelante se explica cómo actúa esta caja.
2. Se efectúa una permutación cíclica de los bytes de las filas 2º, 3º y 4º de la matriz, desplazándolos 1,2 y 3 posiciones a la izquierda, respectivamente.
3. Las columnas se someten a la siguiente transformación de Hill:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} b & c & a & a \\ a & b & c & a \\ a & a & b & c \\ c & a & a & b \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

Donde a, b y c son los bytes: a= 00000001, b= 00000010 y c = 00000011.

4. La matriz resultante se suma con otra matriz de bytes de tamaño 4x4 formada a partir de la clave. Esta matriz es distinta en cada ronda, ha sido obtenida y almacenada antes de empezar a cifrar y su cómputo se efectúa mediante un algoritmo que depende de la longitud de la clave y en el que interviene la <<caja>> del apartado a).

Queda describir la sustitución de bytes del apartado a), la única <<caja>> presente en AES. Esta caja transforma un byte en otro realizando estas dos operaciones:

1. Cada byte se reemplaza por su inverso para el producto. Como el byte nulo no tiene inverso, se deja invariante.
2. Se sustituye el byte resultante del primer paso por otro byte resultado de realizar las siguientes sumas de bits.

$$\begin{cases} b_7 = a_7 \oplus a_6 \oplus a_5 \oplus a_4 \oplus a_3 \\ b_6 = a_6 \oplus a_5 \oplus a_4 \oplus a_3 \oplus a_2 \oplus 1 \\ b_5 = a_5 \oplus a_4 \oplus a_3 \oplus a_2 \oplus a_1 \oplus 1 \\ b_4 = a_4 \oplus a_3 \oplus a_2 \oplus a_1 \oplus a_0 \\ b_3 = a_7 \oplus a_3 \oplus a_2 \oplus a_1 \oplus a_0 \\ b_2 = a_7 \oplus a_6 \oplus a_2 \oplus a_1 \oplus a_0 \\ b_1 = a_7 \oplus a_6 \oplus a_5 \oplus a_1 \oplus a_0 \oplus 1 \\ b_0 = a_7 \oplus a_6 \oplus a_5 \oplus a_4 \oplus a_0 \oplus 1 \end{cases}$$

Al no haber debilidades matemáticas inherentes en el algoritmo, lo que procede es un enfoque de fuerza bruta para "romper" el cifrado. La tabla 2.22 muestra el tiempo necesario para romper el cifrado por fuerza bruta con distintos tamaños de clave. Como se puede observar la clave de 56 bits que utiliza DES, en un sistema que pueda procesar un millón de claves por microsegundos, dejaría de ser computacionalmente seguro. En cambio, con los tamaños de clave que utiliza AES, se garantiza que será computacionalmente seguro ahora y en el futuro.

Tamaño de clave (bits)	Número de claves alternativas	Tiempo necesario a 1 cifrado/μs	Tiempo necesario a 10^6 cifrados/μs
32	$2^{32} = 4{,}3 \times 10^9$	2^{31} μs = 35,8 minutos	2,15 milisegundos
56	$2^{56} = 7{,}2 \times 10^{16}$	2^{55} μs = 1.142 años	10,01 horas
128	$2^{128} = 3{,}4 \times 10^{38}$	2^{127} μs = $5{,}4 \times 10^{24}$ años	$5{,}4 \times 10^{18}$ años
168	$2^{168} = 3{,}7 \times 10^{50}$	2^{167} μs = $5{,}9 \times 10^{36}$ años	$5{,}9 \times 10^{30}$ años
26 caracteres (permutación)	$26! = 4 \times 10^{26}$	2×10^{26} μs = $6{,}4 \times 10^{12}$ años	$6{,}4 \times 10^6$ años

Figura 2.22: Tiempos de rotura de cifrados dependiendo del tamaño de la clave

2.5.3 Funciones Resumen

[24] Una función resumen (función hash) es una función que toma una entrada (mensaje) de cualquier tamaño y genera una cadena de bits de longitud fija n. Matemáticamente, una función hash es una función:

$$H:\{0,1\}^* \rightarrow \{0,1\}^n,\ m \rightarrow h(m)$$

La longitud de n está entre 128 y 512 bits.

Para que una función se considere función hash debe cumplir con estos requisitos:

- La función hash puede aplicarse a un bloque de datos de cualquier tamaño.
- La función hash produce una salida de tamaño fijo.
- La función Hash: H(x) es relativamente fácil de computar para cualquier x dado, haciendo que tanto las implementaciones de hardware como de software sean prácticas.
- Para cualquier valor h dado, es imposible desde el punto de vista computacional encontrar x tal que H(x) = h, se conoce como propiedad unidireccional.
- Para cualquier bloque dado x, es imposible desde el punto de vista computacional, encontrar y != x con H(y) = H(x), lo que se conoce como resistencia débil a la colisión.

- Es imposible desde el punto de vista computacional encontrar un par (x, y) tal que H(x) = H(y), lo que normalmente se conoce como resistencia fuerte a la colisión.

Una función hash que cumpla con las cinco primeras propiedades se denomina función hash débil. Si también posee la sexta propiedad, se denomina función hash robusta.

Hasta hace poco, el algoritmo de hash más usado era el MD5 (Message Digest 5). Pero como el resumen que da es de sólo 128 bits, y aparte se han encontrado otras formas de generar colisiones parciales en el algoritmo, actualmente se recomienda utilizar algoritmos más seguros, como el SHA-1 o SHA-2.

2.5.3.1 SHA

A la familia de funciones SHA pertenecen las funciones SHA-0 publicado en 1993 por el NIST, el algoritmo SHA-1 publicado el 1995 en un estándar del NIST (como revisión de un algoritmo anterior) y SHA-2 publicado en el año 2002 por el NIST. [25].

En 2005, las fallas de seguridad fueron identificadas en SHA-1, es decir, que una debilidad matemática podría existir, lo que provocó la búsqueda de un algoritmo mejor.

SHA-2 consta de un conjunto de cuatro funciones hash con resúmenes de 224, 256, 384 o 512 bits. No se han encontrado fallos de seguridad en SHA-2 y es considerada actualmente una de las funciones más seguras.

A continuación se presentan las características más importantes de los algoritmos de la familia SHA:

Algoritmo	Tamaño de salida	Tamaño estado interno	Tamaño de bloque	Tamaño máximo de mensaje	Tamaño de palabra	Rondas	Operaciones	Colisiones encontradas
SHA-0	160	160	512	$2^{64} - 1$	32	80	+,and,or,xor,rot	SI
SHA-1	160	160	512	$2^{64} - 1$	32	80	+,and,or,xor,rot	Ataque teorico
SHA-224/256	224/256	256	512	$2^{64} - 1$	32	64	+,and,or,xor,shr,rot	NO
SHA-384/512	384/512	512	1024	$2^{128} - 1$	64	80	+,and,or,xor,shr,rot	NO

Tabla 2.4: Algoritmos familia SHA

2.6 Tolerancia a fallos y rendimiento

El concepto tolerancia a fallos en un sistema de almacenamiento se refiere a la capacidad del sistema para resistir fallos en el software o hardware sin la perdida de datos o disponibilidad. En un sistema de almacenamiento en la nube, donde un fichero está almacenado en un único nodo, la tolerancia a fallos para que el sistema continúe proveyendo los servicios respecto a ese fichero es de 0. Ya qué si el nodo falla, el fichero no puede ser alcanzado. [26] [27].

Si se quiere aumentar la tolerancia a fallos, una de las opciones es replicar el fichero a otro nodo, por lo que el fichero estaría almacenado en dos nodos. En este caso, la tolerancia a fallos es de 1. Esto significa que si falla un nodo, el fichero puede ser alcanzado desde otro nodo diferente.

Rendimiento de un sistema de almacenamiento se refiere a la tasa por unidad de tiempo de transacciones de almacenamiento y/o recuperación que se llevan a cabo. Un sistema de almacenamiento que tarde mucho tiempo en almacenar/recuperar información o ficheros, tendrá un rendimiento bajo. Por lo contrario, si es muy rápido tendrá un rendimiento muy alto.

Para el desarrollo de cualquier sistema informático es esencial tener en cuenta estos dos conceptos. Forma parte de las tareas del equipo de desarrollo elegir bien el equilibrio entre la tolerancia a fallos, el rendimiento y el coste monetario.

Estos tres conceptos están muy relacionados, entrando en muchas ocasiones en conflicto. Es posible obtener un sistema barato y de alto rendimiento pero con baja tolerancia a fallos; o un sistema de alta tolerancia a fallos y rápido pero caro; o algo barato y de alta tolerancia a fallos pero de bajo rendimiento.

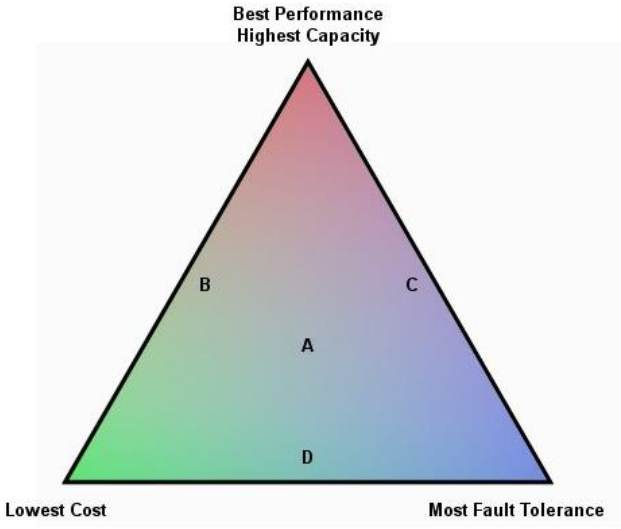


Figura 2.23: Rendimiento, tolerancia a fallos y coste

En la Figura 2.23 se puede ver lo explicado anteriormente: en cada una de las esquinas, uno de los atributos se maximiza a expensas de los otros dos. El punto A representa el equilibrio entre los tres. Los puntos B, C y D representan lo mejor en dos de los atributos a expensas de la tercera, de aquí la frase “Rápido, barato y bueno: elige dos”.

2.7 Replicación

La replicación de datos es una técnica que puede utilizarse para copiar datos a través de una red informática a una o más ubicaciones remotas. Es una técnica para la mejora de los servicios porque proporciona un incremento de su disponibilidad y lo hace tolerante a fallos. Su objetivo es que las operaciones de los usuarios sobre las réplicas se realicen de forma consistente, con un tiempo de respuesta y un caudal satisfactorio. [27].

La replicación de datos se puede implementar utilizando varios tipos, que se describen a continuación.

2.7.1 Replicación síncrona

Cuando se utiliza la replicación síncrona, una actualización hecha a un volumen de datos en el sitio principal, se replica de forma síncrona el volumen de datos a un sitio secundario. Esto garantiza que el sitio secundario tiene una copia idéntica de los datos en todo momento.

La desventaja de la replicación síncrona es que antes de responder a la demanda, el subsistema de almacenamiento debe esperar a que el subsistema secundario complete el proceso, lo que resulta un tiempo de respuesta mayor de la aplicación. Por lo tanto, el rendimiento de la replicación sincrónica está muy afectado por factores tales como: la latencia del enlace y el ancho de banda enlace. El despliegue es sólo práctico cuando el sitio secundario se encuentra cerca del sitio principal.

2.7.2 Replicación asíncrona

En un modo asíncrono de operación, las operaciones de E / S se escriben en el sistema de almacenamiento primario y luego enviada a uno o más sistemas de almacenamiento remoto en algún momento posterior en el tiempo. Debido al tiempo transcurrido, los datos sobre los sistemas remotos no siempre es un reflejo exacto de los datos en el sitio primario. Este modo es ideal para *backups* disco a disco o para tomar instantáneas de datos para procesos fuera de línea, como pruebas o planificación de negocios. El lapso de tiempo permite que los datos se repliquen a través de redes de bajo ancho de banda, pero no proporcionan el mismo nivel de protección que la replicación síncrona.

La replicación asíncrona es menos sensible a la distancia y velocidad de transmisión del enlace. Sin embargo, debido a la replicación podría retrasarse, los datos se pueden perder si se produce un error de comunicación seguida por un corte de sitio primario.

2.7.3 Replicación pasiva

En este modelo para la tolerancia a fallos existe un único gestor de réplicas primario y varios gestores de réplicas secundarios (respaldo o esclavos). En el modelo puramente formal los frontales se comunican con el gestor de réplicas primario para obtener el servicio, éste ejecuta las operaciones y envía copias de los datos actualizados a los gestores de respaldo. Si uno de los gestores de réplicas primario falla, uno de los secundarios toma su lugar.

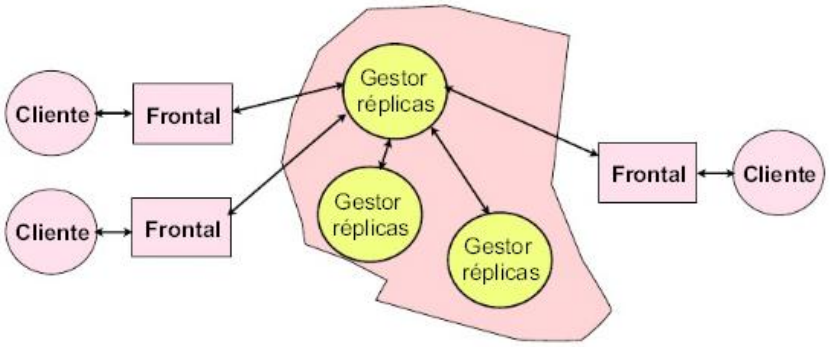


Figura 2.24: Replicación pasiva

La secuencia de eventos cuando un cliente solicita que se realice una operación son:

- Petición: el frontal efectúa la petición, que contiene un identificador único al gestor de réplicas primario.
- Coordinación: el primario acepta cada petición en el orden en que las recibe. Comprueba el identificador único y si ya hubiera ejecutado la petición reenvía la respuesta.
- Ejecución: el gestor de réplicas primario ejecuta la petición y guarda la respuesta.
- Acuerdo: si la petición es una actualización, el primario envía a todas las copias de seguridad el estado actualizado, la respuesta y el identificador. Los respaldos envían a su vez, una confirmación de recepción.
- Respuesta: el gestor de réplica primario responde al frontal y este se encarga de devolver la respuesta al cliente.

El sistema es linealizable ya que si el gestor primario falla, un gestor secundario lo sustituye. Por esto, el sistema conserva la capacidad de secuenciación. Una vez el gestor secundario toma el control, los gestores de replicas se ponen de acuerdo en que operaciones se habían

realizado en el momento en el que el reemplazo del primario tomó el control. La replicación pasiva tiene la desventaja de producir sobrecargas.

2.7.4 Replicación activa

En este modelo para tolerancia a fallos los gestores de réplicas son máquinas de estado que se comportan igual y se organizan como un grupo. Los frontales multidifunden sus peticiones a los grupos de gestores de réplicas, y estos procesan la petición de forma individual pero idéntica y responden. No afecta a la prestación del servicio, la caída de uno de los gestores de réplicas ya que los restantes continúan respondiendo de la forma habitual. La replicación activa puede tolerar fallos bizantinos, ya que el frontal puede recopilar y comparar las respuestas que recibe.

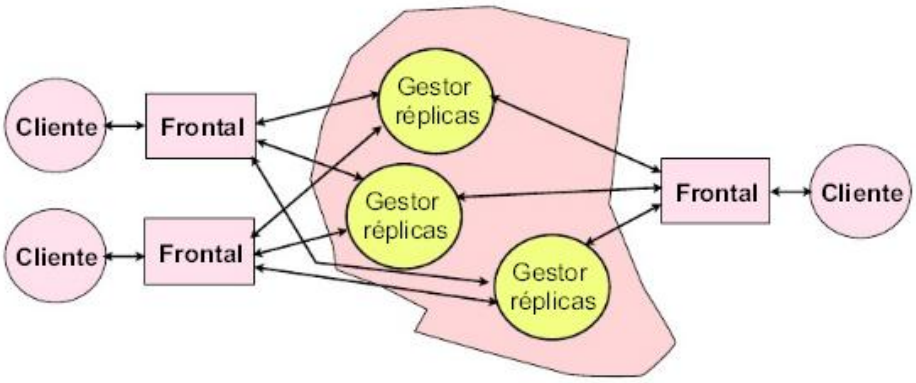


Figura 2.25: Replicación activa

La secuencia de eventos cuando un cliente solicita que se efectúe una operación bajo replicación activa son:

- Petición: el frontal adhiere un identificador único a la petición y la multidifunde al grupo de gestores de réplicas de manera fiable y ordenada. Además no se emitirá una nueva petición hasta haber obtenido la respuesta de la primera.
- Coordinación: el sistema de comunicación en grupo reparte la petición a cada gestor de réplicas correcto en el mismo orden.
- Ejecución: cada gestor de réplicas que recibió la petición la ejecuta, debido a que son máquinas de estado y las peticiones se entregan en el mismo orden total, todos los gestores de réplicas correctos procesan de la misma forma las peticiones. La respuesta contiene el identificador único de la petición del cliente.

- Acuerdo: no es necesario la etapa de acuerdo debido a la semántica de la multidifusión.
- Respuesta: cada gestor de réplicas envía su respuesta al frontal.

Este sistema consigue la consistencia secuencial. Todos los gestores de réplicas correctos procesan la misma secuencia de peticiones. La multidifusión asegura que se procese el mismo conjunto de instrucciones, y el orden total garantiza que las peticiones se procesen en el mismo orden. Como son máquinas estado todos finalizan en el mismo estado después de cada petición. Esto asegura la consistencia secuencial.

El sistema de replicación activa no consigue linealizabilidad debido a que el orden total en el que los gestores procesan las peticiones no es el mismo que el orden impuesto por el tiempo real en que los clientes realizaron sus peticiones.

2.8 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.

2.8.1 Dropbox

Dropbox es un sistema de almacenamiento en la nube que ofrece la subida de ficheros, la sincronización de estos a través de una aplicación de escritorio. Con esta aplicación se puede insertar un fichero y estar disponible al instante en cualquier otro equipo que tenga instalado Dropbox. También ofrece una interfaz Web desde la que es posible acceder desde cualquier ordenador o dispositivo móvil conectado a Internet.

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. Los precios de espacio de almacenamiento son:

- 50 GB por 10$ al mes.
- 100 GB por 20$ al mes.

Las características más destacadas que ofrece Dropbox:

- Ofrece control de versiones.
- Cifrado de archivos.
- Descarga a móvil.
- multimedia.
- Ficheros y directorios colaborativos.
- API Pública.
- Compartir archivos públicamente.

2.8.2 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. Los precios de espacio de almacenamiento en SkyDrive son:

- 20GB por 10$ al año.
- 50GB por 25$ al año.
- 100GB por 50$ al año.

Las características más destacadas que ofrece SkyDrive:

- Control de versiones.
- Cifrado de archivos.
- Sincronización de múltiples directorios.
- Descarga a móvil.
- *Streaming* multimedia.
- Ficheros y directorios colaborativos.
- API Pública.
- Compartir archivos públicamente.

2.8.3 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. Los precios de espacio de almacenamiento en Google Drive son:

- 25 GB por 2,5$ al mes.
- 100 GB por 5$ al mes.
- 1 TB por 50$ al mes.
- 16 TB por 800$ al mes.

Las características más destacadas que ofrece Google Drive:

- Control de versiones.
- Cifrado de archivos.
- Descarga a móvil
- Ficheros y directorios colaborativos.
- API Pública.

3. Marco Regulador

Este capítulo recoge las normativas técnicas y legales que afectan al trabajo.

Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal que se detalla a continuación. [28]

Artículo 1. Objeto.

La presente Ley Orgánica tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar.

Artículo 2. Ámbito de aplicación

1. *La presente Ley Orgánica será de aplicación a los datos de carácter personal registrados en soporte físico, que los haga susceptibles de tratamiento, y a toda modalidad de uso posterior de estos datos por los sectores público y privado.*

   Se regirá por la presente Ley Orgánica todo tratamiento de datos de carácter personal:

   a) *Cuando el tratamiento sea efectuado en territorio español en el marco de las actividades de un establecimiento del responsable del tratamiento.*
   b) *Cuando al responsable del tratamiento no establecido en territorio español, le sea de aplicación la legislación española en aplicación de normas de Derecho Internacional público.*
   c) *Cuando el responsable del tratamiento no esté establecido en territorio de la Unión Europea y utilice en el tratamiento de datos medios situados en territorio español, salvo que tales medios se utilicen únicamente con fines de tránsito.*
2. *El régimen de protección de los datos de carácter personal que se establece en la presente Ley Orgánica no será de aplicación:*
   a) *A los ficheros mantenidos por personas físicas en el ejercicio de actividades exclusivamente personales o domésticas.*
   b) *A los ficheros sometidos a la normativa sobre protección de materias clasificadas.*
   c) *A los ficheros establecidos para la investigación del terrorismo y de formas graves de delincuencia organizada. No obstante, en estos supuestos el responsable del fichero comunicará previamente la existencia del mismo, sus características generales y su finalidad a la Agencia de Protección de Datos.*

3. *Se regirán por sus disposiciones específicas, y por lo especialmente previsto, en su caso, por esta Ley Orgánica los siguientes tratamientos de datos personales:*
    a) *Los ficheros regulados por la legislación de régimen electoral.*
    b) *Los que sirvan a fines exclusivamente estadísticos, y estén amparados por la legislación estatal o autonómica sobre la función estadística pública.*
    c) *Los que tengan por objeto el almacenamiento de los datos contenidos en los informes personales de calificación a que se refiere la legislación del régimen del personal de las Fuerzas Armadas.*
    d) *Los derivados del Registro Civil y del Registro Central de penados y rebeldes.*
    e) *Los procedentes de imágenes y sonidos obtenidos mediante la utilización de videocámaras por las Fuerzas y Cuerpos de Seguridad, de conformidad con la legislación sobre la materia.*

4. Metodología

Las Metodologías de Desarrollo de Software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto software. Se pueden clasificar en dos grupos:

- Metodologías pesadas. Orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Este tipo de metodologías son más eficaces y necesarias cuanto mayor es el proyecto que se pretende realizar respecto a tiempo y recursos que son necesarios emplear, donde una gran organización es requerida.
- Metodologías ágiles. Orientadas a la interacción con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando. Tienen como propósito permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se basan en el manifiesto ágil [1].

El presente TFG es un proyecto sencillo, que dispone de un único trabajador. Con una economía limitada y un presupuesto incluido en el Anexo III. Teniendo en cuenta esto y el limitado tiempo para realizar el TFG, la mejor opción es una metodología ágil.

Existen numerosas metodologías ágiles, la que se va a utilizar en el presente TFG es la denominada “Adaptive Software Development”. ASD es una metodología de desarrollo impulsada por Jim Highsmith y Sam Bayer que hace énfasis en aplicar la adaptación continua del proceso de trabajo. Sus principales características son: iterativo, orientado a los componentes software y tolerante a los cambios.

4.1 Ciclo de vida

El ciclo de vida que propone ASD tiene tres fases esenciales: especulación, colaboración y aprendizaje. En la primera de ellas se inicia el proyecto y se planifican las características del software; en la segunda se desarrollan las características y finalmente en la tercera se revisa su calidad. La revisión de los componentes sirve para aprender de los errores y volver a iniciar el ciclo de desarrollo.

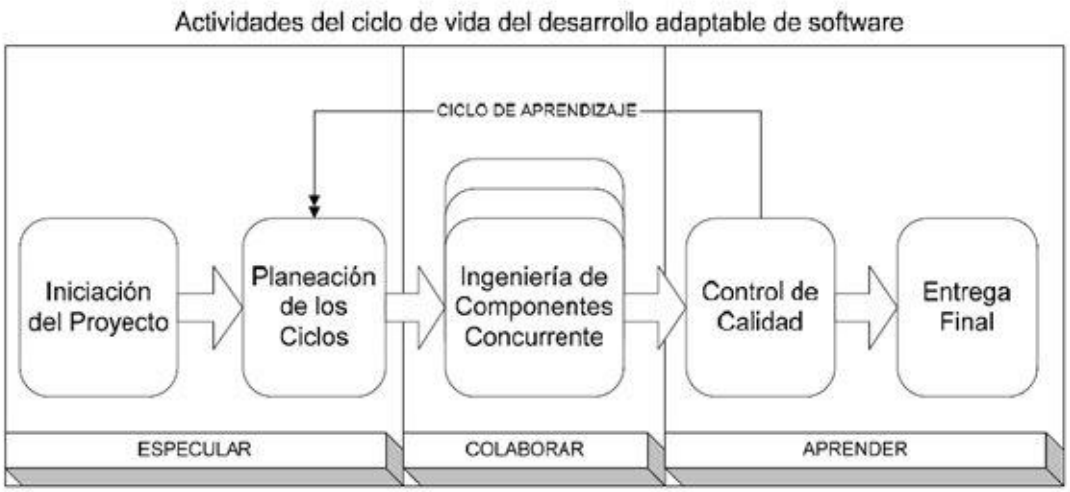


Figura 4.1: Ciclo de vida de ASD [2]

Para el presente trabajo se realizará una iteración. Las tareas que se realizarán en cada fase de dicha iteración se muestran a continuación.

Especulación

En esta fase se inicia el proyecto, por lo tanto es necesario definir los objetivos y realizar la planificación. Así mismo se realizará un aprendizaje de las tecnologías a utilizar.

- Definir objetivos.
- Análisis y aprendizaje de tecnologías a utilizar en el proyecto.
- Planificación de las tareas en el tiempo. El resultado de la planificación se recoge en el diagrama de Gantt, Anexo IV.

Colaboración

En esta fase se llevan a cabo los objetivos marcados en el inicio del trabajo.

- Análisis: se realizará un análisis del sistema de almacenamiento en la nube. Para ello se analizarán los casos de uso y se obtendrán los requisitos y el modelo de la base de datos.
- Diseño: se realizará el diseño del sistema, teniendo en cuenta las diferentes opciones de diseño y el análisis realizado.
- Implementación: finalmente se implementará un prototipo del sistema analizado y diseñado.

Aprendizaje

En esta fase se realiza la revisión la calidad y estado del proyecto. Para ello se analiza el rendimiento, evaluando las características finales del producto. Una vez realizada dicha evaluación se proponen mejoras y se sacan conclusiones.

- Análisis de rendimiento y calidad: evaluación del sistema.
- Conclusiones: En las conclusiones se revisará el estado general del proyecto y si los objetivos propuestos han sido cumplidos.
- Mejoras a realizar (trabajos futuros): una vez realizada la revisión se podrán proponer mejoras a realizar en posibles trabajos futuros.

5. Análisis y Diseño

En este capítulo se muestra el proceso de análisis y diseño del sistema.

5.1 Análisis

El análisis es la descomposición del problema que se desea resolver en partes más pequeñas que permitan su estudio de forma más sencilla. Para ello, se han obtenido los casos de uso, los requisitos de usuario, requisitos de software y el modelo conceptual entidad/relación de la base de datos.

5.1.1 Casos de Uso

Los casos de uso son una técnica para especificar el comportamiento de un sistema. Describen qué hace un sistema pero no cómo lo hace. En la presente sección se recogen de forma gráfica los casos de uso del sistema VoCS.

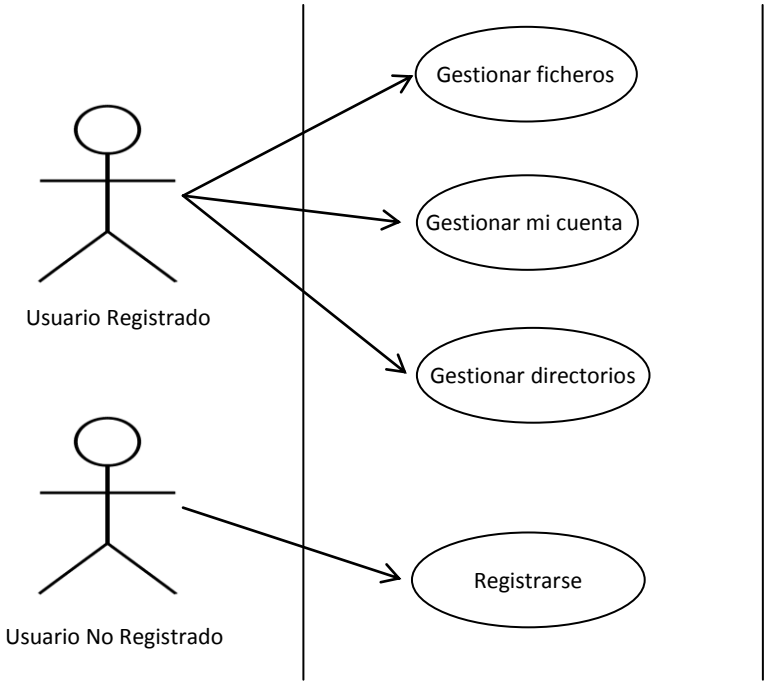


Figura 5.1: Diagrama de Casos de Uso

Tal y como se muestra en la Figura 5.1, existen dos tipos de Usuarios: los Usuarios Registrados y los Usuarios No Registrados. Dado que será necesario que un usuario esté registrado en la aplicación para poder hacer uso de la funcionalidad, los usuarios no registrados sólo tendrán la opción de registrarse. Por el contrario, los usuarios registrados podrán acceder a toda la funcionalidad de la aplicación, que se divide en tres casos de uso generales (mostrados en la Figura 5.1):

- Gestión de Ficheros

- Gestión de Directorios
- Gestión de Cuenta

A continuación se muestra desglosado cada uno de ellos (Figuras 5.2, 5.3 y 5.4).

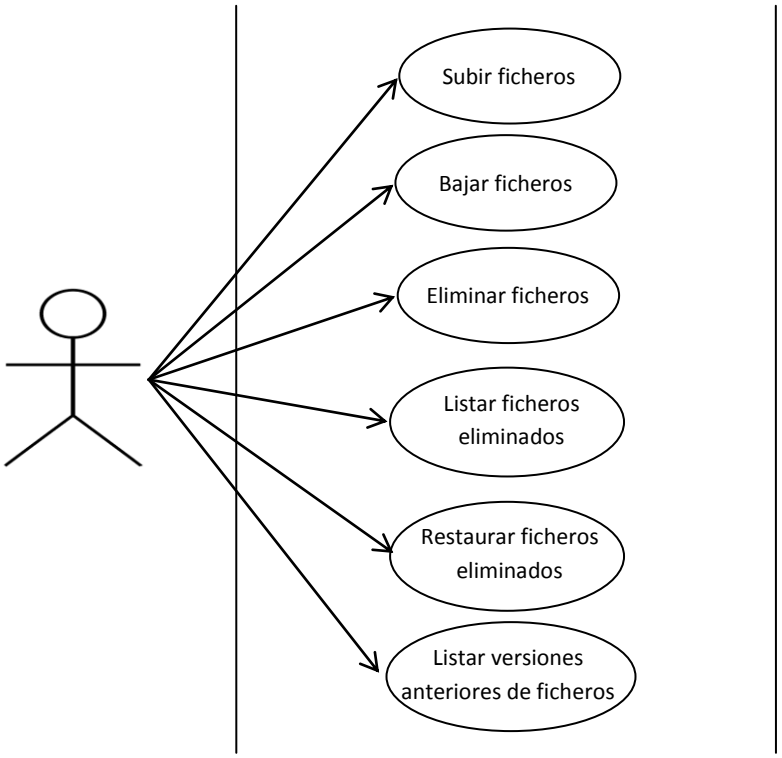


Figura 5.2: Diagrama de Casos de Uso Gestión de Ficheros

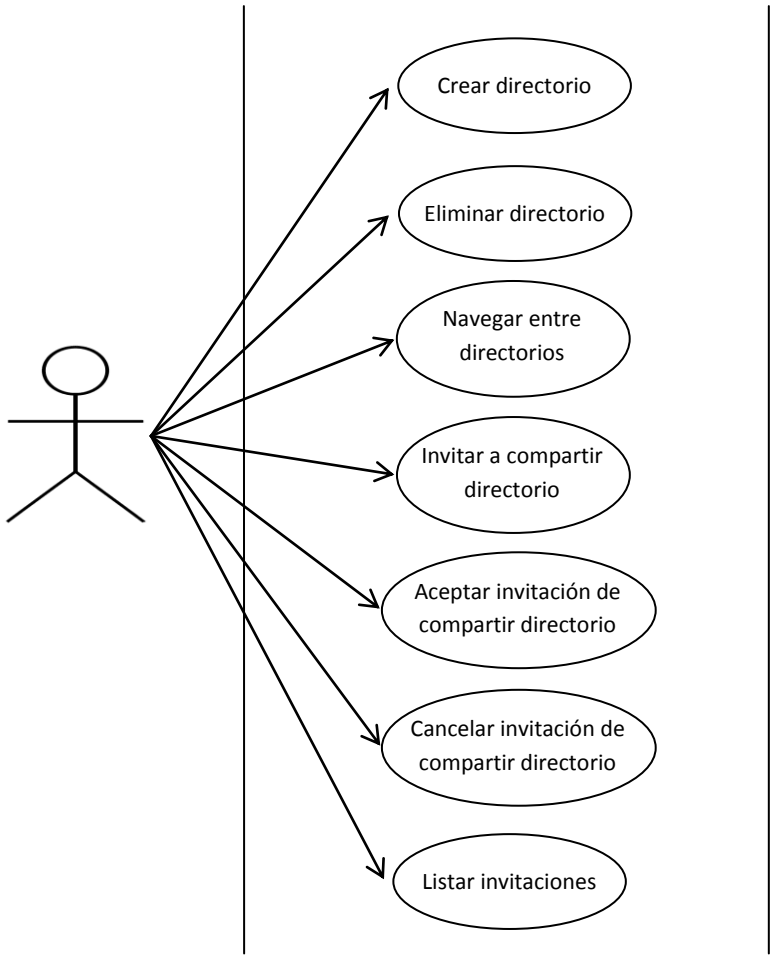


Figura 5.3: Diagrama de Casos de Uso Gestión de Directorios

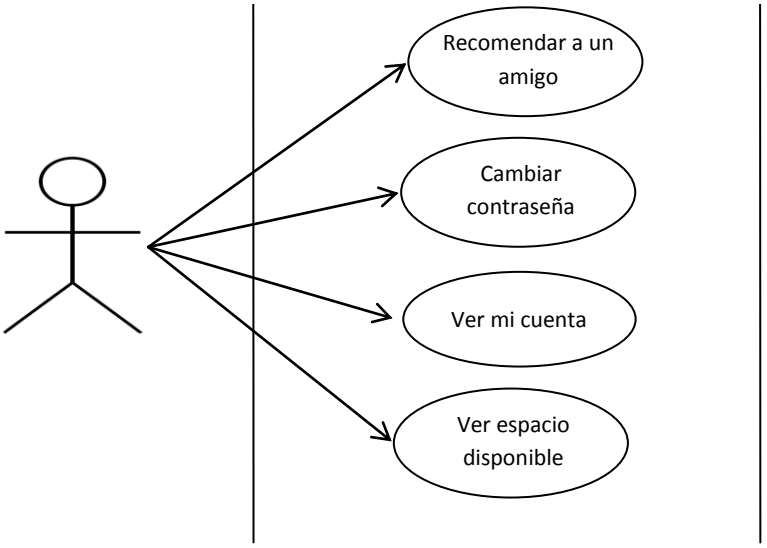


Figura 5.4: Diagrama de Casos de Uso Gestión de Cuenta

5.1.2 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.

5.1.2.1 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.
- 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

RU01: Acceso			
Descripción	El usuario deberá poder acceder a la aplicación vía Web.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.1: Requisito RU01 Acceso a la aplicación

RU02: Inicio de sesión			
Descripción	El usuario podrá autenticarse ante el sistema y acceder a la zona de la aplicación para usuarios autenticados.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.2: Requisito RU02 Inicio de sesión

RU03: Registrar usuario			
Descripción	El usuario podrá registrarse en el sistema para una posterior autenticación.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.3: Requisito RU03 Registrar usuario

RU04: Autenticación			
Descripción	El usuario no podrá autenticarse ante el sistema sin haberse registrado.		
Tipo	Restricción	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.4: Requisito RU04 Autenticación

RU05: Subir ficheros			
Descripción	El usuario podrá subir ficheros al sistema de almacenamiento que proporciona VoCS.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.5: Requisito RU05 Subir ficheros

RU06: Seguridad ficheros			
Descripción	El usuario podrá elegir entre tres niveles de seguridad en los ficheros que almacene la aplicación, en el momento de la subida: • Alta • Media • Baja		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.6: Requisito RU06 Seguridad ficheros

RU07: Estado subida ficheros			
Descripción	En el proceso de subida de ficheros, el usuario podrá ver en todo momento el estado de dicha subida de ficheros.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.7: Requisito RU07 Estado subida de ficheros

RU08: Descargar ficheros			
Descripción	El usuario podrá descargar los ficheros que estén almacenados en el sistema VoCS asociados a su cuenta, ya sea porque los ha subido él o porque otro usuario los ha compartido.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.8: Requisito RU08 Descargar ficheros

RU09: Descargar ficheros no propios			
Descripción	El usuario no podrá descargar ficheros que no estén asociados a su cuenta.		
Tipo	Restricción	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.9: Requisito RU09 Descargar ficheros no propios

RU10: Eliminar ficheros			
Descripción	El usuario podrá eliminar ficheros asociados a su cuenta.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.10: Requisito RU10 Eliminar ficheros

RU11: Eliminar ficheros no propios			
Descripción	El usuario no podrá eliminar ficheros que no estén asociados a su cuenta.		
Tipo	Restricción	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.11: Requisito RU11 Eliminar ficheros no propios

RU12: Listar ficheros eliminados			
Descripción	El usuario podrá listar sus ficheros eliminados.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.12: Requisito RU12 Listar ficheros eliminados

RU13: Listar ficheros eliminados no propios			
Descripción	El usuario no podrá listar ficheros eliminados que no estuvieran asociados a su cuenta.		
Tipo	Restricción	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.13: Requisito RU13 Listar ficheros eliminados no propios

RU14: Restaurar ficheros eliminados			
Descripción	El usuario podrá restaurar sus ficheros eliminados.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.14: Requisito RU14 Restaurar ficheros eliminados

RU15: Restaurar ficheros eliminados no propios			
Descripción	El usuario no podrá restaurar ficheros eliminados que no estuvieran asociados a su cuenta.		
Tipo	Restricción	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.15: Requisito RU15 Restaurar ficheros eliminados no propios

RU16: Listar versiones anteriores			
Descripción	El usuario podrá listar las versiones anteriores de los ficheros almacenados y asociados a su cuenta.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.16: Requisito RU16 Listar versiones anteriores

RU17: Listar versiones anteriores ficheros no propios			
Descripción	El usuario no podrá listar las versiones anteriores de los ficheros almacenados que no estén asociados a su cuenta.		
Tipo	Restricción	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.17: Requisito RU17 Listar versiones de ficheros no propios

RU18: Restaurar versiones anteriores			
Descripción	El usuario podrá restaurar versiones anteriores de los ficheros asociados a su cuenta.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.18: Requisito RU18 Restaurar versiones anteriores

RU19: Restaurar versiones anteriores ficheros no propios			
Descripción	El usuario no podrá restaurar versiones anteriores de ficheros que no estén asociados a su cuenta.		
Tipo	Restricción	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.19: Requisito RU19 Restaurar versiones de ficheros no propios

RU20: Crear directorios			
Descripción	El usuario podrá crear directorios en el sistema de almacenamiento en la nube que proporciona la aplicación.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.20: Requisito RU20 Crear directorios

RU21: Eliminar directorios			
Descripción	El usuario podrá eliminar directorios asociados a su cuenta, ya sean creados por él o compartidos por otro usuario.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.21: Requisito RU21 Eliminar directorios

RU22: Eliminar directorios no propios			
Descripción	El usuario no podrá eliminar directorios que no estén asociados a su cuenta.		
Tipo	Restricción	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.22: Requisito RU22 Eliminar directorios no propios

RU23: Navegar			
Descripción	El usuario podrá navegar a través de su árbol de directorios (directorios asociados a su cuenta) que proporciona la aplicación VoCS.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.23: Requisito RU23 Navegar

RU24: Navegar directorios no propios			
Descripción	El usuario no podrá navegar a través de directorios que no estén asociados a su cuenta.		
Tipo	Restricción	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.24: Requisito RU24 Navegar directorios no propios

RU25: Invitar a compartir directorio			
Descripción	El usuario podrá invitar a otro usuario a compartir un directorio y su contenido.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.25: Requisito RU25 Invitar a compartir

RU26: Aceptar invitaciones			
Descripción	El usuario podrá aceptar las invitaciones a compartir un directorio y su contenido, en las que él sea el destinatario.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.26: Requisito RU26 Aceptar invitaciones

RU27: Aceptar invitaciones no propias			
Descripción	El usuario no podrá aceptar las invitaciones a compartir un directorio y su contenido, en las que él no sea el destinatario.		
Tipo	Restricción	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.27: Requisito RU27 Aceptar invitaciones no propias

RU28: Cancelar invitaciones			
Descripción	El usuario podrá cancelar las invitaciones a compartir un directorio y su contenido, en las que él sea el destinatario.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.28: Requisito RU28 Cancelar invitaciones

RU29: Cancelar invitaciones no propias			
Descripción	El usuario no podrá cancelar las invitaciones a compartir un directorio y su contenido, en las que él no sea el destinatario.		
Tipo	Restricción	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.29: Requisito RU29 Cancelar invitaciones no propias

RU30: Listar invitaciones			
Descripción	El usuario podrá listar las invitaciones a directorios que le han enviado.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.30: Requisito RU30 Listar invitaciones

RU31: Listar invitaciones no propias			
Descripción	El usuario no podrá listar las invitaciones a directorios que no le han enviado a él.		
Tipo	Restricción	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.31: Requisito RU31 Listar invitaciones no propias

RU32: Invitar a VoCS			
Descripción	El usuario podrá invitar a una persona a utilizar VoCS.		
Tipo	Capacidad	Prioridad	Media
Necesidad	Esencial	Verificabilidad	Media

Tabla 5.32: Requisito RU32 Recomendar aplicación

RU33: Cambiar contraseña			
Descripción	El usuario podrá cambiar la contraseña de autenticación de su usuario en la aplicación VoCS.		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.33: Requisito RU33 Cambiar contraseña

RU34: Ver información cuenta			
Descripción	El usuario podrá ver en todo momento la información de su cuenta: • Nombre • Apellidos • DNI • Email • Nombre de usuario • Espacio disponible • Espacio utilizado		
Tipo	Capacidad	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.34: Requisito RU34 Ver información de la cuenta

RU35: Consultar espacio			
Descripción	El usuario podrá ver su espacio disponible, el espacio utilizado y el espacio libre en todo momento.		
Tipo	Capacidad	Prioridad	Media
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.35: Requisito RU35 Consultar espacio

RU36: Cerrar sesión			
Descripción	Los usuarios podrán dar por terminada su sesión dentro de la aplicación y salir de ella en cualquier momento.		
Tipo	Capacidad	Prioridad	Media
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.36: Requisito RU36 Cerrar sesión

Matriz de Trazabilidad: Casos de Uso – Requisitos de Usuario

	CU01: Registrarse	CU02: Subir ficheros	CU03: Descargar ficheros	CU04; Eliminar ficheros	CU05: Listar ficheros eliminados	CU06: Restaurar ficheros eliminados	CU07: Listar versiones anteriores de ficheros	CU08: Crear directorio	CU09: Eliminar directorio	CU10: Navegar entre directorios	CU11: Invitar a compartir directorios	CU12: Aceptar invitación a compartir	CU13: Cancelar invitación a compartir	CU14: Listar invitaciones	CU15: Recomendar a un amigo	CU16: Cambiar contraseña	CU17: Ver cuenta	CU18: Ver espacio disponible
RU01	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RU02	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
RU03	X																	
RU04	X																	
RU05		X																
RU06		X																
RU07		X																
RU08			X															
RU09			X															
RU10				X														
RU11				X														
RU12					X													
RU13					X													
RU14						X												
RU15						X												
RU16							X											
RU17							X											
RU18							X											
RU19							X											
RU20								X										
RU21									X									
RU22									X									
RU23										X								
RU24										X								
RU25											X							
RU26												X						
RU27												X						
RU28													X					
RU29													X					
RU30														X				
RU31														X				
RU32															X			
RU33																X		
RU34																	X	
RU35																		X
RU36	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Tabla 5.37: Matriz de Trazabilidad: Casos de Uso - Requisitos de Usuario

5.1.2.2 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	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.38: Requisito RS01 Acceso a la aplicación

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

Tabla 5.39: Requisito RS02 Interfaz

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

Tabla 5.40: Requisito RS03 Autenticación

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

Tabla 5.41: Requisito RS04 Formulario autenticación

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

Tabla 5.42: Requisito RS05 Medio de autenticación

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	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.43: Requisito RS06 Información de registro

RS07			
Descripción	El sistema almacenará de forma segura la información de registro de los usuarios, así como la información necesaria para autenticarlos.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.44: Requisito RS07 Seguridad información

