Departamento de informática

Universidad
Carlos III de Madrid

Trabajo de Fin de Grado

VoCSDroid

Diseño de un Cliente para Dispositivos Moviles para un Sistema de Almacenamiento Voluntario en la Nube

Autor: Nicolás Cordero Fernández
Tutor: Luis Miguel Sánchez García

Agradecimientos

En primer lugar quiero agradecer a mi tutor, Luis Miguel, por permitirme escoger un proyecto suyo, ayudándome en todo lo que he necesitado y dedicándome parte de su tiempo.

Agradezco a mi familia por haber estado apoyando tanto en los momentos fáciles y en los momentos difíciles.

Agradezco a mi compañero Nicolás el apoyo recibido y la exigencia en el trabajo.

Por último, agradecer a mis amigos el apoyo recibido, desde Las Rozas hasta Monsalupe.

Resumen

VoCSDroid es una aplicación para el sistema operativo Android que permite almacenar y descargar ficheros en la nube de una manera sencilla, segura y óptima.

Cada vez es más frecuente que las personas tengan uno o varios dispositivos móviles como Smartphones y Tablets. Esto produce que las personas tengan acceso a internet desde distintos aparatos electrónicos como PC, Smartphones, etc.

Gracias a que las personas están conectadas desde distintos aparatos electrónicos surge la necesidad de la computación en la nube. Actualmente la computación en la nube esta en auge, lo que provoca que empresas y otros organismos se fijen en ella para proporcionar determinados servicios y basar su modelo de negocio.

El sistema aquí presentado proporciona un servicio de almacenamiento en la nube sencillo, rápido y seguro, que permite subir, descargar, eliminar y restaurar ficheros; crear, eliminar y compartir directorios; un sistema de ficheros distribuido óptimo, escalable y que no requiera muchos recursos; y un sistema de autenticación sencillo para los dispositivos móviles.

Este sistema es un prototipo que esta destinado a futuros proyectos.

Palabras clave: VoCS: Sistema de almacenamiento voluntario en la nube.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

En este capítulo se describen los objetivos del Trabajo de Fin de Grado y se muestra la estructura que va a seguir el documento.

1.1 Definiciones y Acrónimos

API: Aplication Programming Interface. Es una especificación destinada a ser utilizada como una interfaz de componentes de software para comunicarse entre sí. Puede tomar varias formas, desde una biblioteca de un leguaje de programación hasta un Estándar Internacional como POSIX.

Acrónimos:

VoCS: Voluntieer Cloud Storage.
OHA: Open Handset Alliance.
SO: Sistema Operativo
JIT: Pendiente de mirar
NFC: Near Field Communication
FPS: Frames per seconds.
GPU: Graphics Processing Unit
SDK: Software Delevopment Kit
NIC: Network Interface Card
OSI: Open System Interconnection
UDP: User Datagram Protocol
TCP: Transmission Control Protocol
RPC: Remote Procedure Calls
RMI: Remote Method Invocation.
HTTP: HyperText Transfer Protocol
XML: Extensible Markup Languaje
JSON: Java Script Object Notation
SOAP: Simple Object Access Protocol

TFG: Trabajo de Fin de Grado
WSDL: Web Services Description Language
SSL: Secure Socket Layer
TLS: Transport Layer Security
MAC: Message Authentication Code
HMAC: Hash-based Message Authentication Code
RU: Requisito de Usuario.
RS: Requisito de Software

Definiciones:

API: es el conjunto de métodos, funciones y procedimientos que ofrece una biblioteca para ser utilizada por otro software como una capa de abstracción.

Datagrama: Pequeños paquetes que se utilizan en la conmutación de paquetes para transportar los datos en servicios no orientados a conexión. Esta compuesto por una cabecera y los datos.

GET: Comando usado en HTTP para solicitar un recurso ubicado en la URL especificada.

HEAD: Comando que solicita el encabezado del recurso ubicado en la URL especificada

POST: Comando que envía datos al programa ubicado en la URL especificada.

PUT: Comando que envía datos a la URL especificada

DELETE: Comando que borra el recurso ubicado en la URL especificada.

1.2 Motivación

Cada vez es más frecuente que las personas tengan dispositivos móviles con acceso a internet. Se estima que en el año 2012 se venderán más Smartphones que ordenadores portátiles. Hoy en día el uso de Smartphones esta generalizado y la mayoría de las personas tiene uno o varios. Además de los Smartphones se pueden encontrar otros dispositivos con acceso a internet como Tablets, marcos digitales, discos duros externos etc.

Que cada vez haya más personas y dispositivos conectados a internet impulsa a pensar en los servicios ofrecidos en la nube. Esta tecnología esta apareciendo con mucha fuerza ofreciendo servicios muy interesantes como el almacenamiento, el correo electrónico e incluso están apareciendo los primeros Sistemas Operativos en la nube.

De estos pensamientos surge la idea del Proyecto de Fin de Grado llamado VoCS: Voluntieer Cloud Storage, que se trata de un cloud propio que ofrece un servicio de almacenamiento fácil y libre. Este proyecto ofrece la posibilidad de replicar los datos de los usuarios entre los diferentes servidores que compondrán la estructura, la posibilidad de seleccionar entre varios niveles de seguridad a la hora de subir archivos al cloud, un servicio web para acceder al desde cualquier navegador, un API para comunicar al servidor con dispositivos móviles, una aplicación para dispositivos con Sistema Operativo Android y conexiones seguras entre el usuario y el cloud.

1.3 Marco General

En los últimos años la computación en la nube esta experimentando un crecimiento acelerado y en el mercado global competitivo de hoy, las empresas deben innovar y aprovechar al máximo sus recursos. Los servicios en la nube pueden proporcionar un enorme valor para las empresas de cualquier tamaño.

No solo en las empresas esta provocando repercusión, también esta provocando un impacto social ya que con los servicios que proporciona se esta convirtiendo en una referencia del cambio social en las últimas décadas. Con servicios como las redes sociales, almacenamiento virtual desde gestores de correos hasta contenido multimedia, acceso a servicios de contenidos digitales como la música etc.

La computación en la nube proporciona una disminución de costes para grandes y medianas empresas, opciones de almacenamiento escalable, ya que en cualquier momento estos recursos ofrecidos pueden aumentar sin la necesidad de comprar otro tipo de hardware. Las actualizaciones de los sistemas son automáticas por lo que el departamento de IT no tiene que preocuparse por el pago de actualizaciones futuras, acceso remoto a la información por parte de los usuarios en cualquier lugar y cuando lo necesiten, no se necesita instalar ningún tipo de hardware para usarlo, soporta una alta disponibilidad al estar en la web, accesibilidad por diferentes tecnologías como Smartphones, PDAs, Netbooks, Tablets; y por último los centros de cloud computing utilizan menos energía que los centros de datos importantes por lo que contribuye en la ecología.

Pero esta tecnología también tiene desventajas como la privacidad y la seguridad ya que los usuarios que quieran adquirir los servicios de la computación en la nube tienen que tener total confianza en sus proveedores, ya que estos pueden almacenar datos y archivos personales o corporativos y al no tener el control de la infraestructura nunca se puede saber si sus datos y ficheros estan en posesión de un tercero, por eso necesitan hacerlo con total confianza. La

computación en la nube depende de la disponibilidad de acceso a internet por que si se necesita información y no se dispone de acceso a internet no podrá acceder a ella.

Hoy en día se pueden encontrar diversas tecnologías de computación en la nube como Google Apps que brinda servicio de aplicaciones como Gmail, Google Talk, Google Calendar etc. ICloud y Dropbox que son sistemas de almacenamiento en la nube. Microsoft Azure que ofrece servicios de almacenamiento, de mercado online donde vender y comprar aplicaciones etc.

1.4 Objetivo

El objetivo de este proyecto es **el análisis, diseño e implementación de un sistema multiplataforma para el sistema de almacenamiento en la nube VoCS mediante el uso de tecnologías y estándares Web.** Durante el transcurso de construcción del software se creará el presente documento donde se muestra en detalle dicho proceso, que abarca desde la planificación inicial hasta las pruebas.

Esta aplicación permite hacer uso de todas las funciones de VoCS (subir fichero, descargar fichero, crear directorio, eliminarfichero/directorio, restaurar fichero eliminado, etc.). El API REST proporcionará servicio a cualquier dispositivo que lo quiera solicitar, este trabajará sobre HTTPS para garantizar que la conexión entre usuario y dispositivo es segura. Para añadir un nivel de seguridad más a la comunicación el servidor creará token de sesión para cada conexión de login y se implantará un sistema de ficheros escalable acorde con el servidor.

También se persiguen otros objetivos en este proyecto:

- Montaje y preparación de servidores para su correcto funcionamiento.
- Análisis de un sistema multiplataforma para el sistema de almacenamiento en la nube VoCS,
- Análisis de un sistema de comunicaciones REST desde el punto de vista de el cliente como del servidor.
- Diseño de un sistema multiplataforma para el sistema de almacenamiento en la nube VoCS.
- Diseño de un sistema de comunicaciones REST desde el punto de vista de el cliente como del servidor.
- Implementación de un prototipo Android que cumpla con el análisis y diseño de la aplicación.
- Implementación de una API REST.
- Integración de sistemas de seguridad en el sistema de comunicaciones establecido.
- Desarrollo de una aplicación para el Sistema Operativo Android.
- Implementación de un sistema de ficheros que intente ser escalable y que optimice los recursos del dispositivo móvil.
- Evaluación de las operaciones subida y descarga de ficheros desde el dispositivo móvil.

1.5 Estructura del documento

La estructura del documento es la siguiente:

- Capitulo 2 narra el estado de la cuestión, donde se da un repaso al contexto actual de las comunicaciones cliente servidor (Sockets, llamadas a procedimientos

remotos, Servicios Web …) y sistemas operativos móviles, explicando aquellos más utilizados (Android, iOS, Windows Phone…).

- El capitulo 3 explica el marco legal que implica este proyecto.
- El capitulo 4 explica la metodología utilizada.
- En el capitulo 5 se detallará el análisis y diseño del sistema.
- El capitulo 6 explica la implementación del sistema.
- En el capitulo 7 se realizará un análisis del rendimiento del sistema.
- En el capitulo 8 se expondrán las conclusiones, los trabajos futuros, la planificación final y el presupuesto.
- En el capitulo 9 se realizará la bibliografía utilizada.
- En el capitulo 10 se exponen una serie de anexos, como el manual de usuario y el manual de implantación.

2 Estado de la Cuestión

En este apartado se expondrán los Sistemas Operativos móviles de la actualidad más utilizados como son Android, iOS, Windows Phone, Symbian OS. A continuación se explicará el modelo cliente servidor que es la arquitectura más utilizada. Los siguientes puntos adquieren un contexto histórico que explica las tecnologías más importantes y como estas evolucionaron en el ámbito de las comunicaciones distribuidas. El apartado termina con un repaso en la seguridad de las comunicaciones explicando SSL y TLS; y con una comparación entre las tecnologías utilizadas en las comunicaciones distribuidas.

2.1 Sistemas Operativos Móviles

A continuación se expondrán los sistemas operativos de móviles más extendidos en la actualidad.

2.1.1 Android

Android una variante Linux orientada a dispositivos móviles como Smartphones, Tablets, Google TV etc. Inicialmente fue desarrollado por Android Inc. Empresa comprada por Google en 2005, situada en Palo Alto, California. Fundada por Andy Rubin ahora vicepresidente de ingeniería de Google.

El 5 de Noviembre de 2007 se funda la OHA liderada por Google, es una alianza de 84 compañías que se dedica a desarrollar estándares abiertos para dispositivos móviles. Alguno de los miembros más importantes son: Google, HTC, Samsung, Dell, Intel, Motorola, Qualcomm, Nvidia, T-Mobile, Texas Instruments y Toshiba. Ese mismo día también se presento Android una plataforma de código abierto para móviles.

El 12 de Noviembre de 2007 se presentó el kit de desarrollo para que los desarrolladores comenzasen a trabajar con Android. [1]

2.1.1.1 Características de Android

La primera versión de Android fue Apple Pie o Android 1.0. Esta versión fue liberada el 23 de Septiembre de 2008, en el HTC G1. La interfaz era básica y carecía de características como el uso de la cámara, las nuevas versiones de Bluetooth, Flash y soporte para HTML. Pero incluía Widgets y el Market de Android o más conocido ahora como Google Play.

La segunda versión de Android fue Banana Bread o Android 1.1 fue lanzada el 9 de Febrero de 2009 lanzada para el HTC Dream y cambiaron su API además de añadir nuevas funciones como ocultar y mostrar el teclado, guardar archivos adjuntos en mensajes y mostrar detalles y comentarios disponibles cuando un usuario busca empresas en Maps.

Android 1.5 o Cupcake, versión liberada el 30 de Abril de 2009 basada en el kernel de Linux 2.6.27 e incorporaba las siguientes funcionalidades: posibilidad de grabar y reproducir videos a través de l modo camcorder, capacidad de subir videos a Youtube e imágenes a picasa

directamente desde el teléfono, nuevo teclado con predicción de texto y soporte para Bluetooth A2DP y AVRCP.

Figura 2.1: Android Cupcake.

La versión 1.6 o Donut, versión liberada el 15 de septiembre de 2009 basada en el kernel de Linux 2.6.29. Esta versión incluía mejoras como: la posibilidad de correr el sistema operativo en múltiples resoluciones de pantalla, mejora en el Android Market y mejora de la búsqueda de la pantalla de inicio que permite buscar marcadores, historiales, contactos y páginas web.

Figura 2.2: Android Donut.

La versión 2.0/2.1 Eclair fue liberado el 26 de octubre de 2009 basado en el kernel de Linux 2.6.29. Esta versión incluía cambios como: velocidad de hardware optimizada, soporte para más tamaños de pantalla, soporte de flash para la cámara, poder añadir varias cuentas de correo electrónico en el mismo dispositivo, soporte para Microsoft Exchange, Bluetooth 2.1, Motion Event mejorado para captar mejor eventos multi-touch, soporte para HTML5, teclado QWERTY y sistema de navegación de automóviles mediante Google Maps, con guía de voz e información de trafico de forma gratuita.

Figura 2.3: Android Eclar.

La versión 2.2 o Froyo liberado el 20 de mayo de 2010, basado en el kernel de Linux 2.6.32. Esta versión incluyó los siguientes cambios: soporte para Adobe Flash 10.1, optimización general del SO, memoria y rendimiento, mejora en la velocidad de las aplicaciones gracias a la implementación de JIT, permite desactivar tráfico de datos a través de la red del operador e integración del motor de JavaScript V8 de Google Chrome en el navegador del teléfono.

Figura 2.4: Android Froyo.

La versión Android 2.3.x o Gingerbread fue liberado el 6 de diciembre de 2010 y esta basado en el kernel de Linux 2.6.25.7. Esta versión presentaba los siguientes cambios: actualización del diseño de la interfaz de usuario, soporte para pantallas de grandes resoluciones, soporte para telefonía VoIP, reproducción de videos WebM/VP8 y decodificación de audio AAC, soporte para NFC, teclado-multi táctil rediseñado y soporte para más sensores.

Figura 2.5: Android Gingerbread.

La versión 3.0/3.1/3.2 o Honeycomb fue liberada el 22 de Febrero de 2011 estaba basado en el kernel de Linux 2.6.36. Esta versión estaba diseñada para Tablets, esta disponía de un Escritorio 3D con widgets rediseñados, sistema multitarea mejorado, soporte para videochat mediante Google Talk, soporte a gran variedad de periféricos y conexiones USB como teclados, ratones etc. Mejor soporte para redes Wi-Fi.

Figura 2.6: Android Honeycomb.

La versión Android 4.0 o Ice Cream Sandwitch basada en el kernel de Linux 3.0.1 liberada el 19 de octubre de 2011. Esta versión unifica todos los dispositivos Android del mercado, por lo que los Smartphones, Tabletas, Televisores etc. compartían la misma versión de Android. Presenta cambios como multitarea mejorada parecida a la de la versión de Honewcomb, la interfaz aumenta su rapidez y respuesta de forma notable al ser dibujada por la GPU.

Figura 2.7: Android Ice Cream Sandwitch.

La versión 4.1 o Jelly Bean fue liberada el 27 de junio de 2012 y esta basada en el kernel de Linux 3.1.10. Esta versión introduce una mejora fundamental llamada Proyect Butter que mejora la funcionalidad y el rendimiento del sistema operativo. El proyect Butter permite a la CPU y a la GPU correr en paralelo sin que estas choquen, para ello introduce una anticipación al tacto haciendo que toda la interfaz corra a 60 fps y los gráficos son triple buffer. Esta versión incluye una búsqueda por voz más completa, una nueva función llamada Google Now y Google Chrome pasará a ser el navegador por defecto.