RS08			
Descripción	El sistema no permitirá autenticarse a usuarios sin estar registrados.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.45: Requisito RS08 Registro-autenticación

RS09			
Descripción	El sistema permitirá a los usuarios subir ficheros al sistema de almacenamiento en la nube que este proporciona.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.46: Requisito RS09 Subir ficheros

RS10			
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 • Seleccionar el nivel de seguridad del fichero.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.47: Requisito RS10 Interfaz subir ficheros

RS11			
Descripción	El sistema deberá almacenar de forma segura los ficheros subidos.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.48: Requisito RS11 Seguridad de almacenamiento

RS12			
Descripción	El sistema deberá mantener la relación de los ficheros almacenado con los usuarios y el directorio al que pertenecen.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.49: Requisito RS12 Relación fichero-directorio-usuario

RS13			
Descripción	Cuando un usuario realiza la petición de subida de un fichero, el sistema ejecutará el modelo de replicación de ficheros que esté fijado.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.50: Requisito RS13 Replicación subida de ficheros

RS14			
Descripción	El sistema no permitirá subir ficheros a los usuarios que excedan el espacio disponible.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.51: Requisito RS14 Espacio disponible

RS15			
Descripción	El sistema deberá dotar de la seguridad elegida por los usuarios a los ficheros almacenados: • Nivel bajo de seguridad: se almacena el fichero sin cifrado • Nivel medio de seguridad: se almacena el fichero sin cifrado • Nivel alto de seguridad: se almacena el fichero cifrado.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.52: Requisito RS15 Cifrado

RS16			
Descripción	El sistema garantiza, según el nivel de seguridad elegido por los usuarios en la interfaz de subida de ficheros, lo siguiente: • Nivel bajo de seguridad: se garantiza mínimo 0 tolerancia a fallos • Nivel de seguridad medio: se garantiza mínimo 1 tolerancia a fallos • Nivel alto de seguridad: se garantiza mínimo 1 tolerancia a fallos.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.53: Requisito RS16 Mínimo de replicación

RS17			
Descripción	En el proceso de subida de ficheros, el sistema proveerá de información del estado de dicha subida de ficheros.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.54: Requisito RS17 Estado de subida de ficheros

RS18			
Descripción	El sistema se encargará de recopilar la información necesaria para mostrar el estado de la subida de ficheros.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.55: Requisito RS18 Información del estado de subida

RS19			
Descripción	La información mínima sobre el estado de la subida de ficheros es: • Porcentaje subido • Bits por segundo de la subida • Tiempo que queda para terminar.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.56: Requisito RS19 Información mínima del estado de subida

RS20			
Descripción	El sistema permitirá a los usuarios descargar ficheros asociados a su cuenta, ya sea porque los ha subido él o porque han sido compartidos por otro usuario.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.57: Requisito RS20 Descargar fichero

RS21			
Descripción	Cuando un usuario realiza la petición de descarga de un fichero a un servidor, el sistema ejecutará el modelo de replicación que esté actualmente fijado.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.58: Requisito RS21 Replicación en descarga de fichero

RS22			
Descripción	El sistema no permitirá a los usuarios descargar ficheros que no estén asociados a su cuenta.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.59: Requisito RS22 Restricción descarga de ficheros

RS23			
Descripción	El sistema permitirá a los usuarios eliminar ficheros que tengan asociados a su cuenta.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.60: Requisito RS23 Eliminar ficheros

RS24			
Descripción	El sistema deberá marcar el fichero como borrado por el usuario, sin eliminarlo del sistema.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.61: Requisito RS24 Forma de eliminar un fichero

RS25			
Descripción	El sistema no permitirá a los usuarios eliminar ficheros que no estén asociados a su cuenta.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.62: Requisito RS25 Restricción eliminación ficheros

RS26			
Descripción	El sistema permitirá a los usuarios listar sus ficheros eliminados, cuya fecha de borrado no sea anterior a 15 días respecto a la fecha actual.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.63: Requisito RS26 Listar ficheros eliminados

RS27			
Descripción	El sistema deberá poder listar los ficheros eliminados de un usuario en una interfaz gráfica.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.64: Requisito RS27 Interfaz listar ficheros eliminados

RS28			
Descripción	El sistema deberá eliminar completamente los ficheros marcados como borrados por los usuarios, siempre que hayan transcurrido más de 15 días desde la fecha de borrado.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.65: Requisito RS28 Eliminación completa de ficheros

RS29			
Descripción	El sistema no deberá eliminar completamente los ficheros marcados como borrados por los usuarios, si no han transcurrido más de 15 días desde la fecha de borrado.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.66: Requisito RS29 Tiempo de espera eliminación de ficheros

RS30			
Descripción	El sistema no deberá permitir al usuario listar ficheros eliminados que no estuvieran asociados a su cuenta.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.67: Requisito RS30 Restricción listar ficheros eliminados

RS31			
Descripción	El sistema deberá permitir a los usuarios restaurar sus ficheros eliminados.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.68: Requisito RS31 Restaurar ficheros

RS32			
Descripción	El sistema permitirá restaurar los ficheros cuya fecha de borrado no sea anterior a 15 días respecto a la fecha actual.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.69: Requisito RS32 Tiempo restaurar ficheros

RS33			
Descripción	El sistema no permitirá restaurar los ficheros cuya fecha de borrado sea anterior a 15 días respecto a la fecha actual.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.70: Requisito RS33 Restricción tiempo restaurar ficheros

RS34			
Descripción	En el proceso de restauración de un fichero, el sistema deberá generar las relaciones oportunas para que todas las demás versiones almacenadas sean versiones anteriores del fichero restaurado.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.71: Requisito RS34 Versionado

RS35			
Descripción	En el proceso de restauración de un fichero, cuando ese fichero no tiene un directorio asignado, se asigna al directorio raíz.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.72: Requisito RS35 Directorio ficheros restaurados

RS36			
Descripción	El sistema no deberá permitir al usuario restaurar ficheros eliminados que no estuvieran asociados a su cuenta.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.73: Requisito RS36 Restricción restaurar ficheros

RS37			
Descripción	El sistema permitirá al usuario listar las versiones anteriores de los ficheros almacenados.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.74: Requisito RS37 Listar versiones anteriores

RS38			
Descripción	El sistema permitirá listar únicamente las versiones anteriores de los ficheros cuya fecha de modificación no sea anterior a 15 días respecto a la fecha actual.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.75: Requisito RS38 Tiempo listar versiones

RS39			
Descripción	El sistema no permitirá listar las versiones anteriores de los ficheros cuya fecha de modificación sea anterior a 15 días respecto a la fecha actual.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.76: Requisito RS39 Restricciones tiempo listar versiones

RS40			
Descripción	El sistema no permitirá al usuario listar versiones anteriores de ficheros no asociados a su cuenta.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.77: Requisito RS40 Restricción listar versiones

RS41			
Descripción	El sistema deberá permitir a los usuarios restaurar versiones anteriores de sus ficheros.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.78: Requisito RS41 Restaurar versiones

RS42			
Descripción	El sistema permitirá restaurar versiones anteriores de los ficheros cuya fecha de modificación no sea anterior a 15 días respecto a la fecha actual.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.79: Requisito RS42 Tiempo restaurar versiones

RS43			
Descripción	El sistema no permitirá restaurar versiones anteriores de los ficheros cuya fecha de modificación sea anterior a 15 días respecto a la fecha actual.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.80: Requisito RS43 Restricción tiempo restaurar versiones

RS44			
Descripción	En el proceso de restauración de una versión anterior de un fichero, el sistema deberá generar las relaciones oportunas para que todas las demás versiones almacenadas sean versiones anteriores del fichero restaurado.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.81: Requisito RS44 Versionado restauración de versiones

RS45			
Descripción	El sistema proporcionará una interfaz gráfica para el listado de versiones anteriores y la selección de restaurar versión.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.82: Requisito RS45 Interfaz versiones

RS46			
Descripción	El sistema no permitirá al usuario restaurar versiones anteriores de ficheros no asociados a su cuenta.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.83: Requisito RS46 Restricciones restaurar versiones

RS47			
Descripción	El sistema deberá permitir a los usuarios crear directorios en su árbol de directorios que le proporciona la aplicación.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.84: Requisito RS47 Crear directorios

RS48			
Descripción	El sistema deberá permitir al usuario elegir el nombre del directorio en el momento de la creación.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.85: Requisito RS48 Nombre de directorios

RS49			
Descripción	El sistema proveerá de una interfaz gráfica para la creación de directorios que contendrá: • Un formulario con el nombre del directorio • Un botón para crear el directorio.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.86: Requisito RS49 Interfaz crear directorio

RS50			
Descripción	El sistema permitirá al usuario eliminar directorios asociados a su cuenta, ya sean creados por él o compartidos por otro usuario.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.87: Requisito RS50 Eliminar directorios

RS51			
Descripción	En el proceso de eliminación de un directorio, el sistema deberá eliminar el directorio del árbol de directorios del usuario.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.88: Requisito RS51 Proceso de eliminación

RS52			
Descripción	En el proceso de borrado de un directorio, el sistema deberá eliminar los directorios que contenga el directorio borrado.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.89: Requisito RS52 Eliminar sub-directorios

RS53			
Descripción	En el proceso de eliminación de un directorio, el sistema deberá marcar como borrados los ficheros que contenía ese directorio.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.90: Requisito RS53 Eliminar ficheros contenidos

RS54			
Descripción	El sistema no permitirá la eliminación del directorio raíz.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.91: Requisito RS54 Eliminación directorio raíz

RS55			
Descripción	El sistema no permitirá a los usuarios eliminar directorios que no estén en su propiedad o no estén compartidos.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.92: Requisito RS55 Restricción eliminar directorios

RS56			
Descripción	El sistema permitirá al usuario navegar a través de su árbol de directorios.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.93: Requisito RS56 Navegar

RS57			
Descripción	El sistema no permitirá al usuario navegar a través de directorios que no estén asociados a su cuenta.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.94: Requisito RS57 Restricción navegar

RS58			
Descripción	El sistema podrá listar los directorios y carpetas del directorio en el que el usuario se encuentre.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.95: Requisito RS58 Listar directorios y subdirectorios

RS59			
Descripción	El sistema proveerá de una interfaz gráfica que permita la navegación por el árbol de directorios y muestre el listado de directorios y ficheros del directorio en el que el usuario se encuentre. También deberá ofrecer la posibilidad de descarga de los ficheros listados.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.96: Requisito RS59 Interfaz directorios y contenido

RS60			
Descripción	El sistema permitirá al usuario navegar hacia atrás y hacia delante en el árbol de directorios.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.97: Requisito RS60 Navegar atrás-adelante

RS61			
Descripción	El sistema deberá mostrar la ruta en la que se encuentra el usuario.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.98: Requisito RS61 Ruta

RS62			
Descripción	El sistema no permitirá al usuario navegar por directorios que no estén asociados a su cuenta.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.99: Requisito RS62 Restricción navegar directorios

RS63			
Descripción	El sistema deberá permitir al usuario invitar a otro usuario a compartir sus directorios y sus contenidos.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.100: Requisito RS63 Invitar

RS64			
Descripción	El sistema proveerá a los usuarios de los elementos gráficos necesarios para que puedan invitar a otras personas a sus directorios y sus contenidos. Dicha interfaz contendrá: • Un formulario con una entrada para el email del invitado • Un botón para enviar la invitación.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.101: Requisito RS64 Interfaz invitar

RS65			
Descripción	Las invitaciones a directorios y sus contenidos serán a través del email de la persona invitada.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.102: Requisito RS65 Modo de invitación

RS66			
Descripción	En el proceso de invitación a directorios y sus contenidos, cuando el invitado ya es usuario registrado del sistema, se le envía una notificación de la invitación por email y puede aceptarla en la aplicación VoCS.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.103: Requisito RS66 Invitación usuario registrado

RS67			
Descripción	En el proceso de invitación a directorios y sus contenidos, cuando el invitado no es usuario registrado del sistema, se le envía una notificación de la invitación por email e invita a registrarse en la aplicación para poder compartir.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.104: Requisito RS67 Invitación usuario no registrado

RS68			
Descripción	El sistema deberá permitir a los usuarios aceptar las invitaciones a directorios y sus contenidos que les hayan hecho con anterioridad.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.105: Requisito RS68 Aceptar invitación

RS69			
Descripción	El sistema realizará las acciones oportunas para que los usuarios que han querido compartir un directorio, lo compartan.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.106: Requisito RS69 Proceso compartir

RS70			
Descripción	El sistema deberá eliminar las invitaciones que han sido aceptadas.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.107: Requisito RS70 Eliminar invitaciones aceptadas

RS71			
Descripción	El sistema no permitirá al usuario aceptar las invitaciones a compartir un directorio y su contenido, en las que el usuario no sea el destinatario.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.108: Requisito RS71 Restricción aceptar invitación

RS72			
Descripción	El sistema deberá permitir a los usuarios denegar las invitaciones a directorios y sus contenidos que les hayan hecho con anterioridad.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.109: Requisito RS72 Denegar invitación

RS73			
Descripción	El sistema deberá eliminar las invitaciones que han sido denegadas.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.110: Requisito RS73 Eliminar invitaciones denegadas

RS74			
Descripción	El sistema proveerá de una interfaz gráfica para denegar o aceptar las invitaciones. Dicha interfaz contendrá: • Email del usuario que ha invitado • Dos botones: uno para aceptar la invitación y otro para denegarla		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.111: Requisito RS74 Interfaz aceptar o denegar invitación

RS75			
Descripción	El sistema no permitirá al usuario denegar las invitaciones a compartir un directorio y su contenido, en las que el usuario no sea el destinatario.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.112: Requisito RS75 Restricción denegar invitación

RS76			
Descripción	El sistema permitirá a los usuarios listar las invitaciones a compartir directorios.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.113: Requisito RS76 Listar invitaciones

RS77			
Descripción	El sistema proveerá de una interfaz gráfica para visualizar el listado de invitaciones.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.114: Requisito RS77 Interfaz listado invitaciones

RS78			
Descripción	El sistema no permitirá al usuario listar las invitaciones a directorios que no le han enviado a él.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.115: Requisito RS78 Restricción listar invitaciones

RS79			
Descripción	El sistema permitirá al usuario invitar a una persona a utilizar VoCS.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.116: Requisito RS79 Recomendar

RS80			
Descripción	El sistema proveerá de una interfaz gráfica para invitar a otras personas a utilizar VoCS a través de su email. Está interfaz deberá tener un formulario con una entrada para el email de la persona a la que se desea invitar.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.117: Requisito RS80 Interfaz recomendar

RS81			
Descripción	En el proceso de invitación a VoCS, el sistema deberá enviar un correo electrónico, invitando a utilizar VoCS, a la dirección que el usuario indique en la interfaz.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.118: Requisito RS81 Modo recomendar

RS82			
Descripción	El sistema deberá permitir al usuario cambiar la contraseña de autenticación de su usuario en la aplicación VoCS.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.119: Requisito RS82 Cambiar contraseña

RS83			
Descripción	En el proceso de cambio de contraseña el sistema deberá asegurarse que el usuario conoce su contraseña actual.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.120: Requisito RS83 Modo cambio contraseña

RS84			
Descripción	El sistema proveerá de una interfaz gráfica para el cambio de contraseña. Dicha interfaz deberá contener un formulario para la contraseña actual y la nueva contraseña.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.121: Requisito RS84 Interfaz cambio de contraseña

RS85			
Descripción	El sistema deberá asegurarse de que al cambiar la contraseña del usuario el resto del sistema sigue siendo funcional.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.122: Requisito RS85 Seguridad cambio de contraseña

RS86			
Descripción	El sistema deberá permitir al usuario autenticado ver en todo momento la información de su cuenta: • Nombre • Apellidos • DNI • Email • Nombre de usuario • Espacio disponible • Espacio utilizado		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.123: Requisito RS89 Mi cuenta

RS87			
Descripción	El sistema deberá ser capaz de recopilar la información del usuario almacenada en el sistema.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.124: Requisito RS87 Recopilar información de la cuenta

RS88			
Descripción	El sistema proveerá de una interfaz gráfica que muestre la información del usuario.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.125: Requisito RS88 Interfaz mi cuenta

RS89			
Descripción	El sistema deberá permitir al usuario autenticado ver su espacio disponible, el espacio utilizado y el espacio libre en todo momento.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.126: Requisito RS89 Espacio de almacenamiento

RS90			
Descripción	El sistema deberá disponer de una contabilización actualizada del espacio disponible, el espacio utilizado y el espacio libre de los usuarios.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.127: Requisito RS90 Contabilizar espacio de almacenamiento

RS91			
Descripción	El sistema proveerá de una interfaz que muestre los datos actualizados de espacio disponible, el espacio utilizado y el espacio libre del usuario.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.128: Requisito RS91 Interfaz espacio de almacenamiento

RS92			
Descripción	El sistema deberá permitir que los usuarios cierren la sesión de el usuario en la aplicación y que puedan salir de ella desde cualquier página de la interfaz de la aplicación.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.129: Requisito RS92 Cerrar sesión

RS93			
Descripción	El sistema deberá proveer de los mecanismos necesarios para el cierre de sesión en la interfaz gráfica de la aplicación.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.130: Requisito RS93 Mecanismo cerrar sesión

RS94			
Descripción	El sistema deberá mantener la consistencia de los datos almacenados y las relaciones existentes en todo momento, en relación a las acciones de los usuarios y el sistema.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.131: Requisito RS94 Consistencia de datos

RS95			
Descripción	El sistema deberá proporcionar la seguridad necesaria para garantizar la integridad, disponibilidad y confidencialidad de la información.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.132: Requisito RS95 Seguridad

RS96			
Descripción	El sistema deberá ser desarrollado con tecnologías que dispongan de licencias disponibles sin coste monetario.		
Tipo	Funcional	Prioridad	Alta
Necesidad	Esencial	Verificabilidad	Alta

Tabla 5.133: Requisito RS96 Tecnologías

Matriz de Trazabilidad: Requisitos de Usuario – Requisitos de Software Funcionales

RS	RU01	RU02	RU03	RU04	RU05	RU06	RU07	RU08	RU09	RU10	RU11	RU12	RU13	RU14	RU15	RU16	RU17	RU18	RU19	RU20	RU21	RU22	RU23	RU24	RU25	RU26	RU27	RU28	RU29	RU30	RU31	RU32	RU33	RU34	RU35	RU36
01	X																																			
02	X																																			
03		X																																		
04		X																																		
05			X																																	
06			X																																	
07			X																																	
08				X																																
09					X																															
10					X	X																														
11					X																															
12					X																															
13					X																															
14					X																															
15						X																														
16						X																														
17							X																													
18							X																													
19							X																													
20								X																												
21								X																												
22									X																											
23										X																										
24										X																										
25											X																									
26												X																								
27												X																								
28												X																								
29												X																								
30													X																							
31														X																						
32														X																						
33														X																						
34														X																						
35														X																						
36															X																					
37																X																				
38																X																				
39																X																				
40																	X																			
41																		X																		
42																		X																		
43																		X																		

Tabla 5.134: Matriz de Trazabilidad: Requisitos de Usuario - Requisitos de Software (1)

RS	RU01	RU02	RU03	RU04	RU05	RU06	RU07	RU08	RU09	RU10	RU11	RU12	RU13	RU14	RU15	RU16	RU17	RU18	RU19	RU20	RU21	RU22	RU23	RU24	RU25	RU26	RU27	RU28	RU29	RU30	RU31	RU32	RU33	RU34	RU35	RU36
44																		X																			
45																X		X																			
46																			X																		
47																				X																	
48																				X																	
49																				X																	
50																					X																
51																					X																
52																					X																
53																					X																
54																					X																
55																						X															
56																							X														
57																							X														
58																							X														
59																							X														
60																							X														
61																							X														
62																								X													
63																									X												
64																									X												
65																									X												
66																									X												
67																									X												
68																										X											
69																										X											
70																										X											
71																											X										
72																												X									
73																												X									
74																										X		X									
75																													X								
76																														X							
77																														X							
78																															X						
79																																X					
80																																X					
81																																X					
82																																	X				
83																																	X				
84																																	X				
85																																	X				
86																																		X			

Tabla 5.135: Matriz de Trazabilidad: Requisitos de Usuario - Requisitos de Software (2)

RS	RU01	RU02	RU03	RU04	RU05	RU06	RU07	RU08	RU09	RU10	RU11	RU12	RU13	RU14	RU15	RU16	RU17	RU18	RU19	RU20	RU21	RU22	RU23	RU24	RU25	RU26	RU27	RU28	RU29	RU30	RU31	RU32	RU33	RU34	RU35	RU36
87																																		X		
88																																		X		
89																																			X	
90																																			X	
91																																			X	
92																																				X
93																																				X
94					X					X				X				X		X	X				X	X		X					X			
95		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Tabla 5.136: Matriz de Trazabilidad: Requisitos de Usuario - Requisitos de Software (3)

5.1.3 Base de Datos

Para la gestión de los datos del sistema, es necesaria la creación de una Base de Datos. En esta sección se analizan las entidades necesarias para la creación de la BD a través de un modelo entidad-relación. [4]

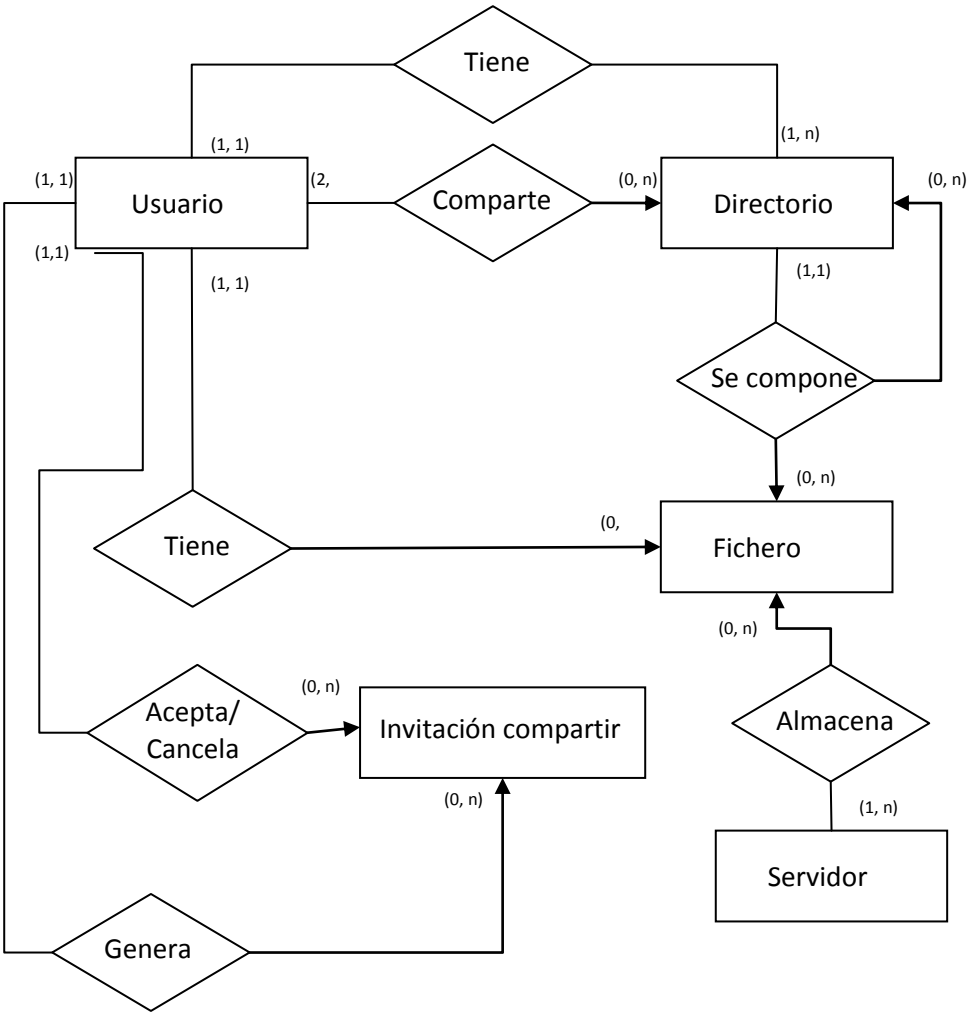


Figura 5.5: Modelo entidad-relación

A continuación se definen las propiedades de cada entidad mostrada en la Figura 5.5.

Nombre	Usuario
Objeto	Almacenar la información de los usuarios registrados en el sistema.
Alcance	Se entiende como usuario a un individuo que hace uso de la aplicación.
Clave primaria	Nombre de usuario (nick).

Tabla 5.137: Entidad Usuario

Nombre	Directorio
Objeto	Almacenar la información de los directorios almacenados en el sistema.
Alcance	Se entiende como directorio a un contenedor virtual en el que se almacenan una agrupación de ficheros y otros subdirectorios.
Clave primaria	Identificador numérico

Tabla 5.138: Entidad Directorio

Nombre	Fichero
Objeto	Almacenar la información de los ficheros almacenados en el sistema.
Alcance	Se entiende como fichero a un conjunto de bits almacenado en un dispositivo.
Clave primaria	Identificador numérico

Tabla 5.139: Entidad Fichero

Nombre	Invitación a compartir
Objeto	Almacenar la información de las invitaciones a compartir directorios que los usuarios realizan.
Alcance	Se entiende como invitación a compartir a las peticiones que realiza un usuario a otro, para dotar al invitado de permisos sobre un directorio perteneciente al anfitrión.
Clave primaria	Compuesta por: una clave primaria de usuario anfitrión, de usuario invitado y de directorio.

Tabla 5.140: Entidad Invitación a compartir

Nombre	Servidor
Objeto	Almacenar la información de los servidores que componen el sistema de almacenamiento.
Alcance	Se entiende como servidor una computadora que, formando parte de una red, provee servicios a otras computadoras denominadas clientes.
Clave primaria	Identificador numérico.

Tabla 5.141: Entidad Servidor

5.2 Diseño

En este capítulo se expondrá el diseño del sistema a desarrollar. Se mostrará la arquitectura detallada del sistema, el modelo relacional de la base de datos, los algoritmos de autenticación y generación de claves, modelos de replicación, las tareas programas, el diagrama de clases y las opciones de implementación.

5.2.1 Arquitectura del sistema

El proyecto VoCS se va a desarrollar en una arquitectura MVC, definida en el estado de la cuestión.

La elección de la arquitectura MVC es debido a que ofrece muchas ventajas que se ajustan a lo requerido en el proyecto:

- Separación de los objetos del negocio, el flujo de la aplicación y la representación de datos. Esto permite una programación ágil de las distintas capas.
- Facilita la inserción de nuevos objetos de negocio.
- Crea independencia de funcionamiento entre capas.
- Facilita la realización de pruebas unitarias de los componentes.
- Facilita el mantenimiento en caso de errores.
- Permite que el sistema sea escalable en caso de ser requerido.
- Reutilización de componentes.
- Facilita el desarrollo de prototipos rápidos.

Se procede al diseño detallado de la arquitectura del proyecto. Todas las funcionalidades del sistema pasan a formar parte de módulos, de tal manera que todos los datos y funciones dentro de cada módulo están semánticamente relacionados. Cada módulo tendrá un objetivo principal y podrá estar compuesto de otros sub-módulos, dirigidos a cumplir parte del objetivo principal. Los módulos que componen el sistema son:

- Módulo de gestión de directorios.
- Módulo de gestión de ficheros.
- Módulo de gestión de usuarios.
- Módulo de inicio.

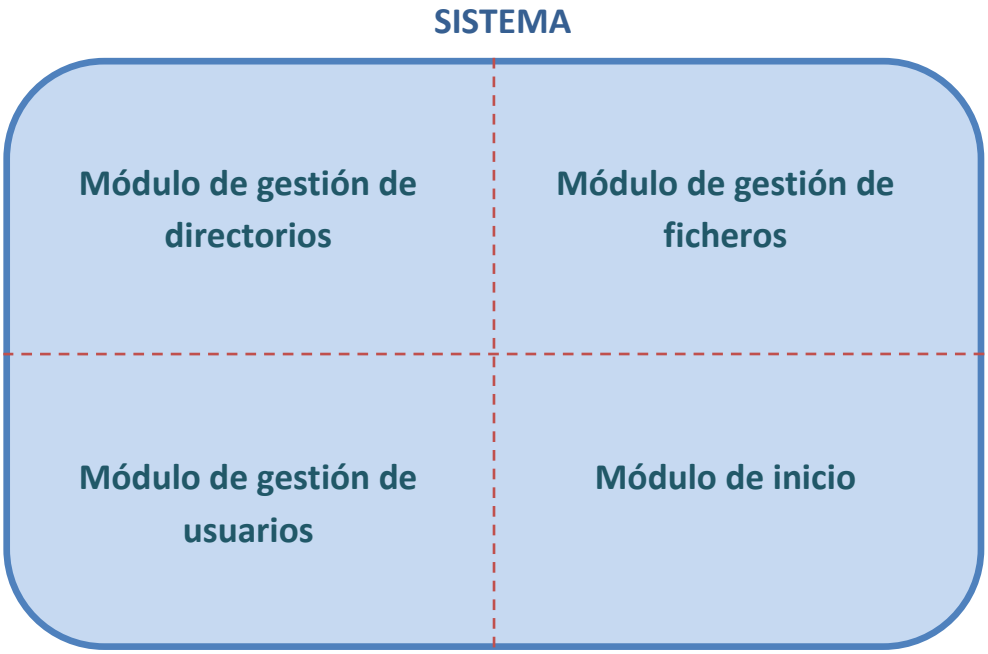


Figura 5.6: Arquitectura

Las funcionalidades que componen los módulos han sido definidas en los casos de uso. Todas ellas tienen un comportamiento general definido en el siguiente diagrama de flujo.

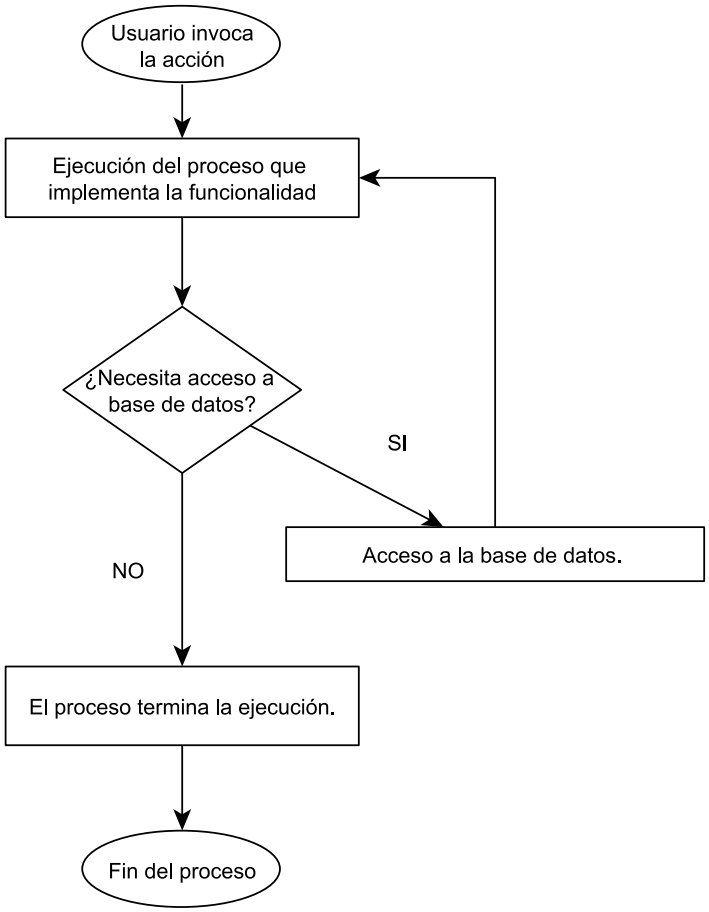


Figura 5.7: Diagrama de flujo de las funcionalidades

5.2.1.1 Modulo de gestión de directorios

El objetivo principal del módulo es proveer al sistema de las funcionalidades necesarias para el control y manipulación de directorios.

Según el análisis realizado anteriormente, en concreto los casos de uso, el módulo de gestión directorios ofrece las siguientes funcionalidades:

- Permitir crear un directorio nuevo al usuario.
- Permitir navegar al usuario por su árbol de directorios.
- Permitir el borrado de directorios.
- Permitir la invitación a compartir directorios.
- Permitir al usuario listar sus invitaciones a compartir un directorio.
- Permitir aceptar la invitación a compartir un directorio.
- Permitir cancelar la invitación a compartir un directorio.

Atendiendo a las funcionalidades que debe aportar el módulo de gestión de directorios, éste se divide en los sub-módulos:

- Crear directorio.
- Eliminar directorio.
- Navegar.
- Invitar.
- Aceptar invitación.
- Cancelar invitación.
- Listar invitaciones.

Para que el módulo pueda cumplir con su objetivo es necesaria la manipulación de entidades pertenecientes al modelo. Las entidades que utiliza son:

- Entidad Directorio.
- Entidad Fichero.
- Entidad Invitación.
- Entidad Usuario.

Crear directorio

El sub-módulo dotará al sistema de la funcionalidad crear directorio. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario. Estas clases se definen en el diagrama de clases.

Provee una interfaz al usuario de creación de directorios. Dicha interfaz se detalla en el punto INTERFACES.

Las funciones que llevará a cabo son:

- Insertar en el sistema el nuevo directorio.
- Asignar el nuevo directorio al usuario.
- Poner el directorio como hijo del directorio actual en el que está el usuario.
- Si el directorio sobre el que se creó está compartido, el nuevo directorio debe ser compartido con los usuarios que comparten el directorio.

El siguiente diagrama de flujo muestra el proceso para crear un directorio en el sistema.

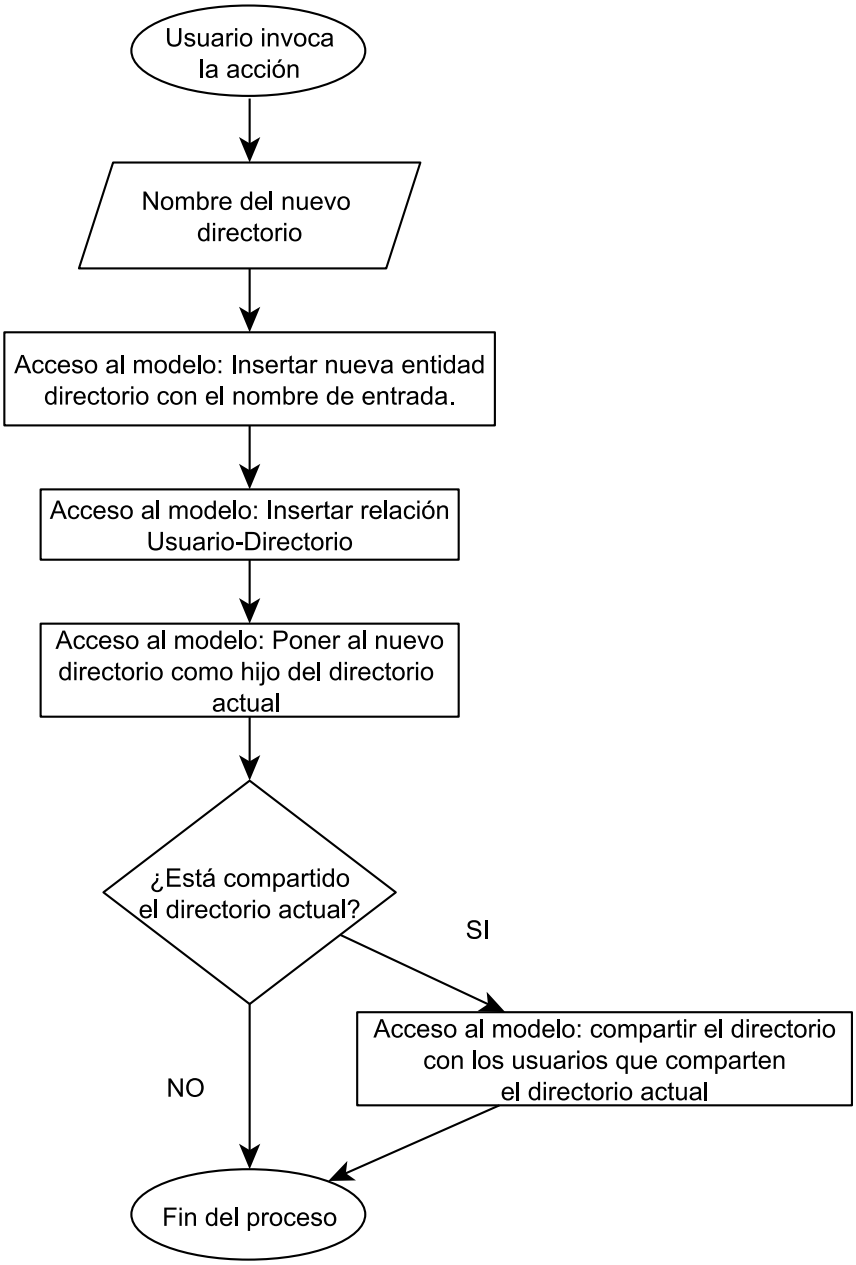


Figura 5.8: Diagrama de flujo. Crear directorio

Eliminar directorio

El sub-módulo dotará al sistema de la funcionalidad eliminar directorio. Para ello, debe implementar las clases y funciones necesarias para la eliminación de directorios por parte de un usuario. Estas clases se detallan en el diagrama de clases.

Provee una interfaz al usuario para eliminar directorios. Dicha interfaz se detalla en el punto INTERFACES.

Las funciones que lleva a cabo son:

- Eliminar un directorio y todo su contenido. Por contenido se entiende a todos los ficheros y directorios contenidos en el directorio a eliminar.

El siguiente diagrama de flujo muestra el proceso para crear un directorio en el sistema.

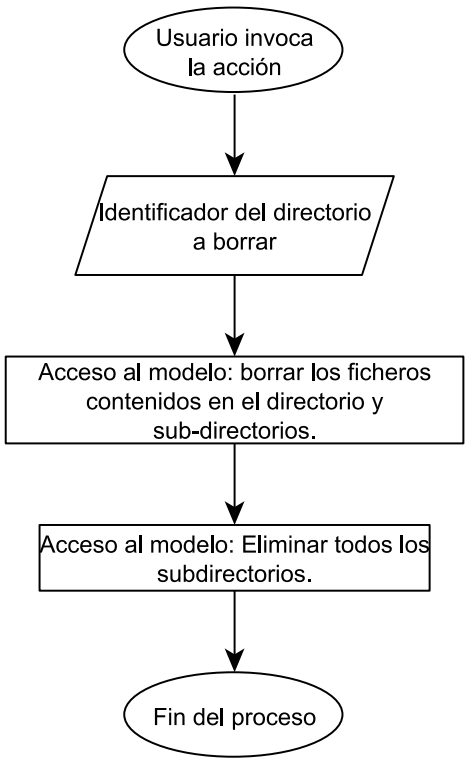


Figura 5.9: Diagrama de flujo. Eliminar directorio

Navegar

El sub-módulo dotará al sistema de la funcionalidad navegar entre directorios. Para ello, debe implementar las clases y funciones necesarias, definidas en el diagrama de clases.

Provee una interfaz al usuario para navegar, detallada en el punto INTERFACES.

Las funciones que lleva a cabo son:

- Listado de directorios y sus contenidos.
- Navegación.

Las acciones que realiza para el listado de directorios y sus contenidos son:

- Obtener los ficheros del directorio actual.
- Obtener los directorios del directorio actual.
- Generar una lista con lo obtenido en los pasos anteriores.

Las acciones que se realiza para la navegación son:

- Cambiar el directorio actual por el nuevo directorio al que se quiere navegar.
- Generar una lista con los directorios padres del directorio actual.

Los siguientes diagramas de flujo muestran las acciones que realizan los diferentes procesos que componen el módulo.

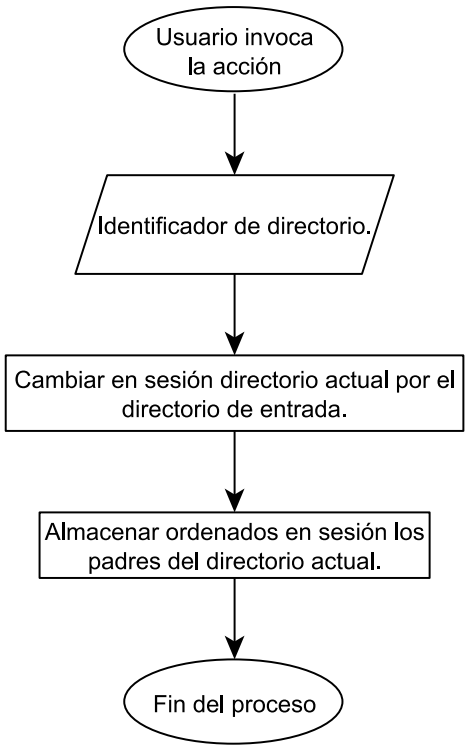


Figura 5.10: Diagrama de flujo. Navegar: cambiar directorio actual

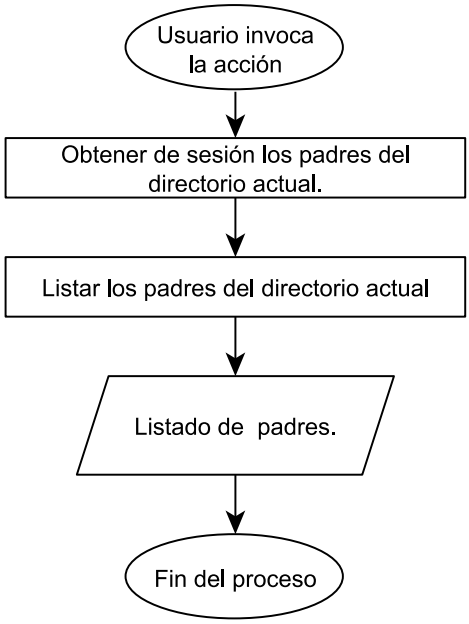


Figura 5.11: Diagrama de flujo. Navegar: Listado de los directorios padres

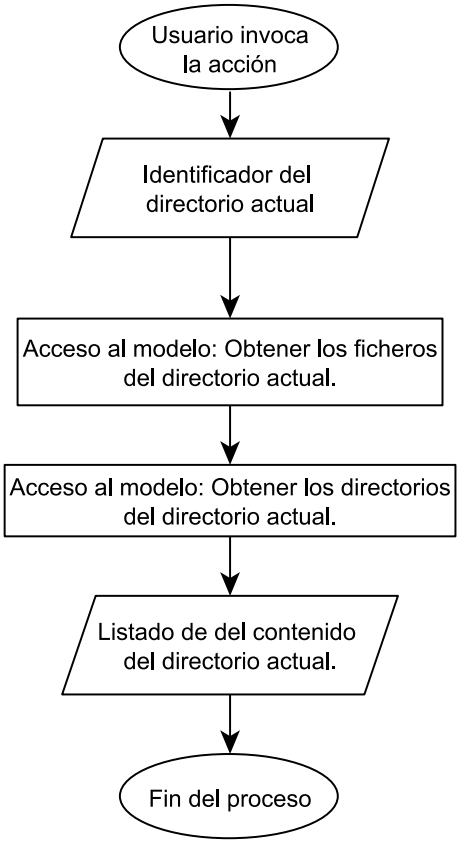


Figura 5.12: Diagrama de flujo. Navegar: Listado del contenido del directorio actual

Invitar

El sub-módulo invitar es el encargado de dotar al sistema de la funcionalidad invitar a compartir un directorio. Para ello, debe implementar las clases y funciones necesarias, definidas en el diagrama de clases.

El sub-módulo provee una interfaz al usuario para invitar a compartir directorios. Dicha interfaz se detalla en el punto INTERFACES.

Las acciones que lleva a cabo son:

- Insertar en el sistema una invitación que relacione al anfitrión, el invitado y el directorio que se quiere compartir.
- Enviar un correo al invitado notificando la invitación.

El siguiente diagrama de flujo muestra el proceso para crear un directorio en el sistema.