Figura 2.8: Android Jelly Bean.

A continuación se puede ver la cuota de Mercado de las versiones de Android.

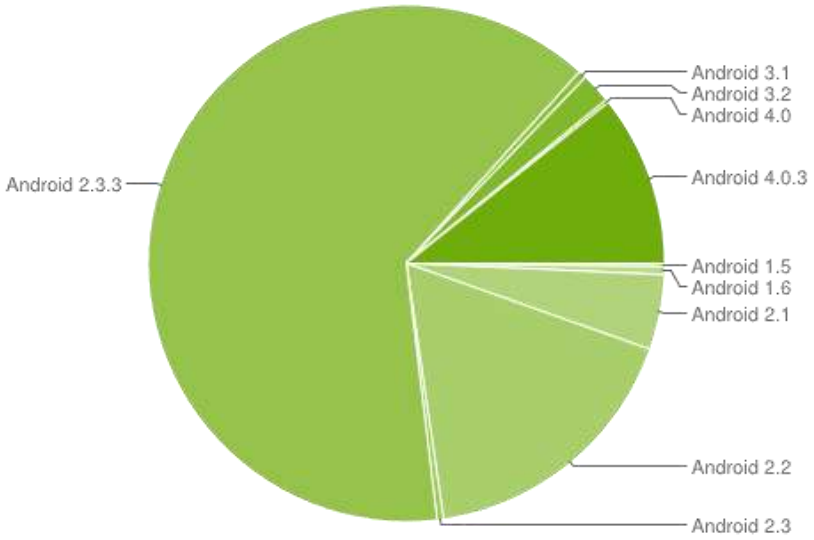


Figura 2.9: Distribución de la cuota de mercado entre las diferentes versiones. [2]

Versión	Nombre	API Level	Distribución
1.5	Cupcake	3	0.2%
1.6	Donut	4	0.5%
2.1	Eclair	7	4.7%
2.2	Froyo	8	17.3%
2.3/2.3.2	Gingerbread	9	0.4%
2.3.3/2.3.7	Gingerbread	10	63.6%
3.1	HoneyComb	12	0.5%
3.2	Honeycomb	13	1.9%
4.0/4.0.2	Ice Cream Sandwitch	14	0.2%
4.0.3/4.0.4	Ice Cream Sandwitch	15	10.7%

Tabla 2.1: Cuota de las versiones de Android en 2 Julio de 2012. [3]

2.1.1.2 Arquitectura Android

Como se puede observar en la Figura 2, la arquitectura Android se compone de 5 capas [1]:

Kernel de Linux: Como se ha dicho antes, el sistema operativo Android esta basado en el Kernel de Linux que usa como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes. Siempre que un fabricante incluye un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde Android es crear las librerías de control o drivers necesarios dentro de este kernel de Linux embebido en el propio Android.

El kernel se encarga de gestionar los recursos del teléfono como el control de energía, memoria etc. Y también se encarga de gestionar los recursos del sistema operativo como la ejecución de procesos, elementos de comunicación etc.

Entorno de Ejecución: Esta situado en la capa superior al kernel de Linux junto con las Bibliotecas de Android. Es un set de bibliotecas esenciales de Android, que incluyen la mayoría de la funcionalidad de las bibliotecas habituales de Java así como otras específicas de Android. El componente principal es la máquina virtual Dalvik que ejecuta todas las aplicaciones no nativas de Android. Las aplicaciones Android se compilan una única vez en un formato especial para la maquina virtual Dalvik y esta las ejecuta teniendo total garantía de que podrán ejecutarse en cualquier dispositivo Android que disponga de la versión mínima del sistema operativo que requiera cada aplicación. Los archivos generados .dex ocupan un 50% menos que los archivos .class generados por java optimizando espacio y el proceso de carga.

Bibliotecas: Se sitúan al mismo nivel que el entorno de ejecución y están escritas en C/C++ y son compiladas para la arquitectura específica del teléfono. Su cometido es proporcionar funcionalidad a las aplicaciones para las tareas que se repiten con frecuencia evitando tener que codificarlas cada vez que se requiera. Algunas de las bibliotecas que se incluyen habitualmente son:

- Libc: Incluye todas las cabeceras y funciones según el estándar C.
- Surface Manager: Esta se encarga de componer los diferentes elementos de navegación de pantalla a partir de capas graficas 2D y 3D. Cada vez que la aplicación pretende dibujar algo en pantalla realiza cambios en las imágenes que almacena en memoria y que después combina para crear la imagen final que se envía a la pantalla, esto permite superposición de elementos, transparencias, animaciones, transacciones etc.
- SGL: desarrollada por Skia y se encarga de representar elementos en dos dimensiones. Es el motor grafico de 2D de Android.
- OpenGL | ES: motor gráfico 3D basado en las APIs de OpenGL ES 1.0, 1.1 y 2.0. Utiliza aceleración hardware o un motor software altamente optimizado cuando no la hay.
- Bibliotecas multimedia: basadas en OpenCORE, proporcionan los códec necesarios para el contenido multimedia permitiendo permiten visualizar, reproducir e incluso grabar numerosos formatos de imagen, vídeo y audio como JPG, GIF, PNG MP3 etc.
- WebKit: motor web utilizado por el navegador. Es el mismo motor que utilizan Safari y Google Chrome.
- SSL (Secure Sockets Layer): proporciona seguridad al acceder a Internet por medio de criptografía.
- FreeType: permite mostrar fuertes tipográficas, tanto basadas en mapas de bits como vectores.
- SQLite: motor de bases de datos relacionales disponibles para todas las aplicaciones.

Framework de Aplicaciones: Representa el conjunto de herramientas de desarrollo de cualquier aplicación. La mayoría de estos componentes son bibliotecas Java que acceden a recursos desde la maquina Dalvik. Toda aplicación que se desarrolle para Android utiliza el mismo conjunto de API y el mismo framework. Las más importantes son las siguientes:

- Administrador de Actividades (Activity Manager): se encarga de gestionar el ciclo de vida de las actividades y la pila de actividades.
- Administrador de ventanas (Windows Manager): se encarga de organizar lo que se muestra en pantalla utilizando la librería Surface Manager.

- Content Provider: Permite encapsular un conjunto de datos que va a ser compartido entre aplicaciones creando una capa de abstracción que hace accesible dichos datos in perder el control sobre cómo se accede a la información. Un ejemplo sería la de compartir la información de los contactos, agenda etc.
- Vistas (Views): Proporciona un gran número de elementos para poder construir interfaces de usuario (GUI), como listas, botones, mosaicos, tamaño de ventanas etc. Incluye vistas estándar para las funcionalidades más frecuentes.
- Administrador de notificaciones (Notification Manager): proporciona servicios para notificar a usuario cuando algo requiera su atención. Esta biblioteca permite incluir un mensaje de alerta que reproduzca sonido, vibre o realice alguna acción en la barra de estados.
- Administrador de telefonía (Telephony Manager): Incluye todas las API vinculadas a las funcionalidades propias de un teléfono como realizar llamadas o enviar y recibir SMS/MMS.
- XMPP Service: Colección API para utilizar este protocolo de intercambio de mensajes basado en XML.

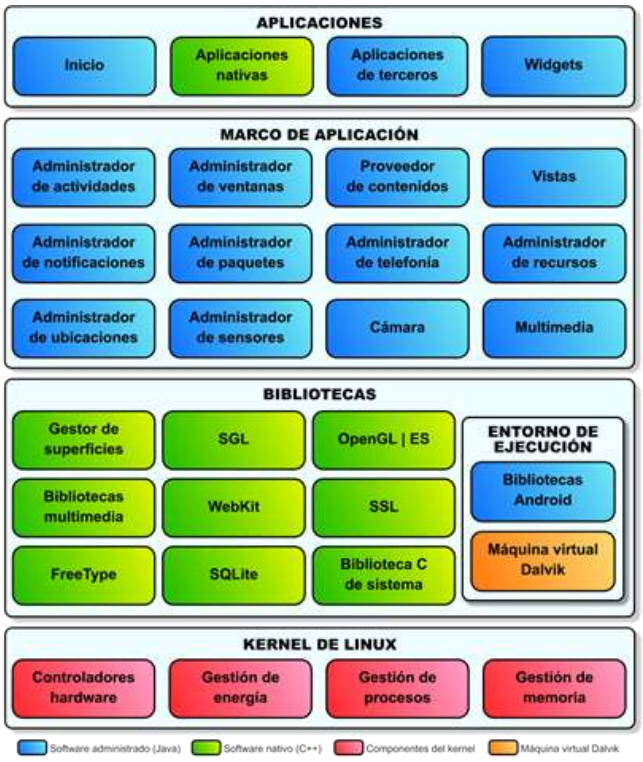


Figura 2.10: Arquitectura Android.

2.1.2 iOS

iOS anteriormente conocido como iPhone OS es un sistema operativo móvil diseñado por Apple Inc. Fue desarrollado para iPhone pero después se adaptó para iPod Touch, iPad y Apple TV. Apple no permite la instalación de iOS en hardware de terceros. Es un derivado de Mac OS X, que a su vez está basado en Darwin BSD.

El sistema operativo se dio a conocer en la Macworld Conference & Expo el 9 de enero de 2007 y publicado en junio de ese año. En un principio las aplicaciones de terceros no fueron apoyadas. El 6 de marzo de 2008 se lanzó la primera beta del SDK y del sistema operativo llamado iPhone OS.

Finalmente en junio de 2010 el sistema operativo pasó a llamarse iOS.

Figura 2.11: iOS 6.

2.1.2.1 Características de iOS

Las características principales de iOS son las siguientes:

- La pantalla principal o Spring Board muestra los iconos de las aplicaciones y el dock en la parte inferior donde se pueden anclar aplicaciones de uso frecuente. La pantalla principal aparece cuando el usuario conecta el dispositivo presiona el botón "Home". En la parte superior la pantalla tiene una barra de estados donde muestra la hora, el tiempo, el nivel de batería, la potencia de señal etc. El resto de la pantalla se usa para la aplicación que se ejecuta en ese momento
- Con la versión 4 de iOS se introdujo un sistema de carpeta, cuando las aplicaciones se encuentra en el modo "*Jiggle*" se pueden crear carpetas moviendo una aplicación sobre otra y se pueden agregar más aplicaciones en las carpetas con el mismo procedimiento.
- Las notificaciones en iOS fueron rediseñadas en la actualización de iOS5. Con Centro de Notificaciones todas las notificaciones aparecerán discretamente en la parte superior. Deslizando con el dedo desde la barra de tareas hacia abajo y se pueden ver las notificaciones pendientes. El Centro de Notificaciones es totalmente configurable. [4]
- Multitarea: Con la versión iOS 4 se introdujo la multitarea para aplicaciones pudiendo dejar en background aplicaciones que se estén utilizando y volver a ellas y continuar desde donde se dejaron. Algunas aplicaciones como GPS, reproducción de audio, sincronización con iTunes, actividad de red y voz sobre ip si siguen ejecutando mientras están segundo plano. En cambio otras aplicaciones se quedan en segundo sin recibir ciclos de reloj.[5]
- Game Center es una red social de juegos lanzado por Apple. En iOS 5 se permite soporte para fotos de perfil. Fue presentado el 8 de abril de 2010 y lanzado el 8 de septiembre de 2010.
- No soporta tecnologías como Flash la cual se explicaba en la carta que Steve Jobs escribía en Abril de 2012.[6]

Figura 2.12: Evolución de iPhone. [7]

2.1.2.2 Arquitectura de iOS

Como se puede observar en la Figura 3, la arquitectura iOS se compone de 4 capas [8]:

Capa del núcleo del SO (Core OS): Controla el sistema de memoria virtual, los hilos, los ficheros del sistema, la red e interproceso la comunicación con los frameworks de la capa del núcleo del sistema operativo. Esta capa rodea el entorno del kernel, los controladores y las interfaces básicas del sistema operativo. Cuenta con la librería libSystem que apoya las especificaciones del API POSIX/BSD 4.4/C99 e incluye un sistema de niveles de APIs para muchos servicios.

Capa del núcleo de servicios (Core Services): Los frameworks en este nivel proveen los servicios básicos tales como la manipulación de cadenas, la gestión de cobro, la creación de redes, los servicios públicos de URL, administración de contactos y preferencias. Ofrece servicios basados en las características hardware del dispositivo como GPS, brújula, acelerómetro y giroscopio. Ejemplos de frameworks de esta capa son Core Location, Core Motion y System Configuration

Media: Los frameworks y servicios de esta capa dependen de la capa del núcleo de servicios y proveen de servicios gráficos y multimedia a la capa Cocoa Touch. Estos incluyen Core Graphics, Open GL | ES, Core Animation, AVFoundation, Core Audio y reproducción de videos.

Cocoa Touch: Los frameworks de esta capa ayudan a las aplicaciones basadas en iOS. Estos incluyen frameworks como el Game Kit, Map kit y el iAd.

La capa de Cocoa Touch y la capa de servicios del núcleo tienen un framework de Objective-C diferentes que son especialmente importantes para los desarrolladores. Estos son:

- UIKit: Este framework proporciona todas las clases que una aplicación necesita construir y gestionar su interfaz de usuario.
- Fundation: Este framework define el comportamiento básico de los objetos, establece los mecanismos para su gestión y proporciona objetos para los tipos de datos primitivos, colecciones y servicios del sistema operativo.

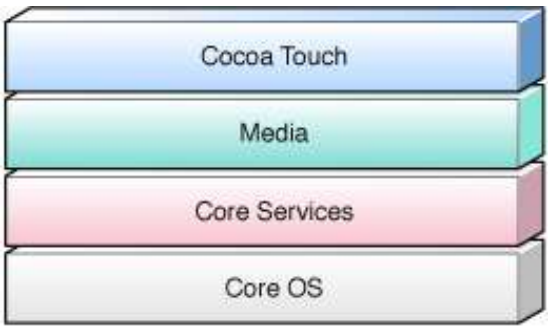


Figura 2.13: Arquitectura de iOS.

2.1.3 Windows Phone

Windows Phone es un sistema operativo móvil diseñado por Microsoft como sucesor de la plataforma Windows Mobile. El primer sistema operativo para móviles desarrollado por Windows fue Windows CE y estaba basado en Windows 95. Solo se introdujo en PDAs. El primer sistema operativo que se diseño para un móvil fue el Windows Mobile 5 pero difería muy poco con las versiones anteriores de Windows CE. La versión anterior a Windows Phone 7 fue Windows Mobile 6.5 que fue la primera versión en contener una interfaz táctil que no necesitaría de lápiz táctil para usarse.

El 1 de septiembre de 2010 se lanzó el Windows Phone7 y el SDK estuvo disponible el 16 de septiembre de 2010.

Con Windows Phone 7 Microsoft quería abarcar un mercado más generalista y debido a esto ya no reutilizaban tanto código de Windows CE abandonando bastantes de las funcionalidades que ofrecía Windows Mobile.

Tras Windows Phone 7.5 y Windows Phone 7.5 Refresh, este mismo año Microsoft ha anunciado la nueva versión del sistema operativo, Windows Phone 8 en la cual abandona completamente el código de Windows CE y utilizan Windows NT.

Figura 2.14: Windows Phone 8.

2.1.3.1 Características de WP

Windows pone incorpora una nueva interfaz llamada "Metro". Esta interfaz cuenta con una pantalla de inicio que se compone de "*Live Tiles*" o baldosas vivas. Estas "*Live Tiles*" son enlaces a aplicaciones, opciones, funciones y elementos fundamentales. Estos son modificables, se pueden borrar, mover y añadir nuevos. Windows Phone utiliza tecnología multitouch y temas oscuros para ahorrar batería ya que los pixeles oscuros no emiten luz [9]. Se pueden ver en la siguiente figura.

Figura 2.15: Windows Phone Lives Tiles.

Windows Phone tiene un centro de Office que organiza y gestiona todas las aplicaciones y documentos de Microsoft Office. El paquete de aplicaciones Microsoft Office Mobile esta formado por Word Mobile, Excel Mobile, Power Point Mobile, One Note Mobile y Outlook Mobile, todas estas aplicaciones son totalmente compatibles con la versión de escritorio de Microsoft Office.

En Windows Phone 7 la multitarea se limita solo a aplicaciones empaquetadas, pero en Windows Phone 7.5 la cosa cambió. Ahora presionando y manteniendo pulsado el botón de retroceso se accede a un conmutador de aplicaciones que muestra 5 aplicaciones que estén activas. Estas aplicaciones que están activas se las reconoce como "*Live Agents*" y las aplicaciones que quedan en reposo rápidamente vuelven al hilo principal de ejecución.

Los requisitos de hardware de Windows Phone estipulan que los dispositivos tienen que incluir un botón de búsqueda dedicado en la parte frontal del dispositivo. Las aplicaciones pueden utilizar este botón para realizar búsquedas internas como contactos etc. El botón también se usa para la aplicación del buscador Bing, que esta permite buscar en web, noticias y localizaciones en mapas.

Windows Phone también tiene una función de reconocimiento de voz que permite realizar búsquedas en Bing, en contactos o iniciar aplicaciones.

Bing Maps ofrece servicios de navegación y Local Scout informa al usuario de los sitios de interés cercanos a este.

Xbox live en Windows Phone proporciona acceso a los juegos en un teléfono con funcionalidad Xbox Live. Esta funcionalidad permite al usuario interactuar con su avatar, ver sus logros, enviar mensajes a sus amigos de Xbox Live etc.

2.1.3.2 Arquitectura de WP

Como se ha comentado antes Windows Phone 7 esta basado en Windows Embedded CE 6.0, dándole suficiente memoria Windows CE 6 puede correr 32000 procesos. [10, 11 y 12]

La gestión de memoria del nuevo sistema operativo es sustancialmente diferente. El espacio de memoria de 32 bits se divide en dos, una parte superior de 2GB dedicada al núcleo y una parte inferior de 2GB dedicado a código fuente que no es del núcleo. La parte superior se divide en dos partes de 1 GB. La parte superior de 1GB se utiliza para diversas estructuras del sistema como las bibliotecas del sistema y la parte inferior de 1GB es para el uso de programas o lo que necesiten.

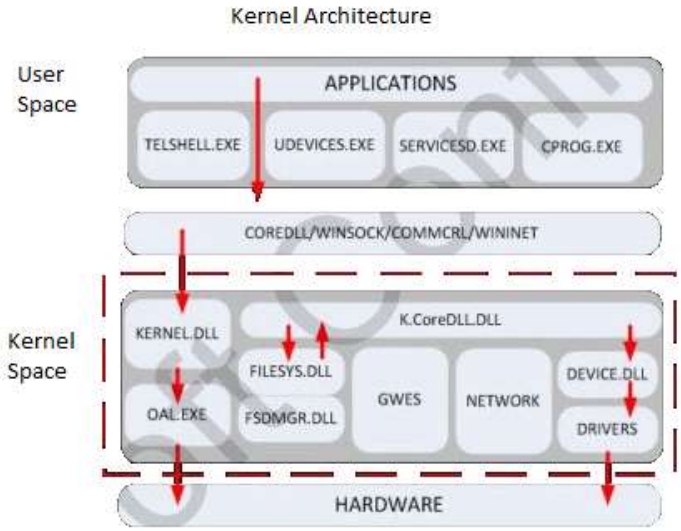


Figura 2.16: Arquitectura del kernel de Windows Phone.

Windows Phone 7 utiliza sistemas dos sistemas de archivos: IMGFS y TextFAT. El primero esta diseñado para almacenar imágenes actualizables en el sistema. El último es una versión del sistema de ficheros exFAT que se utiliza para el almacenamiento del usuario.
Los archivos de usuario se organizan mediante un sistema de almacenamiento unificado que proporcionan aplicaciones que muestra una vista al usuario unificada de los archivos independientemente de su ubicación.

Para los gráficos en 3D, Windows Phone 7 utiliza Direct3D 11, basado en DirectX 10. Los frabricantes de dispositivos tendrán que escribir sus propios controladores en 2D y 3D.

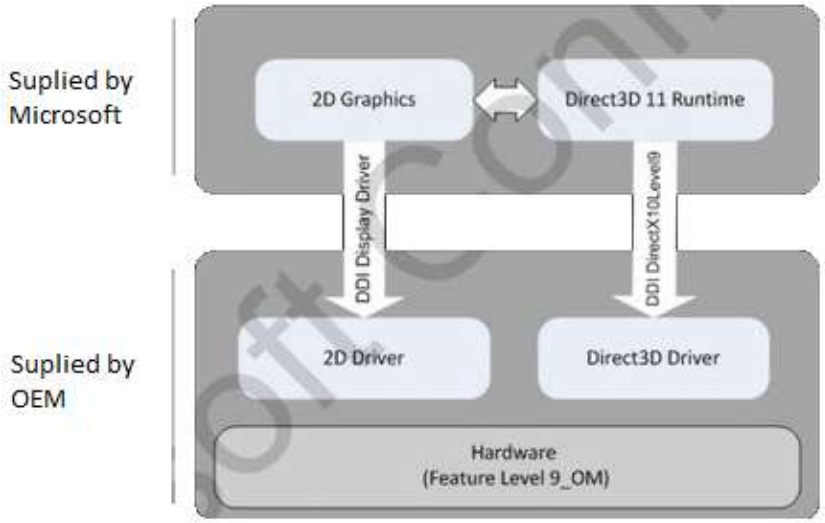


Figura 2.17: Arquitectura de los gráficos de Windows Phone.

2.1.4 Symbian OS

Symbian OS es un sistema operativo diseñado para dispositivos móviles con librerías asociadas, frameworks de interfaz de usuario e implementaciones de referencia de herramientas comunes, producido por Symbian Ltd. Es un descendiente directo de EPOC de Psion y corre

exclusivamente en procesadores ARM. Symbian Ltd. es una compañía de software creada con el propósito de desarrollar una plataforma estándar para teléfonos móviles inteligentes. [13]

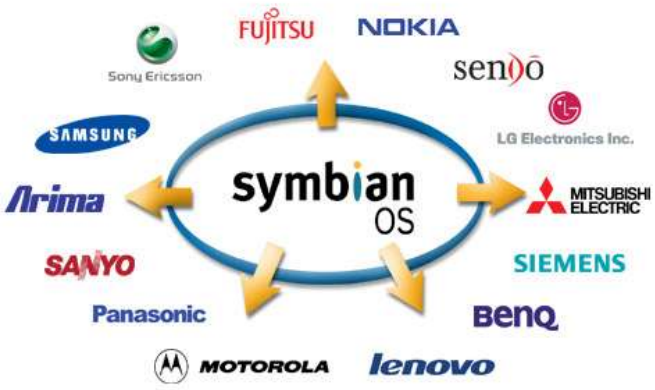


Figura 2.18: Empresas a las que da soporte Symbian.

En 1980 Psion es fundada por David Potter. De 1991-1998 Psion lanza varios dispositivos serie 3 que usan el sistema operativo EPOC16 también conocido como SIBO. En 1997 el dispositivo serie 5 usa la primera muestra del OS EPOC32. Symbian se establece como compañía privada independiente en junio de 1998, propiedad de Ericsson, Nokia, Motorola y Psion. En 1999 Panasonic se une como accionista a Symbian y utiliza su SO. En 2000 Sony y Sanyo utilizan el SO y se anuncia el primer teléfono el Nokia 7650 con el SO Symbian. En 2002 Siemens y Sony Ericsson se unen como accionistas a Symbian; Sendo y Fujitsu utilizan el SO y se lanzan los primeros móviles 3G. En 2003 Samsung se convierte en accionista de Symbian y se lanza la versión 7. En el 2004 se lanza la versión 9 y Lenovo empieza a utilizar el SO Symbian. Desde 2004 hasta 2011 Symbian lanzó versiones del SO hasta la 9.5 y en 2011 lanzaron la versión 10 del SO.

Figura 2.19: Repaso de las interfaces de Symbian. Nokia 3210, N70 y 808

2.1.4.1 Características de Symbian OS

Symbian OS tiene sus raíces en EPOC, el software de Psion, está estructurado como muchos de los sistemas operativos de Escritorio con preventivos multitarea, multilectura y protección de memoria.

La mayor ventaja de Symbian OS es el factor de que fue construido para dispositivos portátiles, con recursos limitados, que deben estar corriendo por meses o años. Pone un fuerte énfasis en la conservación de la memoria, usando idiomas de programación específicos de Symbian tales como desencriptadores y compiladores de limpieza. Junto con otras técnicas esto mantiene el uso de la memoria bajo y las fugas de memoria son raras. Hay varias técnicas para conservar espacio en el disco, los dispositivos Symbian usan memoria flash. Además toda la programación de Symbian está basada en eventos y el CPU está en "OFF" cuando las aplicaciones no están directamente conectadas con un evento. Esto es conseguido a través de un programa llamado Objetos Activos (Active Objects). El uso correcto de estas técnicas ayuda a asegurar la más larga vida de la batería. De esta manera los programadores pueden depurar cuanta energía usan sus aplicaciones.

Todo esto hace el sabor del C++ del Symbian OS muy especializado, sin embargo muchos dispositivos Symbian pueden también ser programados en OPL, Python, Visual Basic, Simkin, Perl junto con Java ME y modalidades de Personal Java y Java.
Symbian OS EKA2 también soporta respuesta en tiempo real suficientemente rápida, tanto que es posible construir un teléfono de núcleo sencillo alrededor de este, un teléfono con un solo procesador ejecuta tanto las aplicaciones de usuario como como las compilaciones de señal. Esta no es una característica disponible de Linux o Windows CE. Esto es permitido por los OS symbian EKA2 para hacerlos más pequeños, baratos y con poder más eficiente.

Symbian ha tenido un conjunto de herramientas gráficas original desde su creación, conocida como AVKON. La última versión se Symbian incluye el Qt framework que es el kit de herramientas de interfaz de usuario recomendada para nuevas aplicaciones. QT también se puede instalar en los dispositivos más antiguos de Symbian al ser compatible.

Figura 2.20: Nueva interfaz Symbian Belle.

2.1.4.2 Arquitectura de Symbian OS

Como se puede ver en la figura 2.21 la arquitectura de Symbian esta basada en capas, todos los servicios prestados por una capa se encuentran en un nivel similar de abstracción. Las capas son cohesivas y autónomas, proporciona servicios a las capas superiores y delega tareas a las capas inferiores. Las dependencias entre capas fluyen de las capas superiores a las capas inferiores, las solicitudes van de las capas superiores a las capas inferiores y las notificaciones de las capas inferiores a las superiores. [13 y 14]

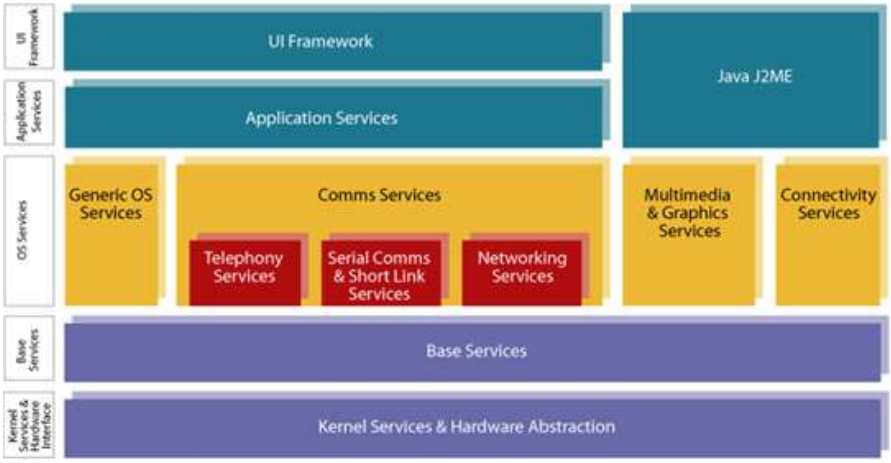


Figura 2.21: Arquitectura de Symbian OS.

En el nivel más bajo se encuentra el núcleo (kernel-EKA1 o EKA2) que apoya suficientemente rápido las respuestas en tiempo real que proporciona un teléfono de un solo núcleo. El núcleo del procesador ejecuta tanto las aplicaciones de usuario como la pila de señalización. Esta basado en una arquitectura de microkernel con las funcionalidades mínimas y más básicas; para ofrecer la

máxima robustez, disponibilidad y capacidad de respuesta. Este contiene un programador de tareas, un gestor de memoria y controladores de dispositivos, con los servicios de redes, telefonía y el sistema de archivos que da soporte a los servicios que se proporcionan en la capa de *OS Services* y la capa *Base Services*. Symbian OS soporta varios tipos de sistemas de archivos incluyendo FAT32 y el sistema específico de Symbian llamado NOR. Los sistemas de archivos no son expuestos generalmente al usuario a través de la interfaz de usuario de teléfono.

En un nivel superior se encuentra la capa *Base Service* que proporciona la abstracción del Hardware del lado del usuario, proporciona DBNS, tienda, repositorio central, servidor de archivos, las bibliotecas del usuario. También incluye el servidor de ventana de texto, una Shell de texto y un framework para drivers de media.

La capa de servicios del SO provee de frameworks y librerías que implementan el kernel del SO, que sirve de apoyo para los gráficos, comunicaciones, conectividad y multimedia, así como frameworks genéricos y las bibliotecas estándar de C/C++. En esta capa se pueden observar 2 grandes bloques: los servicios genéricos del SO, y los servicios comunes que se subdivide en 3 grandes bloques: servicios de comunicación, servicios multimedia y gráficos y servicios de conectividad.

La capa de servicios de aplicación proporciona soporte de interfaz de usuario para aplicaciones en Symbian OS. Aquí se encuentran los servicios que están destinados a las aplicaciones, como por ejemplo protocolos de mensajería, multimedia, gestión de dispositivos etc.

La capa *Framework* de interfaz de usuario es la capa superior de Symbian OS. Symbian OS es entregado a los fabricantes de móviles con una interfaz de usuario llamada *TechView* que no es completa y no tiene mucha calidad. Los fabricantes suelen cambiar la interfaz de usuario por una de calidad producida por ellos mismos.

2.2 Modelo Cliente Servidor

El modelo cliente servidor históricamente es la arquitectura más importante y continua siendo la más ampliamente utilizada. Es un modelo de programación basado en el desarrollo de aplicaciones separadas: un cliente que hace referencia a la aplicación que ve el usuario y que utiliza determinados servicios de la red y un servidor que hace referencia a la aplicación, generalmente remota, que proporciona servicios al cliente [15].

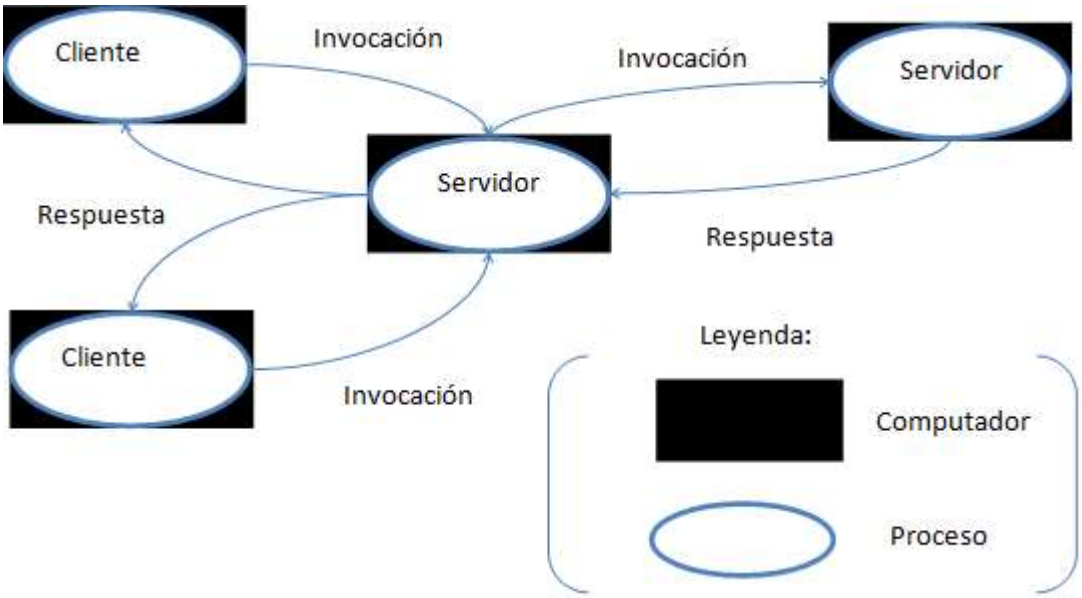


Figura 2.22: Modelo cliente servidor.