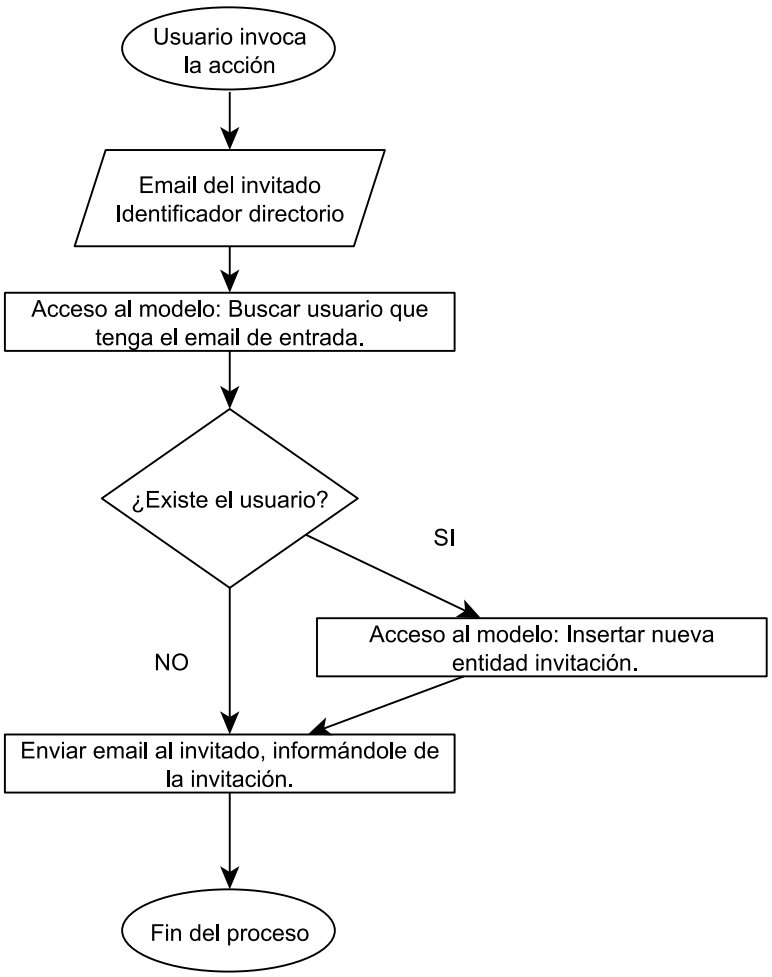


Figura 5.13: Diagrama de flujo. Invitar a compartir

Listar invitaciones

El sub-módulo listar invitaciones es el encargado de dotar al sistema de la funcionalidad listar invitaciones, en las que el usuario que solicita el listado sea el invitado. Para ello, debe implementar las clases y funciones necesarias, definidas en el diagrama de clases.

El sub-módulo provee una interfaz al usuario para la visualización del listado de invitaciones. Dicha se interfaz se detalla en el punto INTERFACES.

Las acciones que lleva a cabo son:

- Buscar las invitaciones del usuario que invocó la acción.
- Generar una lista con las invitaciones encontradas en el paso anterior.

El siguiente diagrama de flujo muestra el proceso para crear un directorio en el sistema.

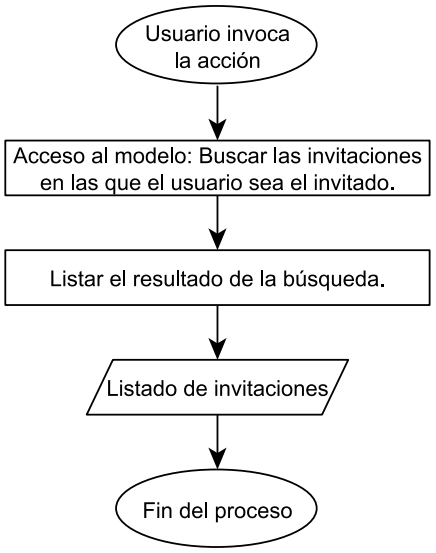


Figura 5.14: Diagrama de flujo. Listar invitaciones.

Cancelar invitación

El sub-módulo es el encargado de dotar al sistema de la funcionalidad cancelar invitación. Para ello, debe implementar las clases y funciones necesarias, definidas en el diagrama de clases. Provee una interfaz al usuario para cancelar invitaciones. Dicha interfaz se detalla en el punto INTERFACES.

Las acciones que lleva a cabo son:

- Eliminar una invitación para compartir un directorio y su contenido.

El siguiente diagrama de flujo muestra el proceso para cancelar una invitación para compartir un directorio.

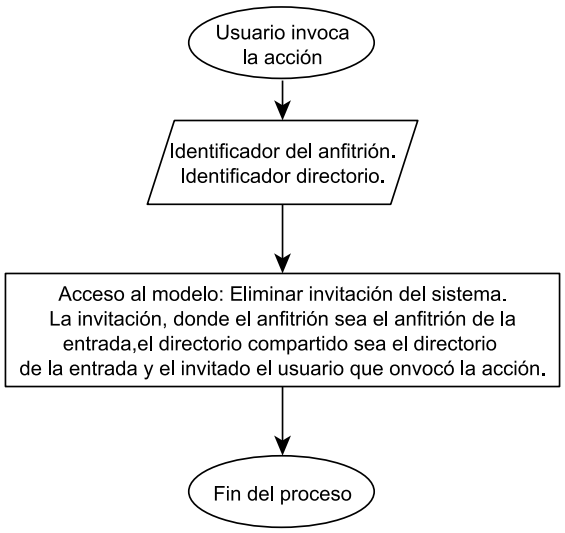


Figura 5.15: Diagrama de flujo. Cancelar invitación.

Aceptar invitación

El sub-módulo es el encargado de dotar al sistema de la funcionalidad cancelar invitación. Para ello, debe implementar las clases y funciones necesarias, definidas en el diagrama de clases. Provee una interfaz al usuario para aceptar invitaciones. Dicha interfaz se detalla en el punto INTERFACES.

Las acciones que lleva a cabo son:

- Realizar las acciones oportunas para que el anfitrión y el invitado compartan el directorio y todo su contenido.
- Eliminar la invitación para compartir el directorio.

El siguiente diagrama de flujo muestra el proceso para aceptar una invitación para compartir un directorio.

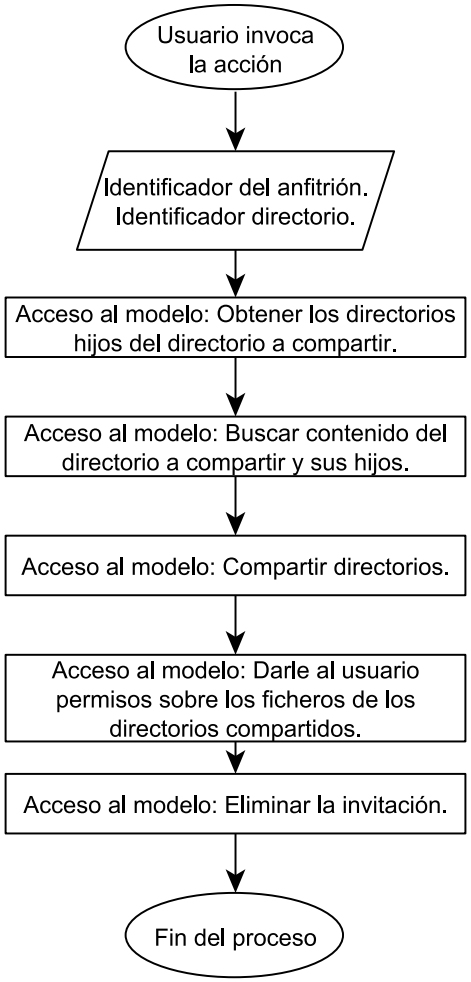


Figura 5.16: Diagrama de flujo. Aceptar invitación

5.2.1.2 Módulo de gestión de ficheros

El objetivo principal del módulo es proveer al sistema de las funcionalidades necesarias para el control y manipulación de ficheros.

Según el análisis realizado anteriormente, en concreto los casos de uso, el módulo de gestión de ficheros ofrece las siguientes funcionalidades:

- Permitir subir un fichero al usuario.
- Permitir descargar un fichero al usuario.
- Permitir el borrado de ficheros.
- Permitir visualizar un listado de ficheros eliminados.
- Permitir visualizar un listado de las versiones anterior de un fichero.
- Permitir al usuario restaurar sus ficheros eliminados.

Atendiendo a las funcionalidades que debe aportar el módulo, éste se divide en los sub-módulos:

- Subir fichero.
- Descargar fichero.
- Eliminar fichero.
- Listar ficheros eliminados.
- Listar versiones anteriores.
- Restaurar ficheros.

Para cumplir con el objetivo marcado en el módulo es necesaria la utilización de los modelos. Las entidades que manejará el módulo de gestión de ficheros son:

- Entidad Fichero.
- Entidad Usuario.
- Entidad Servidor.
- Entidad Directorio.

Subir fichero

El sub-módulo dotará al sistema de la funcionalidad subir fichero. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario, definidas en el diagrama de clases.

Provee una interfaz al usuario para la subida de ficheros. Dicha interfaz se detalla en el punto INTERFACES.

Las funciones que llevará a cabo son:

- Almacenar el fichero en el sistema de almacenamiento físico del servidor.
- Insertar la nueva entidad fichero en el sistema.
- Compartir el fichero si se debe.
- Replicar el fichero.

El siguiente diagrama de flujo muestra el proceso para subir un fichero al sistema.

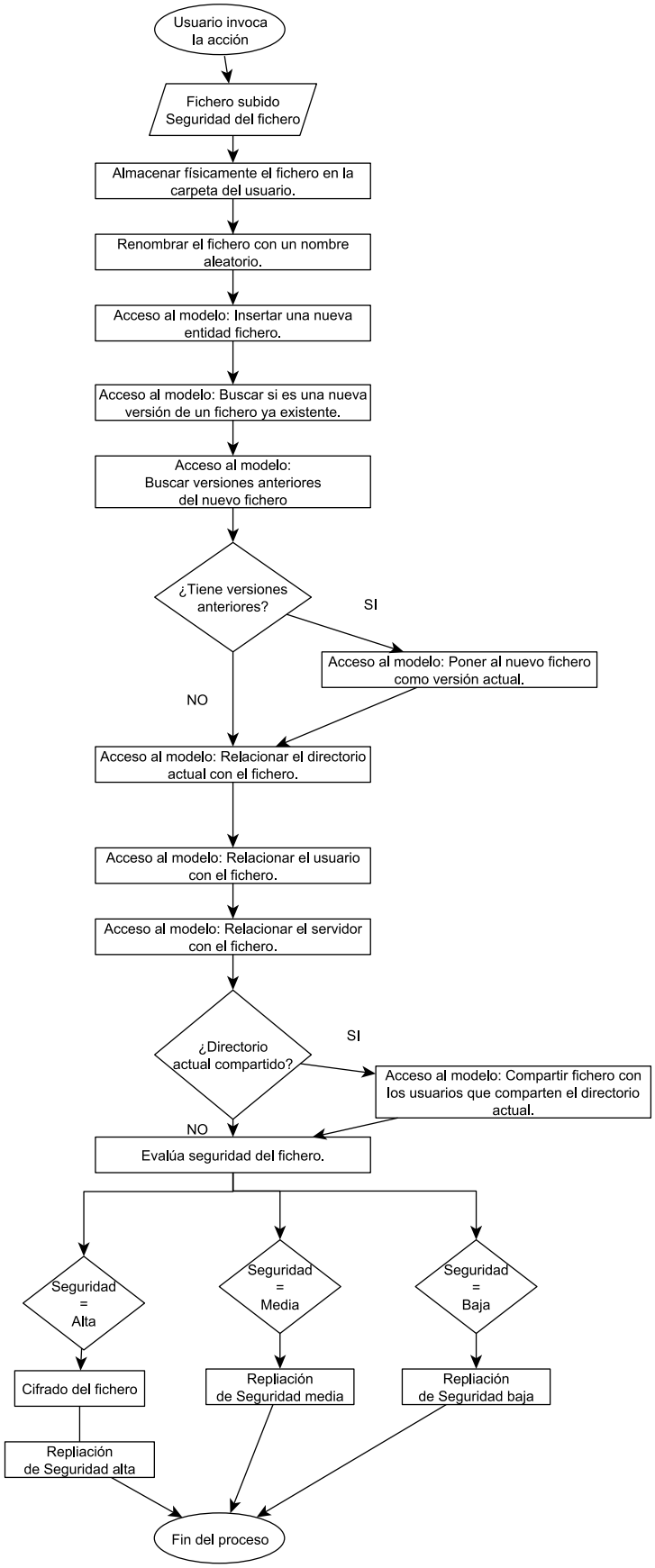


Figura 5.17: Diagrama de flujo. Subir fichero

Descargar fichero

El sub-módulo dotará al sistema de la funcionalidad descargar fichero. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario, definidas en el diagrama de clases.

Provee una interfaz al usuario para la bajada de ficheros. Dicha interfaz se detalla en el punto INTERFACES.

Las funciones que llevará a cabo son:

- Comprobar si el fichero sobre el que se hizo la petición de descarga, está en el servidor sobre el que se realizó la petición.
- Si lo está, lo sirve al usuario.
- Si no lo está, lo pide bajo demanda y sigue la política establecida por el modelo de replicación.

El siguiente diagrama de flujo muestra el proceso para subir un fichero al sistema.

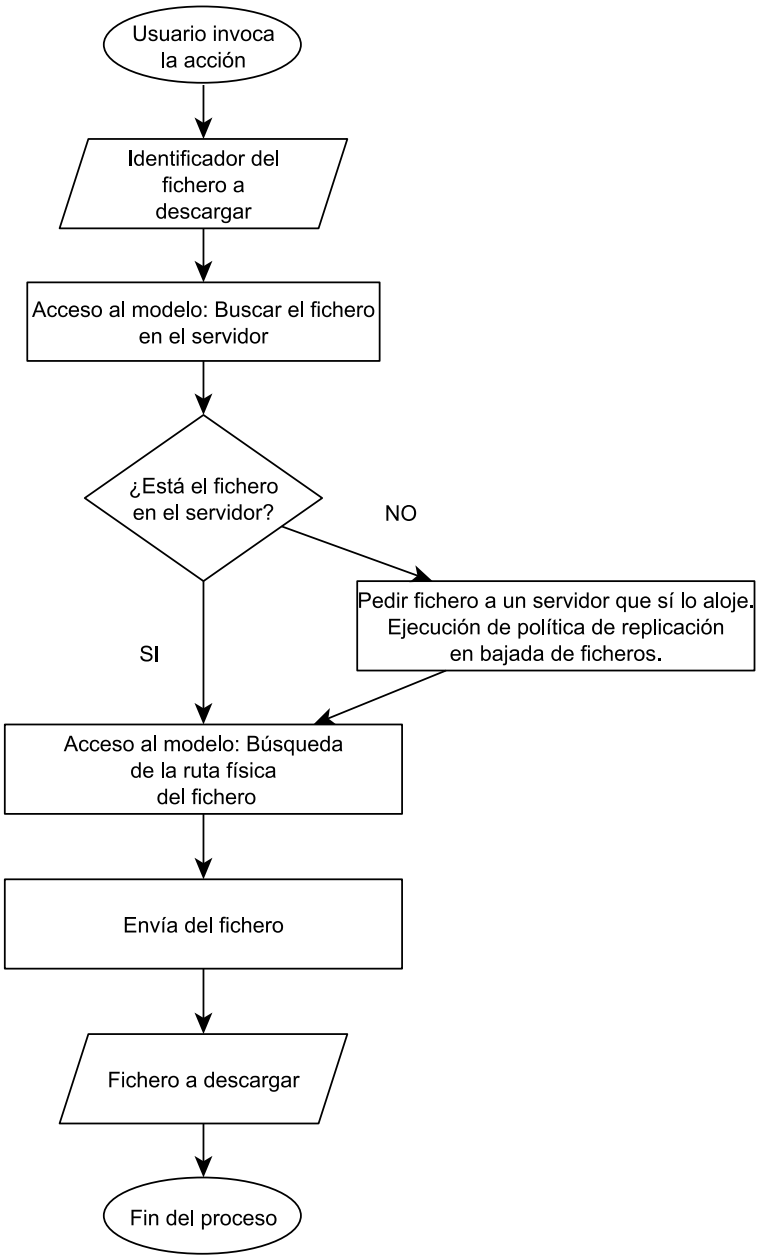


Figura 5.18: Diagrama de flujo. Descargar fichero.

Eliminar fichero

El sub-módulo dotará al sistema de la funcionalidad eliminar fichero. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario, definidas en el diagrama de clases.

Provee una interfaz al usuario para la eliminación de ficheros. Dicha interfaz se detalla en el punto INTERFACES.

Las funciones que llevará a cabo son:

- Marcar en el sistema el fichero como borrado. Pero sin borrarlo físicamente.
- Actualizar espacio utilizado del usuario.

El siguiente diagrama de flujo muestra el proceso para subir un fichero al sistema.

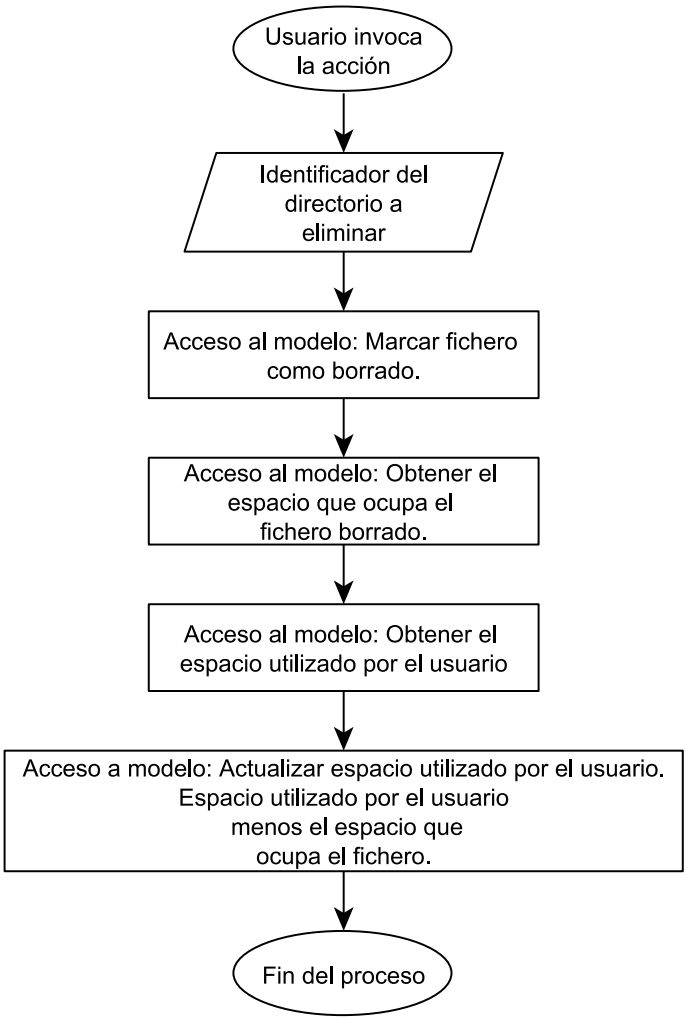


Figura 5.19: Diagrama de flujo. Eliminar fichero

Listar ficheros eliminados

El sub-módulo dotará al sistema de la funcionalidad listar ficheros eliminados. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario, definidas en el diagrama de clases.

Provee una interfaz al usuario para el listado de ficheros eliminados. Dicha interfaz se detalla en el punto INTERFACES.

Las funciones que llevará a cabo son:

- Obtener y listar los ficheros del usuario marcados como borrados en el sistema.

El siguiente diagrama de flujo muestra el proceso para subir un fichero al sistema.

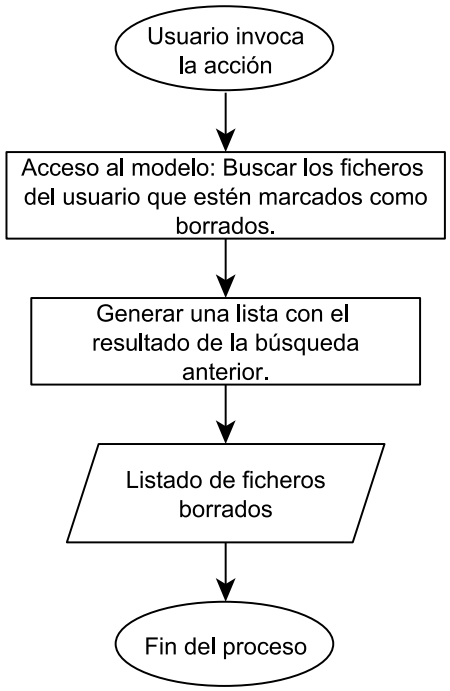


Figura 5.20: Diagrama de flujo. Listar ficheros borrados

Listar versiones anteriores

El sub-módulo dotará al sistema de la funcionalidad listar las versiones anteriores de un fichero. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario, definidas en el diagrama de clases.

Provee una interfaz al usuario para el listado de versiones anteriores de los ficheros. Dicha interfaz se detalla en el punto INTERFACES.

Las funciones que llevará a cabo son:

- Dado un fichero activo, obtener y listar los ficheros que sean sus versiones anteriores.

El siguiente diagrama de flujo muestra el proceso para subir un fichero al sistema.

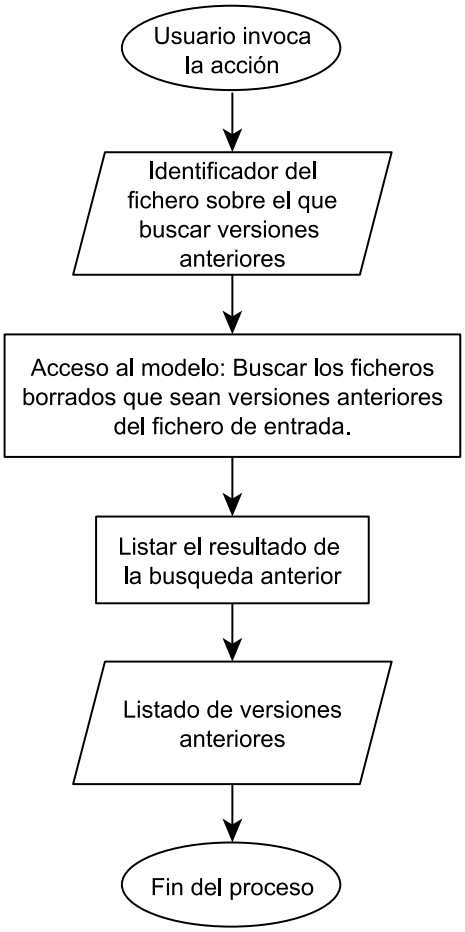


Figura 5.21: Diagrama de flujo. Listar versiones anteriores

Restaurar ficheros

El sub-módulo dotará al sistema de la funcionalidad restaurar un fichero eliminado. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario, definidas en el diagrama de clases.

Provee una interfaz al usuario para la restauración detallada en el punto INTERFACES.

Las funciones que llevará a cabo son:

- Dado un fichero borrado, ponerlo activo.
- Si el fichero tiene versiones, ponerlo como la versión más actual.

El siguiente diagrama de flujo muestra el proceso para subir un fichero al sistema.

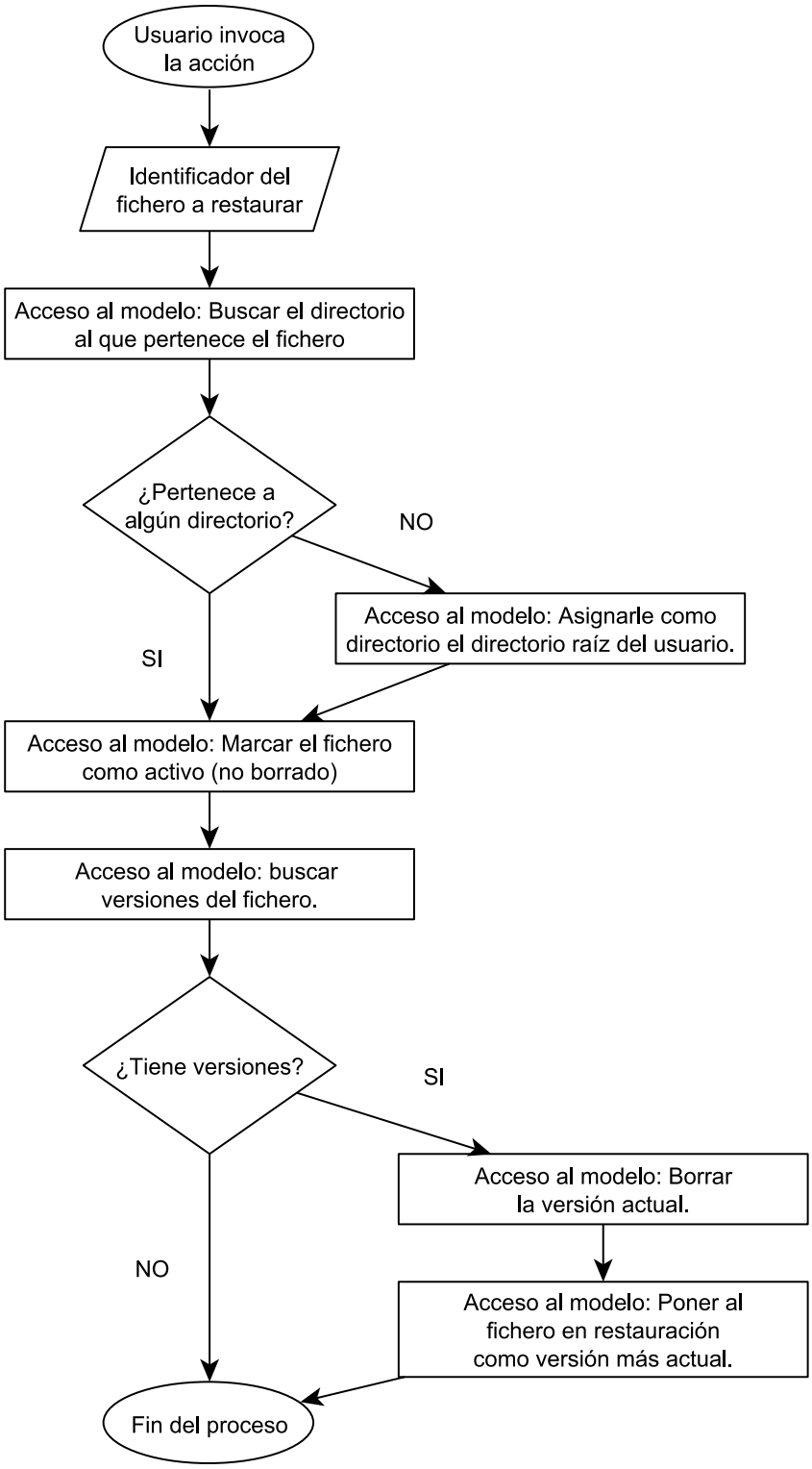


Figura 5.22: Diagrama de flujo. Restaurar fichero

5.2.1.3 Módulo de gestión de usuarios

El objetivo principal del módulo es proveer al sistema de las funcionalidades necesarias para el control y manipulación de la cuenta de usuario.

Según el análisis realizado anteriormente, en concreto los casos de uso, el módulo de gestión de usuarios ofrece las siguientes funcionalidades:

- Permitir al usuario visualizar la información de su cuenta.
- Permitir al usuario cambiar su contraseña de cuenta.
- Permitir al usuario visualizar su espacio de almacenamiento disponible.
- Permitir al usuario recomendar VoCS a otras personas.

Atendiendo a las funcionalidades que debe aportar el módulo, éste se divide en los sub-módulos:

- Ver mi cuenta.
- Cambiar contraseña.
- Ver espacio disponible.
- Recomendar a un amigo.

Para cumplir con el objetivo marcado en el módulo es necesaria la utilización de los modelos. Las entidades que manejará el módulo de gestión de ficheros son:

- Entidad Usuario.

Ver mi cuenta

El sub-módulo dotará al sistema de la funcionalidad ver mi cuenta. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario, definidas en el diagrama de clases.

Provee una interfaz al usuario para la visualización de la información de la cuenta detallada en el punto INTERFACES.

Las funciones que llevará a cabo son:

- Obtener y listar la información del usuario que invocó la acción.

El siguiente diagrama de flujo muestra el proceso para subir un fichero al sistema.

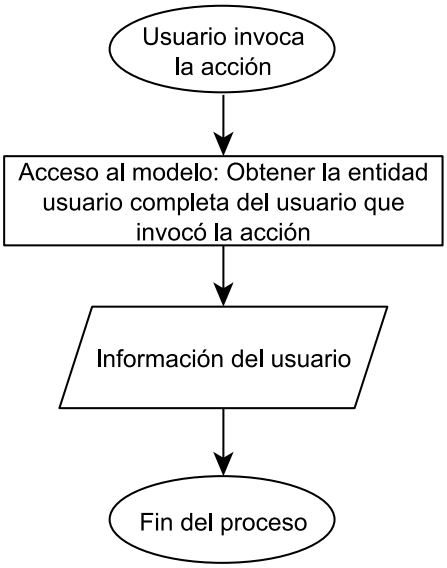


Figura 5.23: Diagrama de flujo. Ver mi cuenta

Ver espacio disponible

El sub-módulo dotará al sistema de la funcionalidad ver el espacio disponible en el sistema de almacenamiento. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario, definidas en el diagrama de clases.

Provee una interfaz al usuario para la visualización del espacio de su cuenta en el punto INTERFACES.

Las funciones que llevará a cabo son:

- Obtener y listar la información del usuario que invocó la acción.

El siguiente diagrama de flujo muestra el proceso para subir un fichero al sistema.

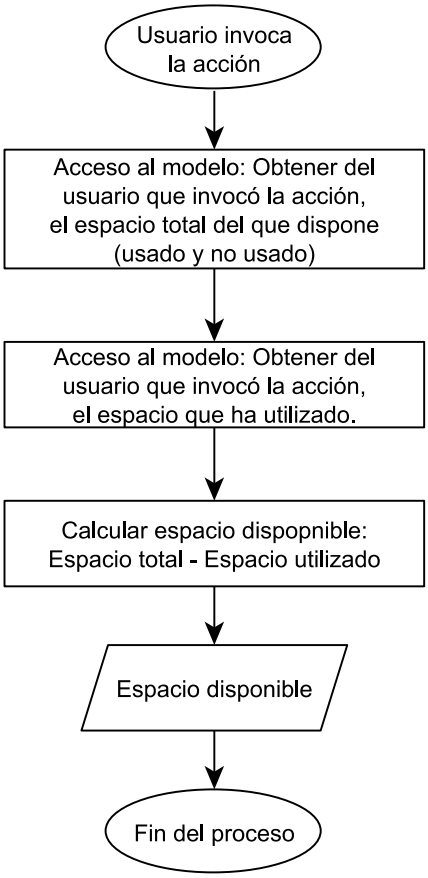


Figura 5.24: Diagrama de flujo. Ver espacio

Recomendar a un amigo

El sub-módulo dotará al sistema de la funcionalidad recomendar a otra persona la aplicación VoCS. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario, definidas en el diagrama de clases.

Provee una interfaz al usuario para que pueda invitar a sus amigos a través de un correo electrónico, detallada en el punto INTERFACES.

Las funciones que llevará a cabo son:

- Dado una dirección, envía un correo electrónico a dicha dirección recomendando el uso de la aplicación VoCS.

El siguiente diagrama de flujo muestra el proceso para subir un fichero al sistema.

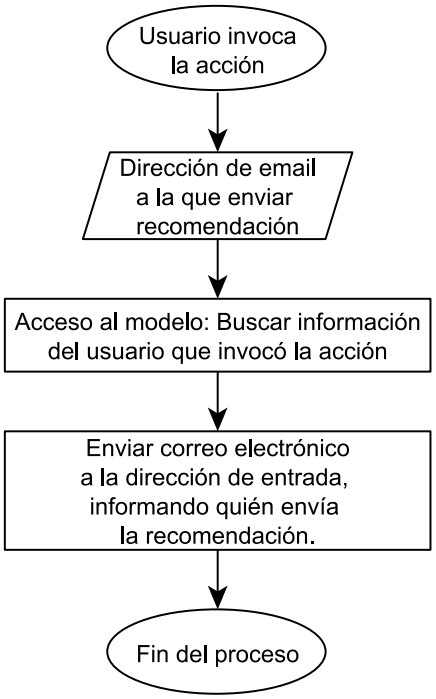


Figura 5.25: Diagrama de flujo. Recomendar a un amigo

Cambiar contraseña

El sub-módulo dotará al sistema de la funcionalidad cambiar la contraseña de la cuenta de un usuario. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario, definidas en el diagrama de clases.

Provee una interfaz al usuario para que pueda cambiar su contraseña, detallada en el punto INTERFACES.

Las funciones que llevará a cabo son:

- Certificar que el usuario que quiere cambiar la contraseña, sabe la actual.
- Actualizar el hash de comparación sacado de la nueva contraseña.
- Cifrar la 2º mitad de la clave de cifrado con el nuevo hash de comparación.

El siguiente diagrama de flujo muestra el proceso para subir un fichero al sistema.

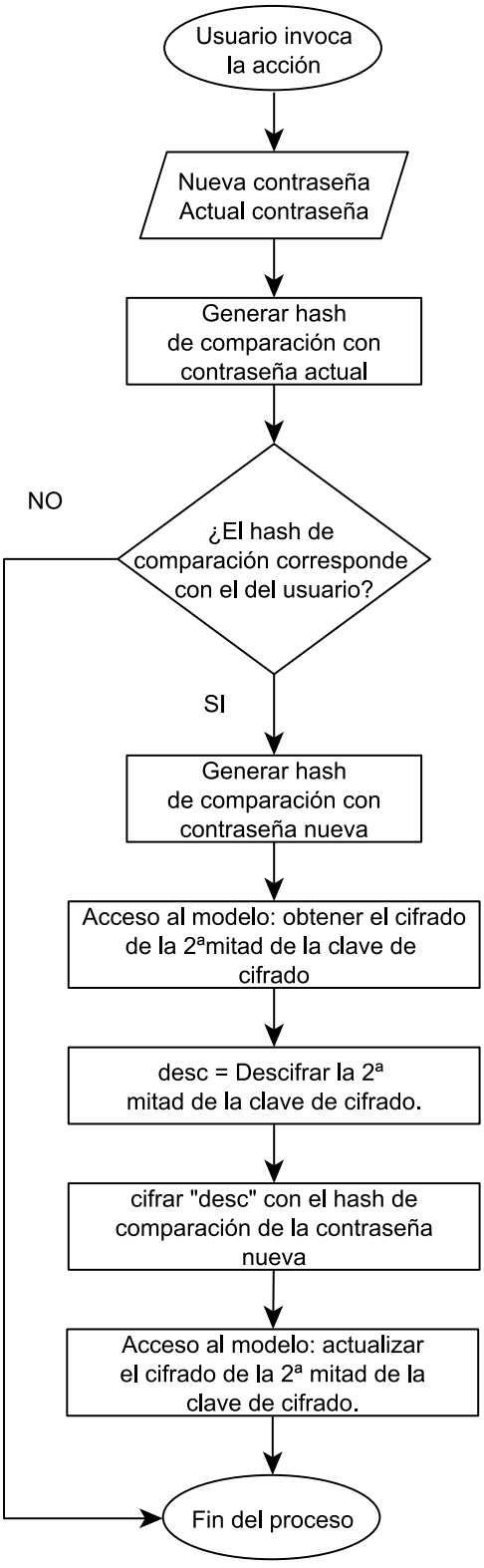


Figura 5.26: Diagrama de flujo. Cambiar contraseña

5.2.1.4 Módulo de inicio

El objetivo principal del módulo es proveer al sistema de las funcionalidades necesarias para que los usuarios puedan iniciar y terminar de usar la aplicación.

Según los casos de uso y requisitos realizados anteriormente, el módulo de inicio ofrece las siguientes funcionalidades:

- Permitir al usuario registrarse en la aplicación.
- Permitir al usuario autenticarse y acceder a la zona de usuarios autenticados.
- Permitir al usuario cerrar sesión y dejar de estar autenticado para el sistema.

Atendiendo a las funcionalidades que debe aportar el módulo, éste se divide en los sub-módulos:

- Registrarse.
- Iniciar sesión.
- Cerrar sesión.

Para cumplir con el objetivo marcado en el módulo es necesaria la utilización de los modelos. Las entidades que manejará el módulo de gestión de ficheros son:

- Entidad Usuario.

Registrarse

El sub-módulo dotará al sistema de la funcionalidad registrarse en el sistema. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario, definidas en el diagrama de clases.

Provee una interfaz al usuario para que pueda cambiar su contraseña, detallada en el punto INTERFACES.

Realiza dos funciones que son:

- Registro a espera de confirmación. En este paso el usuario rellena un formulario de registro. El sistema envía un correo electrónico, a la dirección que indique en el formulario, para que el usuario confirme el registro.
- Confirmación del registro. El usuario recibió un correo electrónico para confirmar el registro que hizo en el paso uno. El correo incluirá un enlace para que pueda hacerlo.

Registro de confirmación realiza las siguientes acciones:

- Genera un hash de comparación para la autenticación del usuario y el cifrado de claves.
- Almacena en el sistema, el hash de comparación junto a la información personal del usuario.
- Deja el estado del usuario como inactivo.

Confirmación del registro realiza las siguientes acciones:

- Poner al usuario como activo.
- Crear directorio raíz del usuario.
- Generar la clave de cifrado y almacenarla cifrada.

El siguiente diagrama de flujo muestra el proceso para subir un fichero al sistema.

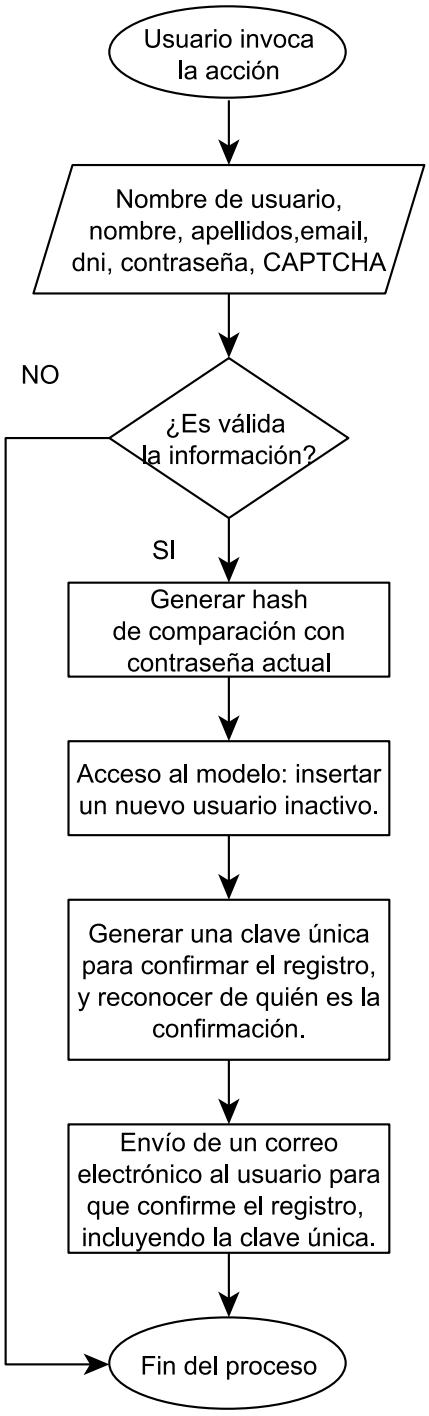


Figura 5.27: Diagrama de flujo. Registro a espera de confirmación

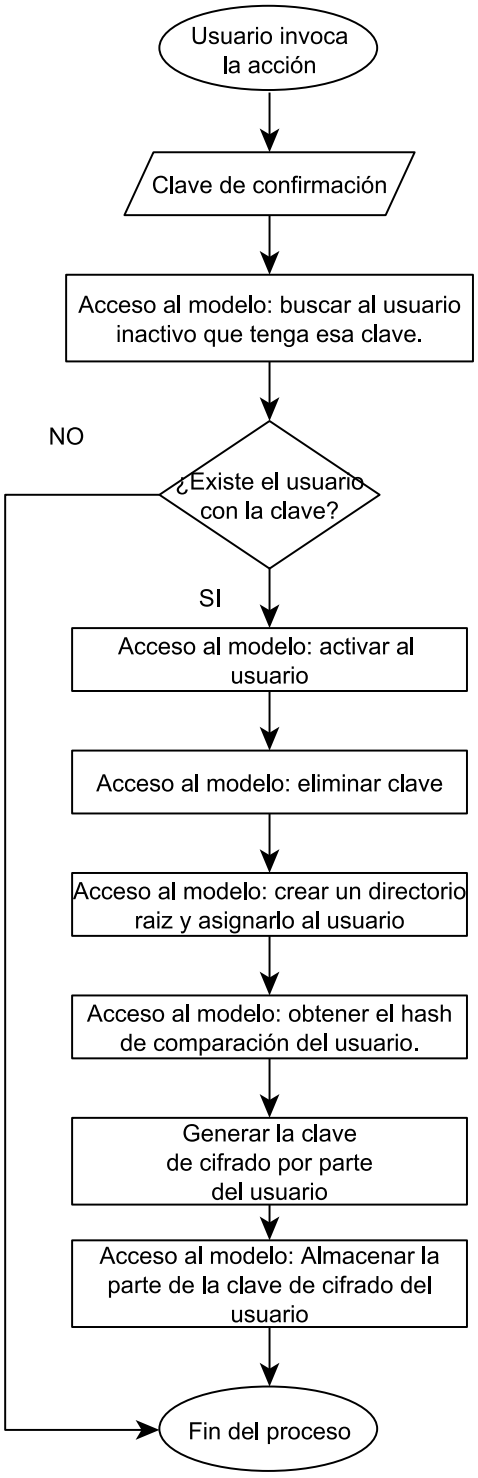


Figura 5.28: Diagrama de flujo. Registrarse confirmación

Iniciar sesión

El sub-módulo dotará al sistema de la funcionalidad iniciar sesión en la aplicación. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario, definidas en el diagrama de clases.

Provee una interfaz al usuario para que pueda cambiar su contraseña, detallada en el punto INTERFACES.

Las funciones que llevará a cabo son:

- Identificar al usuario.
- Inicia la sesión, incluyendo en ella:
    - Rol del usuario.
    - Identificador del usuario.
    - Directorio raíz del usuario.

El siguiente diagrama de flujo muestra el proceso para subir un fichero al sistema.

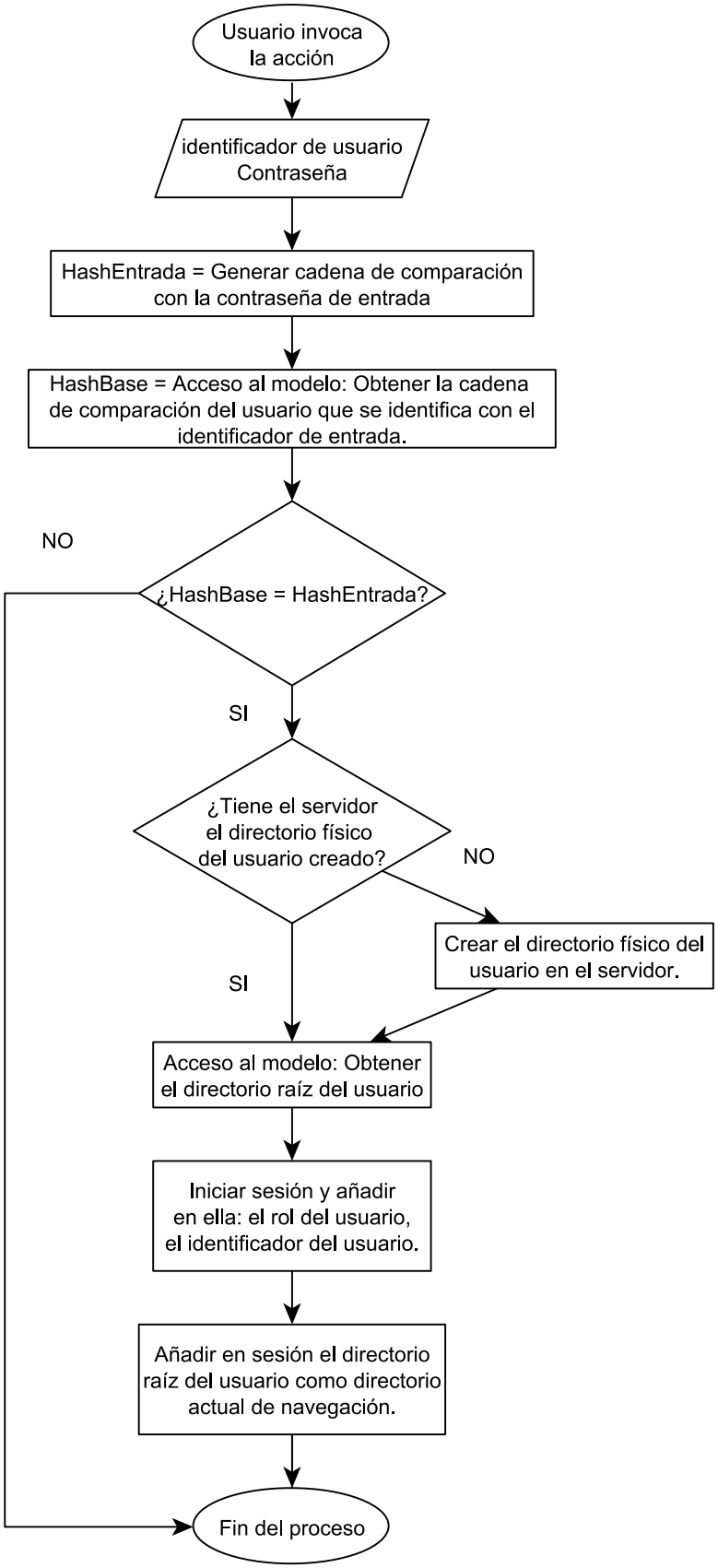


Figura 5.29: Diagrama de flujo. Iniciar sesión

Cerrar sesión

El sub-módulo dotará al sistema de la funcionalidad cerrar sesión en la aplicación. Para ello, debe implementar las clases y funciones necesarias para la creación de directorios por parte de un usuario, definidas en el diagrama de clases.