La Figura 2.22 muestra la sencilla estructura sobre la que interaccionan los procesos clientes con los procesos servidores individuales, en computadores separados, con el fin de acceder a los recursos compartidos que se gestionan.

Los servidores pueden, a su vez, ser clientes de otros servidores, como se indica en la figura. Por ejemplo Los servidores web y la mayoría del resto de los servicios de Internet son clientes del servicio DNS, que traduce Nombres de Dominio de Internet en direcciones de red.

2.3 Sistemas de Comunicación

En este apartado se explicarán los sistemas de comunicación más básicos, el Modelo OSI y los Sockets.

2.3.1 Modelo OSI

A finales de la década de los sesenta, la ISO (Organización Internacional para la Normalización) empezó a desarrollar un modelo conceptual para la conexión en red al que bautizó con el nombre de OSI (*Open Systems Interconnection Reference Model*). En 1984, este modelo pasó a ser el estándar internacional para las comunicaciones de red al ofrecer un marco de trabajo conceptual que permitía explicar el modo en que los datos se desplazaban dentro de una red. [15 y 16]

Como se puede ver en la figura 2.23 el modelo OSI divide en siete capas el proceso de transmisión de la información entre equipos informáticos, donde cada capa se encarga de ejecutar una determinada parte del proceso global.

Figura 2.23: Pila OSI.

La capa de aplicación proporciona la interfaz y servicios que soportan las aplicaciones de usuario. También se encarga de ofrecer acceso general a la red. Esta capa suministra servicios de red como la gestión de mensajes o la transferencia de archivos. La capa de aplicación suministra cada uno de estos servicios a los distintos programas de aplicación con los que cuenta el usuario en su computadora.

La capa de presentación se considera el traductor del modelo OSI. Esta capa toma los paquetes de la capa de aplicación y los convierte a un formato genérico que pueden leer todas las computadoras. La capa de presentación se encarga de cifrar o de comprimir los datos.

La capa de sesión es la encargada de establecer el enlace de comunicación o de sesión entre las computadoras emisora y receptora. Esta capa también gestiona la sesión que se establece entre ambos nodos. Proporciona tolerancia a fallos, comunicación orientada a conexión y comunicación sin conexión.

La capa de transporte es la encargada de controlar el flujo de datos entre los nodos que establecen una comunicación; se encarga de que los paquetes lleguen en la secuencia correcta y de evaluar el tamaño de los paquetes con el fin de que éstos tengan el tamaño requerido por las capas inferiores.

La capa de red encamina los paquetes además de ocuparse de entregarlos. La determinación de la ruta que deben seguir los datos se produce en esta capa, lo mismo que el intercambio efectivo de los mismos dentro de dicha ruta. La capa de red es donde las direcciones lógicas pasan a convertirse en direcciones físicas. Los routers operan en la capa de red.

La capa de enlace de datos transforma los paquetes de datos en tramas que vienen definidas por la arquitectura de red que se esta utilizando. La capa de enlace de datos se encarga de desplazar los datos por el enlace físico de comunicación hasta el nodo receptor, e identifica cada computadora incluida en la red de acuerdo con su dirección de hardware, que viene codificada en la NIC. Esta capa provee de control de errores añadiendo un CRC (Chequeo de redundancia Cíclica) al final de cada trama.

Por último en la capa física, las tramas procedentes de la capa de enlace de datos se convierten en una secuencia única de bits que puede transmitirse por el entorno físico de la red. La capa física también determina las conexiones globales de la computadora hacia la red, tanto en lo que se refiere al medio físico como a la forma de información.

2.3.2 Sockets

Los sockets o conectores se originan en UNIX BSD 4.2 en 1981 y son mecanismos de IPC que proporcionan comunicación entre procesos que se ejecutan en máquinas distintas. Están presentes en la mayoría de las versiones de UNIX, incluido Linux y también Windows NT y Macintosh OS. La comunicación entre procesos consiste en la transmisión de un mensaje entre un conector de un proceso y un conector de otro proceso. Para los procesos receptores de mensajes, su conector debe estar asociado a un puerto local y a una de las direcciones de Internet del computador donde se ejecuta. Los mensajes enviados a la dirección de Internet y a un número de puerto concreto solo pueden ser recibidos por el proceso cuyo conector este asociado a esa dirección y ese a ese puerto. Los procesos pueden utilizar el mismo conector tanto para enviar mensajes como para recibir mensajes. Cada computador permite 2^{16} números de puertos posibles. Cada proceso puede utilizar varios puertos para recibir mensajes, pero un proceso no puede compartir puertos con otros procesos del mismo computador. [15 y 17]

Hay dos tipos de sockets:

- Stream: Los sockets stream utilizan el protocolo de transporte TCP con un flujo de datos bidireccional y están orientados a conexión, debe establecerse una conexión extremo a extremo antes del envío y recepción de datos. Proporciona fiabilidad, paquetes ordenados por secuencia y libre de errores sin duplicación.

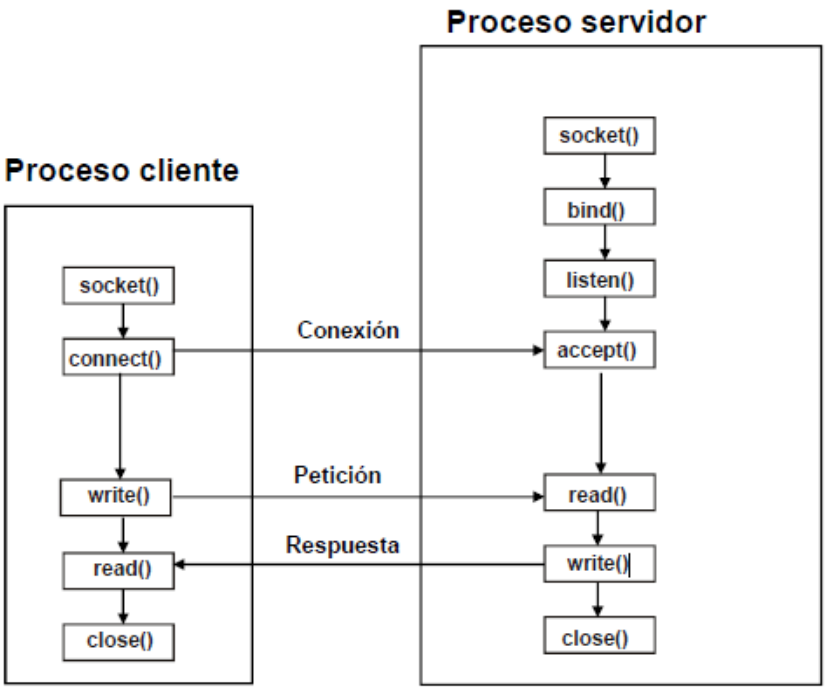


Figura 2.24: Sockets stream

- Datagrama: Los sockets de datagrama utilizan el protocolo de transporte UDP con un flujo de datos bidireccional y no orientado a conexión, no se establece y mantiene una conexión entre los procesos que comunican. Un datagrama es una entidad autocontenida de 64KB de longitud máxima. Mantiene separación de paquetes pero no proporciona fiabilidad.

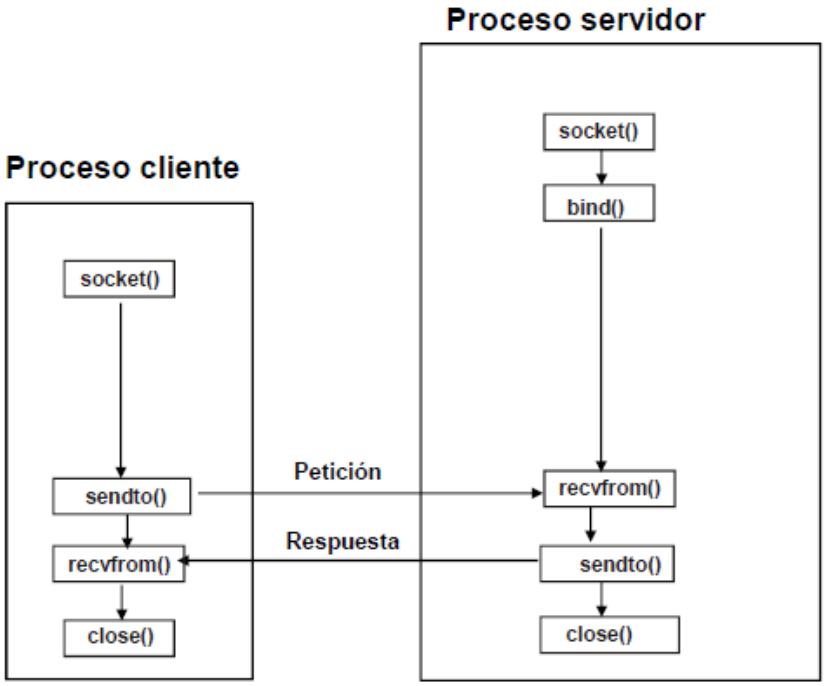


Figura 2.25: Sockets datagramas.

2.4 Llamadas a Procedimientos Remotos

En este apartado se explicarán las llamadas a procedimientos remotos más importantes como son RPC, RMI y XML-RPC.

2.4.1 RPC

RPC (*Remote Procedure Call*) o una llamada a procedimiento remoto, es muy similar a una invocación de un método remoto, en la que un programa cliente llama a un procedimiento de otro programa en ejecución en un proceso servidor. Los servidores pueden ser clientes de otros servidores para permitir cadenas de RPC. [15 y 18]

RPC fue creado por Birrel y Nelson en 1985 en "*Implementing Procedure Calls*". Son un híbrido entre las llamadas a procedimientos en local y los pasos de mensajes. Como se muestra en la figura 2.26 RPC esta presente a partir de la capa de sesión OSI, no depende del protocolo de transporte, por lo que la fiabilidad dependerá del protocolo de transporte elegido al establecer la conexión.

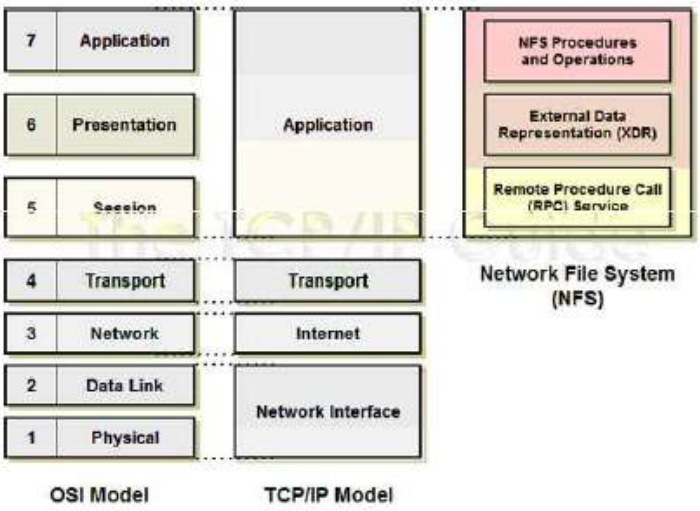


Figura 2.26: Situación de las RPC en la pila OSI.

Las RPC funcionan de la siguiente manera, el proceso que realiza la llamada empaqueta los argumentos en un mensaje, este se los envía a otro proceso y espera el resultado. El proceso que ejecuta el procedimiento, extrae los argumentos del mensaje, realiza la llamada de forma local, obtiene el resultado y se lo envía de vuelta al proceso que realizó la llamada. Se puede observar el ejemplo de este comportamiento en la figura 2.27.

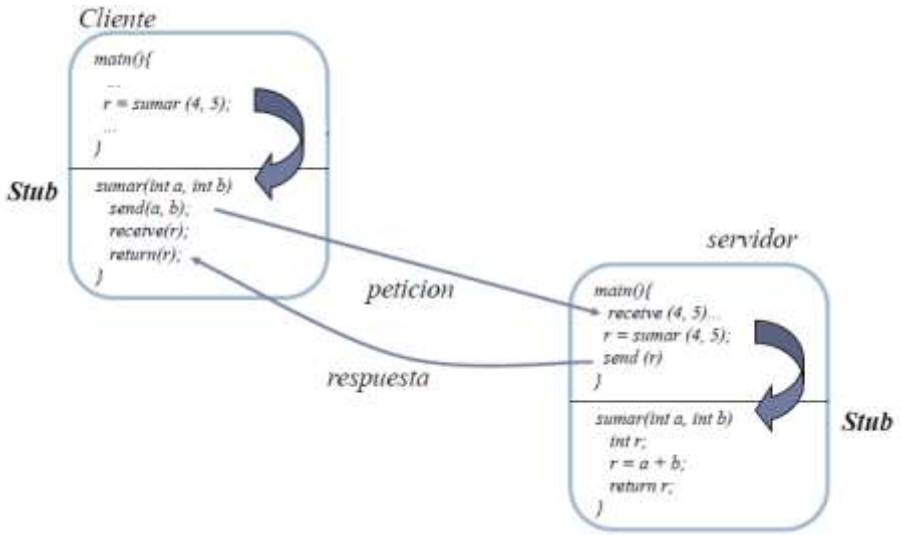


Figura 2.27: Ejemplo de una petición respuesta de una RPC.

En la figura 2.27 se pueden observar 2 *stubs* o suplentes creados en el cliente y en el servidor. Un *stub* es una pieza de código usada en el cliente y en el servidor; generados automáticamente por el software de RPC. Es el responsable de convertir los parámetros de la aplicación cliente/servidor durante una llamada a procedimiento remoto. Los suplentes sólo dependen de la interfaz, por lo que son independientes de la implementación que se haga del cliente y del servidor. Las funciones que realizan los suplentes son las siguientes: suplantar al procedimiento a ejecutar, localizar al servidor, empaquetar los parámetros, construir el mensaje de petición/respuesta, enviar el mensaje de petición/respuesta, esperar mensaje de petición/respuesta y devolver los resultados en caso de que los haya.

La versión 2 de RPC, descrita en la RFC 1057, fue estandarizada por Sun Microsystems, Sun RPC proporciona un lenguaje de interfaz denominado XDR y un conjunto de compilador de interfaces llamado rpcgen cuyo uso esta orientado al lenguaje de programación C.

El lenguaje de definición de interfaz Sun XDR está diseñado para especificar representaciones externas de datos y se extendió para convertirse en un lenguaje de definición de interfaces. Puede utilizarse para definir una interfaz de servicio para Sun RPC especificando un conjunto de definiciones de procedimiento junto a las definiciones de tipos que las soportan. Tiene que cumplir las siguientes características:

- Hay que proporcionar un número de programa y un número de versión. Los números de programa pueden obtenerse de una autoridad central para permitir que cualquier programa tenga un número propio y único. Los números de versión cambian cuando cambia la firma de algún procedimiento. El número de programa y de versión se envían en el mensaje de petición, de modo que el cliente y el servidor puedan concretar la versión que se está usando.
- Una definición de procedimiento especifica una firma de un procedimiento y un número de procedimiento. El número de procedimiento se emplea como un identificador en los mensajes de petición.
- Sólo se permite un parámetro de entrada. De este modo si se quiere incluir más de un parámetro deberán incluirlos como componentes de una estructura.
- Los parámetros de salida de un procedimiento se devuelven como un solo resultado.
- La firma del procedimiento consta de un tipo de resultado, el nombre del procedimiento y el tipo del parámetro de entrada. Tanto el tipo del resultado como el de entrada pueden ser bien un solo valor o una estructura conteniendo varios valores.

Sun RPC lanza un servicio de enlazado denominado enlazador de puertos en un número de puerto bien conocido de cada computador. Cada ejemplar del enlazador de puertos almacena el número de programa, el número de versión y el número de puerto en uso por cada servicio que se ejecuta localmente. Cuando arranca un servidor, registra su número de programa, número de versión y número de puerto frente al enlazador de puertos. Cuando el cliente arranca, encuentra el puerto del servidor mediante una petición remota al enlazador de puertos del servidor huésped, especificando el número del programa y el número de versión.

Los mensajes de petición y respuesta de Sun RPC proporcionan campos adicionales que permiten pasar información de autenticación entre el cliente y el servidor. Es posible dar soporte a diferentes protocolos de autenticación: ninguno, al estilo UNIX que incluye credenciales de autenticación como el uid y el gid del usuario; con una clave compartida que permita firmar los mensajes RPC y al estilo de autenticación de Kerberos. Un campo en la cabecera de RPC indica cuál de ellos se esta empleando.

Las RPC han evolucionado hacia orientación a objetos como RMI o CORBA.