Provee una interfaz al usuario para que pueda cambiar su contraseña, detallada en el punto INTERFACES.

Las funciones que llevará a cabo son:

- Hacer que el usuario deje de estar autenticado en la aplicación.
- Eliminar los datos de sesión.

El siguiente diagrama de flujo muestra el proceso para subir un fichero al sistema.

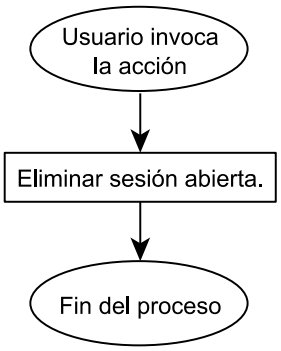


Figura 5.30: Diagrama de flujo. Cerrar sesión

5.2.2 Interfaces

En este apartado se mostrará el diseño de las diferentes vistas que componen la aplicación y permiten al usuario interactuar con ella invocando acciones.

En la Figura 5.31 se muestra un esquema general de la navegación entre las diferentes interfaces, representadas como cajas.

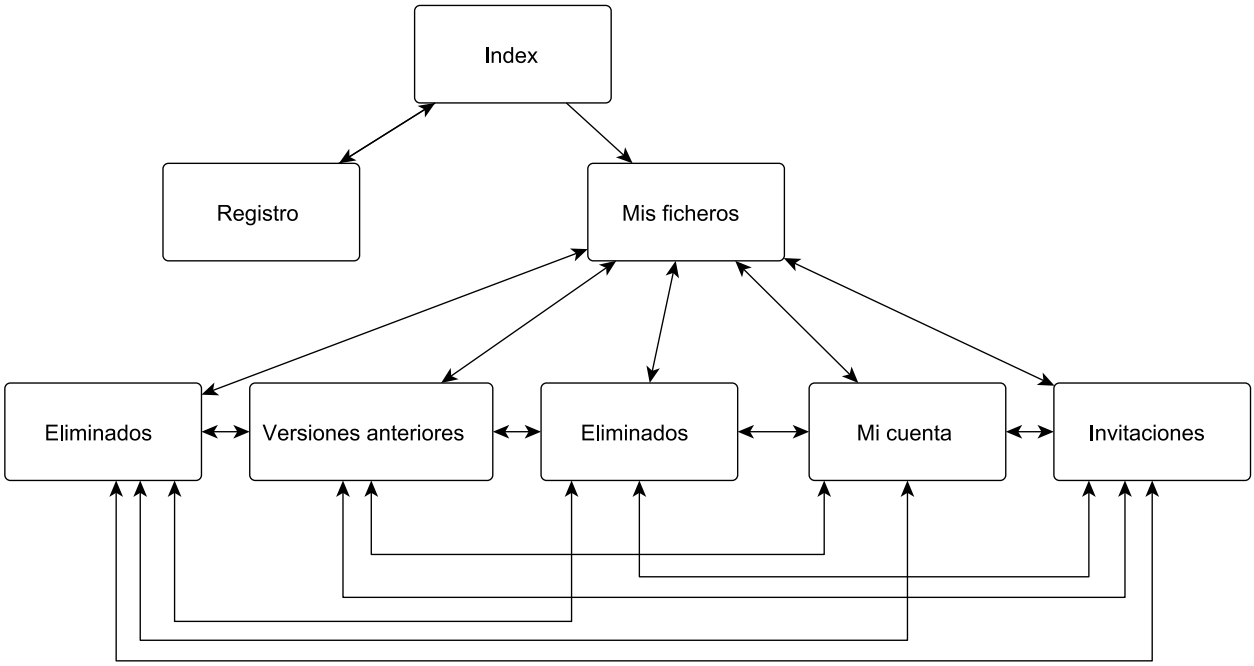


Figura 5.31: Conjunto de interfaces

A continuación se detallará el diseño de cada interfaz.

5.2.2.1 Plantilla o Layout

Se ha diseñado una plantilla que todas las interfaces utilizarán exceptuando algunas.

Figura 5.32 Interfaces: Layout

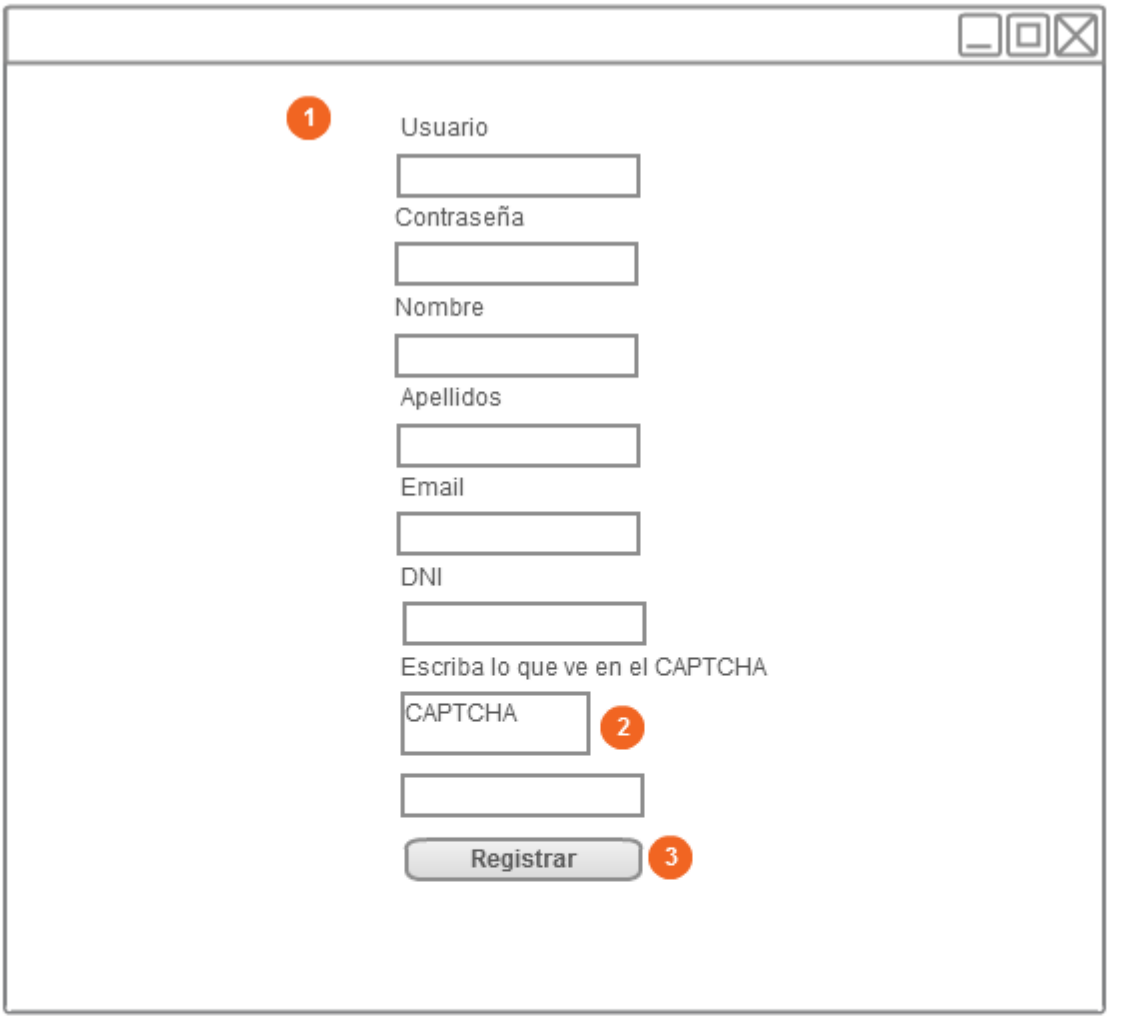
5.2.2.2 Index

Index es la vista de inicio de la aplicación donde se permitirá al usuario iniciar sesión o dirigirse a la vista de registro. No utiliza la plantilla.

Figura 5.33 Interfaces: Index

5.2.2.3 Registro

Registro es la vista que proporciona el usuario el formulario de registro de la aplicación. No utiliza la plantilla.



1. Formulario de registro que contiene los campos: usuario(nick), contraseña, nombre, apellidos, correo electrónico, DNI que corresponde a la información personal y de la cuenta del usuario.
2. CAPTCHA. Campo perteneciente al formulario. Utilizado para evitar que bots se registren.
3. Botón que invoca el proceso de registrarse.

Figura 5.34 Interfaces: registro

5.2.2.4 Mis ficheros

Mis ficheros es la vista que proporciona al usuario la posibilidad de navegar entre sus directorios y manipular su contenido. Utiliza la plantilla.

Figura 5.35 Interfaces mis ficheros

5.2.2.5 Modal crear directorio

Es un modal perteneciente a la vista “mis ficheros” y muestra la interfaz de crear directorio.

Proporciona al usuario la posibilidad de crear nuevos directorios.

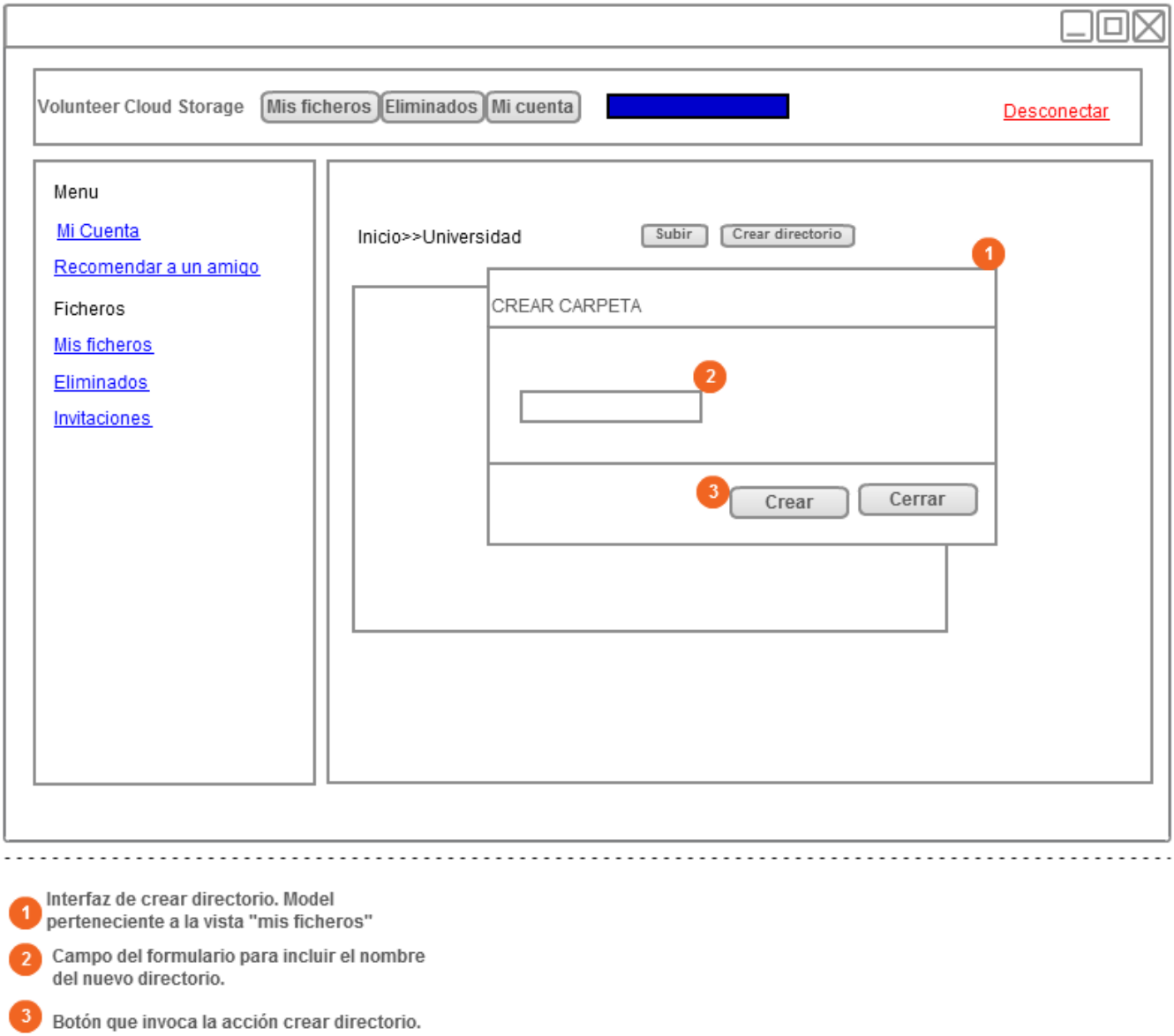


Figura 5.36 Interfaces: Mis ficheros, modal crear directorio

5.2.2.6 Modal subir ficheros

Es un modal perteneciente a la vista “mis ficheros” y muestra la interfaz de subir ficheros. Proporciona al usuario la posibilidad de subir ficheros y borrarlos.

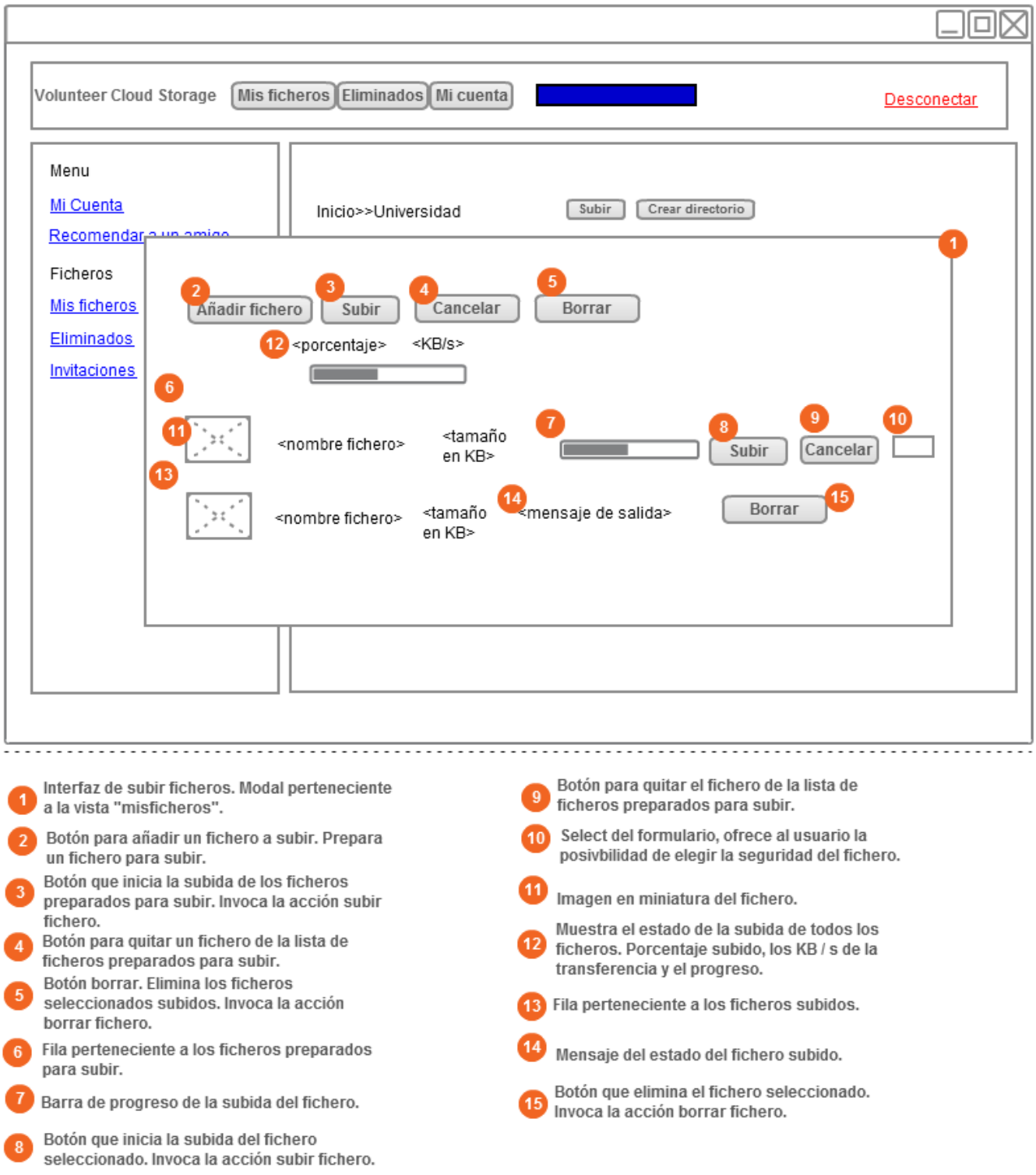
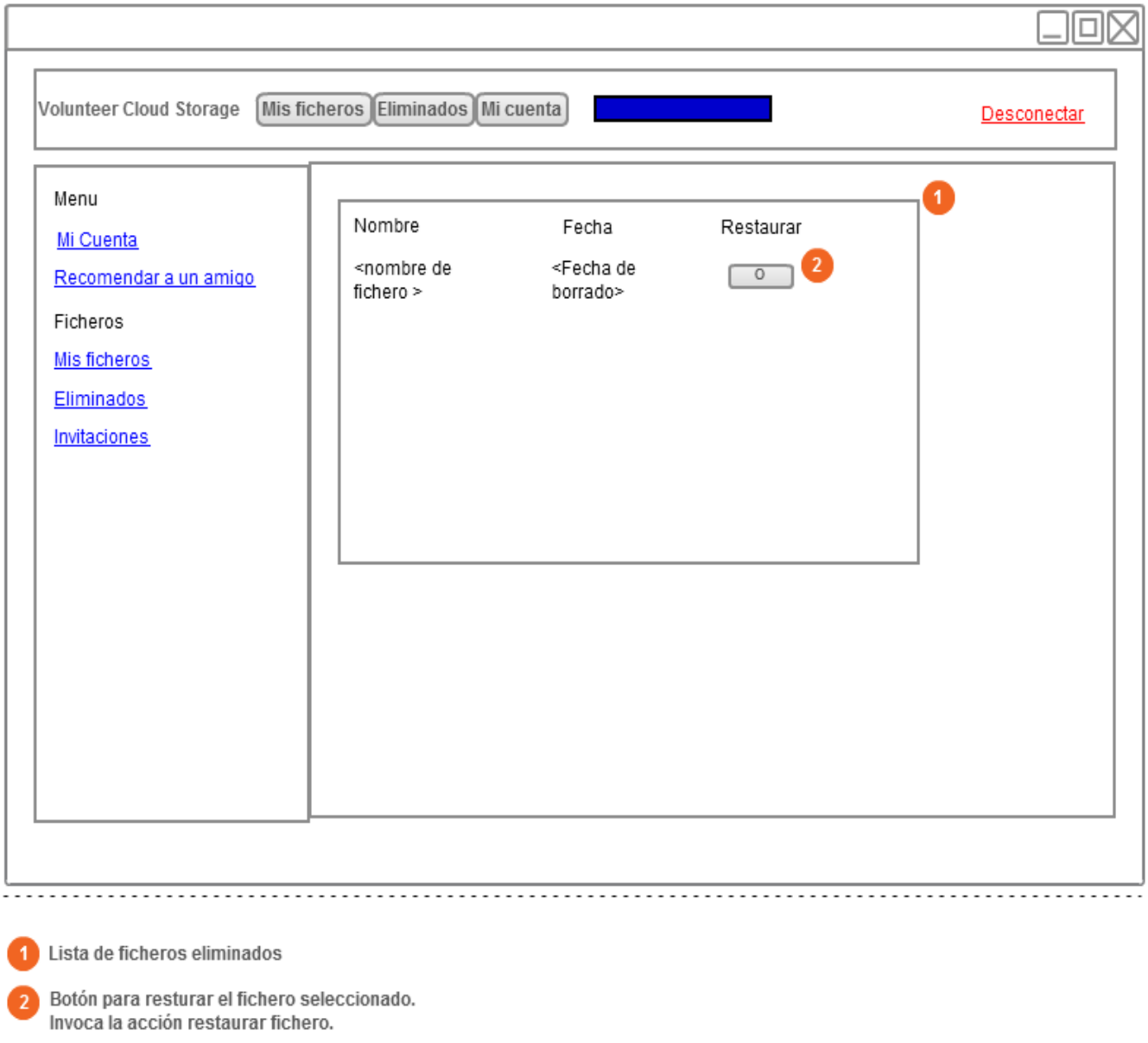


Figura 5.37 Interfaces: mis ficheros, modal subir ficheros.

5.2.2.7 Eliminados

Es la vista que proporciona la usuario una lista de sus ficheros eliminados y la posibilidad de restaurarlos.



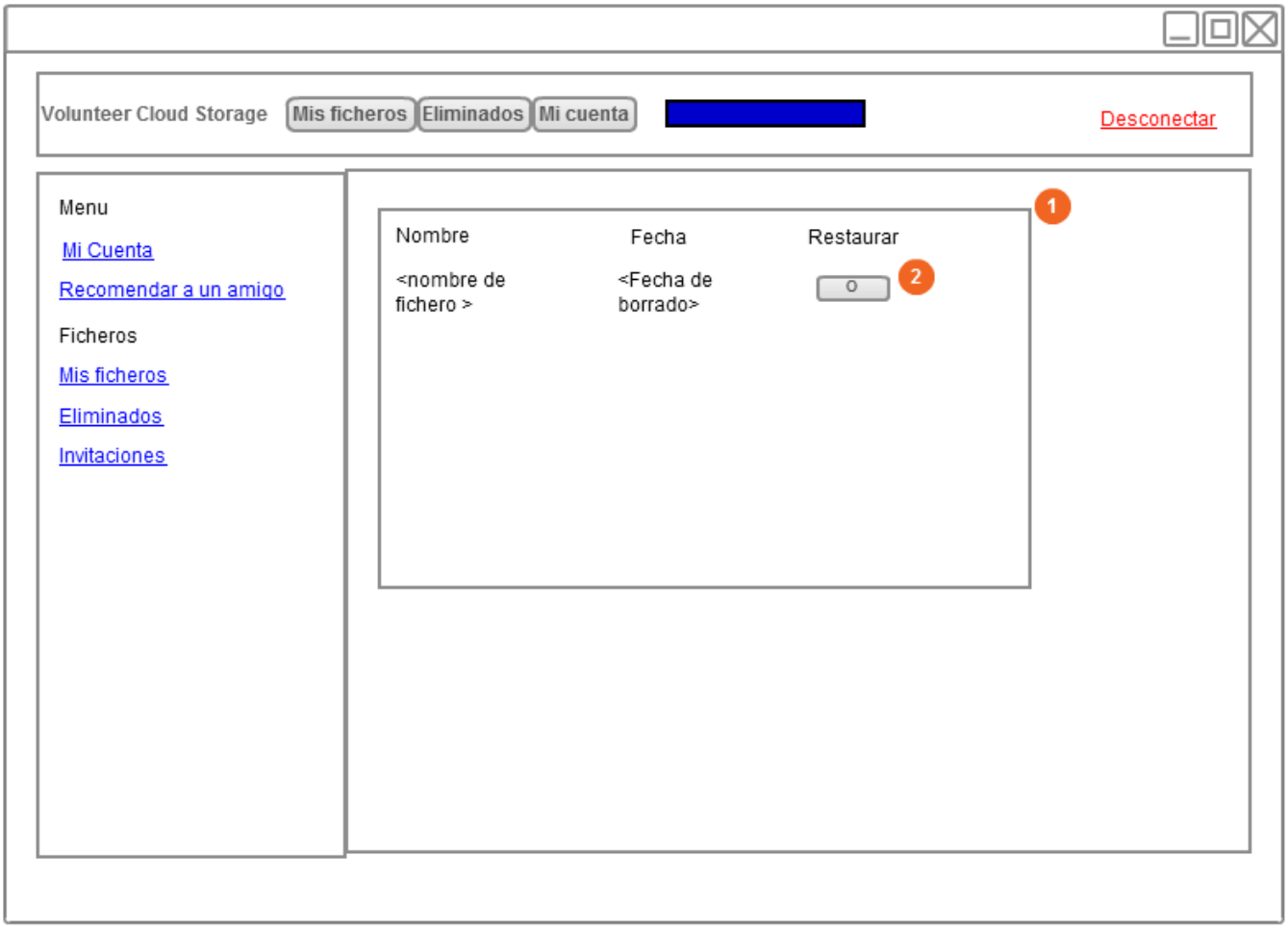
1 Lista de ficheros eliminados

2 Botón para resturar el fichero seleccionado.
Invoca la acción restaurar fichero.

Figura 5.38 Interfaces. Eliminados

5.2.2.8 Versiones anteriores

Es la vista que proporciona al usuario una lista de las versiones anteriores de un fichero elegido en la vista “Mis ficheros” y la posibilidad de restaurarlas.

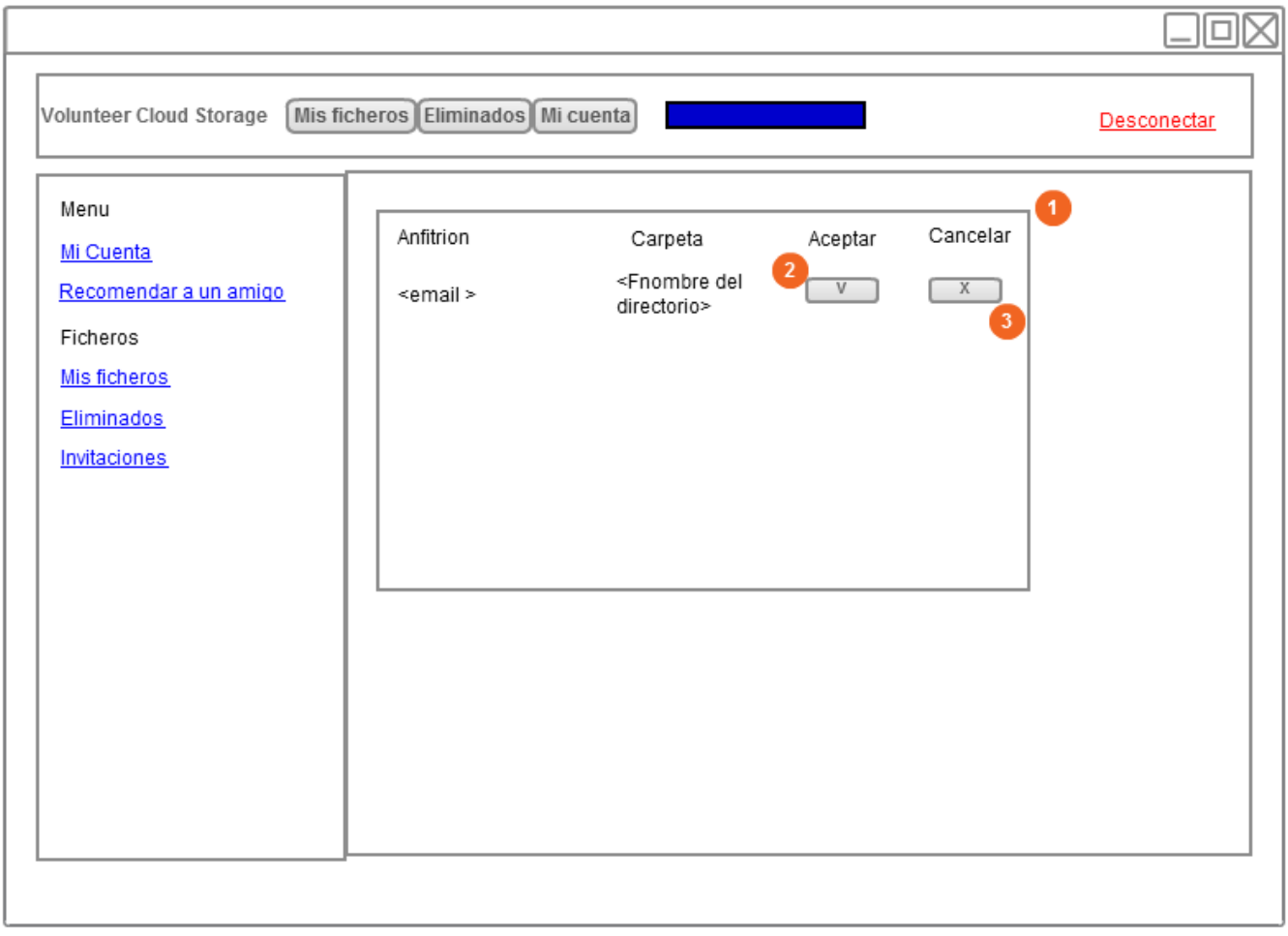


1. Lista de las versiones anteriores de un fichero seleccionado en la vista "mis ficheros".
2. Botón para resturar el fichero seleccionado. Invoca la acción restaurar fichero.

Figura 5.39 Interfaces. Versiones anteriores

5.2.2.9 Invitaciones

Esta interfaz proporciona al usuario la posibilidad de ver las invitaciones que le han realizado y aceptarlas o rechazarlas.



1. Lista de invitaciones donde el usuario que accedió a la vista es el invitado.
2. Botón para aceptar invitación. Invoca la acción aceptar invitación.
3. Botón para cancelar invitación. Invoca la acción cancelar invitación.

Figura 5.40 Interfaces invitaciones

5.2.2.10 *Mi cuenta*

Interfaz que proporciona al usuario la visión de sus datos personales, de cuenta y la posibilidad de cambio de la contraseña de la cuenta.

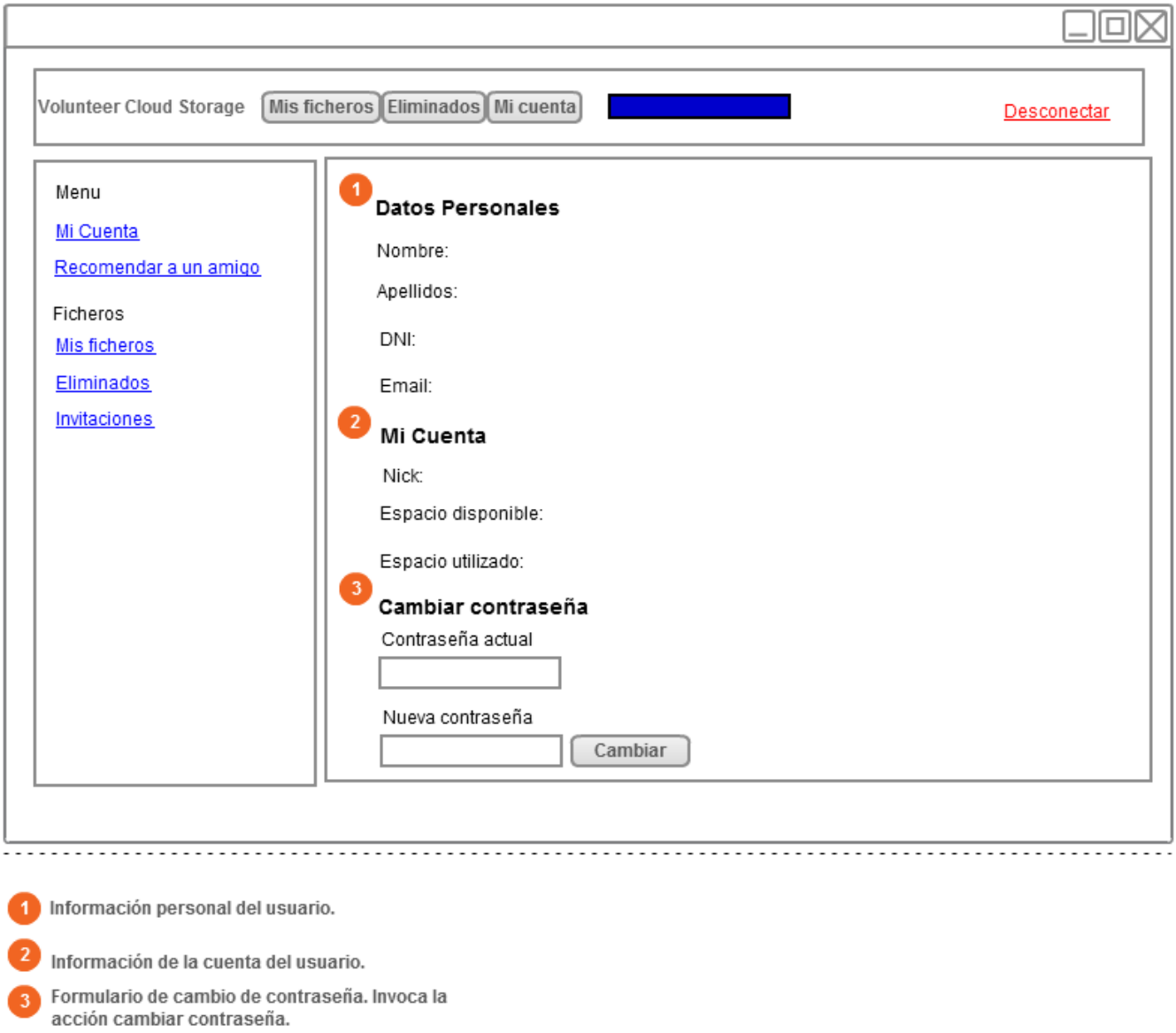


Figura 5.41 Interfaces mi cuenta

5.2.2.11 Recomendar

Esta interfaz proporciona un formulario para que el usuario pueda recomendar la aplicación a sus amigos a través de un email.

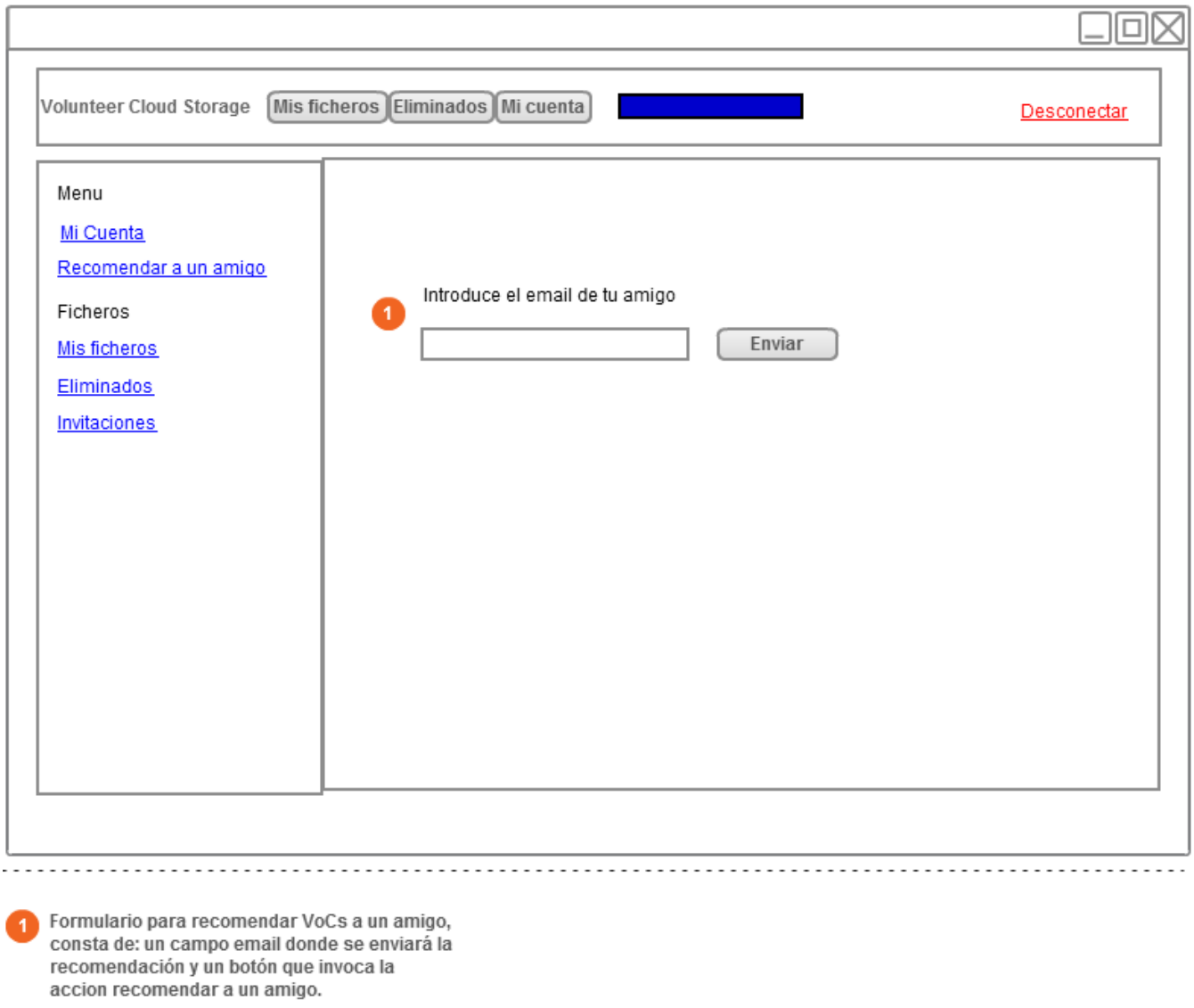


Figura 5.42 Interfaces. Recomendar

5.2.3 Modelo relacional de la base de datos

La Figura 5.43 muestra el diagrama relacional de la base de datos.

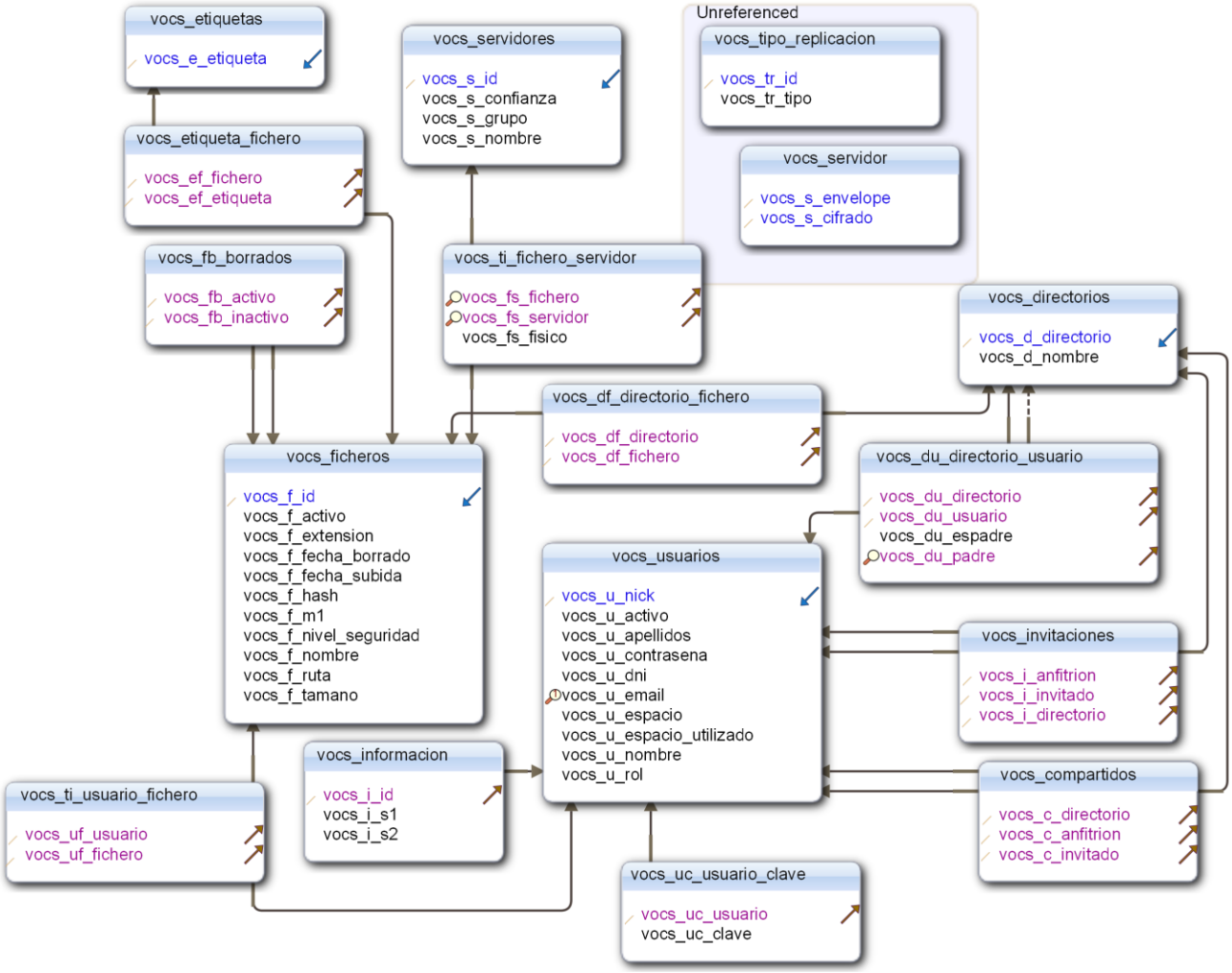


Figura 5.43: Modelo relacional de la base de datos

Los modelos relacionales son modelos convencionales o lógicos que se encuentran instrumentados en la SGBD y orientados a describir los datos a nivel lógico para el SGBD. El modelo relacional está detallado en el estado de la cuestión.

A continuación, se define cada tabla 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 5.142: Tabla vocs_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 5.143: Tabla vocs_df_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 5.144: Tabla vocs_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 5.145: Tabla vocs_du_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 5.146: Tabla vocs_etiqueta_fichero

vocs_etiquetas		
vocs_e_etiqueta	Identificador de la entidad	Cadena de caracteres longitud máxima 255.
Claves		
Clave primaria	(vocs_e_etiqueta)	

Tabla 5.147: Tabla vocs_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 5.148: Tabla vocs_fb_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 5.149: Tabla vocs_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 5.150: Tabla vocs_informacion

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 5.151: Tabla vocs_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 5.152: Tabla vocs_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 5.153: Tabla vocs_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 5.154: Tabla vocs_ti_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 5.155: Tabla vocs_ti_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 5.156: Tabla vocs_tipo_replicacion

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 5.157: Tabla vocs_uc_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 5.158: Tabla vocs_usuarios

5.2.4 Sistema de ficheros

Pueden existir numerosas formas de desarrollar un sistema de almacenamiento, por eso es necesario diseñar detalladamente cómo es el sistema de ficheros que el usuario podrá ver y utilizar dentro de la aplicación.

El sistema de ficheros que VoCS proporciona a sus usuarios, es un sistema virtual donde el control se realiza a través de la base de datos. A continuación se definirá el almacenamiento físico y el almacenamiento en el sistema de ficheros virtual.

Almacenamiento físico

Los servidores de almacenamiento del sistema tienen la función de almacenar los ficheros subidos por lo usuarios. Para ello, el sistema ha creado conjunto de directorios destinados al almacenamiento físico de datos.

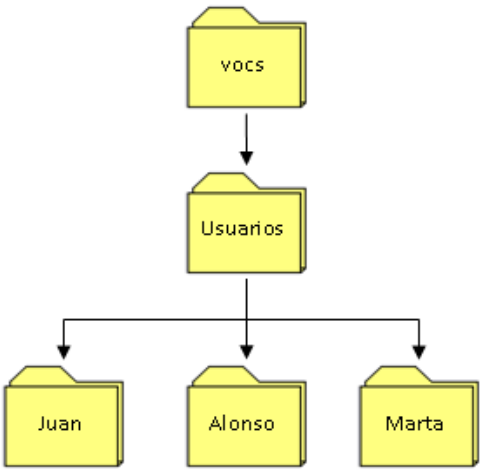


Figura 5.44 Sistema de almacenamiento físico

Las carpetas dedicadas a los usuarios almacenarán todos los ficheros del usuario, independientemente del sistema de ficheros virtual que ofrece VoCS.

Almacenamiento virtual

Para que el usuario pueda manejar sus ficheros como en un sistema de ficheros convencional basado en directorios, el sistema tiene una capa intermedia entre los usuarios y el hardware de almacenamiento.

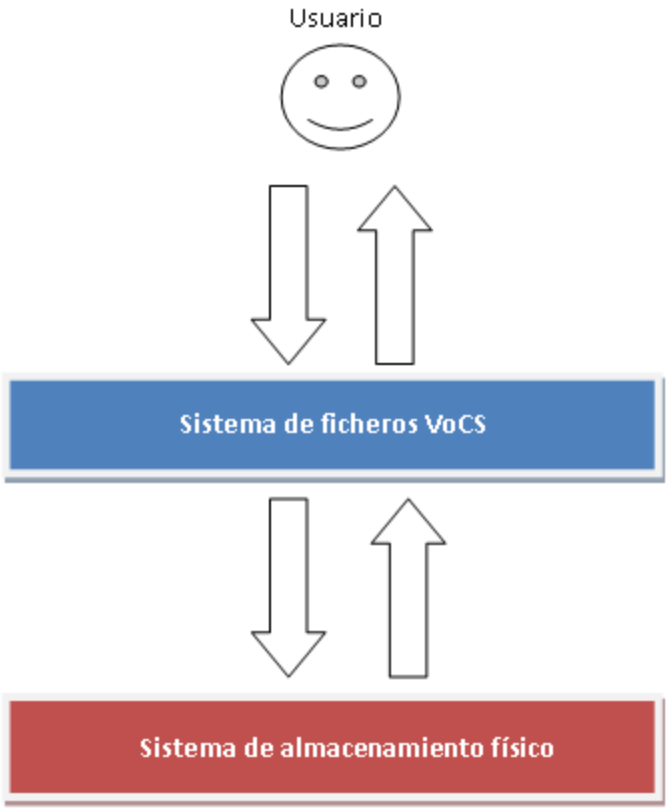


Figura 5.45 Sistema de almacenamiento VoCS

La capa intermedia está desarrollada íntegramente en base de datos. En la base de datos se encuentran las estructuras necesarias para generar directorios, que pueden contener ficheros y otros directorios. A través de consultas SQL se consigue la interacción con el sistema de ficheros.

El siguiente diagrama relacional representa las tablas y las relaciones que forman el sistema de ficheros de la aplicación.

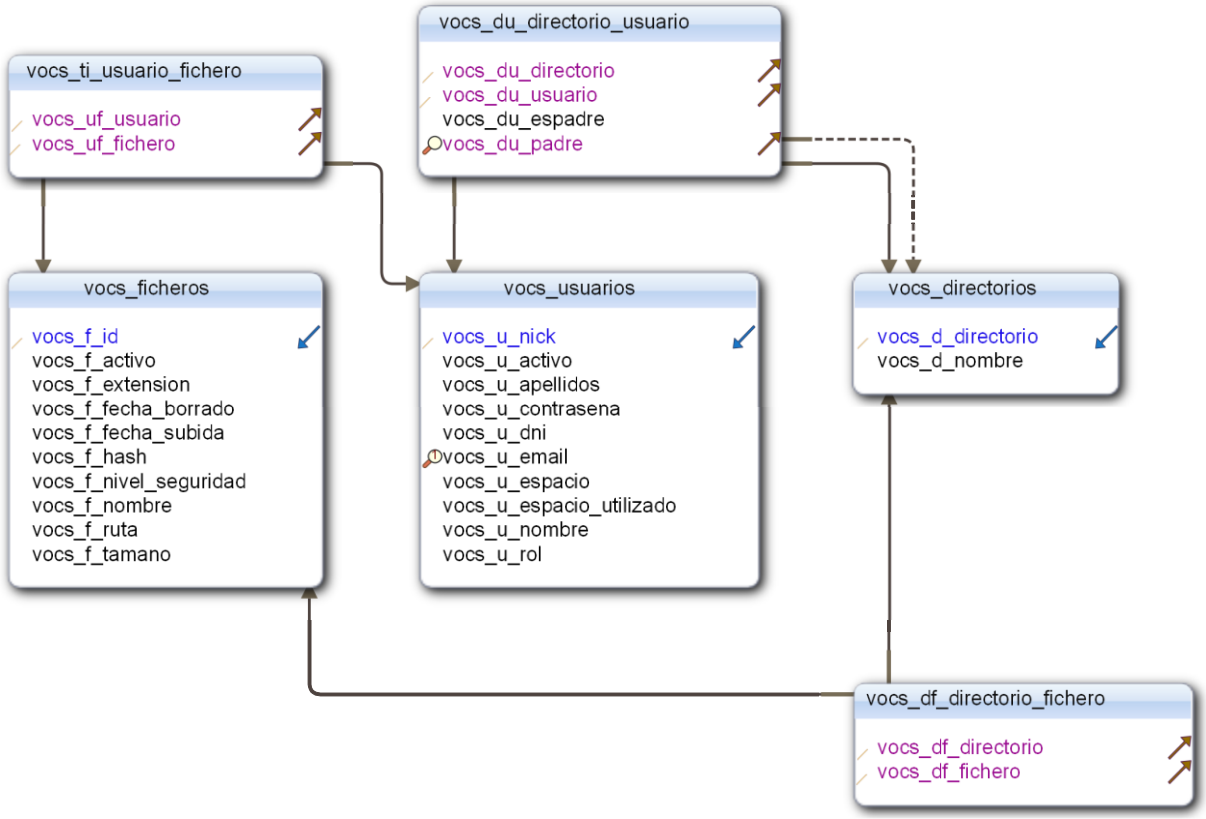


Figura 5.46: Diagrama relacional sistema de ficheros

Como se puede observar en la Figura 5.46, los usuarios tienen relaciones con los directorios (vocs_du_directorio_usuario) que le pertenecen, también en dicha relación se indica el padre del directorio. De esta manera, los directorios pueden ser compartidos al poder asignarles un padre por cada usuario. La Figura 5.47 muestra este concepto.

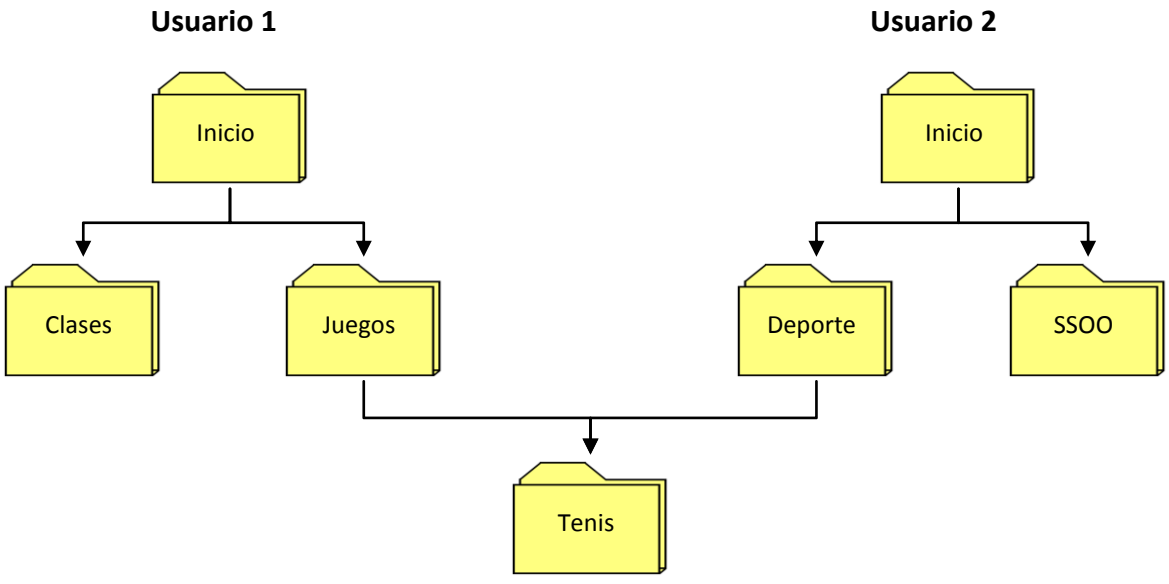


Figura 5.47: Sistema de ficheros. Compartir directorios

La relación usuario-directorio junto con la relación directorio-fichero (vocs_df_directorio_fichero) y con las consultas apropiadas, los usuarios pueden interaccionar con el sistema de ficheros.

Dado un directorio en el que el usuario está navegando (directorio actual), para buscar los directorios que contiene, se realiza lo siguiente:

- Una búsqueda en la relación directorio-usuario de aquellos directorios que tengan como padre el directorio actual y pertenezcan al usuario.

Para la búsqueda de los ficheros que están contenidos en el directorio actual, se realiza lo siguiente:

- Una búsqueda en la relación directorio-fichero de aquellos ficheros que estén relacionados con el directorio actual.

Por otro lado, la relación usuario-fichero, es una relación que indica la propiedad de un fichero por parte de un usuario y está creada para la inclusión, en trabajos futuros, de permisos sobre ficheros.

5.2.5 Autenticación de usuarios y clave de cifrado de datos

El sistema proporciona algoritmos seguros para la autenticación de datos y generación de claves. El diseño de los algoritmos de autenticación y generación de la clave de cifrado de datos se ha basado en los siguientes puntos:

- No almacenar la clave del usuario ni la clave de cifrado.
- Algoritmos suficientemente complejos para evitar todo tipo de ataques.
- Rendimiento. Suficientemente rápidos para los usuarios.
- Fiabilidad.

5.2.5.1 Algoritmo de autenticación

El diseño del algoritmo de autenticación es un proceso seguro para conseguir una cadena de comparación, para que los usuarios puedan ser reconocidos a través de un usuario y una contraseña. La Figura 5.36 muestra los pasos que realiza el algoritmo.

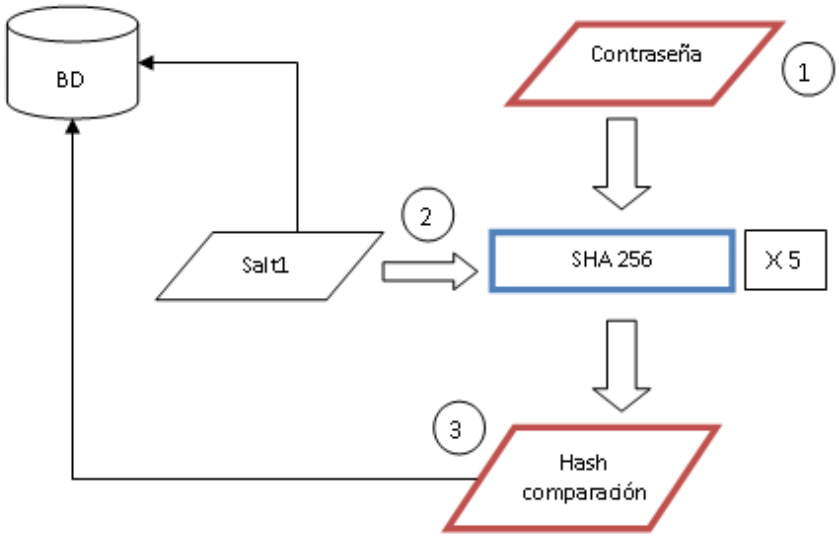


Figura 5.48 Algoritmo de autenticación

A continuación se explica el procedimiento teniendo en cuenta los puntos de la Figura 5.48.

1- Entrada del proceso: contraseña de un usuario.
2- Se genera un salt aleatorio para cada usuario y se almacena en la base de datos. Utilizando SHA256 se crea un hash con el salt y la contraseña del usuario.
3- El hash resultado, es llamado hash de comparación. Es utilizado para autenticar a los usuarios. Se almacena en la base de datos del sistema.

5.2.5.2 Algoritmo de generación de la clave de cifrado

El diseño del algoritmo de generación de la clave de cifrado es un proceso seguro que consigue generar una clave de cifrado de datos única para cada usuario, cada fichero y cada servidor.

La clave de cifrado consta de 2 partes:

- Primera mitad. La primera mitad de la clave de cifrado es una cadena aleatoria cifrada en AES con el hash de comparación. Se genera una clave aleatoria por cada fichero subido. Por lo tanto, la primera mitad es diferente para cada fichero y usuario.
- Segunda mitad. Es una cadena aleatoria cifrada con el certificado OpenSSL del servidor. Se genera una clave aleatoria por cada servidor. Por lo tanto la segunda mitad de la clave es diferente para cada servidor.

A continuación se explica el procedimiento de generación de las mitades de la clave y la obtención de ellas para generar la clave única.

Primera mitad de la clave de cifrado

La primera mitad de la clave tiene el objetivo de ser única para cada fichero y usuario.

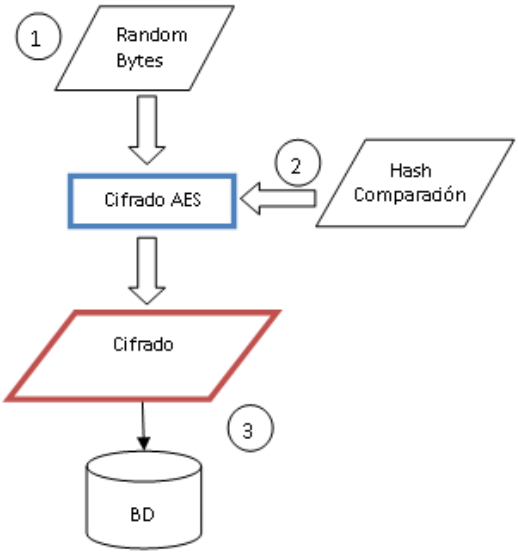


Figura 5.49 Clave de cifrado. Generación de la primera mitad

El proceso de generación de la primera mitad se muestra en la Figura 5.49. La explicación detallada de los puntos de la figura es:

1- Se genera un array de bytes aleatorios en base64. Que servirá como 1ª mitad de la clave de cifrado. Se genera uno por cada fichero subido.
2- Se cifra la primera mitad de la clave de cifrado con el hash aleatorio del usuario.
3- El cifrado de la primera mitad de la clave se almacena en la base de datos.

Segunda mitad de la clave de cifrado

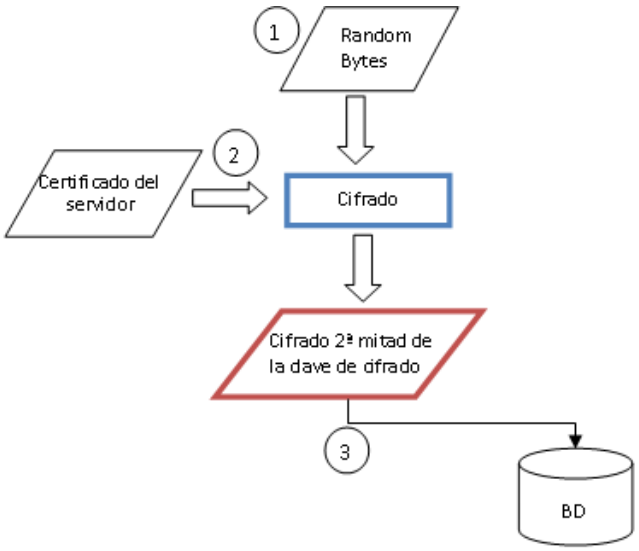


Figura 5.50: Clave de cifrado. Generación de la segunda mitad

El proceso de generación de la segunda mitad se muestra en la Figura 5.50. La explicación detallada de los puntos de la figura es:

1- Se genera un array de bytes aleatorios en base64. Que servirá como 2ª mitad de la clave de cifrado. Se genera uno por cada servidor.
2- Se cifra la segunda mitad de la clave de cifrado con el certificado del servidor.
3- El cifrado de la segunda mitad de la clave se almacena en la base de datos.

Clave de cifrado

En este aparatado se muestra el proceso para la obtención de la clave de cifrado.

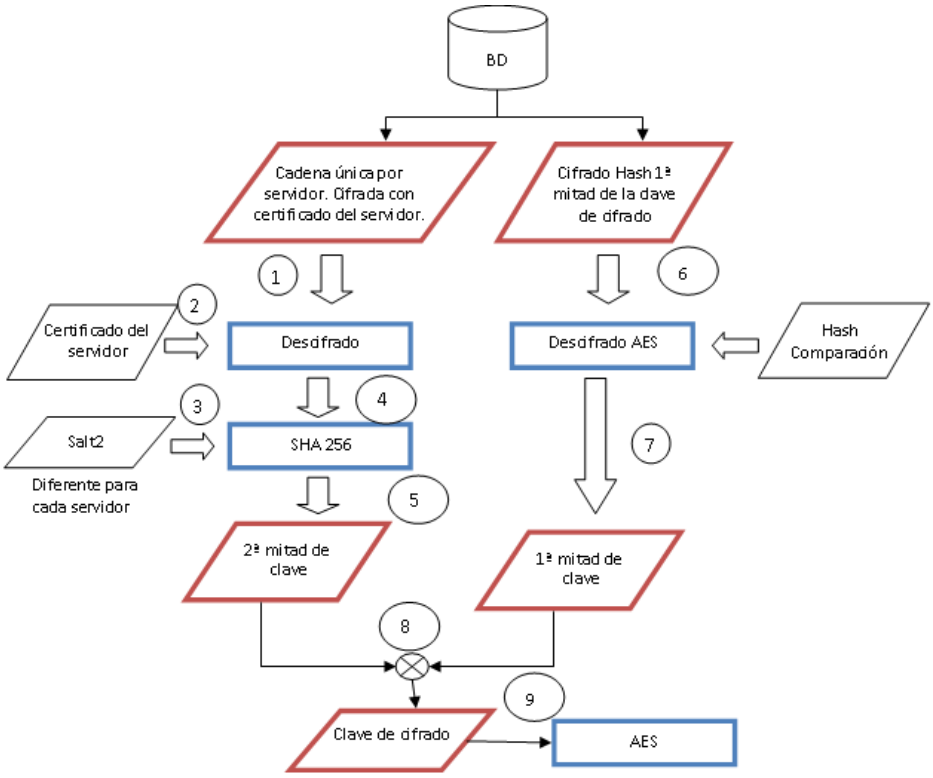


Figura 5.51: Clave de cifrado. Obtención de la clave

El proceso de obtención de la clave de cifrado se muestra en la Figura 5.51. La explicación detallada de los puntos de la figura es:

1- Se obtiene de la base de datos la segunda mitad de la clave cifrada.
2- Se descifra la segunda mitad de la clave con el certificado del servidor.
3- Se genera un hash SHA256 con un salt aleatorio diferente para cada servidor. El salt se almacena en la base de datos.
4- El hash resultado del paso anterior es la segunda mitad de la clave de cifrado.
5- Se obtiene de la base de datos el cifrado de la primera mitad de la clave.
6- Se descifra la primera mitad de la clave con el hash de comparación del usuario.
7- El resultado del descifrado es la primera mitad de la clave.
8- Se realiza la operación XOR con la segunda mitad de la clave y la primera mitad de la clave.
9- El resultado de la operación XOR es la clave de cifrado de datos.

En conclusión, la clave de cifrado se compone de una mitad única para cada servidor y una mitad única para cada fichero y usuario. Por lo que la clave de cifrado también es única para cada servidor, fichero y usuario.

5.2.6 Modelos de replicación

El concepto replicación está detallado en el estado de la cuestión. El sistema utiliza tres modelos de replicación, dos de ellos utilizados para la replicación de ficheros y el otro para la replicación de la información de la base de datos.

5.2.6.1 Modelo de replicación de la base de datos

La replicación de datos consiste en el transporte de datos entre dos o más servidores, permitiendo que ciertos datos de la base de datos estén almacenados en más de un sitio, y así aumentar la disponibilidad de los datos y mejorar el rendimiento de las consultas globales.

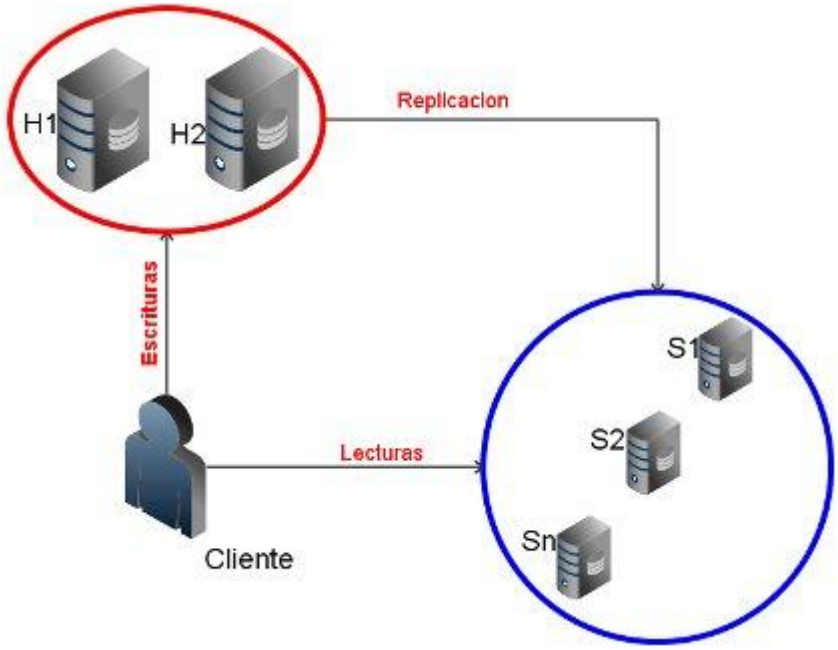


Figura 5.52: Replicación de base de datos

Como se ve en la Figura 5.52:

1- Los servidores que están dentro del círculo rojo, representan los servidores Maestros. Como un esclavo solo puede tener un maestro significa que:
   a. H1 es maestro de H2(H2 es esclavo de H1)
   b. H2 es maestro de H1(H1 es esclavo de H2)

   El diseño permite la inclusión de más maestros.
2- Los servidores que están dentro del círculo azul, representan los servidores esclavos, que se encargarán de hacer las lecturas. Como estos esclavos solo pueden tener un maestro, la mitad de los servidores son esclavos de un maestro y la otra mitad del otro servidor maestro. Si se incluyen más maestros, los esclavos se dividirían entre el número de maestros.

3- Los procesos de escritura en la base de datos, solo se realizarán en aquellos servidores que sean maestros. Esto es así ya que si se produce una escritura en un esclavo, la información escrita no sería replicada al resto de servidores.

4- Los procesos de lectura se pueden realizar en cualquier servidor, pero solo se harán en los esclavos. Esto permite un balanceo de carga en los accesos a la base de datos.

Por lo tanto, el modelo ofrece consistencia en los datos, balanceo de carga entre servidores, disponibilidad, fiabilidad, rendimiento y escalabilidad.

- Disponibilidad. La replicación de la información permite que ésta esté disponible desde cualquiera de los servidores.
- Fiabilidad. Al haber múltiples copias de los datos disponibles en el sistema permite la recuperación en caso de fallos de nodos.
- Rendimiento. Se mejora para las transacciones de consultas.
- Balanceo de carga. Permite distribuir los accesos a la base de datos a diferentes servidores.

5.2.6.2 Replicación de ficheros

Se diseñan dos modelos de replicación de ficheros: modelo VoCS y modelo circular.

Modelo VoCS

El modelo de replicación planteado en este apartado es un hibrido entre el modelo bajo demanda y un modelo basado en un sistema de puntuación de servidores. Es un modelo con replicación síncrona.

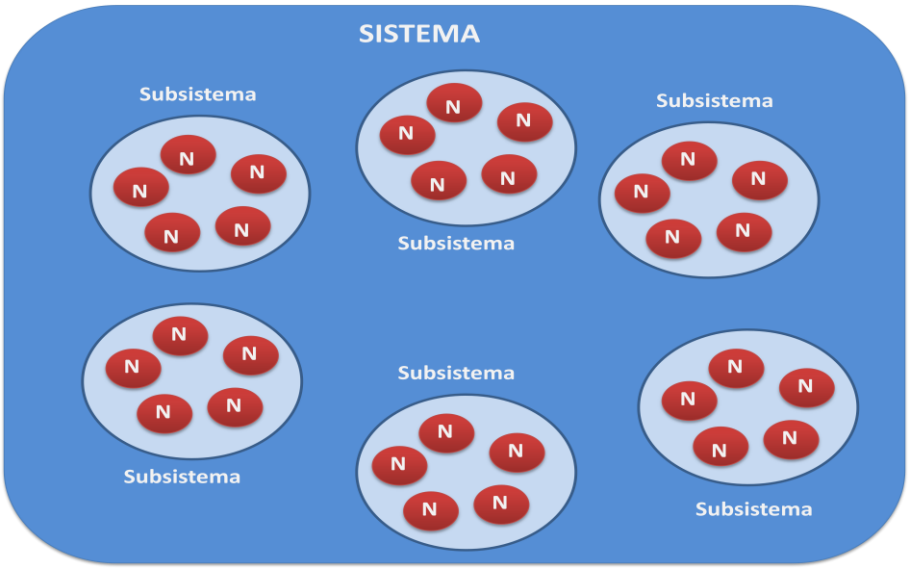


Figura 5.53: Modelos de replicación. Modelo VoCS

Como vemos en la Figura 5.53, el sistema se compone de un conjunto finito de nodos (N) contenidos en subsistemas que permiten un máximo de cinco nodos.

Existen dos casos en los que el modelo de replicación actúa:

- Almacenamiento de ficheros. En el almacenamiento de un fichero el sistema deberá incluirlo en su entramado de servidores garantizando la seguridad y disponibilidad de éste. El modelo ejercerá las siguiente acciones:
    1- El servidor almacenará de forma segura el fichero en su disco de almacenamiento dedicado a VoCS.
    2- El servidor se dispondrá a replicar el fichero garantizando un número N de pérdidas donde N < 5. Si queremos garantizar X perdidas, se deben realizar X+1 replicas.
    3- Para elegir los servidores a los que replicar, éste solo tendrá en cuenta los servidores que estén en su subsistema. Y basado en modelo de puntuación elegirá los más óptimos para recibir la información en ese momento.
    4- Una vez elegidos los servidores para la réplica, los ficheros son replicados.
    5- Espera a que la replicación en los demás servidores finalice y devuelvan una respuesta.
- Petición de ficheros. En la petición para la descarga de un fichero existen dos situaciones, que se explican a continuación:
    1- El fichero se encuentra en el servidor que se conectó el usuario.
        - El servidor envía el fichero al usuario y este lo descarga.
    2- El fichero no se encuentra en el servidor que se conectó el usuario.
        - Actúa bajo el modelo bajo demanda. El modelo bajo demanda solicita al sistema la replicación del fichero cuando se demanda el fichero en ese servidor y no se encuentra en él. En consecuencia, el fichero queda replicado en un servidor más, garantizando la caída de otro nodo.
        - El servidor, una vez obtenido el fichero lo envía al usuario y éste lo descarga.

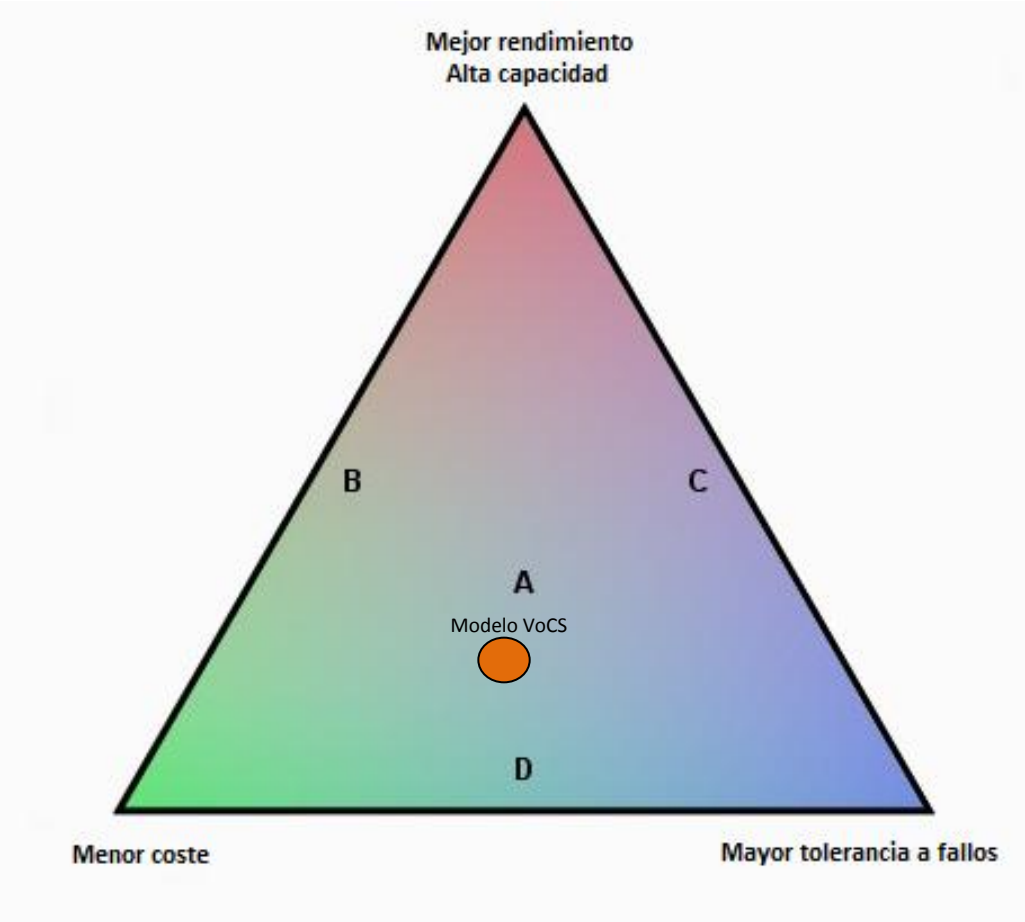


Figura 5.54: Rendimiento, tolerancia a fallos y coste del modelo VoCS

En la Figura 5.54, se puede observar que el modelo (representado con un círculo naranja) está diseñado para aumentar la tolerancia a fallos del sistema a un bajo coste a consecuencia de perder rendimiento. Se disminuye el rendimiento ya que se produce un análisis de todos los servidores del subsistema antes de replicar los datos y realiza una replicación síncrona. Se consigue una buena tolerancia a fallos gracias a que se puede replicar en hasta otros 4 servidores del subgrupo y las replicas bajo demanda.

A continuación se definen el número de replicas a realizar respecto al nivel de seguridad que los usuarios apliquen a sus ficheros. Es necesario mencionar que para el presente TFG, se cuenta con solo dos servidores.

- Seguridad alta, una réplica.
- Seguridad media, una réplica.
- Seguridad baja, ninguna réplica.

Métrica de servidores

En este apartado se explican los parámetros que se tendrán en cuenta para realizar la métrica de servidores.

- Latencia en la red.

  Es el tiempo o lapso necesario para que un paquete de información se transfiera de un host a otro. La latencia, junto con el ancho de banda, son determinantes para la velocidad de una red.

  La latencia es información valiosa para determinar lo accesible que es un servidor en ese momento. Una latencia alta puede significar un servicio limitado, por lo que no es aconsejable cargarlo más.
- Capacidad disponible.

  Los recursos de un servidor son limitados, por lo que el número de conexiones que podrá soportar lo son también. Cualquier petición que se envíe a un servidor saturado será rechazada. Por eso, es necesario tener en cuenta si es posible acceder al servicio.

  Capacidad disponible de un nodo es igual a Cd = c - w, donde c es su capacidad representada por el número de solicitudes de servicio que puede manejar durante una unidad de tiempo, y w es la carga de trabajo representada por el número de sus peticiones recibidas durante una unidad de tiempo.
- Ancho de banda.

  Un servidor tiene un ancho de banda limitado. Es decir, la cantidad de bytes que puede comunicar será limitado. Es necesario saber cuánto ancho de banda está disponible para no saturar la red.
- Capacidad de almacenamiento.

  Los servidores tienen capacidad de almacenamiento limitada. Por lo que es necesario tener en cuenta no sobrepasar ese límite.

Se han elegido estás métricas porque su conjunto abarca los requerimientos necesarios para un servicio de calidad:

- Disponibilidad. La aplicación debe estar lista para su uso en un momento determinado.
- Accesibilidad: Es el grado de capacidad para aceptar una solicitud de servicio. Se puede medir como la probabilidad de éxito de la creación de instancias de servicios de éxito en un punto en el tiempo. Un servidor puede estar disponible pero no accesible.

- Integridad: El sistema debe mantener los datos intactos en la interacción con otros nodos o clientes.
- Rendimiento: Se mide en términos de desempeño y latencia. Un mayor rendimiento y los valores de latencia más bajos representan un buen desempeño. El rendimiento se puede representar como el número de solicitudes asistidas en un periodo de tiempo determinado. La latencia es el tiempo que tomó prestar el servicio, desde el envío de una solicitud hasta la llegada de la respuesta.
  El rendimiento será calculado aparte en pruebas.
- Fiabilidad: característica por la que se mide el tiempo de funcionamiento sin fallos.

La siguiente fórmula será utilizada para valorar los servidores.

- Latencia en red: Lr
- Capacidad disponible: Cd
- Ancho de banda: Ab
- Capacidad de almacenamiento: Ca

Reputación = A*Ca + B*Ab + C*Cd + D*Lr

Modelo en anillo

El modelo de replicación planteado en este apartado es un modelo de replicación en anillo y síncrona.

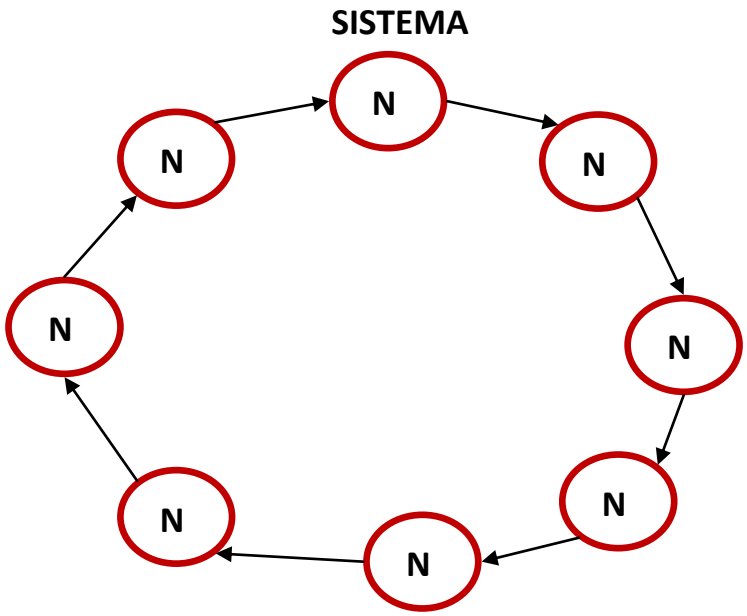


Figura 5.55: Modelos de replicación. Modelo circular

Como se puede observar en la Figura 5.55, el sistema se compone de un conjunto finito de nodos dispuestos de forma circular, de tal manera que un nodo solo replica al nodo que apunta.

Existen dos casos en los que el modelo de replicación actúa:

- Almacenamiento de ficheros. En el almacenamiento de un fichero el sistema deberá incluirlo en su entramado de servidores garantizando la seguridad y disponibilidad de éste. El modelo ejercerá las siguiente acciones:
    1- El servidor almacenará de forma segura el fichero en su disco de almacenamiento dedicado a VoCS.
    2- El servidor se dispondrá a replicar el fichero.
    3- El servidor replicará el fichero al servidor al que apunta. Es decir, el siguiente servidor de la disposición circular.
    4- Espera que la replicación en el otro servidor finalice y devuelva una respuesta.
- Petición de ficheros. En la petición para la descarga de un fichero existen dos situaciones, que se explican a continuación:
    1- El fichero se encuentra en el servidor que se conectó el usuario.
        - El servidor envía el fichero al usuario y este lo descarga.
    2- El fichero no se encuentra en el servidor que se conectó el usuario.
        - Realiza una petición del fichero a otro servidor que si lo contenga. El servidor no almacena el fichero.
        - Sirve el fichero al usuario que solicitó la descarga.

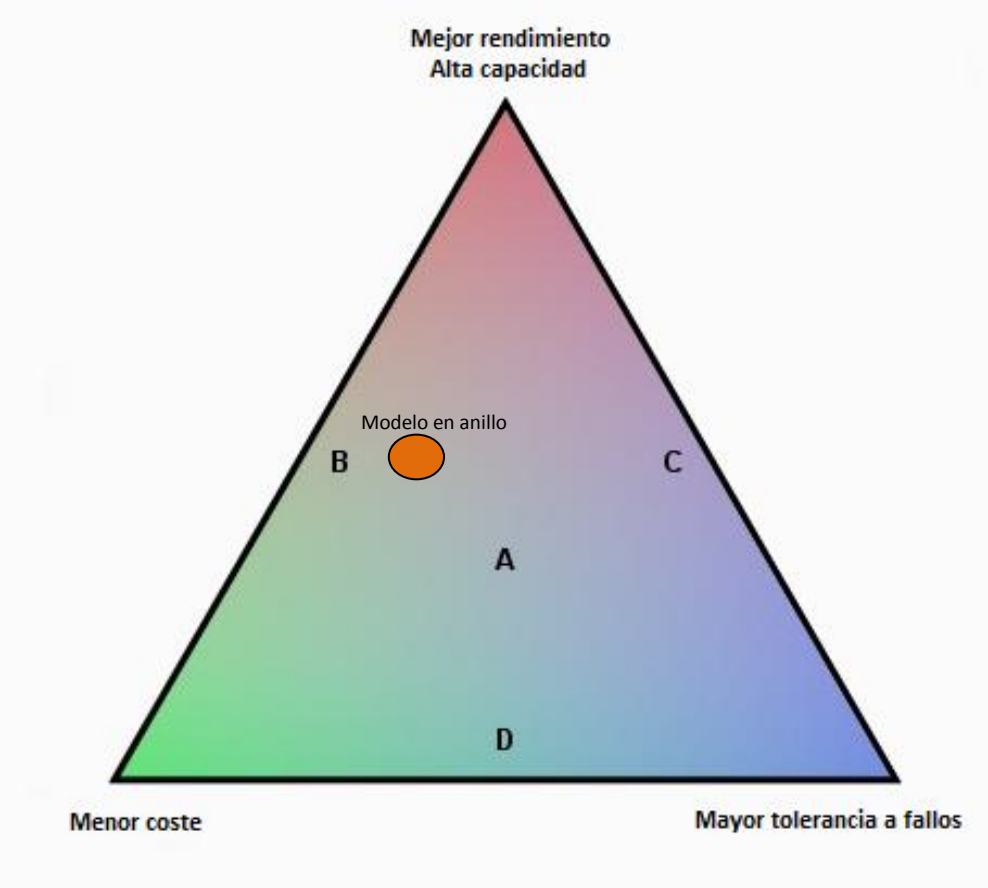


Figura 5.56: Rendimiento, tolerancia a fallos y coste del modelo circular

En la Figura 5.56, se puede observar que el modelo (representado con un círculo naranja) está diseñado para aumentar el rendimiento a un bajo coste a consecuencia de perder tolerancia a fallos. Se disminuye la tolerancia a fallos ya que se realiza solo una réplica. Se aumenta el rendimiento porque solo se realiza una réplica y a diferencia del modelo VoCS no realiza una medición de servidores, reduciendo de esta manera el tiempo de ejecución.

A continuación se definen el número de replicas a realizar respecto al nivel de seguridad que los usuarios apliquen a sus ficheros. Es necesario mencionar que para el presente TFG, se cuenta con solo dos servidores.

- Seguridad alta, una réplica.
- Seguridad media, una réplica.
- Seguridad baja, ninguna réplica.

Comunicación entre servidores

La comunicación entre servidores para realizar las operaciones necesarias en la replicación de ficheros se utilizará REST Webservice.. La clase que contendrá las funciones necesarias para el control de mensajes entre servidores, es RestController y se especifica en el diagrama de clases.

Se utilizará REST Webservice para comunicarse entre servidores. A continuación se muestra el intercambio de mensajes en la replicación de ficheros.

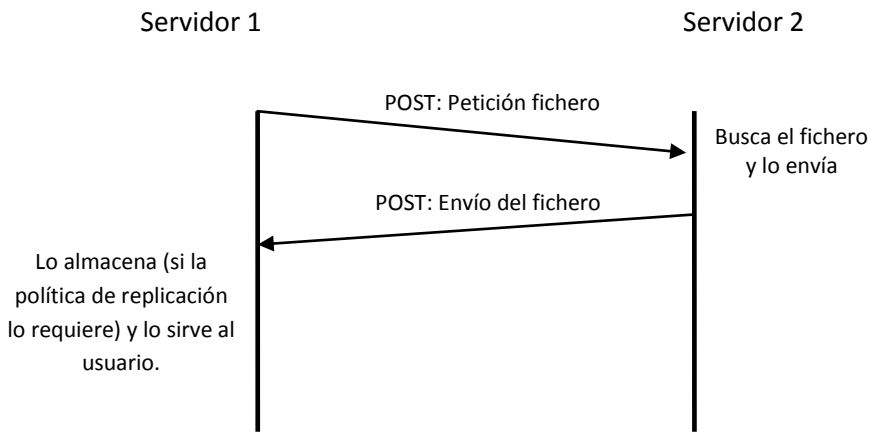


Figura 5.57: Mensajes replicación: Bajada de ficheros (bajo demanda)

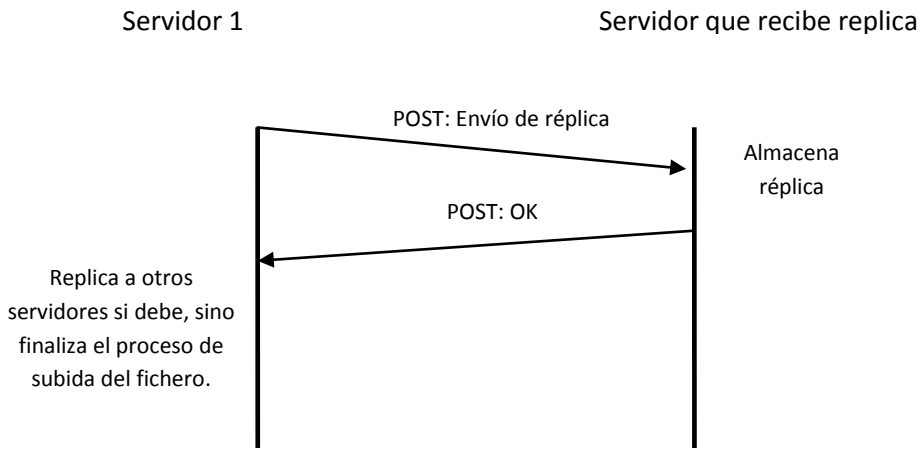


Figura 5.58: Mensajes replicación en subida de ficheros

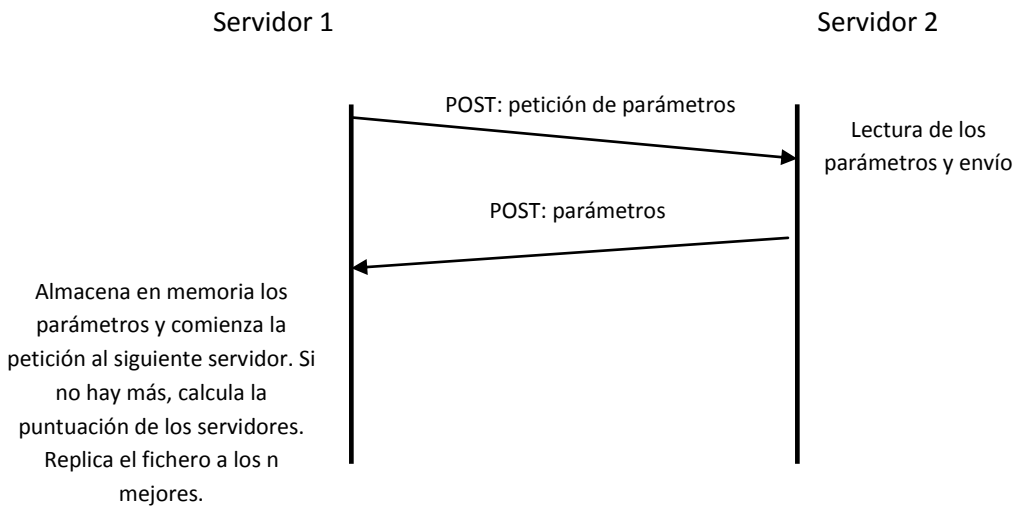


Figura 5.59: Mensajes replicación: petición de métricas

La clase RestController debe implementar las siguientes funcionalidades:

- Obtener parámetros del servidor, funcionalidad que es invocada para obtener los parámetros de métrica de un servidor. Cuando es invocada obtiene los parámetros del fichero definido en el atributo "RUTA_PARAMS" de la clase Metricas. Una vez obtenidos, los envía al servidor que hizo la petición.
- Recibir replica y almacenarla. Funcionalidad que cuando es llamada recibe un fichero y lo almacena.
- Recibir replica. Funcionalidad que recibe un replica de un fichero y no lo almacena.
- Petición fichero. Funcionalidad que recibe la petición de un fichero, lo busca y lo envía.

5.2.7 Diagrama de clases

Este apartado muestra el diagrama de clases generado gracias al proceso de análisis y diseño.