2.4.2 RMI

Las RMI (*Remote method invocation*) o las invocaciones de metodos remotos, son invocaciones de métodos entre objetos en diferentes procesos, tanto si es en el mismo computador o no. A los objetos que pueden recibir invocaciones remotas los conocemos como objetos remotos. Todos los objetos pueden recibir invocaciones locales. Los dos conceptos fundamentales siguientes son el corazón de RMI [15 y 19]:

- Referencia de objeto remoto: otros objetos pueden invocar los métodos de un objeto remoto si tienen acceso a su referencia de objeto remoto.
- El objeto remoto donde se recibe la información de método remoto se especifica mediante una referencia a objeto remoto.
- Las referencias a objetos remotos pueden pasarse como argumentos y resultados de las invocaciones de métodos remotos.
- Interfaz remota: cada objeto remoto tiene una interfaz remota que especifica cuáles de sus métodos pueden invocarse remotamente. La clase de un objeto remoto implementa los métodos de su interfaz remota. Los objetos en otros procesos pueden invocar solamente los métodos que pertenezcan a su interfaz remota. En Java RMI las interfaces remotas se definen de la misma forma que cualquier interfaz en Java

La implentación de RMI se basa en varios módulos separados. El módulo de comunicación realiza el protocolo de petición-respuesta que transmite los mensajes de petición y respuesta entre el cliente y el servidor. El módulo de comunicación emplea el tipo de mensaje, su idPetición y la referencia remota del objeto que se invoca. Los módulos de comunicación son los encargados de proporcionar la semántica de comunicación y en el servidor selecciona el distribuidor para la clase del objeto que se invoca, pasando su referencia local, que se obtiene del módulo de referencia remota en respuesta al identificador de objeto remeto en el mensaje petición.

El módulo de referencia remota es el responsable de traducir las referencias entre objetos locales y remotos, y de crear referencias a objetos remotos. Suele ser llamado por el software de RMI cuando realizan el empaquetado y desempaquetado de las referencias a objetos remotos. Para soportar sus responsabilidades, el módulo de referencia remota de cada proceso tiene una tabla de objetos remotos que almacena la correspondencia entre referencias a objetos locales en ese proceso y las referencias a objetos remotos. Las acciones del módulo de referencia ocurren de la siguiente manera: Cuando se pasa un objeto remoto por primera vez, como argumento o

resultado, se le pide al módulo de referencia remota que cree una referencia a un objeto remoto, que se añade a su tabla. Cuando llega una referencia a un objeto remoto, en un mensaje de petición o de respuesta, se le pide al módulo la referencia al objeto local correspondiente, que se referirá a un proxy o a un objeto remoto. En el caso de que el objeto no esté en la tabla, el software RMI crea un nuevo proxy y pide al módulo que lo añada a la tabla.

El software de RMI consiste en crear una capa de software entre la aplicación y los módulos de comunicación y de referencia remota. Cuenta con los siguientes elementos:

- Proxy: el papel del proxy es hacer que la invocación al método remoto sea transparente para los clientes, y para ello se comporta como un objeto local para el que invoca; pero en lugar de ejecutar la invocación, dirige el mensaje al objeto remoto. Oculta los detalles de la referencia al objeto remoto, el empaquetado de los argumentos, el desempaquetado de los resultados y el envío y recepción de los mensajes desde el cliente. Cada método del proxy empaqueta una referencia hacia el objeto remoto, su propio idMetodo y sus argumentos en un mensaje de petición y lo envía al objetivo, espera el mensaje de respuesta, lo desempaqueta y devuelve los resultados a quien lo invocó.
- Distribuidor: cada servidor tiene un distribuidor y un esqueleto para cada clase que representa a un objeto remoto. El distribuidor recibe el mensaje de petición desde el módulo de comunicación. Emplea el idMetodo para seleccionar el método apropiado del esqueleto, pasándole el mensaje de petición. El distribuidor y el proxy emplean los mismos métodos de asignación de cada idMetodo para los métodos de la interfaz remota.
- Esqueleto: la clase de un objeto remoto tiene esqueleto, que implementa los métodos de la interfaz remota. Se encuentran implementados de forma muy diferente de los métodos del objeto remoto. Un método del esqueleto desempaqueta los argumentos del mensaje de petición e invoca el método correspondiente en el objeto remoto. Espera la consumación de la invocación y después empaqueta el resultado, junto con las excepciones producidas.

Las clases para cada proxy, distribuidor y esqueleto empleado en RMI se generan automáticamente mediante un compilador de interfaces. Un ejemplo de la arquitectura de RMI se encuentra en la figura 2.28.

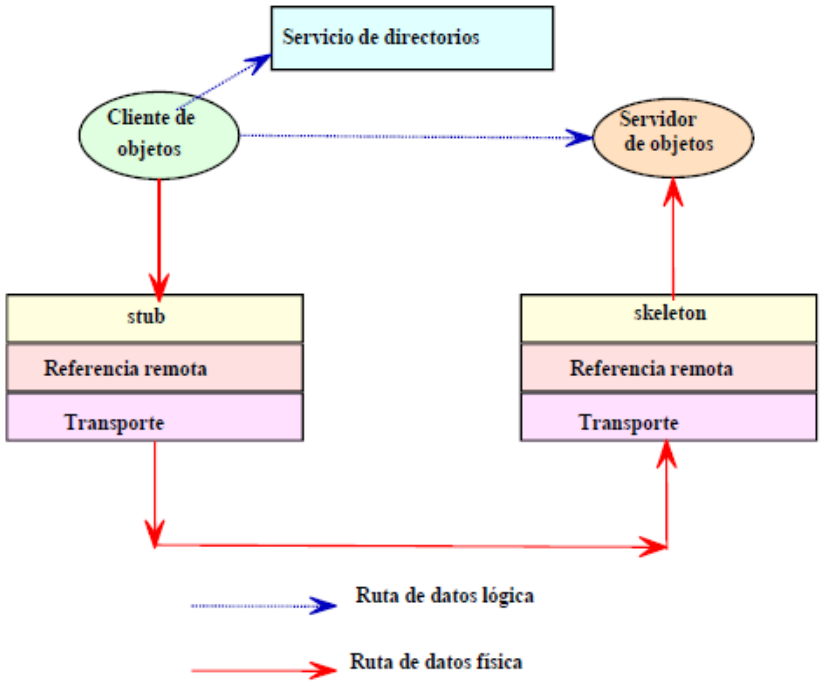


Figura 2.28: Funcionamiento de la arquitectura RMI.

El enlazador (binder) en un sistema distribuido es un servicio separado que da soporte a una tabla que contiene relaciones con nombres textuales y referencias a objetos remotos. Se emplea por parte de los servidores, para registrar sus objetos remotos mediante su nombre y, por parte de los clientes, para buscarlos por su nombre.

Algunas aplicaciones necesitan que su información sobreviva durante un periodo de tiempo largo y es muy ineficiente tener los objetos que representan esta información se mantengan en ejecución constantemente, por ello un objeto remoto se dice que esta activo cuando está disponible para su invocación en el interior de un proceso en ejecución, mientras que se denomina pasivo si no esta activo actualmente pero puede activarse. Un objeto pasivo consta de dos partes: La implementación de sus métodos y su estado en forma empaquetada. La activación consiste en la creación de un objeto activo a partir de un objeto pasivo instanciando su clase y la iniciación de sus variables de instancia desde el estado almacenado. Un activador se encarga de registrar los objetos pasivos que estén disponibles para su activación, arrancar procesos de servicio con nombre, activar los objetos remotos de su interior y mantener la pista de las ubicaciones de los servidores de los objetos remotos que ya han sido activados.

Un ejemplo de RMI es Java RMI esta basado en las interfaces y clases definidas por los paquetes java.rmi y java.rmi.server. RMI ofrece mecanismos para crear servidores y objetos cuyos métodos se puedan invocar remotamente, mecanismos que permiten a los clientes localizar los objetos remotos y servicio de directorios, rmiregistry, servicio de directorios de Java.

Los programas RMI son más fáciles de diseñar y poseen un servidor concurrente, pero los sockets tienen menos sobrecarga y RMI sólo funciona en plataformas Java.

2.4.3 XML-RPC

XML-RPC es una antesala de los servicios web y hace que sea fácil para los equipos llamar a procedimientos en otros equipos. XML-RPC utiliza el lenguaje XML que proporciona un vocabulario para describir RPC, que se transmiten entre los ordenadores mediante HTTP. [20 y 21]

Fue creado por Dave Winer de la empresa UserLand Software en asociación con Microsoft en el año 1998. Microsoft consideró que era bastante simple por lo que le añadió nuevas funcionalidades y se convirtió en SOAP.

Las llamadas XML-RPC se realizan entre dos partes: el cliente y el servidor. Realizan los siguientes pasos:

- El programa cliente realiza una llamada a un procedimiento mediante el cliente de XML-RPC, especificando el nombre del método, los parámetros y un servidor destino.
- El cliente XML-RPC toma el nombre del método y los parámetros; y los empaqueta en un XML. A continuación, el cliente emite una solicitud HTTP POST al servidor de destino.
- Un servidor HTTP en el servidor de destino recibe la petición POST y pasa el contenido XML a un detector de XML-RPC.
- El oyente de XML-RPC analiza el XML para obtener el nombre del método y los parámetros para llamar al método apropiado y pasarle los parámetros.
- El método devuelve una respuesta al proceso de XML-RPC y el proceso XML-RPC empaqueta la respuesta en un XML.
- El servidor web retorna el XML generado como respuesta a la petición HTTP POST.
- El cliente XML-RPC analiza el XML para extraer el valor de retorno y, a continuación analizar el valor de regreso en el programa cliente.

El programa cliente continúa procesando el valor de retorno.

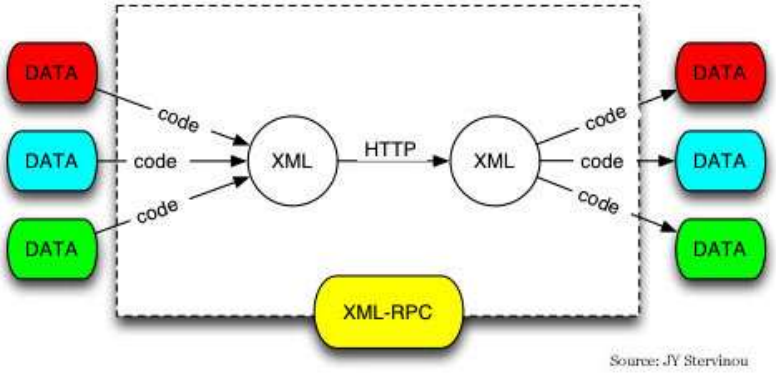


Figura 2.29: Comportamiento XML-RPC.

El uso de HTTP significa que las peticiones XML-RPC deben ser síncronas y sin estado.

Existen implementaciones de XML-RPC para varios sistemas operativos, lenguajes de programación, licencias comerciales y de software libre como: C/C++, Delphi, Java, Microsoft .NET, Perl, PHP, Python, TLC, Web Objects Zope…

Algunas de las implementaciones más conocidas son:

- Apache XML-RPC, implementado en Java.
- XMLRPC-EPI, implementado en C.
- XML-RPC-C, implentación en C y C++.

2.5 Servicios Basados en HTTP

En este apartado se explicará brevemente el protocolo HTTP. [16]

2.5.1 HTTP

El protocolo de transferencia de hipertexto (HTTP, *HyperText Transfer Protocol*) es el protocolo de la capa de aplicación de la Web y se encuentra en el corazón de la Web. El protocolo HTTP es un método de diálogo estandarizado entre dos máquinas distintas, generalmente un cliente y un servidor que se comunican mediante mensajes HTTP. En éste diálogo la máquina cliente suele solicitar un determinado recurso (por ejemplo un documento) mediante una URL a la máquina servidora, y esta responde con una determinada información (generalmente el documento solicitado).

HTTP utiliza TCP como su protocolo de transporte subyacente, esto provoca que HTTP no tenga que preocuparse por las perdidas de datos o por los detalles de cómo TCP recupera datos perdidos, ordena paquetes.

HTTP no mantiene ninguna información acerca de los clientes, se dice que HTTP es un protocolo sin memoria de estado.

HTTP puede utilizar dos tipos de conexiones con TCP dependiendo de sus necesidades, ya que las peticiones de los clientes a una aplicación pueden hacerse una detrás de otra, periódicamente a intervalos regulares o de forma intermitente; se debe decidir si se requiere de una conexión con TCP para atender todas estas peticiones y respuestas o por el contrario se requiere que cada petición establezca una conexión TCP. Si se decide la primera de ellas se habla de HTTP con conexiones persistentes; si se emplea la segunda opción entonces se habla de HTTP con conexiones no persistentes.

- HTTP con conexiones no persistentes:

  Estos son los pasos para transferir una página web desde un servidor a un cliente en el caso de conexiones no persistentes:

  1º El proceso cliente HTTP inicia una conexión TCP con el servidor en el puerto número 80, que es el número por defecto para HTTP. Asociados con la conexión TCP habrá un socket en el cliente y otro en el servidor.

  2º El cliente HTTP envía un mensaje de solicitud HTTP al servidor a través de su socket.

  3º El proceso servidor HTTP recibe el mensaje de solicitud a través de su socket, recupera el objeto de su medio de almacenamiento (RAM o disco), encapsula el objeto en un mensaje de respuesta HTTP y lo envía al cliente a través de su socket.

  4º El proceso servidor HTTP indica a TCP que cierre la conexión TCP.

  5º El cliente HTTP recibe el mensaje de respuesta. La conexión TCP termina. El mensaje indica que el objeto encapsulado es un archivo HTML. El cliente extrae el archivo del mensaje y examina el HTML.

  6º Los cuatro primeros pasos se solicitan en el caso de que se localice la referencia, en el HTML extraído, de algún otro objeto.

- HTTP con conexiones persistentes

  Con las conexiones persistentes, el servidor deja la conexión TCP abierta después de enviar una respuesta. Las subsiguientes solicitudes y respuestas que tienen lugar entre el mismo cliente y el servidor pueden enviarse a través de la misma conexión. En concreto una página web completa se puede enviar en una misma conexión TCP persistente. Además varias páginas web que residan en el mismo servidor pueden enviarse a través de la misma conexión TCP persistente. Normalmente el servidor HTTP cierra una conexión cuando no se ha utilizado durante cierto tiempo.

Las especificaciones de HTTP incluyen las definiciones de los formatos de los mensajes HTTP de solicitud y de respuesta.

Mensaje de solicitud HTTP:

El mensaje de solicitud de HTTP esta escrito en código ASCII por lo que cualquier persona con conocimientos informáticos podrá leerlo y entenderlo sin problemas. La primera línea del mensaje HTTP se llama línea de solicitud y las siguientes líneas son las líneas de cabecera.

La línea de solicitud consta de tres campos, el campo del método, el campo URL y el campo de la versión de HTTP. El campo que indica el método puede tomar diferentes valores, GET, PUT, POST, DELETE y HEAD.

En un mensaje HTTP pueden encontrase varias líneas de cabecera, unos ejemplos son Connection, que indica como es la conexión de HTTP, si es persistente o no, HOST que indica el nombre del host al que se le quiere solicitar el objeto o User Agent que indica el nombre del agente de usuario que hace la petición HTTP, un ejemplo sería el nombre y versión del navegador.

Además de las líneas de solicitud y cabecera, hay un salto de línea y otro campo llamado cuerpo de entidad, que esta dedicado a los métodos como POST que necesitan enviar ciertos datos al servidor como los introducidos en un formulario.

Mensaje de Respuesta HTTP:

El mensaje de respuesta de HTTP al igual que el mensaje de solicitud esta escrito en código ASCII y posé la misma estructura que el mensaje de solicitud, una línea de solicitud, varias líneas de cabecera una línea en blanco y el cuerpo de entidad.

En el caso del mensaje de respuesta la línea de solicitud esta formado por un código de estado que indica el resultado de la solicitud. Alguno de los códigos de estado y sus frases asociadas más utilizados son:

- 200 Ok: La solicitud se ha ejecutado con éxito y se ha devuelto la información en el mensaje de respuesta.
- 301 *Moved Permanently*: El objeto ha sido movido de forma permanente; el nuevo URL se encuentra en la cabecera Location: del mensaje de respuesta.
- 400 *Bad Request*: Se trata de un código de error genérico que indica que la solicitud no ha sido comprendida por el servidor.
- 404 *Not Found*: El documento solicitado no existe en el servidor.
- 505 *HTTP Version Not Supported*: La versión de protocolo HTTP solicitada no es soportada por el servidor.