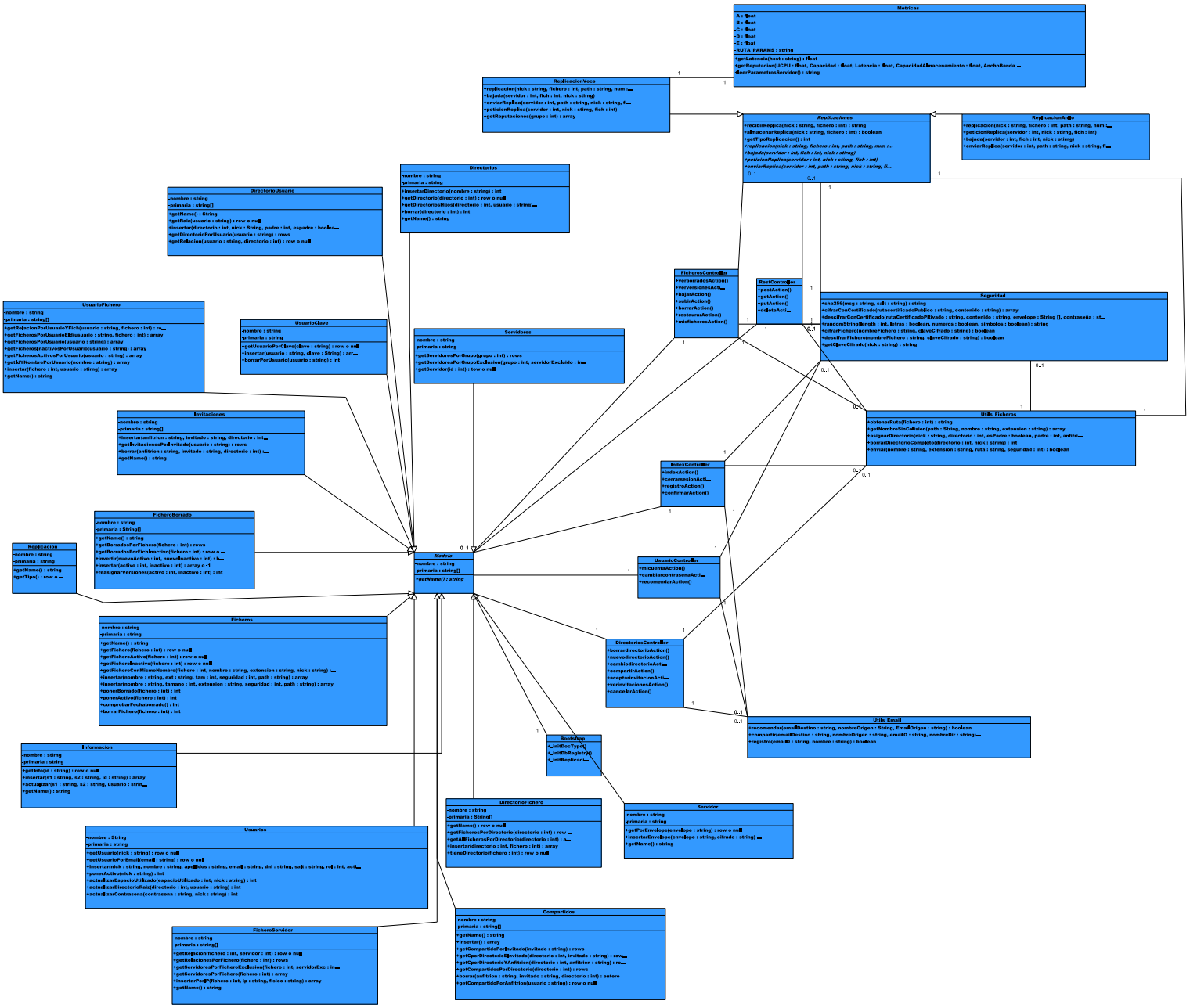


Figura 5.60 Diagrama de clases

5.2.8 Tareas programadas

En este apartado se describirán las dos tareas programadas del sistema:

- Tarea de eliminación de aquellos ficheros cuya fecha de borrado sea anterior a 15 días respecto a la fecha actual.
- Tarea de recopilación de métricas del servidor.

Tarea de eliminación

Los ficheros que eliminan los usuarios se marcan como borrados, pero estos pueden restaurarlos hasta 15 días después de la fecha de borrado. Aquellos ficheros que sobrepasen los 15 días como borrados, serán eliminados completamente del sistema. Para ello, se genera una tarea programada que revisa todas las fechas de borrado de los ficheros y elimina física y virtualmente aquellos ficheros que sobrepasen los 15 días como borrados.

La tarea programada ejecutará un código en el lenguaje del lado del servidor elegido para el desarrollo.

Recopilación de métricas

El modelo de replicación VoCS utiliza un sistema de métricas de servidores. Los servidores deben obtener sus métricas y tenerlas preparadas para cuando sean reclamadas. Para ello, se crea una tarea programada que obtiene, mediante comandos de consola, los parámetros que requiere el modelo de replicación y genera un fichero XML con el siguiente formato:

```
<?xml versión=”1.0” encoding = “UTF-8”?>
        <mensaje>
                <cpu> <datos></datos></cpu>
                <cd> <datos></datos></cd>
                <ca> <datos></datos></ca>
                <ab> <datos></datos></ab>
        </mensaje>
```

El fichero XML será enviado por webservice a los servidores que pidan las métricas.

5.2.9 Opciones de implementación

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

5.2.9.1 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.

Se desea desarrollar un prototipo de código abierto por lo que la programación debe ser ágil, rápida y de coste nulo. Entre los aspectos particulares del proyecto, se tienen en cuenta aquellos lenguajes que ofrezcan frameworks con una implementación del patrón MVC, ya que la arquitectura del sistema es MVC. Por último, aquellos lenguajes en los que el equipo de desarrollo tenga experiencia se valorarán más.

Con la información recopilada en el estado de la cuestión sobre lenguajes de programación del lado del servidor, concluimos que PHP se ajusta a los requisitos del proyecto por estas razones:

- 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.

A pesar de que JSP y ASP.NET ofrecen similares características que PHP, JSP es más lento por la maquina virtual que necesita y no es independiente del entorno de desarrollo y ASP.NET tiene licencia propietaria y genera costes. Por esto, PHP es el lenguaje elegido.

La arquitectura del sistema es MVC, por lo que el proyecto se desarrollará en un framework que implemente el patrón MVC. Teniendo en cuenta la información detallada en el estado de la cuestión sobre frameworks de PHP, Zend Framework se ajusta a la perfección a las necesidades del TFG ya que nos ofrece lo siguiente:

- Implementa el patrón MVC.
- Es de código abierto.
- Tiene una extensa documentación.
- Permite la utilización de todo tipo de bases de datos con el mismo código de programación.

- Y a diferencia del resto de frameworks, el equipo de desarrollo de la aplicación tiene experiencia en Zend Framework.

A pesar de que CakePHP ofrece similares características, el equipo de desarrollo no cuenta con experiencia en el framework. Por esto, el elegido es Zend Framework.

5.2.9.2 Lenguajes del lado del cliente

El lenguaje del lado del cliente elegido es JavaScript que es actualmente el más utilizado, con mucho soporte y actualización. También ofrece gran variedad de frameworks con características avanzadas que pueden ser utilizadas fácilmente. Uno de los frameworks más utilizados por los desarrolladores es jQuery.

Se utilizará jQuery por las características descritas en el estado de la cuestión.

Otros lenguajes del lado del cliente que se utilizarán son:

- CSS para el diseño de las páginas.
- HTML para la representación de la información y estructura de las páginas.

5.2.9.3 Base de datos

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. Es posible ver una comparativa de sistemas gestores de bases de datos relacionales en el estado de la cuestión. [7][8].

El sistema gestor elegido es MySQL por las siguientes razones:

- Es fácil de usar, instalar y mantener. Está bien documentado, tiene un buen soporte a través del grupo de usuarios, y también ofrece soporte comercial.
- Dispone de varios modelos de almacenamiento, como InnoDB, MyISAM, y texto completo.
- MySQL es uno de los gestores de bases de datos con mejor rendimiento gracias a la velocidad en que realiza las operaciones.
- MySQL proporciona tolerancia a fallos, equilibrio de carga y la seguridad a través de la replicación de datos.
- No utiliza demasiados recursos, por lo que es posible la implementación de bases de datos en computadores con recursos muy limitados.
- Tiene licencia GNU/GPL y no genera costes.

- Cuenta con muchas herramientas que facilitan el análisis, diseño e implementación de bases de datos.
- PHP tiene mucho soporte para MYSQL.

5.2.9.4 Servidor

El servidor elegido para el desarrollo es Apache porque sus características son las ideales para un proyecto como el del presente TFG, su uso no genera costes, es el más utilizado y el equipo de desarrollo cuenta con experiencia en Apache.

5.2.9.5 LAMP

Se conoce con el término LAMP a la plataforma para desarrollo y ejecución de aplicaciones web de alto rendimiento dada por el uso en conjunto de Linux, Apache, MySQL y PHP. El nombre LAMP se debe justamente a las iniciales de estas 4 tecnologías. LAMP está considerada como una de las mejores herramientas para desarrollar y ejecutar aplicaciones web, sus tecnologías por separado son de altísima calidad y en conjunto sus virtudes se multiplican. LAMP no tiene una corporación detrás de su desarrollo, sino una comunidad ingenieros y programadores de primer nivel que trabajan para mejorar las diferentes tecnologías que componen LAMP.

Figura5.61 LAMP

6. Implementación

El apartado está dedicado a los detalles de implementación de la aplicación analizada y diseñada. Se explicará los detalles de implementación de los diferentes módulos diseñados.

Las versiones de las tecnologías que se han utilizado se muestran en la Tabla 6.1.

Tecnología	Versión	Tecnología	Versión
PHP	5.3.6	CSS	2.0 y 3.0
Zend Framework	1.11.10	HTML	4.0
jQuey	1.8.0	Ubuntu	11.10
MySQL	5.1		

Tabla 6.1 Versiones de las tecnologías utilizadas

6.1 Módulo de gestión de directorios

En este apartado se definen los ficheros y clases que implementan las funcionalidades del módulo gestión de directorios, así como la interacción del usuario con las diferentes funcionalidades.

Crear directorio

Los ficheros que implementan la funcionalidad son:

- misficheros.phtml que implementa la interfaz "Mis ficheros".
- DirectoriosController.php que implementa la clase DirectoriosController.

Dada la interfaz "Mis ficheros", el usuario dispondrá de un formulario para indicar el nombre del nuevo directorio y un botón enviar. Mediante el método POST, se envía la información del formulario a la acción "crearAction()" de la clase DirectoriosController. Ésta acción implementa la funcionalidad "Crear directorio" descrita en el diseño.

Eliminar directorio

Los ficheros que implementan la funcionalidad son:

- misficheros.phtml que implementa la interfaz "Mis ficheros".
- DirectoriosController.php que implementa la clase DirectoriosController.

Dada la interfaz "mis ficheros", el usuario dispone de un botón situado al lado de cada directorio que sirve para eliminarlo. Mediante el método GET, se envía el identificador del directorio elegido a la acción "eliminarAction()" de la clase DirectorioController. Ésta acción implementa la funcionalidad "Eliminar directorio" descrita en el diseño.

Navegar

Los ficheros que implementan la funcionalidad son:

- misficheros.phtml que implementa la interfaz "Mis ficheros".
- DirectoriosController.php que implementa la clase DirectoriosController.
- FicherosController.php que implementa la clase FicherosController.

Dada la interfaz "mis ficheros", el usuario dispone de una lista ordenada con los padres del directorio actual y una lista del contenido del directorio actual. Tanto en los padres como en los directorios contenidos en el directorio actual, hay enlaces que permiten al usuario navegar dentro de ellos.

Pulsado un enlace de navegación, mediante el método GET, se envía el identificador del directorio seleccionado a la acción "cambiardirectorioAction()" de la clase DirectoriosController. La acción implementa la funcionalidad "Navegar: cambiar directorio actual" descrita en el diseño.

Para la visualización del contenido del directorio actual, cuando el usuario accede a la vista "mis ficheros" se ejecuta la acción misficherosAction() de la clase FicherosController. La acción implementa la funcionalidad "Navegar: Listar contenido del directorio actual" descrita en el diseño. La vista se encarga de mostrar al usuario el listado del contenido del directorio actual.

La acción "misficherosAction" también provee a la vista del listado de padres del directorio actual, por lo que implementa la funcionalidad "Navegar: Listado de directorios padres", descrita en el diseño. La vista se encarga de mostrar al usuario el listado del contenido del directorio actual.

Invitar

Los ficheros que implementan la funcionalidad son:

- misficheros.phtml que implementa la interfaz "Mis ficheros".
- DirectoriosController.php que implementa la clase DirectoriosController.

Dada la interfaz "mis ficheros", el usuario dispone de un botón al lado de cada directorio que sirve para invitar a compartirlo. Mediante el método POST, se envía el identificador del directorio seleccionado a la acción "invitarAction()" de la clase DirectoriosController. La acción

implementa la funcionalidad "Invitar", descrita en el diseño del módulo de gestión de directorios.

Listar invitaciones

Los ficheros que implementan la funcionalidad son:

- verinvitaciones.phtml que implementa la interfaz "Invitaciones".
- DirectoriosController que implementa la clase DirectoriosController.

Cuando un usuario realiza una petición para obtener la vista invitaciones se ejecuta la acción "verinvitacionesAction()" de la clase DirectoriosController. La acción implementa la funcionalidad "Listar invitaciones" descrita en el diseño. La vista se encarga de mostrar al usuario el listado de invitaciones.

Cancelar invitación

Los ficheros que implementan la funcionalidad son:

- Invitaciones.phtml que implementa la interfaz "Invitaciones".
- DirectoriosController que implementa la clase DirectoriosController.

Dada la interfaz "invitaciones", el usuario dispone de un botón al lado de cada invitación listada. El botón es utilizado para cancelar la invitación que se encuentra en la misma fila que el botón. Cuando es pulsado, mediante la método GET, se envía el identificador del directorio seleccionado a la acción "cancelarInvitacionAction()" de la clase DirectoriosController. La acción implementa la funcionalidad "Cancelar invitación" descrita en el diseño.

Aceptar invitación

Los ficheros que implementan la funcionalidad son:

- Invitaciones.phtml que implementa la interfaz "Invitaciones".
- DirectoriosController que implementa la clase DirectoriosController.

Dada la interfaz "invitaciones", el usuario dispone de un botón al lado de cada invitación listada. El botón es utilizado para aceptar la invitación que se encuentra en la misma fila que el botón. Cuando es pulsado, mediante el método GET, se envía el identificador de la invitación seleccionada a la acción "aceptarInvitacionAction()" de la clase DirectoriosController. La acción implementa la funcionalidad "Aceptar invitación" descrita en el diseño.

6.2 Módulo de gestión de ficheros

En este apartado se definen los ficheros y clases que implementan las funcionalidades del módulo gestión de ficheros, así como la interacción del usuario con las diferentes funcionalidades.

Subir fichero

Los ficheros que implementan la funcionalidad son:

- misficheros.phtml que implementa la interfaz "Mis ficheros".
- FicherosController.php que implementa la clase FicherosController.

Dado el modal "subir ficheros" de la interfaz "Mis ficheros", el usuario dispone de un panel de control para seleccionar ficheros, cancelar selección, subir y borrar ficheros. Para cada una de estas acciones dispone de un botón:

- Botón seleccionar ficheros abre un navegador al usuario que permite seleccionar ficheros dentro de su disco duro. Una vez lo selecciona mediante JavaScript se incluye dentro de la lista de ficheros seleccionados para subir.
- Botón cancelar selección. Permite al usuario cancelar los ficheros seleccionados a subir. Mediante JavaScript se eliminan de la lista aquellos ficheros cancelados.
- Botón subir fichero envía los ficheros seleccionados al servidor utilizando el método POST. Invoca la acción "subirAction()" de la clase FicherosController, que implementa la funcionalidad "subir fichero" descrita en el diseño.
- Botón borrar fichero. Envía el identificador del fichero que se quiere borrar utilizando el método POST. Invoca la acción "borrarAction()" de la clase FicherosController, que implementa la funcionalidad "borrar fichero" descrita en el diseño.

En la subida de fichero hay que tener en cuenta la política de replicación establecida. Para éste caso, la función que implementa la política es "replicación()" que debe ser implementada por todas las clases de replicación. La clase "ReplicacionVoCS", en su función "replicación()", implementa la política de subida diseñada en el punto 5.2.6.2 Replicación de ficheros. La clase "ReplicacionAnillo", en su función "replicación()", implementa la política de subida diseñada en el punto 5.2.6.2 Replicación de ficheros.

La funcionalidad subir fichero ha sido implementada utilizando AJAX. La biblioteca utilizada es "BlueImp jQuery FileUpload" que permite realizar las operaciones descritas mediante AJAX.

El uso de la biblioteca es el siguiente:

- Generar en la vista el formulario y la presentación necesaria definida en el "modal subir ficheros". Al formulario se le incluye un identificador para poder inicializar el módulo de subida con AJAX.

```
<form id="fileupload" action="/ficheros/switch" method="POST"
enctype="multipart/form-data">
```

Debido a la biblioteca utilizada la implementación se ve forzada a crear una nueva acción. El formulario se utiliza tanto para la subida como para el borrado, por eso se incluye una nueva acción para diferenciar entre las dos operaciones. Su implementación es la siguiente:

```
if ($this->_request->isPost()) {
      $id = $this->_request->getParam('files');
      if($id == null){
            $this->upload( $this->session->user['id'],
                        $this->session->user['email'] );
      }else{
            $this->delete( $this->session->user['id'],
                        $this->session->user['email'],$id);
      }
}
```

En el caso que el POST incluye el identificador de un fichero se trata de la acción borrar fichero y en otro caso se trata de subir fichero.

- Inicializar el módulo e indicar sus opciones. El código en lenguaje JavaScript utilizado para ello es:

```
$('#fileupload').fileupload();
$('#fileupload').fileupload('option', {
      maxFileSize: 5368709120

});
```

Para el cifrado de ficheros se utilizará AES modo COUNTER. Se hará uso de la biblioteca "phpseclib". Para el cifrado de un contenido de un fichero se utilizan las siguientes líneas de código:

```
$cipher = new Crypt_AES(CRYPT_AES_MODE_CTR);
$cipher->setKey($claveCifrado);
$cifrado = $cipher->encrypt($contents);
```

Para el descifrado se utilizan las siguientes líneas de código:

```
$cipher = new Crypt_AES(CRYPT_AES_MODE_CTR);
$cipher->setKey($claveCifrado);
$texto = $cipher->decrypt($cifrado);
```

Descargar fichero

Los ficheros que implementan la funcionalidad son:

- misficheros.phtml que implementa la interfaz "Mis ficheros".
- FicherosController.php que implementa la clase FicherosController.

Dada la interfaz "mis ficheros", el usuario dispone de un listado del contenido del directorio actual. En el nombre de cada fichero listado, hay un enlace para la descarga de dicho fichero. Éste enlace mediante el método GET, envía el identificador del fichero seleccionado a la acción "bajarAction()" de la clase FicherosController, que implementa la funcionalidad "bajar ficheros" descrita en el diseño.

También es posible descargar ficheros desde el modal "subir ficheros" de la interfaz "mis ficheros". El nombre de los ficheros subidos son enlaces de similares características a los enlaces explicados anteriormente.

En el caso que la petición del fichero se haga sobre un servidor en el qué no esté alojado, se ejecuta la función "bajada()" que debe ser implementada por toda clase de replicación. La clase "ReplicacionVoCS", en su función "bajada()", implementa la política de bajada diseñada en el punto 5.2.6.2 Replicación de ficheros. La clase "ReplicacionAnillo", en su función "bajada()", implementa la política de bajada diseñada en el punto 5.2.6.2 Replicación de ficheros.

Eliminar fichero

Los ficheros que implementan la funcionalidad son:

- ficheros.phtml que implementa la interfaz "Mis ficheros".
- FicherosController.php que implementa la clase FicherosController.

Dada la interfaz "mis ficheros", el usuario dispone al lado de cada fichero listado un botón o enlace para eliminarlo. Pulsado el botón o enlace, mediante el método GET, se envía el

identificador del fichero a la acción "borrarAction()" de la clase FicherosController, que implementa la funcionalidad "eliminar fichero" descrita en el diseño.

Listar ficheros eliminados

Los ficheros que implementan la funcionalidad son:

- verborrados.phtml que implementa la interfaz "Eliminados".
- FicherosController que implementa la clase DirectoriosController.

Cuando un usuario realiza una petición para obtener la vista de ficheros eliminados se ejecuta la acción verborradosAction() de la clase FicherosController. La acción implementa la funcionalidad "Listar ficheros eliminados" descrita en el diseño. La vista se encarga de mostrar al usuario el listado de ficheros eliminados del usuario.

Listar versiones anteriores

Los ficheros que implementan la funcionalidad son:

- verversiones.phtml que implementa la interfaz "Versiones anteriores".
- FicherosController que implementa la clase DirectoriosController.

Cuando un usuario realiza una petición para obtener la vista de versiones anteriores de un fichero se ejecuta la acción verversionesAction() de la clase FicherosController. La acción implementa la funcionalidad "Listar versiones anteriores" descrita en el diseño. La vista se encarga de mostrar al usuario el listado las versiones del fichero seleccionado por el usuario.

Restaurar ficheros

Los ficheros que implementan la funcionalidad son:

- verversiones.phtml que implementa la interfaz "Versiones anteriores".
- verborrados.phtml que implementa la interfaz "Eliminados".
- FicherosController.php que implementa la clase FicherosController.

Dadas las interfaces "Versiones anteriores" y "Eliminados", el usuario dispone de un botón o enlace al lado de cada fichero eliminado o versión anterior. Pulsado el botón, mediante el método GET, se envía el identificador del fichero seleccionado a la acción "restaurarAction()" de la clase FicherosController que implementa la funcionalidad "Restaurar ficheros" descrita en el diseño.

6.3 Módulo gestión de usuarios

En este apartado se definen los ficheros y clases que implementan las funcionalidades del módulo gestión de usuarios, así como la interacción del usuario con las diferentes funcionalidades.

Ver mi cuenta

Los ficheros que implementan la funcionalidad son:

- micuenta.phtml que implementa la interfaz "Mi cuenta".
- UsuarioController.php que implementa la clase UsuarioController.

Cuando un usuario realiza una petición para obtener la vista de mi cuenta se ejecuta la acción micuentasAction() de la clase FicherosController. La acción implementa la funcionalidad "Ver mi cuenta" descrita en el diseño. La vista se encarga de mostrar al usuario la información de la cuenta del usuario.

Ver espacio disponible

Los ficheros que implementan la funcionalidad son:

- micuenta.phtml que implementa la interfaz "Mi cuenta".
- layout.phtml que implementa la interfaz "Plantilla o layout".
- UsuarioController.php que implementa la clase UsuarioController.

Cuando un usuario realiza una petición para obtener la vista de mi cuenta o una vista que contenga layout, se ejecuta la acción correspondiente a esas vistas que implementan la funcionalidad "Ver espacio disponible" descrita en el diseño. La vista se encarga de mostrar al usuario el espacio disponible.

Recomendar a un amigo

Los ficheros que implementan la funcionalidad son:

- recomendar.phtml que implementa la interfaz "Recomendar".
- UsuarioController.php que implementa la clase UsuarioController.

Dada la interfaz "Recomendar", el usuario dispone de un formulario para añadir la dirección de correo electrónico del invitado y un botón para enviar. Pulsado el botón enviar, mediante el método POST, se envía la dirección a la acción "recomendarAction()" de la clase

UsuarioController, que implementa la funcionalidad "Recomendar a un amigo" descrita en el diseño.

Cambiar contraseña

Los ficheros que implementan la funcionalidad son:

- micuenta.phtml que implementa la interfaz "Mi cuenta".
- UsuarioController.php que implementa la clase UsuarioController.

Dada la interfaz "Mi cuenta", el usuario dispone de un formulario para añadir cambiar su contraseña de aplicación. Cuando el usuario pulsa el botón "Enviar", mediante el método POST, se envía la información que escribió en la inputs del formulario a la acción "cambiarcontrasenaAction()" de la clase UsuarioController, que implementa la funcionalidad "Cambiar contraseña" descrita en el diseño.

6.4 Módulo de inicio

En este apartado se definen los ficheros y clases que implementan las funcionalidades del módulo de inicio, así como la interacción del usuario con las diferentes funcionalidades.

Registrarse

Los ficheros que implementan la funcionalidad son:

- registro.phtml que implementa la interfaz "Mi cuenta".
- IndexController.php que implementa la clase IndexController.

Dada la interfaz "Registro", el usuario dispone de un formulario en el que incluir su información personal y de cuenta. Cuando el usuario pulsa el botón "Registrar", mediante el método POST, se envía la información que escribió en la inputs del formulario a la acción "registroAction()"de la clase UsuarioController, que implementa la funcionalidad "Registrarse" descrita en el diseño.

Iniciar sesión

Los ficheros que implementan la funcionalidad son:

- index.phtml que implementa la interfaz "Index".
- IndexController.php que implementa la clase IndexController.

Dada la interfaz "Index", el usuario dispone de un formulario en el que incluir su nombre de usuario y su contraseña. Cuando el usuario pulsa el botón "Entrar", mediante el método POST, se envía la información que escribió en la inputs del formulario a la acción "indexAction()"de la clase UsuarioController, que implementa la funcionalidad "Iniciar sesión" descrita en el diseño.

Cerrar sesión

Los ficheros que implementan la funcionalidad son:

- layout.phtml que implementa la interfaz "Plantilla o Layout".
- IndexController.php que implementa la clase IndexController.

Dado el layout, el usuario dispone de un enlace para desconectarse de la aplicación. Pulsado el enlace se invoca la acción "cerrarsesionAction()"de la clase UsuarioController, que implementa la funcionalidad "Cerrar sesión" descrita en el diseño.

6.5 Webservice REST

Para la implementación de los servicios Webservice diseñados, se ha utilizado la clase abstracta Zend_Rest_Controller que aporta Zend Framework. La clase RestController especificada en al diagrama de clases, extiende de Zend_Rest_Controller implementando las funciones:

- postAction(), función que se ejecuta siempre que el servicio web reciba un POST vía HTTP.
- getAction(), función que se ejecuta siempre que el servicio web reciba un GET vía HTTP.
- deleteAction(), función que se ejecuta siempre que el servicio web reciba un DELETE vía HTTP.
- putAction(), función que se ejecuta siempre que el servicio web reciba un DELETE vía HTTP.

La clase se encuentra en el fichero RestController.php.

El código necesario para recibir un fichero es el siguiente:

```
$adapter = new Zend_File_Transfer_Adapter_Http();
$adapter->setDestination($user_path);
```

Se utiliza la clase Zend_File_transfer_Adapter_Http, que permite obtener un objeto capaz controlar los ficheros enviados por HTTP.

Para generar un cliente REST y poder llamar a los servicios, se utiliza la clase Zend_Http_Client. A continuación, se muestra el código necesario para enviar un fichero a un servicio web.

```
$httpClient = new Zend_Http_Client('http://'.$servidor.'/rest');
$httpClient->setParameterPost('metodo','peticionReplicaService');
            $httpClient->setFileUpload($path_del_fichero,'upload');
            return  $httpClient->request('POST');
```

6.6 Interfaces

En este aparatado se mostrará la implementación de las interfaces diseñadas.

6.6.1 Plantilla o Layout

La implementación de la interfaz diseñada en el punto 5.2.2.1 Plantilla o Layout se muestra en la Figura 6.2.

Figura 6.1 Implementación interfaces: Layout

6.6.2 Index

La implementación de la interfaz diseñada en el punto 5.2.2.2 Index se muestra en la Figura 6.3.

Figura 6.2 Implementación interfaces: Index

6.6.3 Registro

La implementación diseñada en el punto 5.2.2.3 Registro se muestra en la Figura 6.4.

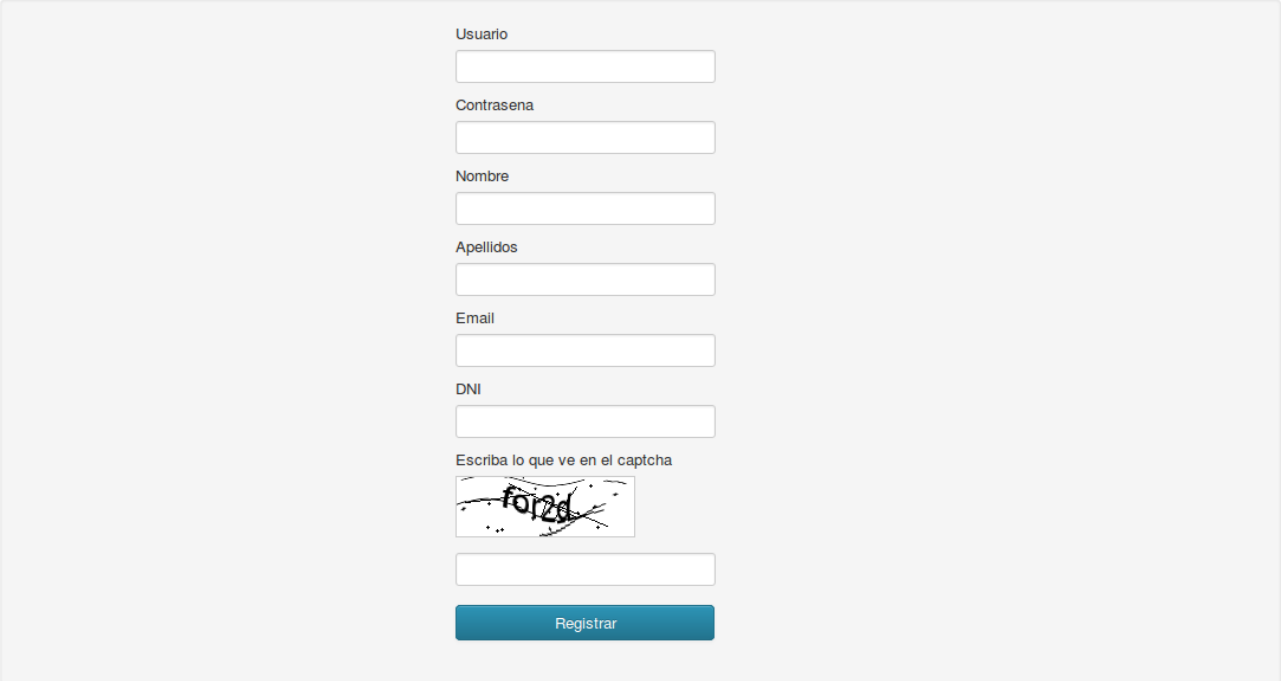


Figura 6.3 Implementación interfaces: Registro

6.6.4 Mis ficheros

La implementación de la interfaz diseñada en el punto 5.2.2.4 Mi ficheros se muestra en la Figura 6.5.

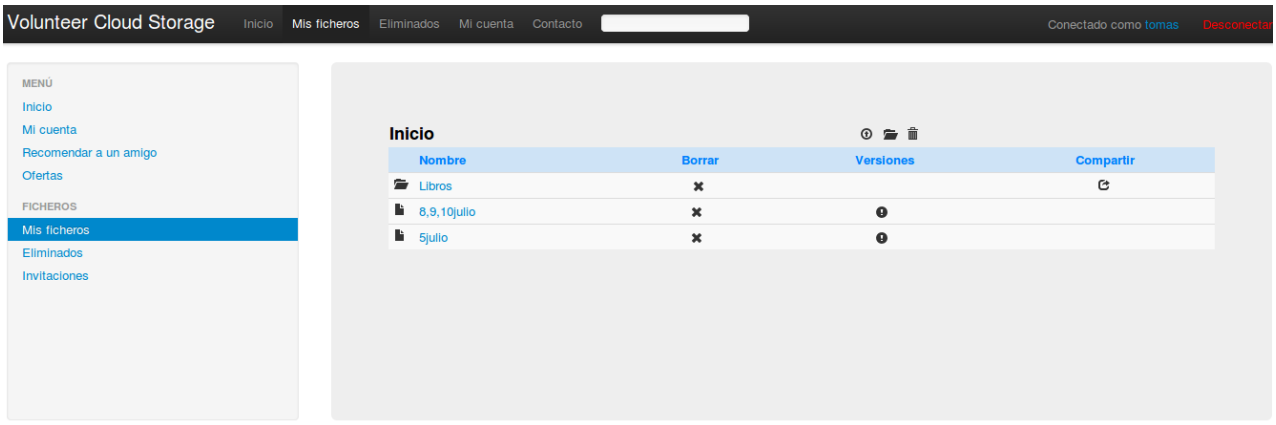


Figura 6.4 Implementación interfaces: Mis ficheros

6.6.5 Modal crear directorio

La implementación del modal diseñado en el punto 5.2.2.5 Modal crear directorio se muestra en la Figura 6.6.

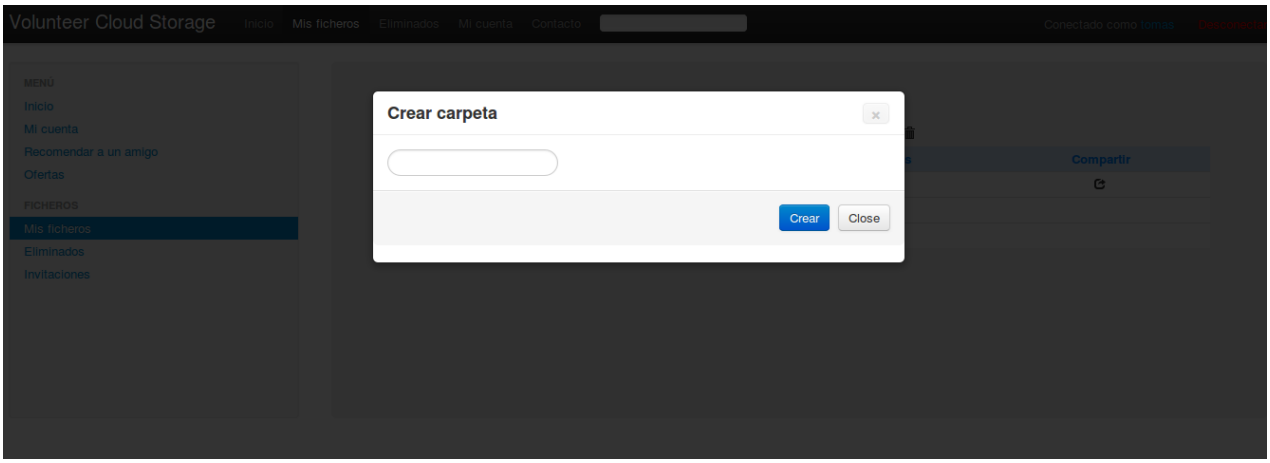


Figura 6.5 Implementación interfaces: Modal crear directorio

6.6.6 Modal subir ficheros

La implementación del modal diseñado en el punto 5.2.2.6 Modal subir ficheros se muestra en la Figura 6.7.

Figura 6.6 Implementación interfaces: Modal subir ficheros

6.6.7 Eliminados

La implementación de la interfaz diseñada en el punto 5.2.2.7 Eliminados se muestra en la Figura 6.8.

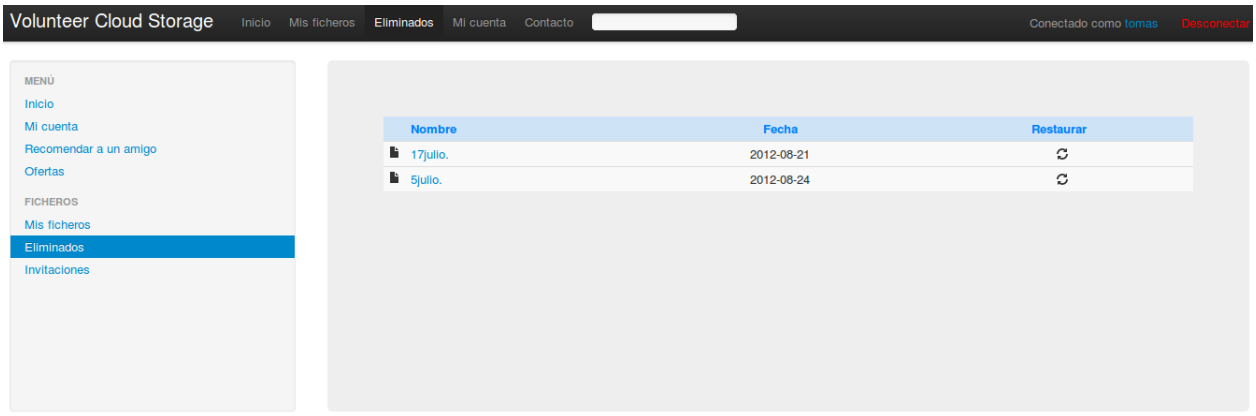


Figura 6.7 Implementación interfaces: Eliminados

6.6.8 Versiones anteriores

La implementación de la interfaz diseñada en el punto 5.2.2.8 Versiones anteriores se muestra en la Figura 6.9.

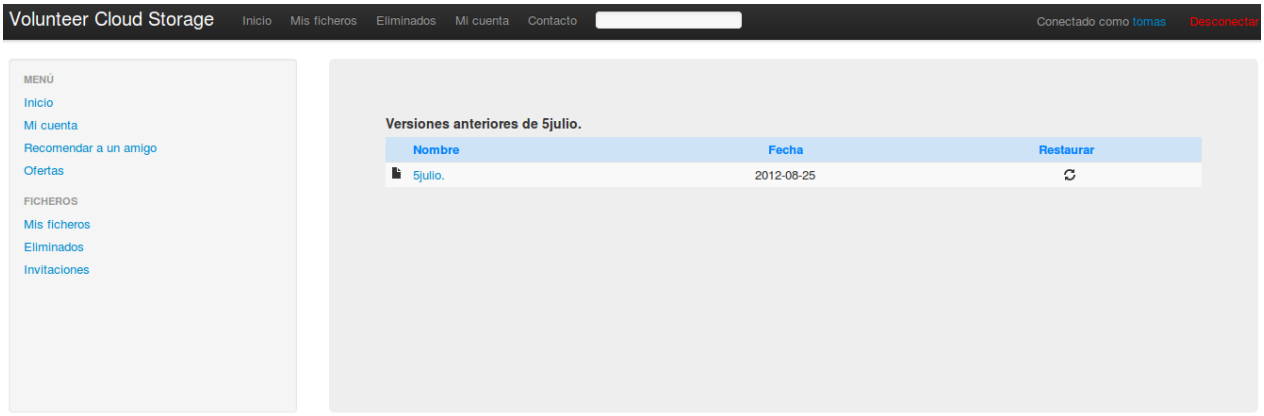


Figura 6.8 Implementación interfaces: Versiones anteriores

6.6.9 Invitaciones

La implementación de la interfaz diseñada en el punto 5.2.2.9 Invitaciones se muestra en la Figura 6.10.

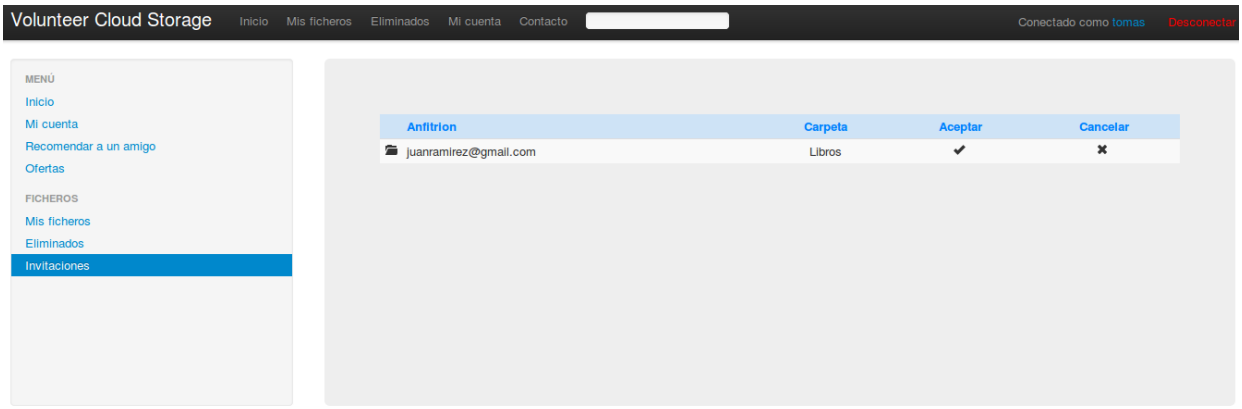


Figura 6.9 Implementación interfaces: Invitaciones

6.6.10 Mi cuenta

La implementación de la interfaz diseñada en el punto 5.2.2.10 Mi cuenta se muestra en la Figura 6.11.

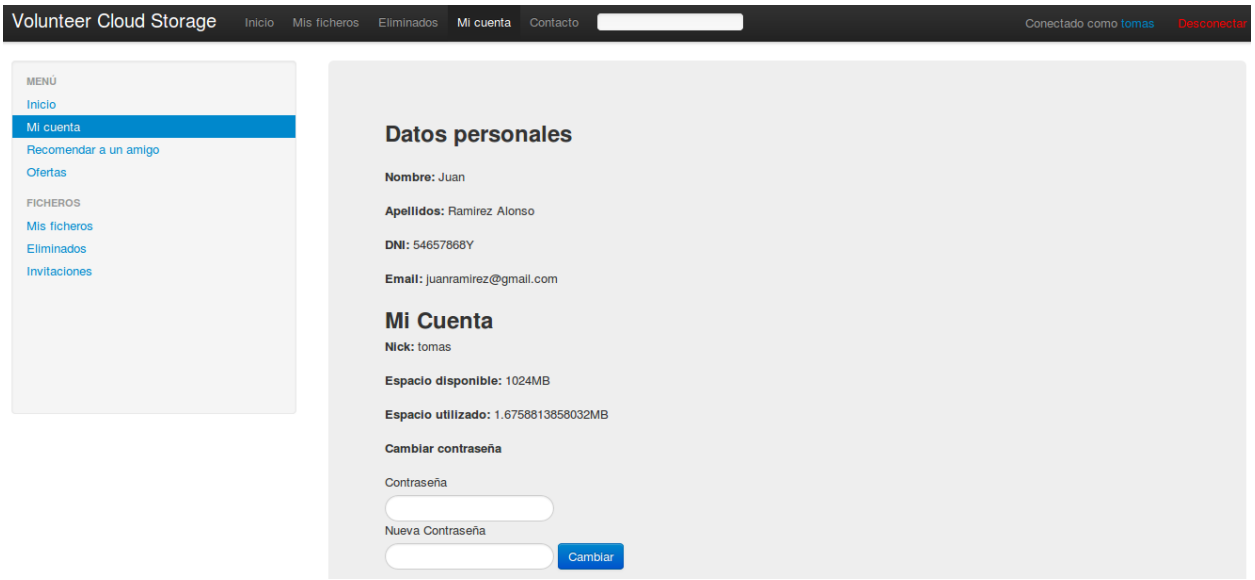


Figura 6.10 Implementación interfaces: Mi cuenta

6.6.11 Recomendar

La implementación de la interfaz diseñada en el punto 5.2.2.11 Recomendar se muestra en la Figura 6.12.

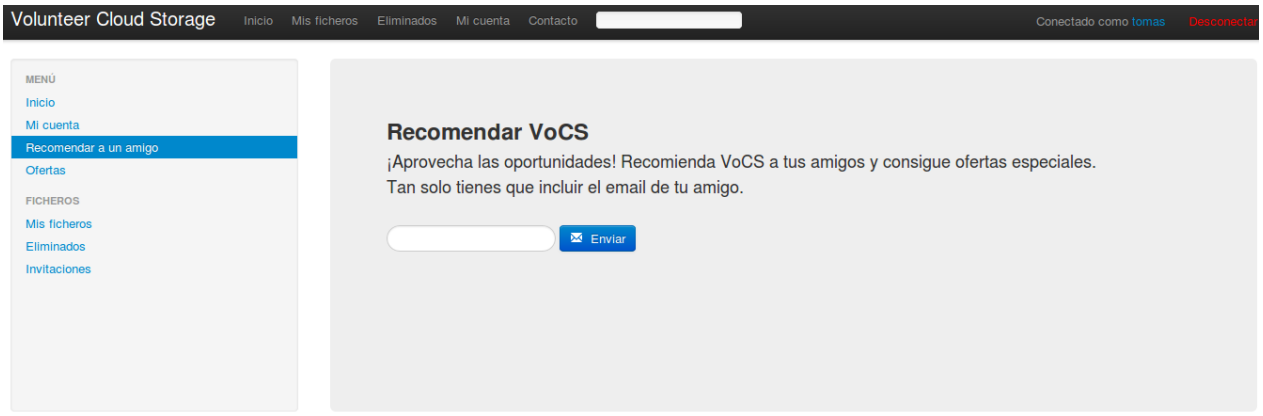


Figura 6.11 Implementación interfaces: Recomendar

6.7 Base de datos

En este aparatado se detalla la implementación de la base de datos diseñada en 5.2.3 Modelo relacional de la base de datos. En este aparatado, solo se tendrán en cuenta los detalles que difieran del diseño, por las características únicas de MySQL: los tipos de datos y el modelo de almacenamiento.[4]

Todas las tablas utilizan el modelo de almacenamiento InnoDB.

vocs_compartidos		
vocs_c_directorio	Identificador del directorio.	BIGINT (20), sin signo, no nulo.
vocs_c_anfitrion	Identificador del usuario anfitrión.	VARCHAR (255), no nulo.
vocs_c_invitado	Identificador del usuario invitado.	VARCHAR (255), no nulo.