2.6 Servicios Web

El consorcio W3C define los Servicios Web como sistemas software diseñados para soportar una interacción interoperable maquina a maquina sobre una red. Los Servicios Web suelen ser APIs Web que pueden ser accedidas dentro de una red (principalmente Internet) y son ejecutados en el sistema que los aloja. [20, 22, 23 y 24]

Los Servicios Web surgieron por la necesidad de ofrecer aplicaciones distribuidas utilizando software que ejecuta en diferentes sistemas operativos y arquitecturas, escrito utilizando diferentes lenguajes y herramientas de programación; y desarrollado de forma independiente.
Los principios básicos de diseño de los Servicios Web son los siguientes:

Interoperabilidad en entornos heterogéneos

Los Servicios Web están basados en protocolos y mecanismos estándar como HTTP, SOAP, XML, UDDI, WSDL. Un ejemplo de combinación de varios Servicios Web es la observada en la figura 2.30.

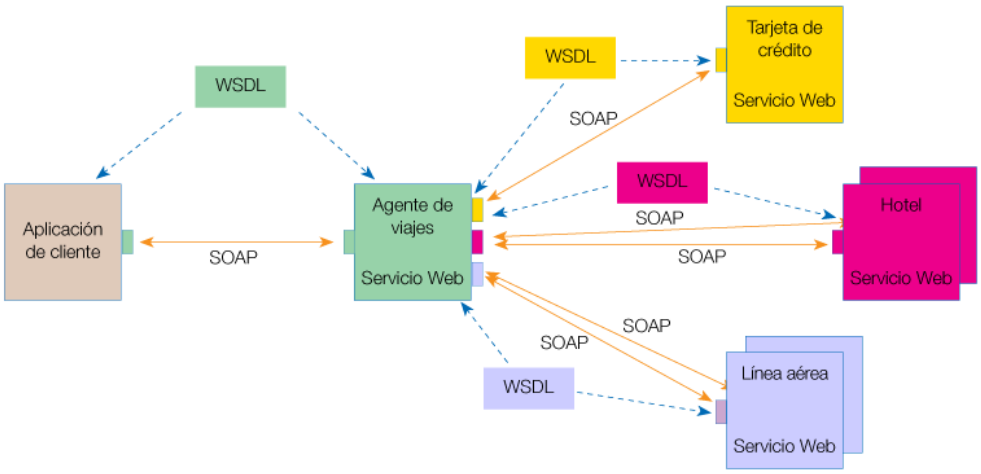


Figura 2.30: Servicios Web.

Diferentes Patrones de comunicación

Patrón petición-respuesta síncrono como modelos basados en RPC. Por otro lado patrones asíncronos.

Independencia en el modelo de programación

Los Servicios Web ofrecen un modelo de computación en Internet que proporciona independencia en el lenguaje de programación y en el modelo de programación.

Representación de mensajes

El caso común de los Servicios Web es que los clientes y servidores se comuniquen mediante mensajes XML y que sigan el estándar SOAP.

Referencias de servicios

Cada servicio Web tiene una URI (*uniform resource identifier*) que contiene una URL (*uniform resource locator*), que incluye la localización del recurso, y una URN (*uniform resource name*), que incluye nombres de recursos que no incluyen localización.

La URI es utilizada por los clientes para referenciar al servicio

Activación de servicios

El Servicio Web se solicita por medio de la URL, ya que es identificado por esta y el Servicio Web puede residir en ese computador o en otro computador para mejorar las prestaciones ofrecidas.

La activación del Servicio Web puede realizarse de dos maneras, bajo demanda y este puede ejecutar conjuntamente.

Empleo de los Servicios Web

Los Servicios Web permiten el uso directo entre clientes y servidores de SOAP y XML a través de un API que oculta los detalles del SOAP y del XML.

La invocación se puede realizar de una manera estática mediante un suplente o un proxy; o de una manera dinámica mediante la operación genérica que convierte la llamada y los argumentos a SOAP y XML dinámicamente.

Para poder comprender la circulación de información vista en la figura 2.30, hay que explicar las siguientes tecnologías:

2.6.1 SOAP

SOAP son las siglas de *Simple Object Access Protocol* y deriva de XML-RPC como se dijo en su sección y fue creado por Microsoft e IBM y actualmente se encuentra bajo protección de W3C. SOAP es un protocolo estándar que mediante el intercambio de datos XML dos objetos en diferentes procesos pueden comunicarse.

SOAP específica cómo representar los mensajes XML, cómo combinar mensajes SOAP para un modelo petición respuesta, cómo procesar los elementos de los mensajes y cómo utilizar el transporte para enviar mensajes SOAP.

El protocolo SOAP esta formado por nodos SOAP que transmiten, reciben, procesan y responden un mensaje SOAP. Puede haber emisor SOAP, receptor SOAP y un intermediario.

Los mensajes SOAP son la unidad de comunicación entre los nodos SOAP, esta formado por una cabecera, que es opcional e incluye información de control como un identificador de mensajes, un nombre de usuario, una clave pública, etc. y por el cuerpo, que incluye el mensaje y la referencia al XML que describe el servicio. Estos elementos son definidos como un esquema en el espacio de nombres XML. El mensaje se envía en un *envelope* o sobre, que envuelve el mensaje.

La comunicación se realiza cliente-servidor mediante RPC donde el elemento body contiene una petición o una respuesta. El protocolo para el transporte de mensajes SOAP puede ser HTTP al estilo RPC con peticiones en HTTP POST y respuestas en la respuesta al POST; o con HTTP POST y HTTP GET. SOAP puede usar el protocolo SMTP donde la propia especificación indica cómo encapsular mensajes SOAP en mensajes SMTP.

2.6.2 XML

XML (*Extensible Markup Languaje*) según W3C es un formato sencillo de texto muy flexible derivado de SGML (ISO 8879). Originalmente diseñado para satisfacer los desafíos de la gran escala de la publicación electrónica, XML también está desempeñando un papel cada vez más importante en el intercambio de una amplia variedad de datos en la Web y en otros lugares. [25]

XML es extensible y permite a los usuarios crear sus propias etiquetas cosa que no permite HTML.

Los mensajes XML están formados por etiquetas de tal forma que estas pueden contener uno u varios atributos con valores dentro de ellas. Todos los mensajes XML tienen que empezar con la siguiente línea “<?xml versión=”1.0”?>”. Además XML permite introducir comentarios en sus mensajes de la siguiente forma “<!—Comentario-->”.

XML también esta formado por un espacio de nombres que permite solucionar las ambigüedades existentes entre elementos que se llaman igual pero provienen de dos vocabularios de XML distintos. Los nombres dentro de un espacio de nombres deben ser únicos.

XML esta formado por XML Sherma que es un lenguaje de esquemas que describe la estructura y restricciones de los contenidos de los documentos XML de una forma precisa. Fue desarrollado por el W3C. Este se compone de 3 partes:

- **XML Schema: Fundamentos**: es un documento no-normativo hecho para proporcionar una descripción fácilmente legible de las posibilidades del Esquema XML, y está orientado a un entendimiento rápido sobre cómo crear esquemas utilizando el lenguaje Esquema XML.
- **XML Schema: Estructuras:** especifica la definición del lenguaje XML Schema, que cuenta con las instalaciones para la descripción de la estructura y restringir el contenido de documentos XML1.0, incluidos aquellos que utilizan el espacio de nombres de XML.
- **XML Schema: Tipos de datos:** se definen las facilidades para definir tipos de datos para ser utilizas en XML Schemas así como en otras especificaciones XML. El lenguaje de tipo de datos que se representa en XML 1.0, proporciona un superconjunto de las capacidades que se encuentran en las definiciones de los DTDs (*document type definitions*) para especificar los tipos de datos sobre los elementos y atributos.

2.6.3 WSDL

WSDL (*Web Services Description Language*) es un lenguaje basado en XML, que se usa para describir servicios de red como un conjunto de variables que operan en los mensajes que contenga cualquier información orientada a documentos u procedimientos. Las operaciones y los mensajes se describen de manera abstracta, y luego se unen a un protocolo concreto y a un formato de mensaje para definir un punto final. Estos puntos finales son combinados en puntos finales abstractos más conocidos como servicios. WSDL es extensible para permitir las descripciones de los puntos finales y sus mensajes sin importar los formatos de los mensajes o los protocolos de red que se están usando para la comunicación. De esta manera, WSDL describe la interfaz pública para el servicio Web.

WSDL se utiliza a menudo en combinación con SOAP y XML Schema para proporcionar los Servicios Web a través de Internet. Un programa cliente que se conecta a un Servicio Web puede leer el archivo WSDL para determinar que operaciones es tan disponibles en el servidor. El cliente puede utilizar SOAP para llamar realmente a una de las operaciones que figuran en el archivo WSDL usando HTTP o XML.

Hoy día la versión actual de la especificación es la 2.0, aprobada por el W3C. Ofrece un mejor soporte para Web Service REST y es mucho más sencillo de implementar.

2.7 Rest

REST (*Representational State Transfer*) es un estilo de arquitectura de software para hipermedias distribuidos tales como la Web. El término fue introducido en la tesis doctoral de Roy Fielding en 2000. [24 y 26]

Los servicios Web basados en REST intentan emular al protocolo HTTP o protocolos similares mediante la restricción de establecer la interfaz a un conjunto conocido de operaciones estándar como GET, POST, DELETE y PUT. Por lo que este estilo se centra más en interactuar con recursos con estado, que con mensajes y operaciones.

Hay que destacar que REST no es un estándar, es un estilo de arquitectura, pero está basado en estándares como HTTP, URL, representación de los recursos (XML, JSON, HTML, GIF…) y tipos MIME (text/xml, text/json, text/html,…).

REST es un estilo de arquitectura subyacente a la Web y tiene los siguientes objetivos:

- Escalabilidad de la iteración con los componentes. La Web ha crecido exponencialmente sin degradar su rendimiento. A ella pueden acceder muchos tipos de clientes, desde estaciones de trabajo a dispositivos móviles.
- Generalidad de interfaces. Cualquier cliente puede interactuar con cualquier servidor HTTP sin ninguna configuración especial gracias al protocolo HTTP. Con los métodos tradicionales GET, PUT, POST, DELETE
- Permite que servidores antiguos se comuniquen con clientes actuales y que a su vez los servidores nuevos puedan mantenerse en el tiempo.
- Permite compatibilidad con componentes intermedios como es el caso de caches, proxys, firewalls, Gateway. Al ser compatibles con estos intermediarios, permite reducir la latencia de interacción, reforzar la seguridad y encapsular otros sistemas.

REST funciona de la siguiente manera:

- Identifica y manipula los recursos a través de representaciones, ya que utiliza HTTP, que es un protocolo centrado en URIs. Cómo los recursos son objetos lógicos a los que se les envían mensajes, estos recursos no pueden ser accedidos o modificados, por eso se trabaja con representaciones de ellos. Las representaciones son el estado del recurso que nos gustaría que fuera. Internamente el estado del recurso puede ser cualquier cosa, desde una base de datos relacional hasta un fichero de texto. Como se puede ver en la figura 2.31.
- REST quiere mensajes autodescriptivos, dicta que los mensajes HTTP deberían ser tan descriptivos como sea posible. Esto permite que los intermediarios interpreten los mensajes y ejecuten servicios en nombre del usuario. HTTP logra esto por medio de métodos estándares, encabezamientos y mecanismos de direccionamiento.
- REST utiliza hipermedia como un mecanismo del estado de la aplicación. El servidor conoce el estado de sus recursos, aunque no intenta seguirle la pista a las sesiones individuales. De este modo el estado de la aplicación Web es capturada en uno o varios documentos de hipertexto que residen en el cliente y en le servidor.

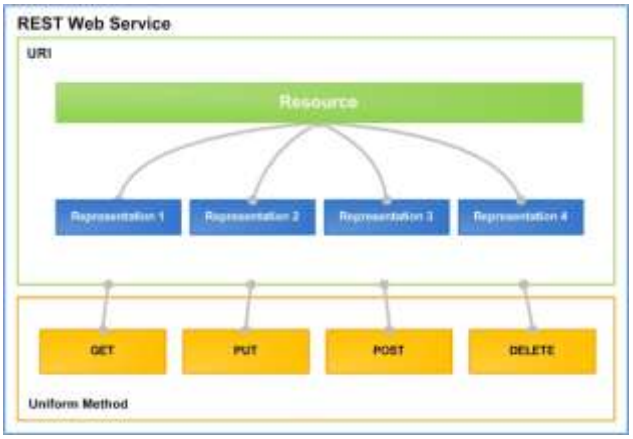


Figura 2.31: REST Web Service

2.7.1 JSON

JSON (*Java Script Object Notation*) es un formato ligero de intercambio de datos que es simple interpretarlo y generarlo para las máquinas. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, Java Script, Perl, Phyton etc. Estas propiedades hacen a JSON un lenguaje ideal para el intercambio de datos.

JSON está construido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocidos como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras. [27]

2.8 Seguridad SSL y TLS

El protocolo de Capa de Socket Segura (*Secure Socket Layer*, SSL) fue desarrollado originalmente por Netscape Corporation y se propuso como un estándar diseñado específicamente para resolver las necesidades de se seguridad. La versión 3 del protocolo se diseñó con la participación pública y con el apoyo de la industria y se publicó como borrador de internet. Cuando se alcanzó el conceso de presentar el protocolo para su estandarización se creó el grupo de trabajo TLS dentro de la IETF con el objetivo de desarrollar un estándar común. La primera versión publicada de TLS puede verse como SSL v3.1 y es muy similar y compatible con SSL v3. [28 y29]

La mayor parte de esta sección esta dedicada a SSL v3. Al final de la misma se describen las principales diferencias entre SSLv3 y TLS.

2.8.1 Arquitectura SSL

SSL está diseñado de forma que utilice TCP para proporcionar un servicio fiable y seguro extremo a extremo. SSL no es un protocolo simple si no que tiene dos niveles de protocolos.

El protocolo Record de SSL proporciona servicios de seguridad básica a varios protocolos de nivel más alto, en particular el HTTP. Como se puede observar en la figura 2.32, se definen tres protocolos de nivel más alto como parte de SSL *Handshake Protocol*, *Change Cipher Spec Protocol* y *Alert Protocol*.

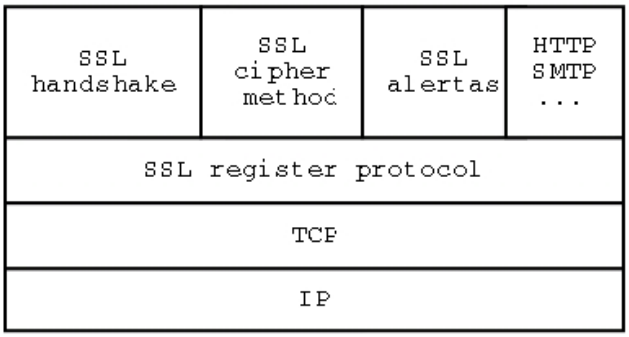


Figura 2.32: Protocolos SSL.

Dos conceptos importantes de SSL son la sesión SSL y la conexión SSL, definidos en las especificaciones de la siguiente manera:

- Conexión: una conexión es un transporte que proporciona un tipo de servicio idóneo. Para SSL, tales conexiones son relaciones de igual a igual. Las conexiones son transitorias. Cada conexión esta asociada con una sesión.
- Sesión: una sesión SSL es una asociación entre un cliente y un servidor. Las sesiones la crea el protocolo *Handshake* o el apretón de manos. Las sesiones definen un conjunto de parámetros criptográficos de seguridad, que se pueden compartir entre múltiples conexiones. Se usan para evitar el costoso proceso de negociación de parámetros de seguridad para la conexión.

Una fase de sesión se define por los siguientes parámetros:

- Identificación de sesión: secuencia arbitraria de bytes elegida por el servidor para identificar un estado de sesión activo o reanudable.
- Certificado de la entidad par: certificado X509.v3 del par.
- Método de compresión: el algoritmo usado para comprimir datos antes del cifrado.
- Especificación del cifrado: especifica el grueso del algoritmo de cifrado y un algoritmo hash usado para el cálculo del MAC.
- Clave maestra: contraseña de 48 bytes compartida entre cliente y servidor.
- Es reanudable: indicador que refleja si la sesión se puede utilizar para iniciar nuevas conexiones.

Un estado de conexión se define por los siguientes parámetros:

- Valores aleatorios del servidor y del cliente: secuencias de bytes elegidas por el servidor y el cliente para cada conexión.
- Clave secreta para MAC de escritura del servidor: la clave secreta usada en operaciones MAC sobre datos enviados por el servidor.
- Clave secreta para MAC de escritura del cliente: la clave secreta usada en operaciones MAC sobre datos enviados por el cliente.
- Clave de escritura del servidor: la clave de cifrado convencional para los datos cifrados por el servidor y descifrardos por el cliente.
- Clave de escritura del cliente: la clave de cifrado convecional para los datos cifrados por el cliente y descifrados por el servidor.
- Vector de inicialización (IV): cuando se usa un cifrador de bloque en modo CBC, se necesita un vector de inicialización para cada clave.
- Números de secuencia: cada parte mantiene números de secuencia separados para los mensajes transmitidos y recibidos en cada conexión.

2.8.2 Protocolo Record de SSL

El protocolo *Record* proporciona dos servicios a las conexiones SSL:

- Confidencialidad: el protocolo *Handshake* define una clave secreta compartida que se usa para cifrado convencional de carga útil de SSL.
- Integridad de mensajes: el protocolo *Handshake* también define una clave secreta compartida que se usa para formar un código de autentificación de mensajes.

La figura 2.33 indica las diferentes operaciones del protocolo *Record*. El protocolo toma un mensaje de la aplicación que se va a transferir, fragmenta los datos en bloques manejables, opcionalmente los comprime, le añade un MAC, realiza el cifrado, le añade una cabecera y transmite la unidad resultante en un segmento TCP. Los datos recibidos se descifran, se verifican, se descomprimen y se ensamblan. Luego se nevían a la aplicación receptora en el nivel superior.

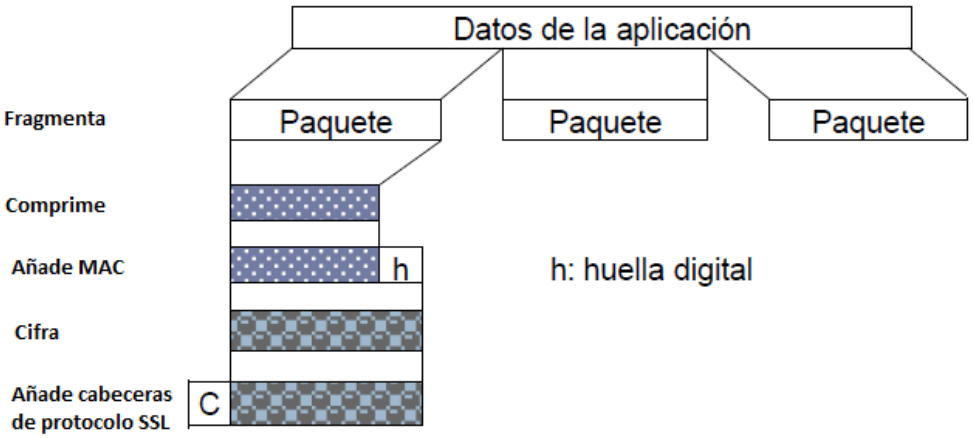


Figura 2.33: Fases protocolo Record

El primer paso es la fragmentación, cada mensaje se fragmenta en bloques de 2^{14} bytes o más pequeños. Después opcionalmente se comprime y se debe realizar sin perdidas y no debe incrementar la longitud de los contenidos en más de 1024 bytes. El siguiente paso es calcular un código de autentificación de mensaje (MAC) sobre los datos comprimidos. Para este propósito se usa una clave secreta compartida. Lo siguiente es cifrar el mensaje comprimido más el MAC usando cifrado simétrico. El cifrado no puede incrementar la longitud del contenido en más de 1024 bytes para que la longitud total no exceda de 2^{14} + 2048. Permite algunos algoritmos de cifrado de bloque como IDEA, RC2-40, DES-40, DES, 3DES y Fortezza; y cifradores de flujo como RC4-40 y RC4-128.

El paso final consite en aladir una cabecera compuesta por los siguientes campos:

- Tipo de contenido (8 bits): el protocolo de nivel superior usado para procesar el fragmento interno.
- Versión mayor (8 bits): indica la versión mayor de SSL en uso. En SSL es 3.
- Versión menor (8 bits): indica la versión menor en uso. En SSL es 0.
- Longitud de compresión (16 bits): la longitud en bytes del fragmento de texto en claro o comprimido. El valor máximo es 2^{14} + 2048.

2.8.3 Protocolo Change Cipher Spec

El protocolo *Change Cipher Spec* es uno de los tres protocolos específicos de SSL que utilizan el protocolo *Record* y es el más simple. Este protocolo se compone de un único mensaje, que

contiene un único byte con el valor 1. El único propósito de este mensaje es hacer que un estado pendiente se copie en el estado operativo, lo cual actualiza la suite de cifrado que se usará en esta conexión.

2.8.4 Protocolo Alert

El protocolo *Alert* se usa para transmitir las alertas relacionadas con SSL a la entidad par. Los mensajes de alerta se comprimen y se cifran, según se especifica en el estado en operativo. Cada mensaje está formado por dos bytes. El primer byte toma el valor de aviso (1) o fatal (2) para saber la gravedad del mensaje. Si el nivel es fatal SSL termina la conexión inmediatamente. El segundo byte contiene un código que indica la alerta específica. Primero se van a enumerar las alertas que siempre son fatales:

- Mensaje inesperado: se recibió un mensaje inapropiado.
- Mac de registro erróneo: se recibió un MAC incorrecto.
- Fallo de descompresión: entrada inadecuada en la función de descompresión.
- Fallo de negociación: el emisor fue incapaz de negociar un conjunto de parámetros de seguridad aceptables.
- Parámetro ilegal: un campo en un mensaje de negociación estaba fuera incapaz de rango o era inconsistente con otros campos.

El resto de alertas son las siguientes:

- Notificación de cierre: notifica al receptor que el emisor no enviará más mensajes en esta conexión.
- No verificado: puede enviarse como respuesta a una solicitud de certificado cuando no hay certificado disponible.
- Certificado erróneo: un certificado recibido no es correcto.
- Certificado no permitido: el tipo de certificado recibido no está permitido.
- Certificado revocado: un certificado ha sido revocado por su firmante.
- Certificado caducado: un certificado ha sido caducado.
- Certificado desconocido: algún otro asunto no especificado se detectó durante el tratamiento del certificado, considerándose éste inaceptable.

2.8.5 Protocolo Handshake

La parte más compleja del SSL es el protocolo *Handshake*. Este protocolo permite la autenticación mutua de servidor y cliente; negociar un algoritmo de cifrado y de cálculo del MAC y las claves criptográficas que se utilizarán para proteger los datos enviados en un registro SSL. Este protocolo *Handshake* se utiliza antes de que se transmita cualquier dato de aplicación.

El protocolo *Handshake* consiste en una serie de mensajes intercambiados entre servidor y cliente. Todos los mensajes tienen los siguientes campos:

- Tipo (1 byte): indica uno de los 10 mensajes que tiene definidos SSL.
- Longitud (3 bytes): la longitud del mensaje en bytes.
- Contenido (mayor o igual a 1 byte): los parámetros asociados con el mensaje.

El protocolo *Handshake* consta de 4 fases:

- Establecimiento de las capacidades de seguridad, incluyendo versión del protocolo, ID de sesión, suite de cifrado, método de compresión y los números aleatorios iniciales.

- El servidor puede enviar un certificado, intercambio de clave y solicitud de certificado. El servidor señala el final de la fase del mensaje "*hello*".
- El cliente envía certificado, si se le solicita el intercambio de clave y puede que envíe la verificación de certificado.
- Intercambio de suite de cifrado y finalización de protocolo *Handshake*.

2.8.6 TLS

TLS (*Transport Layer Security*) es una iniciativa de estandarización de la IETF cuyo objetivo es producir una versión estándar de internet de SSL. TLS se define como un Estándar Propuesto de Internet en el RFC 2246. El RFC 2246 es muy similar a SSLv3. En esta sección se destacarán las diferencias. [28]

- Número de versión:

El formato del registro de TLS es el mismo que el de SSL, y los campos de cabecera tienen el mismo significado. La única diferencia radica en los valores de versión. En la versión actual de TLS, la versión mayor es 3 y la versión menor es 1.

- Código de Autentificación de Mensaje

Hay dos diferencias entre los esquemas MAC de SSLv3 y TLS. TLS utiliza el algoritmo HMAC definido en el RFC 2104. SSL v3 usa el mismo el mismo algoritmo, excepto que los bytes de relleno se concatenan con la clave secreta en vez de aplicárseles el XOR con la clave secreta rellena hasta la longitud del bloque. El nivel de seguridad debería ser el mismo en ambos casos. TLS añade un campo más al calculo del MAC, la versión del protocolo que se esta empleando.

- Función pseudoaleatoria

TLS usa una función pseudoaletoria conocida PRF (*PseudoRandom Function*) para expandir valores secretos en bloques de datos útiles para generación de claves o validación. El objetivo es utilizar un valor secreto compartido relativamente pequeño, pero generar bloques de datos más grandes de forma segura contra los ataques a las funciones hash y a los MAC.

- Suites de cifrado

Hay pequeñas diferencias entre las suites de cifrado disponibles en SSLv3 y en TLS:
  - Intercambio de clave: TLS permite todas las técnicas de intercambio de claves permitidas en SSL v3 con la excepción de Fortezza.
  - Algoritmos de cifrado simétrico: TLS incluye todos los algoritmos de cifrado simétricos encontrados en SSLv3, con la excepción de Fortezza.

- Mensajes *certificate_verify* y *finished*

En el mensaje certificate_verify de TLS, los hash MD5 y SHA-1 se calculan solamente sobre los mensajes de negociación. Para SSL el cálculo del hash también incluía la clave maestra y valores de relleno. Estos campos parecen no suponer mayor seguridad.

Como con el mensaje de finalización en SSL, TLS es un hash basado en la clave maestra compartida, los mensajes previos de la negociación y una etiqueta que identifica al servidor o al cliente. Los cálculos son ligeramente diferentes.

- Relleno

En SSL la cantidad de relleno añadido antes del cifrado de los datos de usuario es la mínima necesaria, de manera que el tamaño total de los datos que se van a cifrar es un múltiplo de la longitud del bloque de cifrado. En TLS la cantidad se relleno puede ser cualquiera cantidad, como máximo hasta 255 bytes, para que el total sea múltiplo de la longitud del bloque de cifrado.

2.9 Conclusión

En este apartado se compararán las características fundamentales de las tecnologías citadas en el Estado de la Cuestión.

	Rendimiento	**Facilidad de uso**	**Número de aplicaciones que lo usan**	**Limitación a una plataforma**	**Fácil de diseñar**
Sockets	Los Sockets tienen un rendimiento mejor al producir menos sobrecarga.	Son fáciles de utilizar pero a nivel de aplicación no son tan usables.	Son usados en SSL, WebSockets…	Están presentes en la mayoría de las versiones de UNIX, incluido Linux y también Windows NT y Macintosh OS.	Son fáciles de desarrollar debido a que están presentes en muchas plataformas y hay bastante documentación de ellos.
RPC	Los Sockets tienen un mejor rendimiento que las RPC. Tienen un mejor rendimiento que RMI.	Tiene una norma muy simple que hace que sea muy fácil de utilizar.	RPC son usadas en tecnologías como NFS y en NIS.	RPC están basadas en C.	Abstrae la comunicación a nivel de invocación de procedimientos, por lo que son adecuadas para programación estructurada basada en librerías
RMI	Tienen un rendimiento mejor que los Servicios Web SOAP. Pero peor que los Sockets. RMI sólo tiene un mejor rendimiento si utiliza hilos para invocar varios métodos simultáneos.	Son simples y fáciles de utilizar.	No se han encontrado.	RMI esta basado en Java.	Java RMI es un mecanismo que facilita el desarrollo para cualquier programador acostumbrado a usar Java.
XML-RPC	XML-RPC posee un rendimiento muy parecido a RPC. Ya que utiliza	Tiene una norma muy simple que hace que sea	Aplicaciones que utilizan XML-RPC son Wordpress para iPhone y Android;	Existen implementaciones de XML-RPC para varios sistemas operativos,	La simplicidad de XML-RPC es su mayor característica que

	el protocolo RPC.	muy fácil de utilizar.	Windows Live Writer, WB Editor, Deepest Sender.	lenguajes de programación, licencias comerciales y de software libre como: C/C++, Java, Microsoft .NET, Perl, PHP…	hace que sea fácil de implementar y depurar.
Servicios Web (SOAP)	Su rendimiento es bajo comparado con otras tecnologías como REST o RMI.	Es muy fácil de utilizar.	Amazon pose un estilo de Web Service con SOAP. Mapquest, UPS, FedEx…	Los Servicios Web SOAP, están soportados por la mayoría de tecnologías. Una de sus características fundamentales es que añada interoperabilidad entre plataformas.	El concepto es fácil de entender, incluso existen toolkits de vendedores como IBM o Microsoft que permiten a los desarrolladores crearlos rápido y fácil.
Servicios Web (REST)	REST ofrece un bajo consumo de recursos y ofrece una gran escalabilidad.	Es algo más complejo de utilizar que SOAP porque las URIs pueden resultar liosas para el usuario.	Amazon posee un estilo de Web Service con REST, eBay posee una interfaz REST, Yahoo y Dropbox entre otras ofrecen APIs REST para desarrolladores.	Los Servicios Web REST, están soportados por la mayoría de las tecnologías. Una de sus características fundamentales es que añada interoperabilidad entre plataformas.	Una de sus características principales es que es muy fácil de construir y acoplar.

Tabla 2.2: Comparativa de tecnologías.

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. [33]

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

El objetivo principal del presente Trabajo de Fin de Grado es el análisis y diseño de una API y una aplicación Android de VoCS. Se realizará el desarrollo de un prototipo ágil que cumpla con la funcionalidad básica del sistema analizado. Se va a optar por una metodología ágil ya que se debe tener en cuenta las características del prototipo: ágil y orientado a validación de análisis y diseño.

Para la metodología se va a utilizar Scrum, a continuación se explicarán las características de susodicha aplicación.

4.1 Scrum

Scrum es una metodología ágil que presenta un proceso adaptativo, rápido y auto-organizado de desarrollo de productos. Fue presentada en 1986 por los japoneses Nonaka y Takeuchi en el artículo "The New Proudct Development Game". Scrum surgió como práctica en el desarrollo de productos tecnológicos y en 1933 Jeff Sutherland aplicó el modelo al desarrollo de software en la Easel Corporation.

En Scrum se identifican tres fases: planificación del Sprint, seguimiento del Sprint y revisión del Sprint. La figura 3.1 muestra los pasos que se siguen en Scrum.

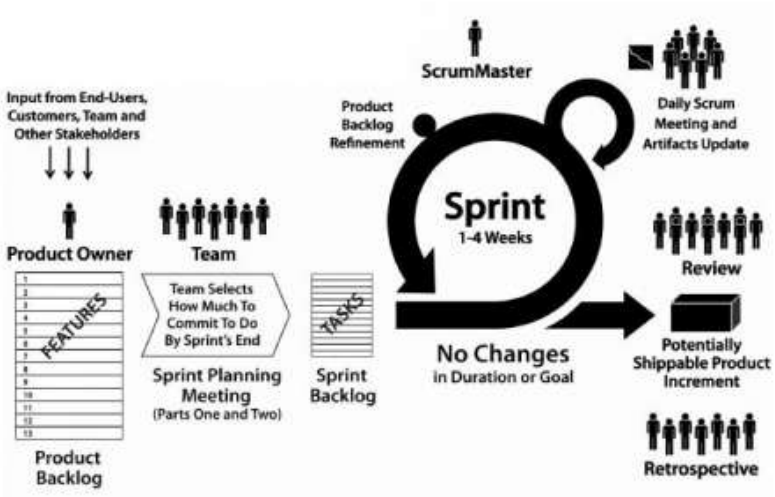


Figura 4.1: Fases de Scrum. [34]

- Planificación del Sprint: En esta fase se define el Product Backlog el cual consiste en una lista de requisitos del sistema que puede ser actualizada. En cada iteración, se revisa el Product Backlog. También se realiza la planificación del primer Sprint, en la cual se determinan cuales son los objetivos y el trabajo que se tienen que realizar en esa iteración. La lista de tareas obtenidas se denomina Sprint Backlog.
- Seguimiento del Sprint: En esta fase se llevan a cabo reuniones diarias para comprobar el avance de las tareas que se tienen que realizar del Sprint. En estas reuniones están presentes el Scrum Master y el equipo de desarrollo, se suelen preguntar que trabajo se ha

realizado desde la reunión anterior, que trabajo se va a hacer hasta la próxima reunión y por último que impedimentos se han tenido a la hora de realizar estas tareas.
- Revisión del Sprint: En esta fase se realiza un análisis y revisión del Sprint terminado. Se presentan los resultados finales y si se tiene una demo se presenta. La demo es importante ya que se mejora el feedback con los interesados, hay un reconocimiento del trabajo y un esfuerzo por finalizar las cosas o un correctivo en caso de ser una demo mala.