Tabla 6.2 Implementación: Tabla vocs_compartidos

vocs_df_directorio_fichero		
vocs_df_directorio	Identificador del directorio que contiene un fichero.	BIGINT (20), sin signo, no nulo.
vocs_df_fichero	Identificador del fichero que está contenido en un directorio.	BIGINT(20), sin signo, no nulo.

Tabla 6.3 Implementación: Tabla vocs_df_directorio_fichero

vocs_directorios		
vocs_d_directorio	Identificador de la entidad	BIGINT (20), sin signo, no nulo y autoincremental.
vocs_d_nombre	Nombre del directorio.	VARCHAR(255), no nulo.

Tabla 6.4 Implementación: Tabla vocs_directorios

vocs_du_directorio_usuario		
vocs_du_directorio	Identificador del directorio que posee el usuario.	BIGINT (20), sin signo, no nulo.
vocs_du_usuario	Identificador del usuario que posee un directorio.	VARCHAR(255), no nulo.
vocs_du_espadre	BIT que indica si el directorio que posee el usuario es padre o no. 0=> no padre, 1=> padre.	TINYINT(1)
vocs_du_padre	Identificador del directorio padre.	BIGINT (20), sin signo, no nulo.

Tabla 6.5 Implementación: Tabla vocs_du_directorio_usuario

vocs_etiqueta_fichero		
vocs_ef_fichero	Identificador del fichero que tiene una etiqueta.	BIGINT (20), sin signo, no nulo.
vocs_ef_etiqueta	Identificador de la etiqueta del fichero.	VARCHAR (255), no nulo.

Tabla 6.6 Implementación: Tabla vocs_etiqueta_fichero

vocs_etiquetas		
vocs_e_etiqueta	Identificador de la entidad	VARCHAR (255), no nulo.

Tabla 6.7 Implementación: Tabla vocs_etiquetas

vocs_fb_borrados		
vocs_fb_activo	Identificador del fichero que es primera versión.	BIGINT (20), sin signo, no nulo.
vocs_fb_inactivo	Identificador del fichero que es versión antigua.	BIGINT (20), sin signo, no nulo.

Tabla 6.8 Implementación: Tabla vocs_fb_borrados

vocs_ficheros		
vocs_f_id	Identificador del fichero que es primera versión.	BIGINT (20), sin signo, no nulo, autoincremental.
vocs_f_activo	Indica si el fichero ha sido borrado o no. 0 indica borrado, 1 indica activo.	TINYINT (1), no nulo.
vocs_f_extension	Extensión del fichero	VARCHAR (9)
vocs_f_fecha_borrado	Fecha de borrado del fichero. NULL si nunca fue borrado.	DATE
vocs_f_fecha_subida	Fecha de subida.	DATE, no nulo
vocs_f_hash	Hash del contenido del fichero.	VARCHAR (255).
vocs_f_nivel_seguridad	Nivel de seguridad aplicado al fichero.	INTEGER, no nulo.
vocs_f_nombre	Nombre del fichero.	VARCHAR (255), no nulo.
vocs_f_ruta	Ruta física del fichero.	VARCHAR (512), no nulo.
vocs_f_tamano	Tamaño en Bytes del fichero.	DOUBLE, no nulo.
vocs_f_m1	Cifrado de la primera mitad de la clave de cifrado del usuario.	VARCHAR (255), no nulo.

Tabla 6.9 Implementación: Tabla vocs_ficheros

vocs_informacion		
vocs_i_id	Identificador del usuario.	VARCHAR (255), no nulo
vocs_i_s1	Salt para la generación de la cadena de autenticación (comparación).	VARCHAR (255).
vocs_i_s2	Salt para la generación de segunda mitad de la clave de cifrado	VARCHAR (255).

Tabla 6.10 Implementación: Tabla vocs_informacion

vocs_invitaciones		
vocs_i_anfitrion	Identificador del usuario anfitrión de la invitación.	VARCHAR (255), no nulo.
vocs_i_invitado	Identificador del usuario invitado.	VARCHAR (255), no nulo.
vocs_i_directorio	Identificador del directorio al que se invita.	BIGINT(20), sin signo, no nulo.

Tabla 6.11 Implementación: Tabla vocs_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	VARCHAR (255), no nulo.
vocs_s_cifrado	Cifrado de la cadena de entrada utilizada en la generación de la segunda mitad de la clave de cifrado	VARCHAR (255), no nulo.

Tabla 6.12 Implementación: Tabla vocs_servidor

vocs_servidores		
vocs_s_id	Identificador del servidor.	INTEGER, no nulo, sin signo y autoincremental.
vocs_s_confianza	Columna que indica si el servidor es de confianza.	TINYINT (1), no nulo.
vocs_s_grupo	Grupo de servidores al que pertenece.	INTEGER, no nulo.
vocs_s_nombre	Nombre ó IP del servidor.	VARCHAR (255), no nulo.

Tabla 6.13 Implementación: Tabla vocs_servidores

vocs_ti_fichero_servidor		
vocs_fs_fichero	Identificador del fichero.	BIGINT (20), sin signo y no nulo.
vocs_fs_servidor	Identificador del servidor donde está alojado el fichero.	INTEGER, sin signo, no nulo.
vocs_fs_fisico	Nombre físico del fichero.	VARCHAR (255), no nulo.

Tabla 6.14 Implementación: Tabla vocs_ti_fichero_servidor

vocs_ti_usuario_fichero		
vocs_uf_usuario	Identificador de usuario.	VARCHAR (255), no nulo.
vocs_uf_fichero	Identificador del fichero perteneciente al usuario.	BIGINT(20), sin signo y no nulo.

Tabla 6.15 Implementación: Tabla vocs_ti_usuario_fichero

vocs_tipo_replicacion		
vocs_tr_id	Identificador de replicación.	INTEGER, sin signo, no nulo y autoincremental.
vocs_tr_tipo	Indica el tipo de replicación del sistema. 0=> Replicacion VoCS. 1=> Anillo	INTEGER, sin signo, no nulo.

Tabla 6.16 Implementación: Tabla vocs_tipo_replicacion

vocs_uc_usuario_clave		
vocs_uc_usuario	Identificador del usuario.	VARCHAR (255), no nulo.
vocs_uc_clave	Clave de confirmación de registro de usuario.	VARCHAR (255), no nulo.

Tabla 6.17 Implementación: Tabla vocs_uc_usuario_clave

vocs_usuarios		
vocs_u_nick	Identificador del usuario	VARCHAR (255), no nulo.
vocs_u_activo	Marca si el usuario está activo o inactivo. 0=inactivo, 1=activo	TINYINT (4) y no nulo.
vocs_u_apellidos	Apellidos del usuario	VARCHAR (50).
vocs_u_contrasena	Contraseña del usuario	VARCHAR (255), no nulo.
vocs_u_dni	DNI del usuario	VARCHAR (9), no nulo.
vocs_u_email	Correo electrónico del usuario	VARCHAR (255), no nulo.
vocs_u_espacio	Espacio Total del que dispone el usuario en MB	DOUBLE, no nulo.
vocs_u_espacio_utilizado	Espacio utilizado en MB	DOUBLE, no nulo.
vocs_u_nombre	Nombre	VARCHAR (45), no nulo.
vocs_u_rol	Rol del usuario.	TINYINT(4) y no nulo.

Tabla 6.18 Implementación: Tabla vocs_usuarios

7. Análisis de Rendimiento

En este apartado se realizará la evaluación del rendimiento del sistema en el almacenamiento de ficheros y descarga de ellos.

Para ello, se evaluarán las diferentes situaciones en cada uno de los dos modelos diseñados e implementados en el TFG, detallados en el punto 5.2.6.2 Replicación de ficheros.

- Subida de ficheros.
  - Seguridad baja. Sin replicaciones y sin cifrado.
  - Seguridad media. Con una replicación y sin cifrado.
  - Seguridad alta. Con una replicación y con cifrado.
- Descarga de ficheros.
  - El fichero está alojado en el servidor al que se realiza la petición de descarga.
  - El fichero no está alojado en el servidor al que se realiza la petición de descarga.

El entorno utilizado para la realización de las pruebas se indica en la tabla 7.1.

Computador 1	
Procesador	Pentium 4
CPU	2.4 GHz
RAM	1 GB
Disco duro	78,7 GB
Caché	512 KB
Computador 2	
Procesador	AMD Anthlon 64
CPU	800 MHz
RAM	1 GB
Disco duro	265,6 GB
Caché	1 GB
Red de conexión	
Velocidad subida	7.57 Mbps
Velocidad bajada	7.81 Mbps
Tipo	Wi-Fi (Eduroam)
Red interna del sistema	
Tipo	Ethernet 10-100 Mbps

Tabla 7.1: Entorno de pruebas

7.1 Subida de ficheros con el modelo de replicación VoCS

En este punto se mostrará el rendimiento de subida y descarga de ficheros utilizando el modelo VoCS.

	Seguridad baja (segundos)	Seguridad media (segundos)	Seguridad alta (segundos)
0,5 MB	2,081	3,676	19,587
10 MB	10,995	15,889	341,898
100 MB	99,264	133,777	3422,198
250 MB	305,056	691,105	9585,332
500 MB	710,573	1443,641	23134

Tabla 7.2: Pruebas replicación VoCS

Los datos recogidos en la tabla 7.2 generan la gráfica de la Figura 7.1

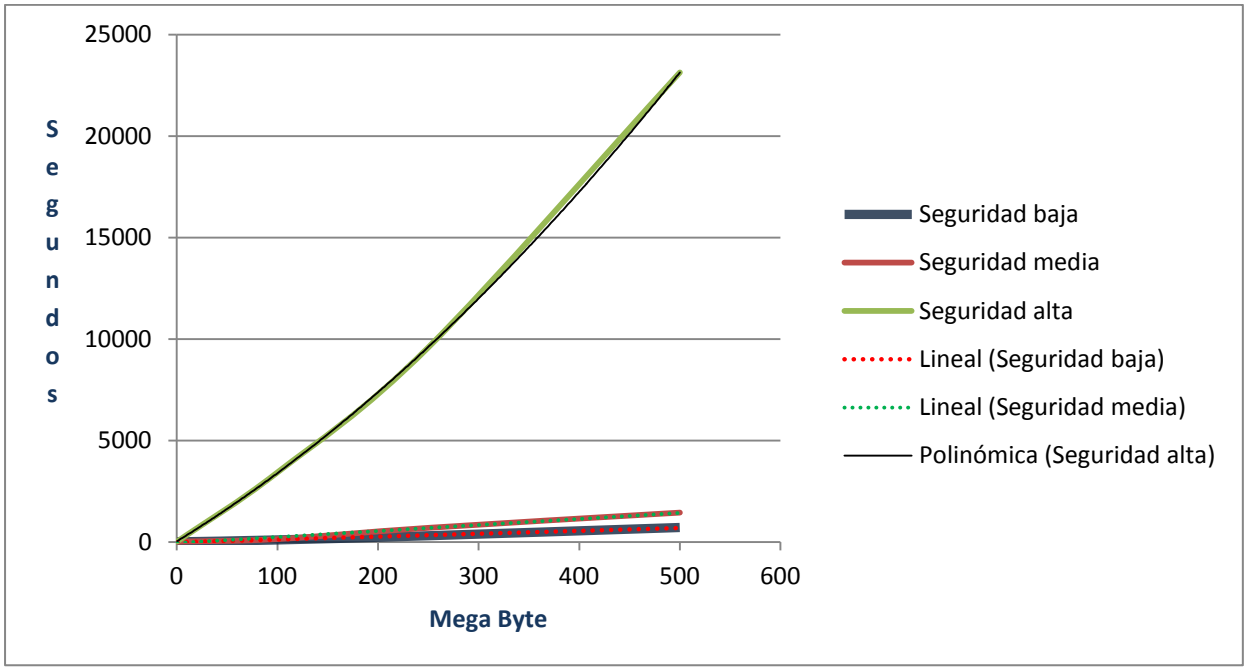


Figura 7.1: Rendimiento replicación VoCS

Como se puede observar en la Figura 7.1, la subida con seguridad media tiene un rendimiento inferior a la subida con seguridad baja, esto es debido a la replicación del fichero subido. En este caso, la diferencia no es muy alta en ficheros pequeños, pero en ficheros grandes puede llegar a ser casi hasta el doble de tiempo. La subida de ficheros con seguridad baja ofrece un buen rendimiento, ya que del tiempo utilizado solo 0,98625 segundos es de ejecución del servidor y el resto es por la transferencia del fichero entre cliente-servidor. La subida con seguridad media ofrece un rendimiento aceptable, pero en ficheros muy grandes el tiempo utilizado puede llegar a ser el doble o más que con seguridad baja.

Por otro lado, la subida de ficheros con seguridad alta tiene un rendimiento aceptable para ficheros muy pequeños. Pero en ficheros de tamaño medio o grande el rendimiento no es aceptable.

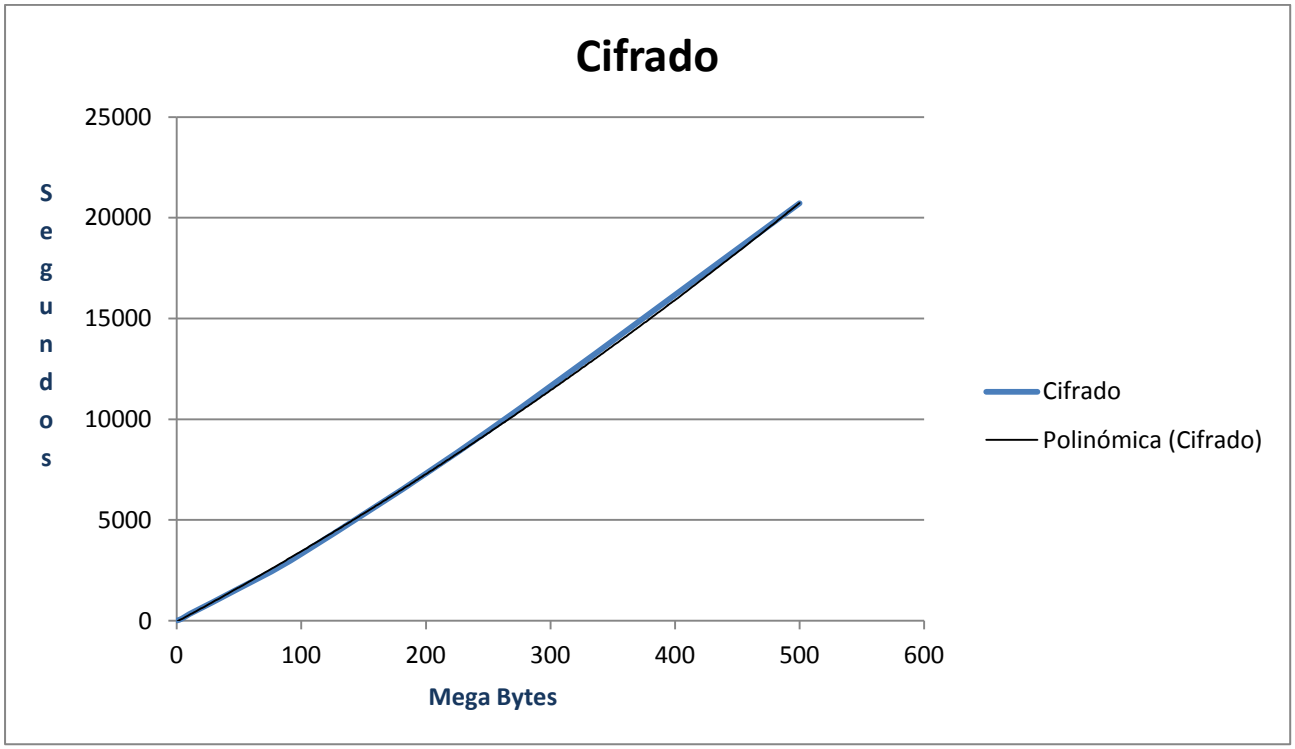


Figura 7.2: Cifrado

El bajo rendimiento de la subida de ficheros con seguridad alta es debido al cifrado de ficheros. Como se muestra en la Figura 7.2, el algoritmo de cifrado tiene una tendencia polinómica, pero es bastante lento para utilizarlo en el almacenamiento de ficheros.

7.2 Subida de ficheros con el modelo de replicación anillo

En este punto se mostrará el rendimiento de subida y descarga de ficheros utilizando el modelo en anillo.

	Seguridad baja (segundos)	Seguridad media (segundos)	Seguridad alta (segundos)
0,5 MB	1,983	2,395	17,271
10 MB	10,304	12,023	333,328
100 MB	101,47	128,021	3402,1332
250 MB	302,346	678,105	9455,8663
500 MB	690,153	1421,141	22234

Tabla 7.3: Pruebas replicación en anillo

Los datos recogidos en la tabla 7.3 generan la gráfica de la Figura 7.1

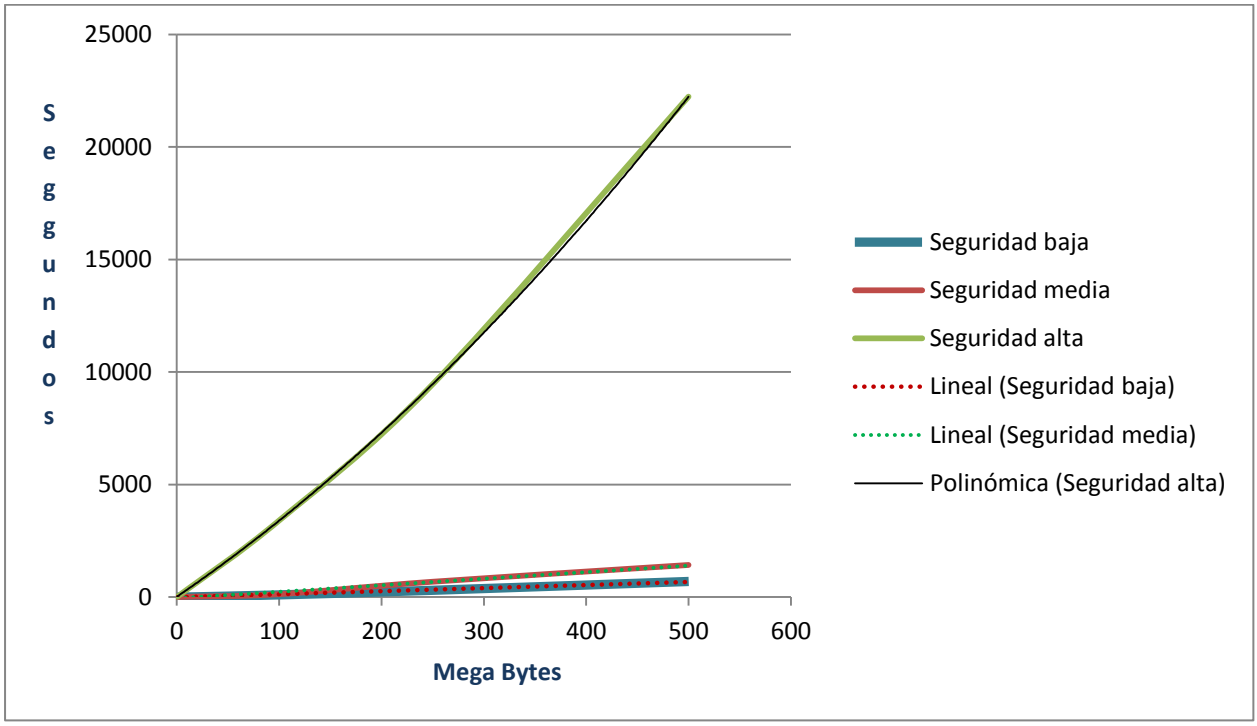


Figura 7.3: Rendimiento replicación en anillo.

Como se ve en la figura 7.3, la subida de ficheros con el modelo de replicación en anillo es muy parecida a la subida de ficheros con el modelo de replicación VoCS. De la misma manera, ofrece un buen rendimiento en subidas con seguridad baja y aceptable con seguridad media. La seguridad alta no ofrece un buen rendimiento por el tiempo que tarda en cifrar ficheros, como se ve en la figura 7.2,

7.3 Comparativa modelos de replicación.

En este punto se va a comparar las diferencias en el rendimiento de la subida y descarga de ficheros de los modelos de replicación del sistema.

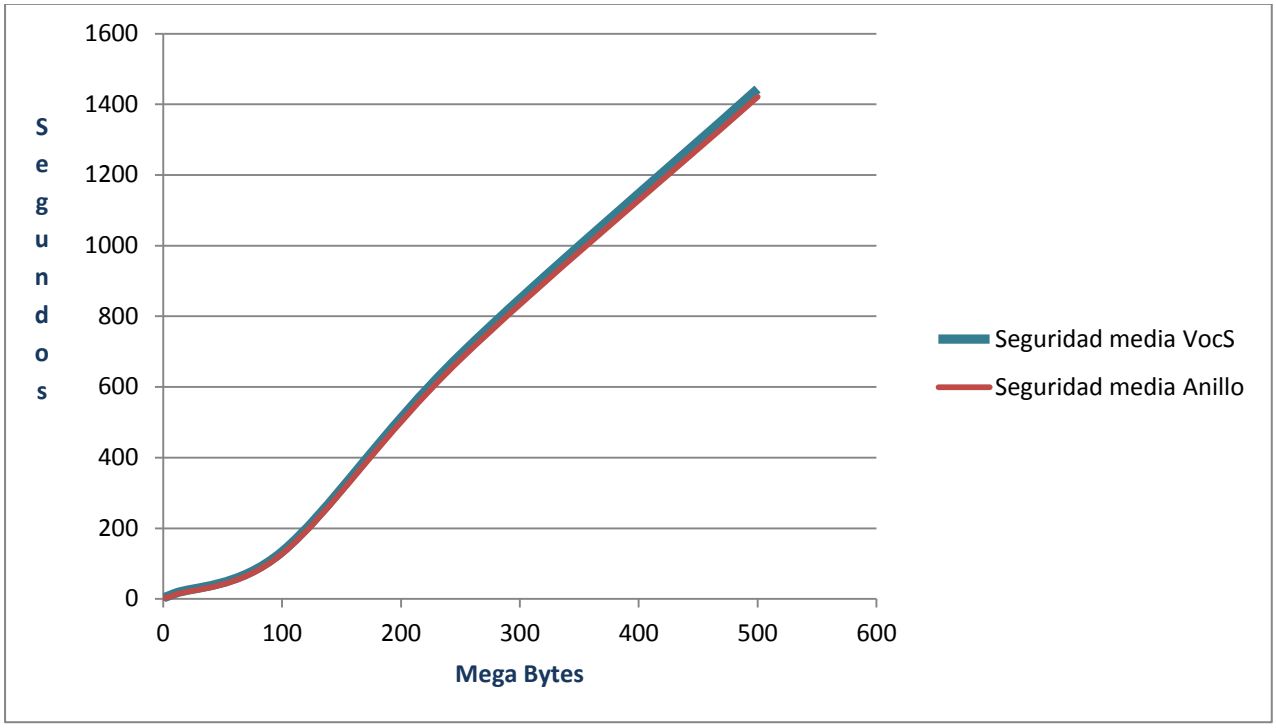


Figura 7.4: Comparativa seguridad media

La Figura 7.4 muestra el rendimiento de la subida de ficheros con seguridad media. Como se puede observar, el modelo de replicación en anillo ofrece un rendimiento ligeramente superior a la replicación VoCS. Esto se debe a que el modelo VoCS, realiza una petición y evaluación de métricas de los servidores de su subgrupo. En cambio, el modelo en anillo directamente replica.

	Alojado en el servidor.	No alojado en el servidor. Modelo VoCS.	No alojado en el servidor. Modelo en anillo.
0,5 MB	0,654	2,312	1,828
10 MB	6,306	8,502	8,323
100 MB	62,757	72,681	70,342
250 MB	169,56	252,64	241,33
500 MB	325,952	523,1172	511,659

Tabla 7.4: Pruebas bajada de ficheros

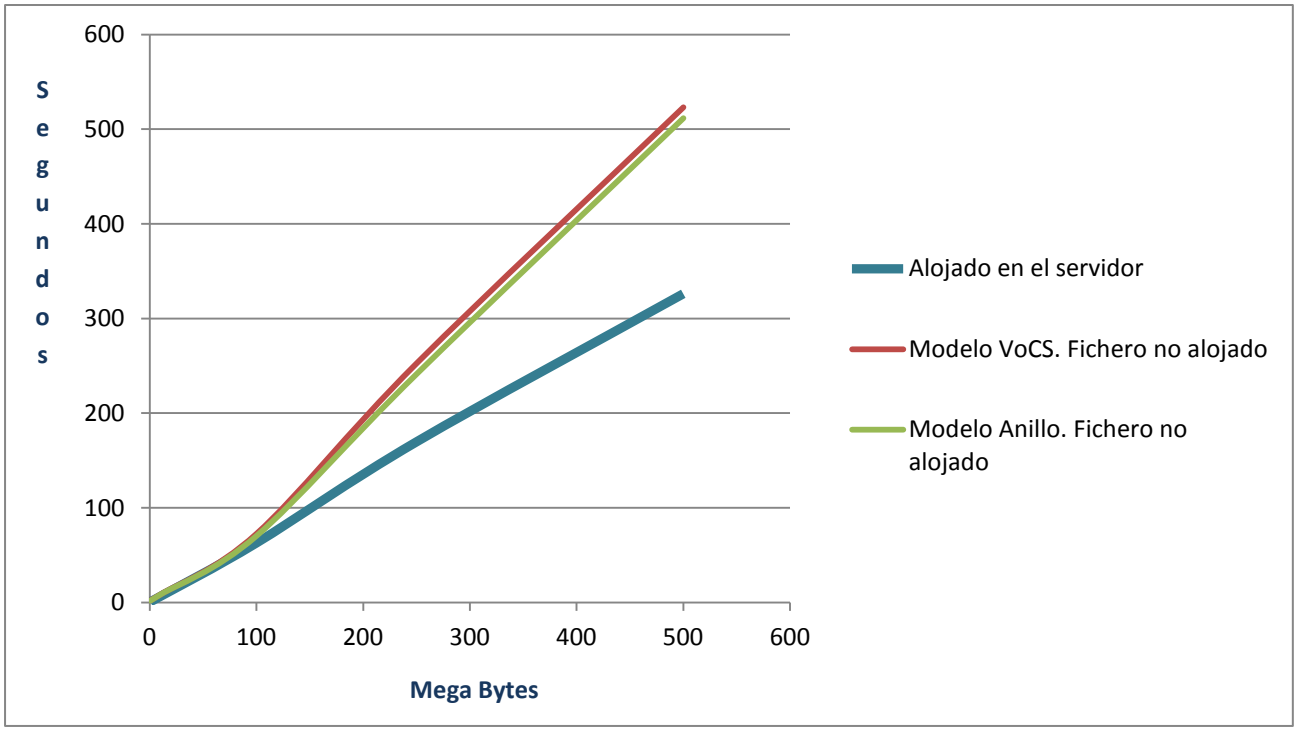


Figura 7.5: Comparativa descarga de ficheros.

En la Figura 7.5 se puede observar los tiempos de descargas utilizando los diferentes modelos de replicación. La línea azul muestra los tiempos cuando se realiza una petición de descarga a un servidor que tiene alojado el fichero, la línea roja representa cuando se realiza la petición a un servidor que no lo tiene alojado y el sistema utiliza el modelo VoCS, la línea verde representa cuando se realiza la petición a un servidor que no lo tiene alojado y utiliza el modelo en anillo.

Como se observa, el rendimiento ofrecido cuando se realiza la petición a un servidor que contiene el fichero, es mucho mejor ya que no tiene que buscarlo y pedirlo a otro servidor. En cambio, cuando se realiza la petición a servidores que no contienen el fichero, se realiza la transferencia del fichero al servidor, lo que empeora el rendimiento.

La diferencia que se muestra entre los diferentes modelos cuando se realiza una descarga de un fichero que no está alojado en el servidor, es debido a que el modelo VoCS realiza una réplica de éste en el sistema, en cambio el modelo en anillo solo lo sirve al usuario, sin replicarlo.

8. Conclusiones

En este apartado se exponen las conclusiones obtenidas en relación con los objetivos propuestos, los trabajos futuros que suponen mejoras del sistema y el presupuesto y planificación del TFG.

8.1 Conclusiones

Se ha cumplido el objetivo principal del proyecto consistente en realizar el análisis, diseño e implantación de un sistema de almacenamiento voluntario en la nube con las características indicadas en el punto 1.2 Objetivos. En los apartados 5.1 Análisis, 5.2 Diseño y 6 Implementación se pueden ver las fases del proyecto que han permitido cumplir con el objetivo principal del proyecto.

Al finalizar el presente TFG, se concluye que los sub-objetivos marcados también han sido alcanzados:

- Análisis y elección de las tecnologías Web actuales, que permitan el desarrollo de la aplicación VoCS y cumplan con sus requisitos y objetivos. Se ha realizado el análisis de las tecnologías que permitían el desarrollo del sistema, siendo elegidas: Linux, Apache, MySQL, PHP con Zend Framework, JavaScript con jQuery, HTML y CSS. Ver punto 5.2.9 Opciones de implementación.
- Se ha construido un entorno de desarrollo e implantación detallado en la tabla 7.1.
- Se ha realizado el análisis, diseño e implementación de un sistema de almacenamiento en la nube de acceso público, en el que se dispone de las funcionalidades: registrarse, iniciar sesión, cerrar sesión, ver información de la cuenta de usuario, ver espacio disponible en el sistema, cambiar contraseña de usuario, recomendar la aplicación a un amigo, crear directorios, eliminar directorios, compartir directorios, subir ficheros, descargar ficheros, eliminar ficheros, restaurar ficheros eliminados y restaurar versiones anteriores de los ficheros. Ver puntos 5.1 Análisis, 5.2 Diseño y 6 Implementación.
- Se ha desarrollado una base de datos que almacena la información relevante del sistema y permite la gestión de la información de forma sencilla y rápida. Ver puntos 5.1.3 Base de datos, 5.2.3 Modelo relacional de la base de datos, 6.7 Base de datos.
- Se ha diseñado un sistema de ficheros basado en bases de datos, en el que los usuarios son capaces de hacer uso de las funcionalidades comunes que aporta un sistema de ficheros convencional basada en directorios. Ver punto 5.2.4 Sistema de ficheros.

- Se ha desarrollado un sistema que permite compartir directorios y su contenido entre usuarios de la aplicación. Ver punto 5.2.4 Sistema de ficheros.
- Se han desarrollado y evaluado modelos de replicación de ficheros propios: Modelo de replicación VoCS y modelo en anillo. También se ha desarrollado un modelo de replicación de datos de la base de datos, basado en los modelos maestro-maestro y maestro-esclavo. Ver punto 5.2.6 Modelos de replicación.
- Se han desarrollado un conjunto de interfaces usables e intuitivas para el uso de la aplicación por parte de los usuarios. Ver puntos 5.2.2 Interfaces y 6.6 Interfaces.
- Se han desarrollado algoritmos que garantizan la seguridad de la cuenta del usuario y la generación de una clave segura para el cifrado de ficheros. Ver punto 5.2.5 Autenticación de usuarios y clave de cifrado de datos.
- Se ha evaluado la subida y descarga de ficheros en la aplicación con los diferentes modelos de replicación desarrollados. Ver punto 7. Análisis de Rendimiento.

En conclusión, se han alcanzado todos los objetivos propuestos en el inicio del presente Trabajo de Fin de Grado, a excepción de la subida de ficheros con seguridad alta que ofrece un bajo rendimiento debido al algoritmo de cifrado.

8.2 Trabajos Futuros

En este apartado se expondrán los trabajos que se pueden realizar en el futuro para mejorarlo y dotarlo de nuevas funcionalidades.

Nuevos modelos de replicación

Como trabajo futuro se pueden diseñar, implementar y evaluar nuevos modelos de replicación de ficheros que mejoren las prestaciones de los dos modelos ya desarrollados.

Nuevas funcionalidades

Desarrollo de nuevas funcionalidades como:

- Mover directorios, funcionalidad que permite modificar la ubicación de los directorios.
- Mover ficheros, funcionalidad que permite modificar la ubicación de los ficheros.
- Restaurar directorios, funcionalidad que permite restaurar los directorios eliminados.
- Renombrar directorios y ficheros.
- Generar permisos sobre directorios y ficheros. Los usuarios podrían tener permisos de escritura y/o lectura sobre los ficheros y directorios.
- Poder generar directorios públicos visibles para todos los usuarios.

- Buscar, funcionalidad que busca el fichero o directorio que el usuario indique.
- Modificar los datos personales y de cuenta.

Aplicación de escritorio

Como trabajo futuro se puede desarrollar una aplicación para ser instalada en el sistema operativo del usuario. Ésta aplicación sincroniza los ficheros y directorios del sistema VoCS con la aplicación de escritorio, por lo que el usuario podrá ver en su disco el contenido que tiene en VoCS.

Seguridad alta

Como se ha visto en el punto 7. Análisis de Rendimiento, la subida de ficheros con seguridad alta ofrece un bajo rendimiento debido al algoritmo de cifrado. Como trabajo futuro, se puede mejorar el algoritmo de cifrado o diseñar un nuevo procedimiento para ofrecer la confidencialidad de los ficheros subidos.

Sistema de fichero semántico

Un sistema de ficheros semánticos, que facilite la organización del espacio de nombres y la localización de los archivos almacenados en el sistema, permitiendo al usuario el acceso transparente al sistema de ficheros.

Cesión de espacio

Funcionalidad que permite a los usuarios que se registren ceder parte de su disco duro que será utilizado por el sistema para el almacenamiento de datos.

Integración con redes sociales

Permitir que la aplicación se integre con las redes sociales permitiendo compartir ficheros, subir fotos, notificar, etc.

Evaluación

Realizar una evaluación más completa utilizando más equipos, otras redes y más recursos que permitan un mejor análisis del rendimiento del sistema y los modelos de replicación.

8.3 Presupuesto y Planificación

En este punto se presenta el presupuesto total del TFG y la planificación resultante.

El presupuesto del Trabajo Fin de Grado se muestra en el Anexo III. Los sueldos de los trabajadores se han obtenido de [29].

Por otra parte, la planificación se encuentra recogida en el diagrama de Gantt expuesto en el Anexo V.

9. Bibliografía

[1] José H. Canós, Patricio Letelier y Mª Carmen Penadés (2003). *Metodologías Ágiles en el Desarrollo de Software*. Actas de las VIII Jornadas de Ingeniería del Software y Sases de Datos, JISBD (pp. 1-8)

[2] *Adaptive Software Development* (http://aidanamx.blogspot.com.es/) Agosto 2012

[3] Ramez Elmasri y Shamkant B. Navathe. *Fundamentos de Sistemas de Bases de Datos*. (5ª edición). Editorial Pearson Addison Wesley

[4] Adoración de Miguel, Mario Piattini y Esperanza Marcos, *Diseño de Bases de Datos Relacionales*. Editorial RA-MA.

[5] C.J Date, *Introducción a los sistemas de bases de datos* (7ª edición). Editorial Pearson Prentice Hall.

[6] Cyril Thibaud, *MySQL 5. Instalación, implementación, administración y programación.* Ediciones Eni.

[7] Di Giacomo, M., *MySQL: lessons learned on a digital library, (*http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1438321) *Junio 2012.*

[8] *Oracle and MySQL compared* (http://docs.oracle.com/cd/E12151_01/doc.150/e12155/oracle_mysql_compared.htm) Junio 2012.

[9] *HTML 4 Specification* (http://www.w3.org/TR/html401/about.html) Junio 2012.

[10] Francisco Charte Ojeda, *HTML La Bilbia.* Editorial Anaya.

[11] Larry Ullman, *PHP.* Editorial Pearson.

[12] *PHP* (http://php.net) Junio 2012.

[13] *Cookbook CakePHP (*http://book.cakephp.org*)* Junio 2012.

[14] *Zend Framework* (http://www.zend.com) Junio 2012.

[15] John Coggeshall y Morgan Tocker, *Zend Enterprise PHP Patterns*. Publicado por Apress.

[16] David Flanagan, *JavaScript: The Definitive Guide* (5ª edición). Editorial O'Reilly.

[17] Luc Van Lancker, XHTML y CSS – *Los nuevos estándares del código fuente (2ª Edición).* Ediciones Eni.

[18] *Cherokee* (http://www.cherokee-project.com/) Junio 2012.

[19] *Apache* (http://httpd.apache.org/) Junio 2012.

[20] James F. Kurose y Keith W. Ross, *Redes de computadoras un enfoque descendente* (5ª edición). Editorial Pearson.

[21] Joaquín García Alfaro y Xavier Perramón Tornil, *Aspectos avanzados de la seguridad en redes*. Publicado por Universidad Oberta de Cataluña.

[22] William Stallings, *Fundamentos de seguridad en redes.* Editorial Pearson.

[23] Jesús J. Ortega Triguero,Miguel Ángel López Guerrero,Eugenio C. García del Castillo Crespo, *Introducción a la criptografía: Historia y actualidad.* Universidad de Castilla la Mancha.

[24] William Stallings, *Fundamentos de seguridad en redes: Aplicaciones y estándares.* Editorial Pearson.

[25] *Secure Hash Standard FIPS 180-3* (http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf) Junio 2012.

[26] *Trade off* (http://www.pcguide.com/ref/hdd/perf/raid/whyTradeoffs-c.html) Junio 2012.

[27] *The Arte of Data Replication* (http://www.oracle.com/technetwork/articles/systems-hardware-architecture/o11-080-art-data-replication-487868.pdf) Junio 2012.

[28] Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal. (http://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750) Agosto 2012.

[29] Salario medio de un programador (http://www.tufuncion.com/trabajo-programador) Agosto 2012.

10. Anexos

10.1 Anexo I: Manual de Usuario

Este manual proporciona las instrucciones necesarias para el uso de todas las funcionalidades que el sistema proporciona a los usuarios.

VoCS un sistema de almacenamiento voluntario en la nube de código abierto y seguro, que ofrece al usuario acceso ubicuo a sus ficheros y funcionalidades para la manipulación de estos. A continuación se proporciona una guía de uso de las funcionalidades del sistema.

10.1.1 Registrarse

Para poder hacer uso de la aplicación es necesario ser usuario registrado. Para ello se accede a la URL donde se haya desplegado la aplicación, añadiendo el controlador “Inicio” y la acción “Registro”.

- https://url-de-la-aplicación/index/registro

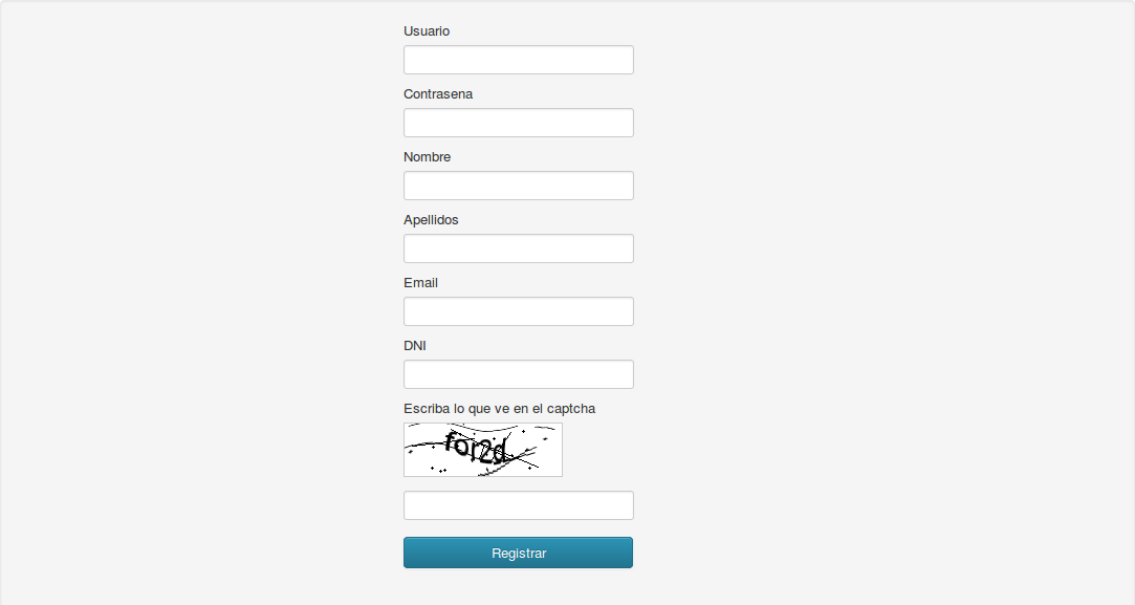


Figura 10.1: MU. Registro

Una vez accedido, se verá una pantalla similar a la Figura 10.1, donde se debe rellenar los datos que pide el formulario y pulsar el botón “Registrar”. Una vez pulsado, como se muestra en la Figura 10.2, la aplicación pedirá acceder a la dirección de correo electrónico indicada en el formulario y confirmar el registro.

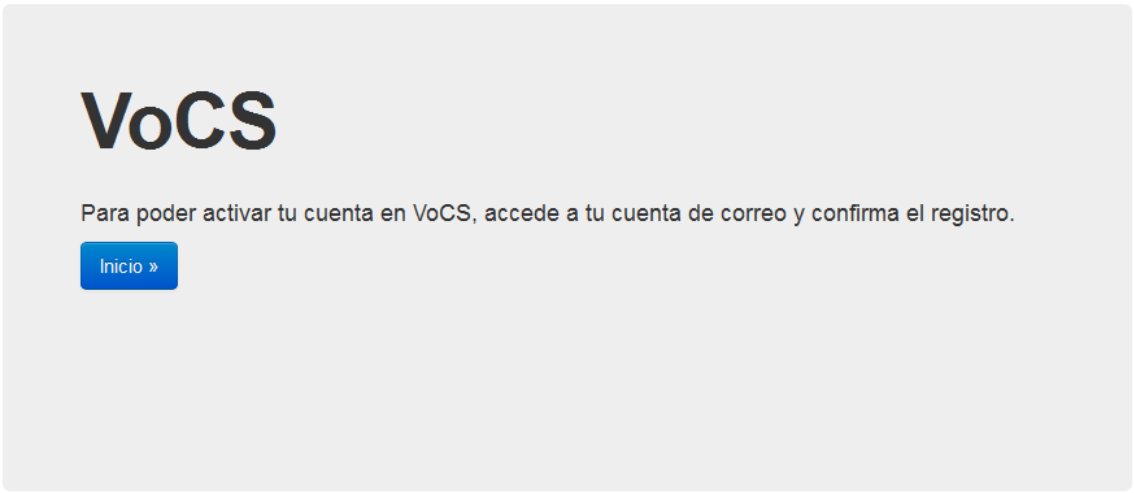


Figura 10.2: MU. Fin registro paso 1

El correo electrónico proveniente del equipo de VoCS, solicitará acceder a una URL para confirmar el registro. El contenido del correo electrónico es similar a este:

> Debe confirmar el registro, haga click en:
> 163.117.XXX.XXX/index/confirmar/0e4f1213e6398bb3788c1212a04f17d693912c3fa9cfc613036fc6d52e6e1a31

Si no es posible hacer clic en dicha URL, se copia y pega en la barra de direcciones del navegador.

Figura 10.3: MU. Cuenta activada

Una vez se ha accedido a la URL indicada en el correo electrónico, como se muestra en la Figura 10.3, la cuenta de usuario se activa y ya es posible acceder a la aplicación.

10.1.2 Iniciar sesión

Para acceder a la aplicación como usuario registrado se debe acceder al controlador “Index” y acción “Index”.

- https://url-de-la-aplicación/index/index

Figura 10.4: MU. Iniciar sesión

Una vez accedido, se verá una pantalla similar a la Figura 10.4, donde se debe introducir el nombre y la contraseña del usuario. Una vez introducidos los dos campos, se pulsa el botón “Entrar”.

10.1.3 Cerrar sesión

Para salir de la aplicación y cerrar la sesión de usuario abierta, se debe pulsar el enlace “cerrar sesión” que aparece en la esquina superior derecha de todas las páginas de los usuarios que han iniciado sesión. En la Figura 10.5 se muestra dicho enlace marcado con un círculo rojo.

Figura 10.5: MU. Cerrar sesión

10.1.4 Ver mi cuenta

Para poder acceder a la información de la cuenta de usuario se debe pulsar el enlace “Mi cuenta” del menú que aparece en las páginas de usuarios que han iniciado sesión. Los enlaces de acceso se pueden ver en la Figura 10.6 marcados con círculos rojos. Otra forma de acceder es accediendo a la URL:

- https://url-de-la-aplicación/usuario/micuenta

Una vez dentro, se mostrará una pantalla como se muestra en la Figura 10.6.

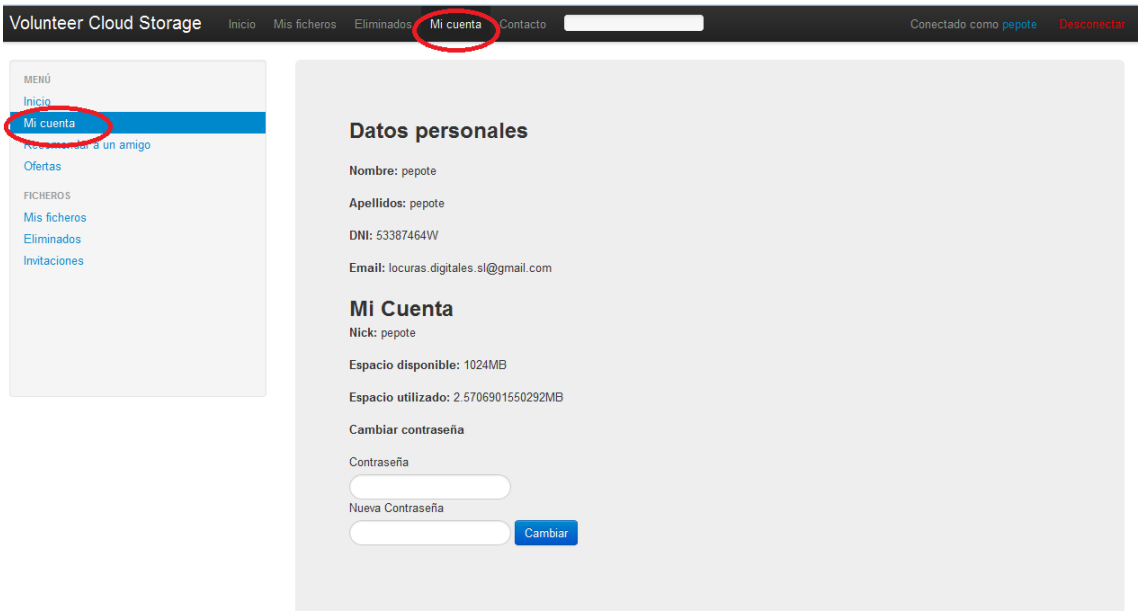


Figura 10.6: MU. Mi cuenta

10.1.5 Ver espacio disponible

Para ver el espacio disponible del usuario en el sistema, se puede observar desde “Mi cuenta” detallada en el apartado anterior o en el menú horizontal de todas las páginas de los usuarios que han iniciado sesión, como muestra la Figura 10.7.

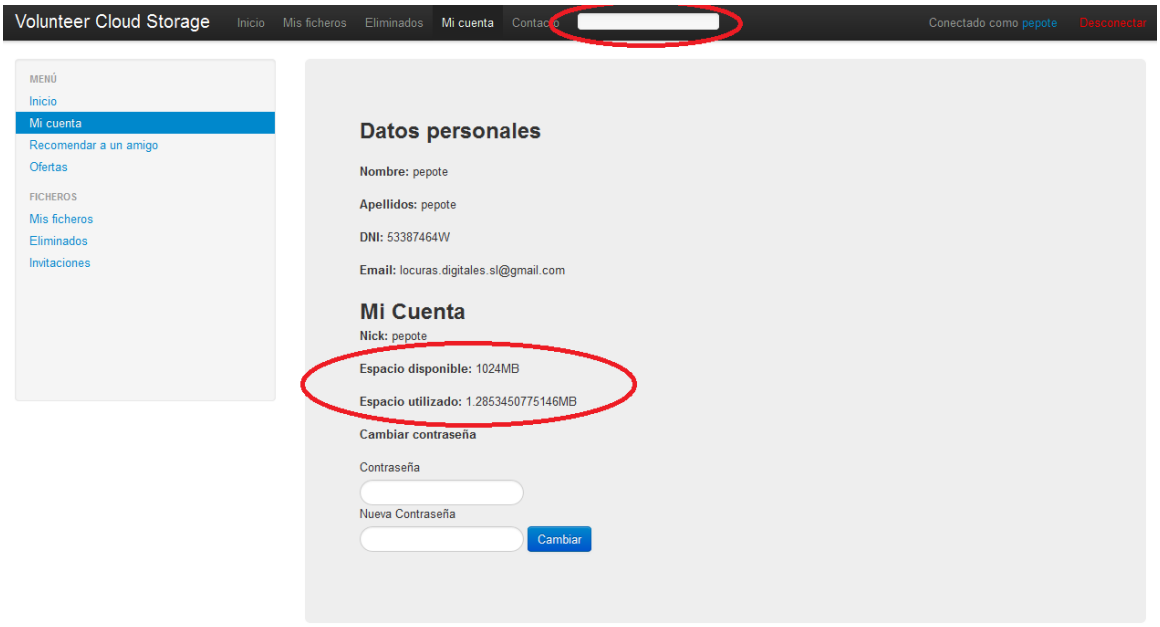


Figura 10.7: MU. Espacio disponible

10.1.6 Cambiar contraseña

Para cambiar la contraseña de usuario, es necesario acceder a “Mi cuenta” (el acceso a “Mi cuenta”, esta detallado en el apartado Ver mi cuenta). Una vez dentro, como muestra la Figura 10.8, en la parte baja de la página hay un formulario para cambiar de contraseña. Se debe insertar la contraseña actual del usuario y la nueva contraseña. Una vez introducidos se pulsa el botón cambiar.

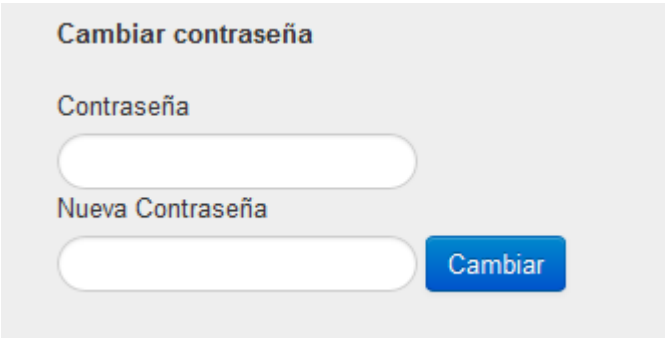


Figura 10.8: MU. Formulario cambiar contraseña

10.1.7 Recomendar a un amigo

Para poder recomendar a un amigo la aplicación, se debe acceder al controlador “Usuario” y a la acción “recomendar”, para ello se puede pulsar el enlace “Recomendar a un amigo” del menú que aparece en las páginas de usuarios que han iniciado sesión. Dicho enlace se puede ver en el menú vertical de la Figura 10.9. O acceder directamente a la URL:

- https://url-de-la-aplicación/usuario/recomendar

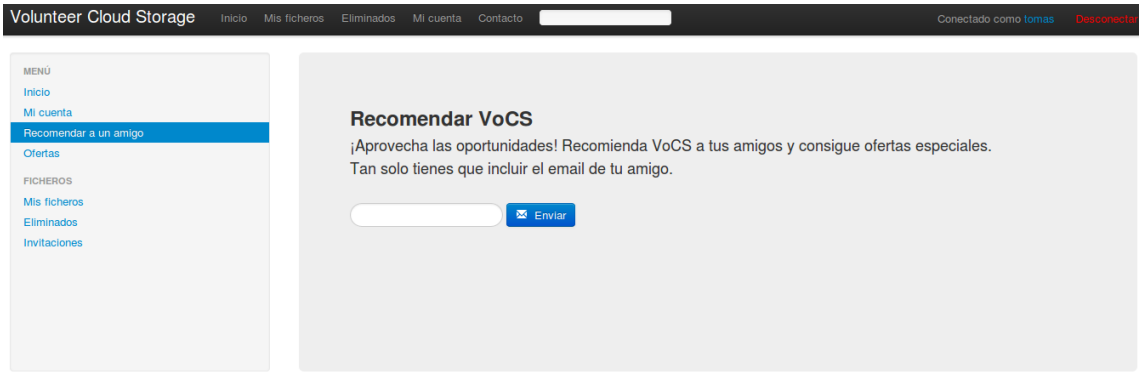


Figura 10.9: MU. Recomendar

Una vez dentro, se mostrará una pantalla como la que se muestra en la Figura 10.9, donde se debe incluir la dirección de correo electrónico de la persona a la que se quiere recomendar la aplicación. Una vez insertada la información se pulsa el botón “Enviar”.

10.1.8 Crear directorio

Para poder crear un directorio, se debe acceder al controlador “Ficheros” y a la acción “Misficheros”, para ello se puede pulsar el enlace “Mis ficheros” del menú vertical de todas las páginas de los usuarios que han iniciado sesión. El enlace se muestra en la Figura 10.10. También se puede acceder a través de la URL:

- https://url-de-la-aplicación/ficheros/misficheros

Una vez dentro, aparecerá una pantalla como muestra la Figura 10.10.

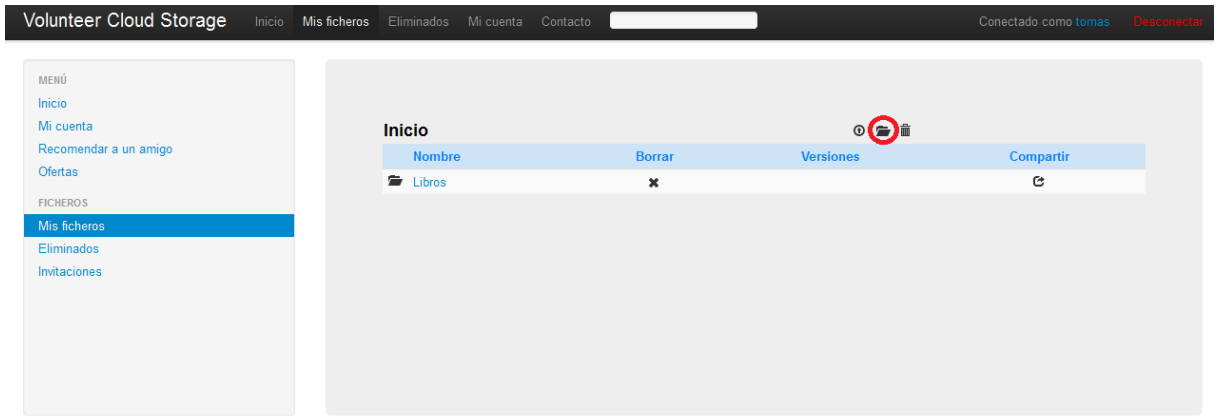


Figura 10.10: MU. Botón crear directorio

A continuación, se debe pulsar el botón señalado con un círculo rojo. Se abrirá un modal como se muestra en la Figura 10.11.

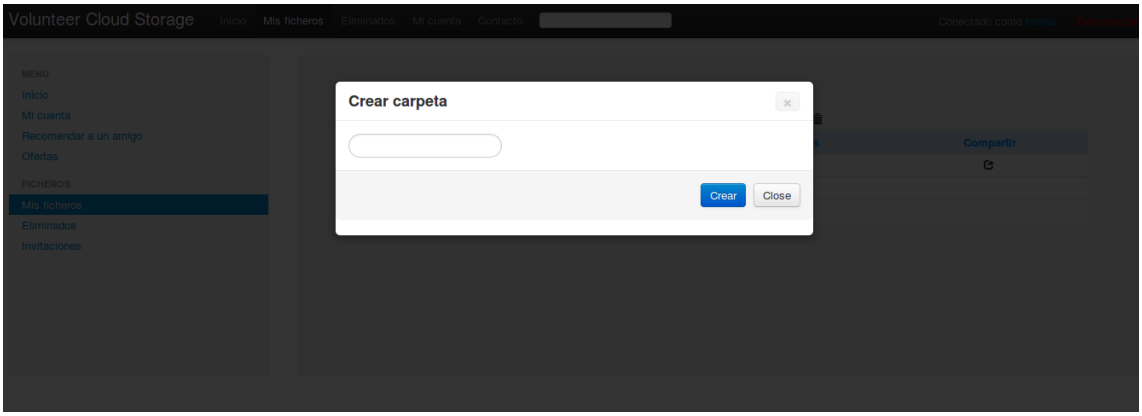


Figura 10.11: MU. Modal crear directorio

En el modal hay que introducir el nombre del directorio que se quiere crear y pulsar el botón “Crear”.

10.1.9 Eliminar directorio

Para eliminar un directorio hay que acceder a “Mis ficheros” desde el menú vertical. Se puede observar el menú en la Figura 10.12. También se puede acceder a través de la URL:

- https://url-de-la-aplicación/ficheros/misficheros

Una vez dentro, aparecerá una pantalla como muestra la Figura 10.12.

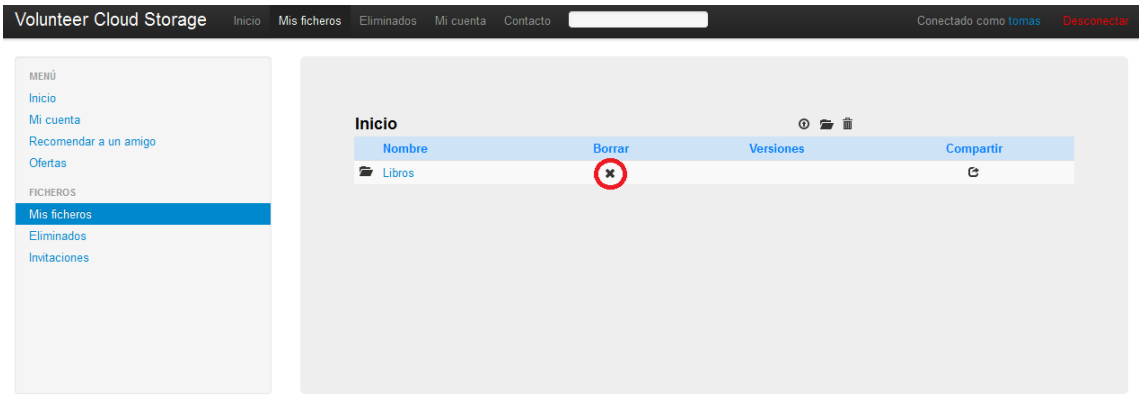


Figura 10.12: MU. Eliminar directorio

Como se ve en la figura, para eliminar el directorio “Libros” se ha de pulsar el botón señalado con un círculo rojo.

Acceder a un directorio

Para acceder a un directorio hay que situarse en la página “Mis ficheros”, esto se puede hacer desde el menú vertical. Se puede observar el menú en la Figura 10.13. También se puede acceder a través de la URL:

- https://url-de-la-aplicación/ficheros/misficheros

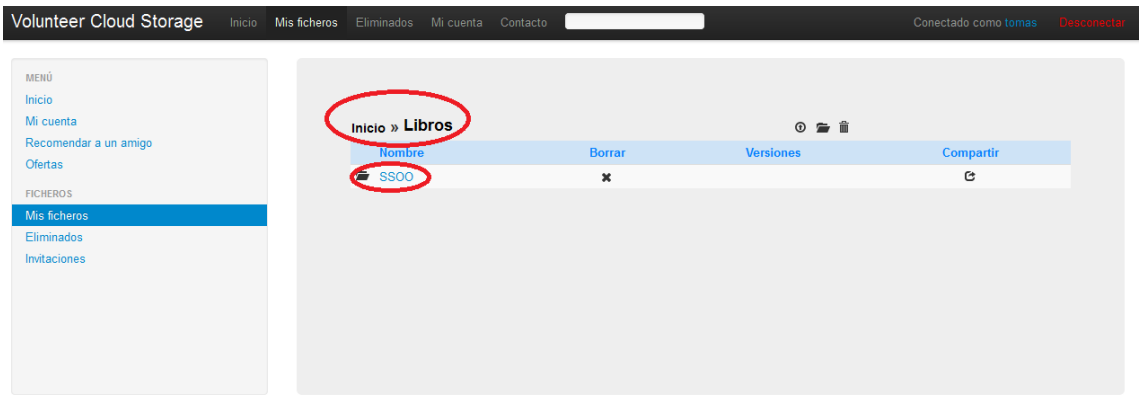


Figura 10.13: MU. Navegar

Una vez dentro, aparecerá una pantalla como muestra la Figura 10.13, los enlaces marcados con círculos rojos permiten navegar entre directorios, tanto a los padres del directorio actual como a sus hijos. Para acceder a uno de ellos, hay que pulsar los enlaces.

10.1.10 Invitar a compartir un directorio

Para invitar a otro usuario a compartir un directorio es necesario acceder a "Mis ficheros", esto se puede hacer desde el menú vertical. Se puede observar el menú en la Figura 10.14. También se puede acceder a través de la URL:

- https://url-de-la-aplicación/ficheros/misficheros

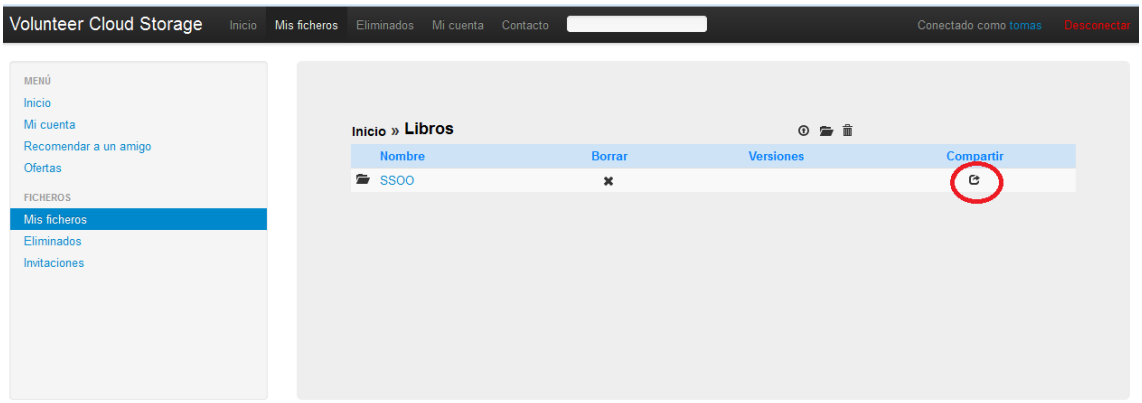


Figura 10.14: MU. Acceso invitar

Una vez dentro, aparecerá una pantalla como muestra la 10.14. Para invitar a compartir el directorio "SSOO", hay que pulsar el botón marcado con un círculo rojo. Una vez pulsado se abrirá un modal, como el que se muestra en la Figura 10.15.

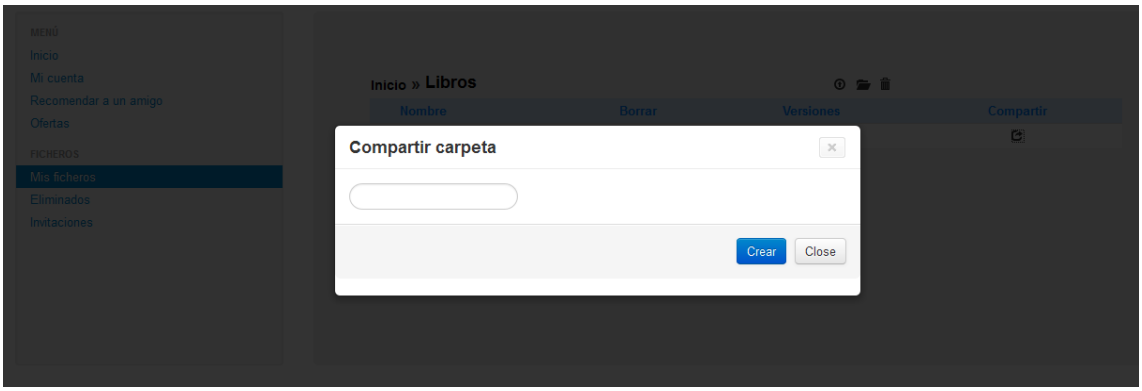


Figura 10.15: MU. Invitar

En el modal hay que introducir la dirección de correo electrónico del usuario al que se quiere invitar. Una vez introducidos los datos, se pulsa el botón "Crear".

10.1.11 Aceptar invitación

Para aceptar invitaciones a compartir, hay que acceder a “Mis invitaciones”, esto se puede hacer a través del menú vertical (se puede observar el menú en la Figura 10.18) o a través de la URL:

- https://url-de-la-aplicación/directorios/invitaciones

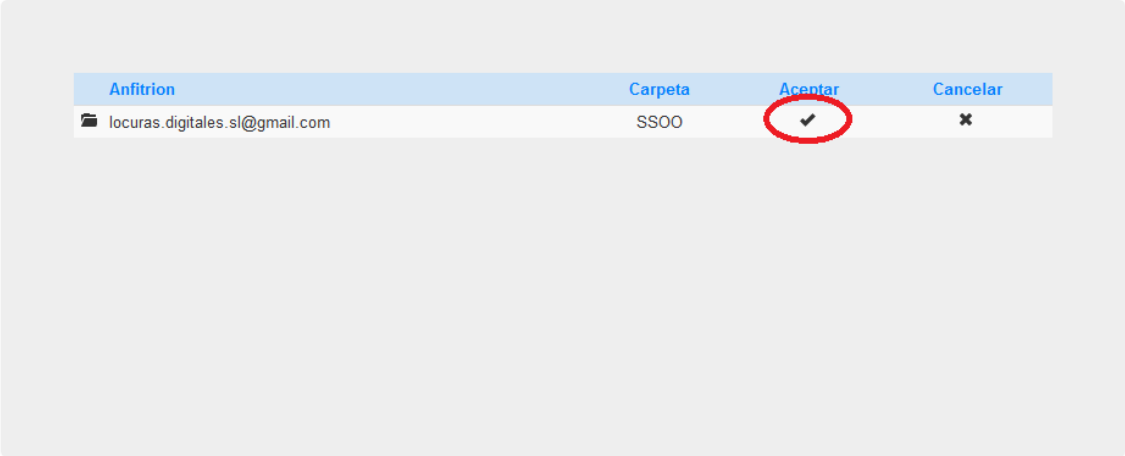


Figura 10.16: MU. Aceptar invitaciones

Para aceptar las invitaciones que aparecen en pantalla, se debe pulsar el botón aceptar que se puede observar en la Figura 10.16, marcado con un círculo rojo.

10.1.12 Cancelar invitación

Para cancelar invitaciones a compartir, hay que acceder a “Mis invitaciones”, esto se puede hacer a través del menú vertical. Se puede observar el menú en la Figura 10.13. También se puede acceder a través de la URL:

- https://url-de-la-aplicación/directorios/invitaciones

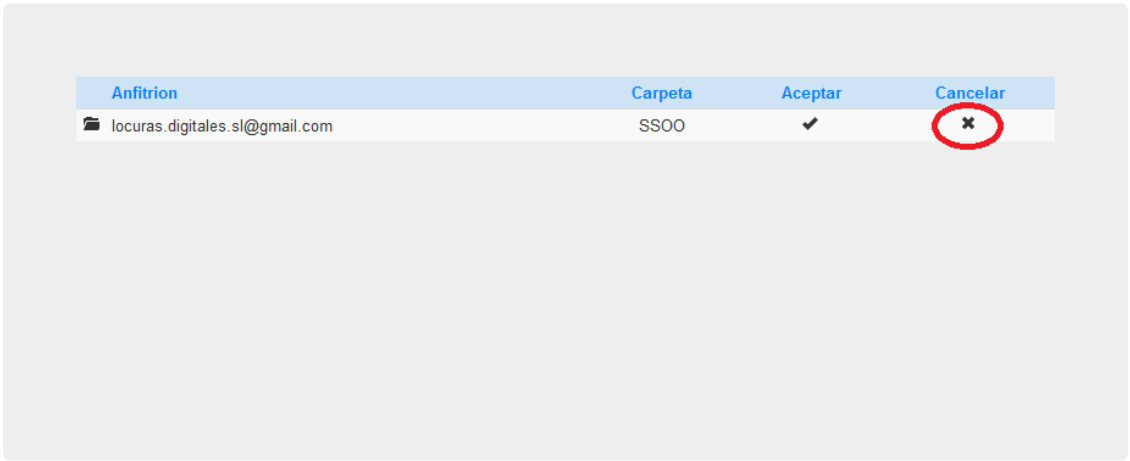


Figura 10.17: MU. Cancelar invitaciones

Para aceptar las invitaciones que aparecen en pantalla, se debe pulsar el botón cancelar. Esto se puede observar en la Figura 10.17, donde el botón está marcado con un círculo rojo.

10.1.13 Subir fichero

Para subir un fichero es necesario acceder a “Mis ficheros”, esto se puede hacer desde el menú vertical. Se puede observar el menú en la Figura 10.18. También se puede acceder a través de la URL:

- https://url-de-la-aplicación/ficheros/misficheros

Figura 10.18: MU. Subir fichero

Una vez dentro, aparecerá una pantalla como muestra la Figura 10.18. Para subir un fichero se ha de pulsar el botón marcado con un círculo rojo. Una vez pulsado se abrirá un modal, como el que se muestra en la Figura 10.19.

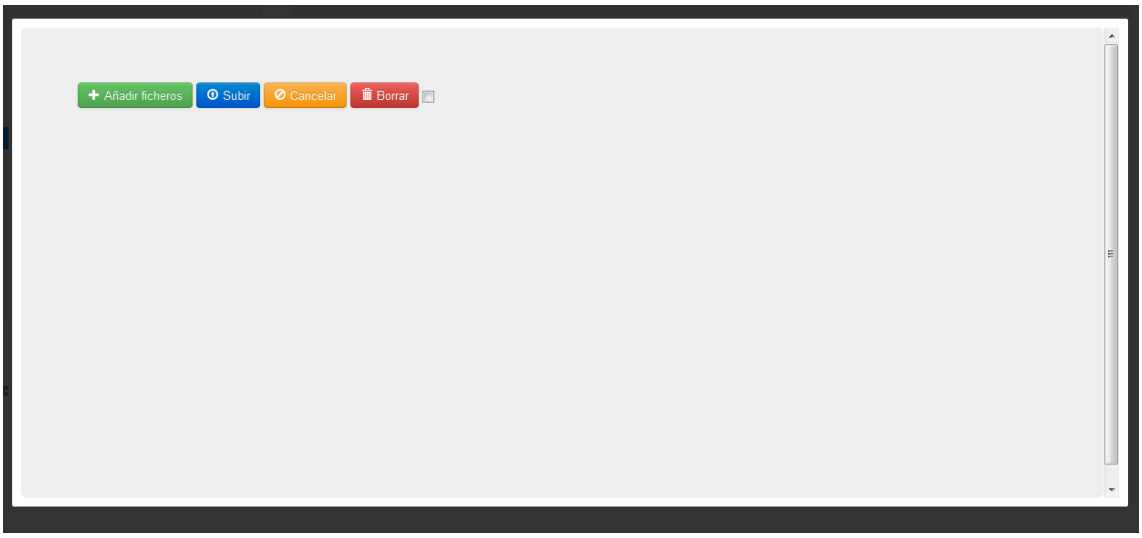


Figura 10.19: MU. Modal subir fichero

Dentro del modal, se dispone de un botón verde en el que se puede seleccionar ficheros para subir, un botón azul para subir los ficheros seleccionados, un botón amarillo para cancelar las selecciones y un botón rojo para eliminar los ficheros subidos.

10.1.14 Descargar ficheros

Para descargar un fichero es necesario acceder a "Mis ficheros", esto se puede hacer desde el menú vertical. Se puede observar el menú en la Figura 10.20. También se puede acceder a través de la URL:

- https://url-de-la-aplicación/ficheros/misficheros

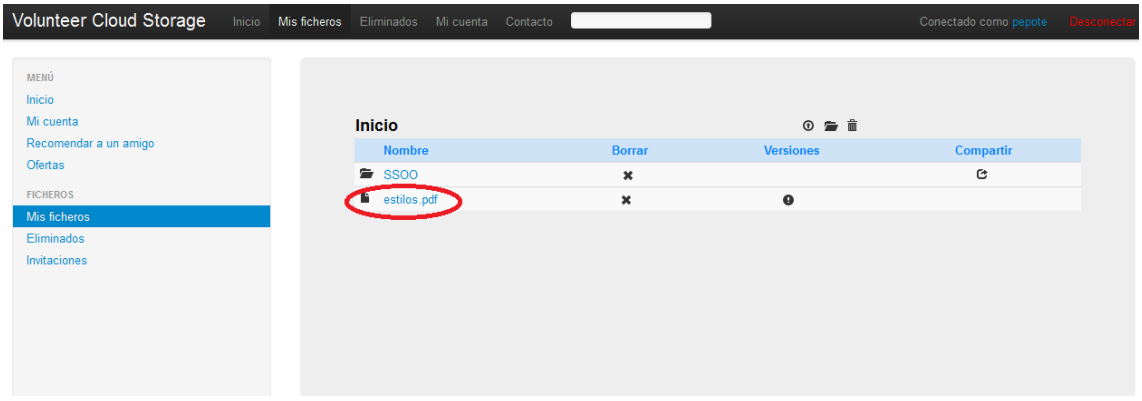


Figura 10.20: MU. Descargar fichero

Una vez dentro, aparecerá una pantalla como muestra la Figura 10.20. Para descargar un fichero hay que pulsar el enlace que hay en el nombre del fichero que se desea descargar. Un ejemplo se puede ver en la Figura 10.20 marcado con un círculo rojo.

También se puede descargar ficheros desde el modal de subir ficheros, explicado en el punto anterior.

Figura 10.21: MU. Descargar fichero en modal subir fichero

Como se ve en la Figura 10.21, se pueden descargar los ficheros recién subidos pulsando sobre los enlaces que hay en su nombre, un ejemplo de enlace de descarga se puede ver marcado con un círculo rojo.

10.1.15 Eliminar fichero

Para eliminar un fichero es necesario acceder a “Mis ficheros”, esto se puede hacer desde el menú vertical. Se puede observar el menú en la Figura 10.22. También se puede acceder a través de la URL:

- https://url-de-la-aplicación/ficheros/misficheros

Figura 10.22: MU. Eliminar fichero

Una vez dentro, aparecerá una pantalla como muestra la Figura 10.22. Para eliminar un fichero hay que pulsar sobre el botón borrar que se muestran marcados con un círculo rojo en la Figura 10.22.

10.1.16 Restaurar fichero eliminado

Para restaurar un fichero eliminado es necesario acceder a “Eliminados”, esto se puede hacer desde el menú vertical u horizontal. Se pueden observar los enlaces marcados con círculos rojos en la Figura 10.23. También se puede acceder o a través de la URL:

- https://url-de-la-aplicación/ficheros/vereliminados

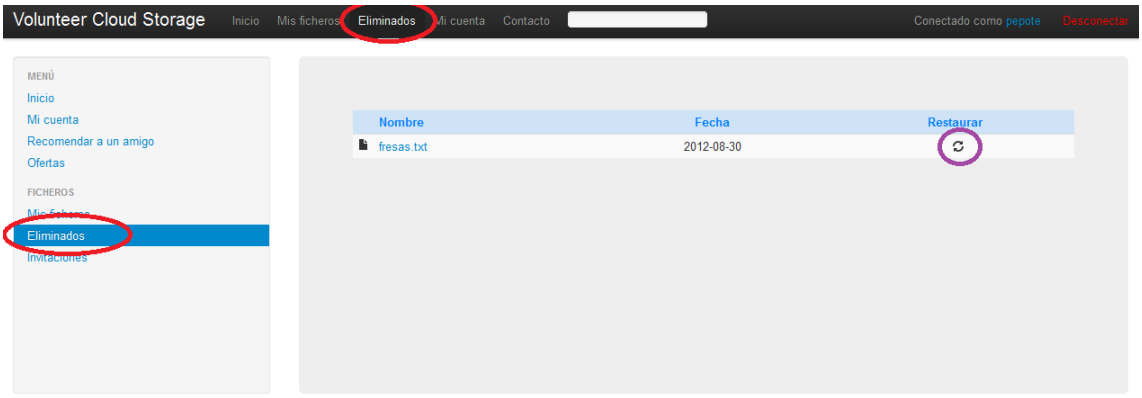


Figura 10.23: MU. Restaurar fichero

Una vez dentro, aparecerá una pantalla como muestra la Figura 10.23. Para restaurar un fichero hay que pulsar sobre el botón restaurar que se muestra marcado con un círculo morado en la Figura 10.23.

Restaurar una versión anterior

Para restaurar una versión de un fichero es necesario acceder a “Mis ficheros”, esto se puede hacer desde el menú vertical u horizontal. Se pueden observar los enlaces marcados con círculos rojos en la Figura 10.24. También se puede acceder a través de la URL:

- https://url-de-la-aplicación/ficheros/misficheros

Figura 10.24: MU. Versiones

Una vez dentro, para restaurar un fichero se debe pulsar el botón “versiones” situado al lado de cada fichero. Una vez pulsado se accede a la URL:

- https://url-de-la-aplicación/ficheros/verversiones

Figura 10.25: MU. Restaurar versiones

Aparecerá una pantalla con las versiones anteriores del fichero. Para restaurar una se debe pulsar el botón restaurar que hay al lado de cada versión. Se puede ver el botón en la Figura 10.25, marcada con un círculo rojo.

10.2 Anexo II: Manual de Instalación VoCS

En este apartado se va a explicar la instalación del sistema VoCS y los componentes necesarios para su correcto funcionamiento.

MySQL

Para instalar MySQL se ejecuta los siguientes comandos en la consola Linux:

```
$ sudo apt-get install  mysql-server

$ sudo apt-get install  mysql-client
```

Apache

Para instalar el servidor web Apache se tiene que ejecutar el siguiente comando:

```
$sudo apt-get install apache2
```

PHP

Para instalar PHP se ejecuta la siguiente línea de comando:

```
$sudo apt-get install php5
```

También se ha de ejecutar el módulo de MySQL para PHP con el siguiente comando:

```
$sudo apt-get install php5-mysql
```

Zend Framework

Para instalar Zend Framework se siguen los siguientes pasos:

1- Agregar el repositorio

```
$ sudo add-apt-repository ppa:zend-framework/ppa
```

2- Actualizar la lista de paquetes

```
$ sudo apt-get update
```

3- Instalar Zend Framework

```
$ sudo apt-get install zend-framework
```

Instalación de VoCS

Una vez instalados los componentes se despliega el proyecto en el servidor Web Apache. Los pasos a seguir son:

1- Copiar el fichero codigo.rar que se puede encontrar en la carpeta “src” del CD.

2- Pegar el fichero en el directorio “/var/www/”.

3- Descomprimir el fichero.

10.3 Anexo III: Presupuesto

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1. Autor:
   Ignacio Nicolás Schiavón Raineri
2. Departamento:
   Informática
3. Descripción del Proyecto:
   - Título **VoCS: Sistema de almacenamiento voluntario en la nube**
   - Duración (meses) 3
   - Tasa de costes Indirectos: 20%
4. Presupuesto total del Proyecto (valores en Euros):
   11.388,00 Euros

5. Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar solo a título informativo)	Categoría	Dedicación (hombres mes) [a)]	Coste hombre mes	Coste (Euro)
Schiavón Raineri, Ignacio Nicolás		Analista	1	3.000,00	3.000,00
Schiavón Raineri, Ignacio Nicolás		Diseñador	1	2.400,00	2.400,00
Schiavón Raineri, Ignacio Nicolás		Programador	1	1.400,00	1.400,00
Schiavón Raineri, Ignacio Nicolás		Control Calidad	1	1.400,00	1.400,00
	Hombres mes 3			**Total**	8.200,00

[a)] 1 Hombre mes = 131.25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8.8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d)]
Ordenador Intel(R) Core™ i3	500,00	100	3	60	25,00
Pentium 4	150,00	100	3	60	7,50
AMD Athlon 64	150,00	100	3	60	7,50
				Total	40,00

[d)] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} xCxD$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)

C= coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO [e)]

Descripción	Empresa	Coste imputable
Internet	Movistar	120,00
Microsoft Office 2007	Microsoft	130,00
Luz	Iberdrola	200,00
Dietas	Otros	500,00
	Total	950,00

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

6. Resumen de costes

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	8.200
Amortización	40
Subcontratación de tareas	0
Costes de funcionamiento	950
Costes Indirectos	1.838
Total	11.028

10.4 Anexo IV: Planificación Inicial

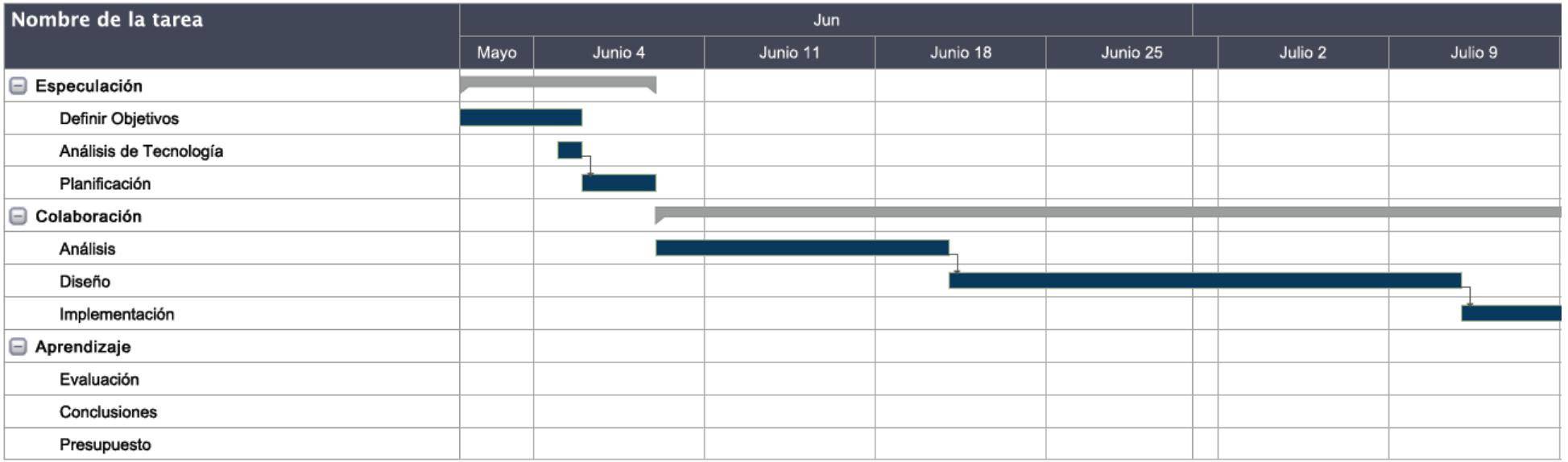


Figura 10.26: Planificación Inicial (1)

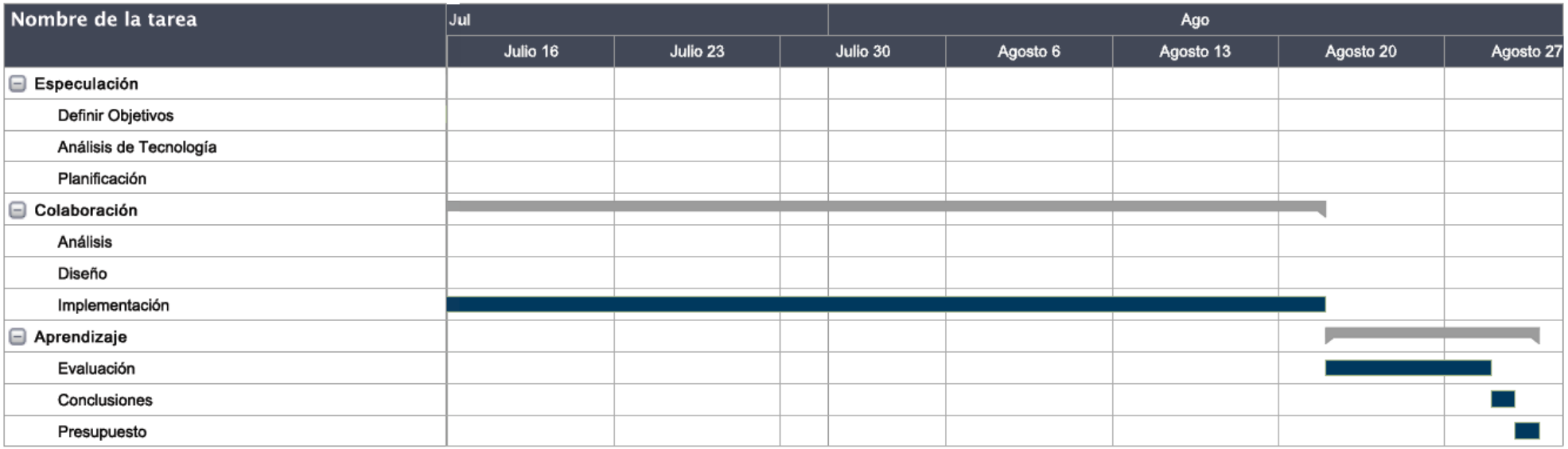


Figura 10.27: Planificación Inicial (2)

10.5 Anexo V: Planificación Final

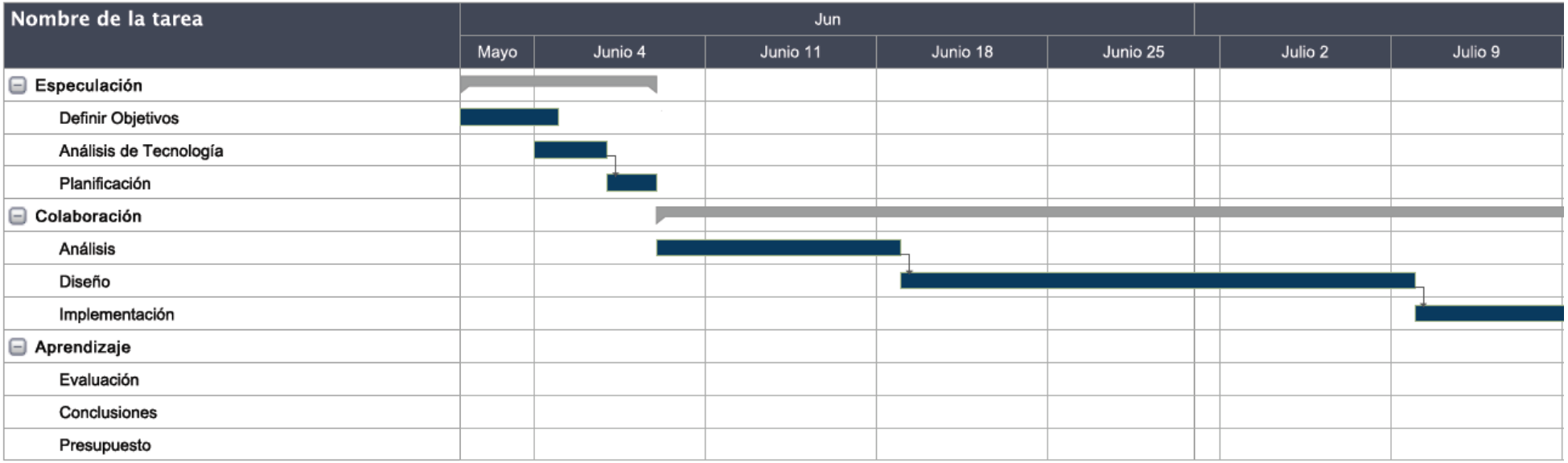


Figura 10.28: Planificación Final (1)

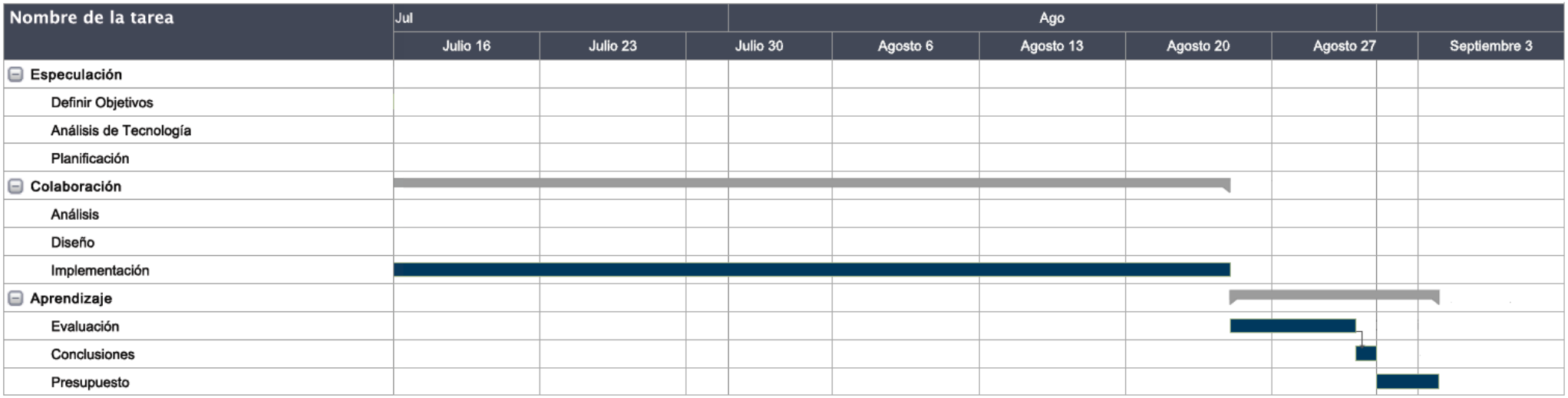


Figura 10.29: Planificación Final (2)