Para que esta metodología sea efectiva es necesario tener una serie de roles adquiridos a la hora de realizar el proyecto.

- Propietario del producto: es la persona involucrada del proyecto que conoce el entorno de negocio del cliente y es el responsable de obtener el mejor resultado posible para el cliente. Proporciona la financiación necesaria, toma las decisiones que afectan en el resultado final, elige fechas de lanzamiento y el retorno de inversión.
- Scrum Master: es el encargado de garantizar el funcionamiento de los procesos y de la metodología. En el Scrum Master recae toda la responsabilidad de funcionamiento de modelo y es fundamental elegir una persona cualificada. El Scrum Master tiene que interactuar con el equipo de desarrollo, con el cliente y con los gestores.
- Equipo de desarrollo: Es el equipo encargado de desarrollar el sistema y tiene la autoridad de decidir las acciones necesarias y auto-organizarse con la finalidad de alcanzar los objetivos del Sprint. El equipo se involucra en la planificación del Product Backlog, en la creación de los Sprint etc.
- El cliente participa en la creación del Product Backlog.
- El gestor: se encarga de la toma de decisiones finales, participa en la elección de objetivos y requisitos, elige al propietario del producto y junto con el Scrum Master se encarga de reducir si es necesario el Product Backlog.

A continuación se exponen los productos necesarios para llevar a cabo el proyecto:

- El Product Backlog es donde se reflejan los requisitos y funcionalidades del sistema y lo que se va a hacer a lo largo de este proyecto.
- El Sprint Backlog es donde se reflejan las tareas que tienen que hacer el equipo de desarrollo.
- El Sprint Planning Meeting tiene como objetivo planificar el Sprint a partir del Product Backlog.
- El Sprint Review tiene como objetivo realizar las pruebas necesarias para comprobar que el Sprint se ha realizado correctamente.
- El Sprint Retrospective tiene como objetivo revisar que se han cumplido los objetivos y se remarcarán los aspectos positivos y negativos del Sprint realizado.

Una vez explicada la metodología, comenzamos a explicar que se realizó en cada fase:

Planificación del Sprint

- Se realiza la planificación, que se recoge en el diagrama de Gaant.
- Se realiza el Product Backlog, que se puede observar en el apartado Análisis y diseño del sistema.
- Se realiza el Sprint Planning Meeting donde se detalla que se realizará un único Sprint por lo que todas las funcionalidades, tareas y requerimientos obtenidos en el Product Backlog se harán en susodicho Sprint.

Seguimiento del Sprint

En esta fase se realizará el desarrollo del sistema. El apartado Desarrollo explica como se desarrollaron todas las funcionalidades, tareas y requerimientos del sistema VoCS.

Revisión del Sprint

- Se realiza el Sprint Review que se puede observar en el apartado Análisis de Rendimiento, donde se realizarán pruebas del sistema para comprobar su correcto funcionamiento y el rendimiento que proporciona.
- Se realiza el Sprint Retrospective que se puede observar en el apartado de las Conclusiones. En este apartado se reflejan las cosas positivas y negativas del sistema, si se llegaron a alcanzar los objetivos propuestos y se realizarán propuestas de mejoras que se podrán realizar en trabajos futuros.

5 Análisis y Diseño

Este capítulo se compone del análisis y el diseño del Trabajo de Fin de Grado.

5.1 Análisis

En este apartado se expone el análisis realizado. Dicho análisis está formado por los casos de uso de la aplicación VoCS, los requisitos obtenidos a partir de dichos casos de uso.

5.1.1 Casos de Uso

En este apartado se muestra una representación de la funcionalidad del sistema a analizar, diseñar y desarrollar.

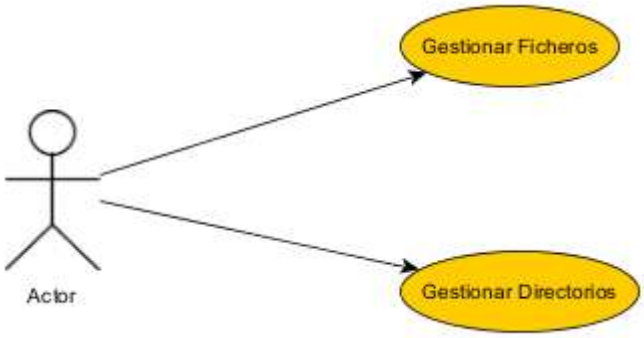


Figura 5.1: Diagrama de casos de uso.

Como se muestra en la figura 5.1 se pueden observar los casos de uso Gestionar Ficheros y Gestionar Directorios que son dos casos de uso generales. En las figuras 5.2 y 5.3 se pueden observar los casos de uso de la figura 5.1 desglosados.

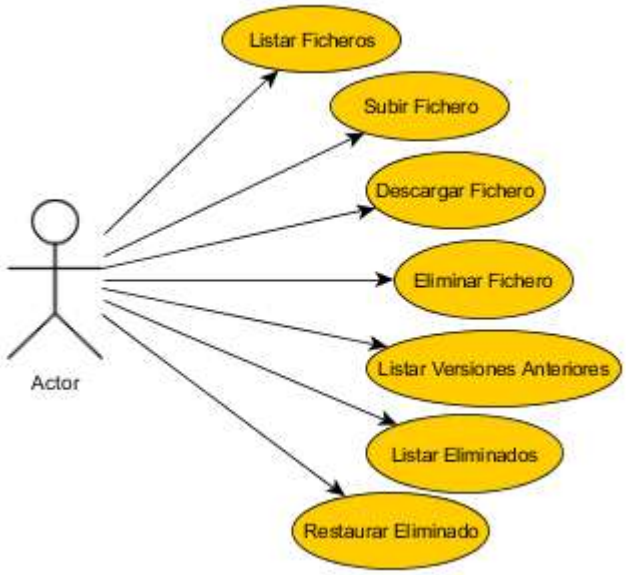


Figura 5.2: Cases de uso de Gestionar Ficheros.

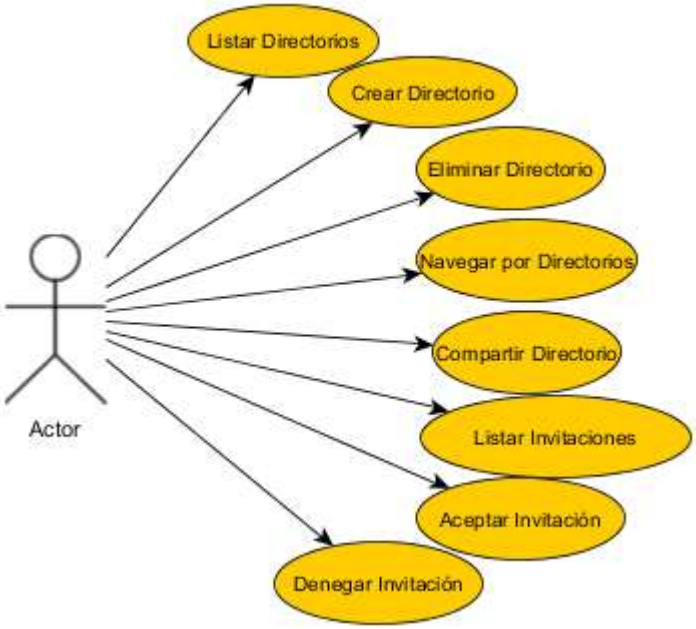


Figura 5.3: Cases de uso de Gestionar Directorios.

5.1.2 Requisitos

Tras el análisis de los casos de uso expuestos en la sección anterior, se han extraído los requisitos de la aplicación VoCS. Dichos resquicitos se dividen en Requisitos de Usuario y 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) o Restricción (lo que no puede hacer).
- Necesidad: describe si el requisito es esencial u opcional.
- Prioridad: describe la prioridad de implementación del requisito. Posibles valores: Baja, Media y Alta
- Verificabilidad: describe en qué medida el requisito es verificable. Posibles valores: Baja, Media y Alta.

RU01: Registrar Usuario			
Descripción	El usuario podrá registrarse en la aplicación.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.1: Requisito RU01: Registrar Usuario.

RU02: Loguear Usuario			
Descripción	El usuario podrá autenticarse en la aplicación.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.2: Requisito RU02: Loguear Usuario.

RU03: Salir			
Descripción	El usuario podrá salir de la aplicación.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.3: Requisito RU03: Salir.

RU04: Listar Ficheros	
Descripción	El usuario podrá listar los ficheros que tiene alojados en el servidor.

Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.4: Requisito RU04: Listar Fichero.

RU05: Subir Fichero S0			
Descripción	El usuario podrá subir un fichero que este alojado en el dispositivo móvil con un nivel de seguridad 0.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.5: Requisito RU05: Subir Fichero S0.

RU06: Subir Fichero S1			
Descripción	El usuario podrá subir un fichero que este alojado en el dispositivo móvil con un nivel de seguridad 1.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.6: Requisito RU06: Subir Fichero S1.

RU07: Subir Fichero S2			
Descripción	El usuario podrá subir un fichero que este alojado en el dispositivo móvil con un nivel de seguridad 2.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.7: Requisito RU 07: Subir Fichero S2.

RU08: Tamaño			
Descripción	El usuario no podrá subir un fichero si este ocupa más del tamaño disponible que tiene el usuario.		
Tipo	Restricción	**Necesidad**	Esencial
Prioridad	Media	**Verificabilidad**	Alta

Tabla 5.8: Requisito RU08: Tamaño.

RU09: Descargar Fichero			
Descripción	El usuario podrá descargar un fichero que haya subido con anterioridad.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.9: Requisito RU09: Descargar Fichero.

RU10: Eliminar Fichero			
Descripción	El usuario podrá eliminar un fichero que haya subido con anterioridad.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.10: Requisito RU10: Eliminar Fichero.

RU11: Listar Versiones

Descripción	El usuario podrá listar las versiones anteriores de un fichero, en caso de que tuviera alguna.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.11: Requisito RU11: Listar Versiones.

RU12: Listar Eliminados			
Descripción	El usuario podrá listar los ficheros eliminados por el usuario, en caso de que tuviera alguno.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.12: Requisito RU12: Listar Eliminados.

RU13: Eliminados 15 días			
Descripción	El usuario no podrá listar los ficheros eliminados que estén más de 15 días en el servidor.		
Tipo	Restricción	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.13: Requisito RU13: Eliminados 15 días.

RU14: Restaurar Fichero			
Descripción	El usuario podrá restaurar la versión anterior de un fichero o podrá restaurar un fichero que halla sido borrado.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.14: Requisito RU14: Restaurar Fichero.

RU15: Listar Directorios			
Descripción	El usuario podrá listar los directorios que tiene alojados en el servidor.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.15: Requisito RU15: Listar Directorios.

RU16: Crear Directorio			
Descripción	El usuario podrá crear un directorio.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.16: Requisito RU16: Crear Directorio.

RU17: Eliminar Directorio			
Descripción	El usuario podrá eliminar el directorio con todo su contenido.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.17: Requisito RU17: Eliminar Directorio.

RU18: Directorio Raíz			
Descripción	El usuario no podrá eliminar el directorio raíz.		
Tipo	Restricción	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.18: Requisito RU18: Directorio Raíz.

RU19: Recuperar Directorio			
Descripción	El usuario no podrá recuperar el directorio eliminado.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.19: Requisito RU19: Recuperar Directorio.

RU20: Navegar por los Directorios			
Descripción	El usuario podrá navegar por los directorios que el usuario haya creado.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.20: Requisito RU20: Navegar por los Directorios.

RU21: Compartir Directorio			
Descripción	El usuario podrá compartir con otro usuario un directorio con todo su contenido. Mandando una invitación.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Media	**Verificabilidad**	Alta

Tabla 5.21: Requisito RU21: Compartir Directorio.

RU22: Compartir Directorio			
Descripción	El usuario no podrá compartir con un usuario que no este registrado en la aplicación.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Media	**Verificabilidad**	Alta

Tabla 5.22: Requisito RU22: Compartir Directorio.

RU23: Listar Invitaciones			
Descripción	El usuario podrá listar las invitaciones que haya recibido por parte de otro usuario que quiere compartir un directorio.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Media	**Verificabilidad**	Alta

Tabla 5.23: Requisito RU23: Listar Invitaciones.

RU24: Aceptar Invitación			
Descripción	El usuario podrá aceptar la invitación para compartir un directorio de otro usuario.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Media	**Verificabilidad**	Alta

Tabla 5.24: Requisito RU24: Aceptar Invitación.

RU25: Denegar Invitación			
Descripción	El usuario podrá denegar la invitación para no compartir un directorio de otro usuario.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Media	**Verificabilidad**	Alta

Tabla 5.25: Requisito RU25: Denegar Invitación.

RU26: Recuperar Invitación			
Descripción	El usuario no podrá recuperar una invitación que haya denegado.		
Tipo	Capacidad	**Necesidad**	Esencial
Prioridad	Media	**Verificabilidad**	Alta

Tabla 5.26: Requisito RU26: Recuperar Invitación.

Matriz de trazabilidad: Casos de Uso-Requisitos de usuario.

	CU1: Listar Ficheros	CU2: Subir Fichero	CU:3 Descargar Fichero	CU:4 Eliminar Fichero	CU:5 Listar Versiones Anteriores	CU:6 Listar Eliminados	CU:7 Restaurar Eliminados	CU8: Listar Directorios	CU9: Crear Directorio	CU10: Eliminar Directorio	CU11: Navegar por Directorios	CU:12 Compartir Directorio	CU:13 Listar Invitaciones	CU14: Aceptar Invitación	CU15: Denegar Invitación
RU01	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
RU03	X	X	X	X	X	X	X	X	X	X	X	X	X	X	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

Tabla 5.27: Matriz de trazabilidad: Casos de Uso-Requisitos de usuario.

5.1.2.2 Requisitos de 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 'RU' seguidas del número de requisito.
- Descripción: contiene una explicación más amplia 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. Posibles valores: Baja, Media y Alta
- Verificabilidad: describe en qué medida el requisito es verificable. Posibles valores: Baja, Media y Alta

RS01			
Descripción	El sistema deberá proveer un formulario de registro con los siguientes campos: • Nick (texto) • Nombre (texto) • Apellidos (texto) • Email (texto) • DNI (texto) • Contraseña (texto) • Captcha (texto)		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.28: Requisito RS01.

RS02			
Descripción	El sistema no permitirá registrarse a un usuario que introduzca un Nick que ya exista en el sistema.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.29: Requisito RS02.

RS03			
Descripción	El sistema deberá proveer de un formulario de autenticación con los siguientes campos: Nick (texto) Contraseña (texto)		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.30: Requisito RS03.

RS04	
Descripción	Al introducir la contraseña en el campo correspondiente, el texto de la

	misma será oculto.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.31: Requisito RS04.

RS05			
Descripción	La información relativa a los usuarios se almacenará en la BBDD		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.32: Requisito RS05.

RS06			
Descripción	El software mostrará con texto en la pantalla principal cuánto espacio ha ocupado y su espacio máximo en MB, acto seguido mostrará una barra progresiva que indicará al usuario cuanto espacio ha consumido del total.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.33: Requisito RS06.

RS07			
Descripción	El software mostrará en la pantalla principal los directorios del usuario y después los ficheros almacenados en el directorio raíz en forma de lista.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.34: Requisito RS07.

RS08			
Descripción	El software permitirá a los usuarios que tengan muchos directorios y ficheros que la pantalla se pueda desplazar hacia arriba para mostrarlos todos.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.35: Requisito RS08.

RS09			
Descripción	El software mostrará un menú con las siguientes opciones: • Subir • Crear directorio • Opciones • Salir		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.36: Requisito RS09.

RS10			
Descripción	El software sólo deberá mostrar en la pantalla principal el menú.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.37: Requisito RS10.

RS11			
Descripción	El software deberá mostrar opciones de navegación cuando se navegue por un directorio que no sea el directorio raíz.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.38: Requisito RS11.

RS12			
Descripción	El software deberá permitir al usuario poder navegar por el sistema de ficheros.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.39: Requisito RS12.

RS13			
Descripción	El software deberá mostrar al usuario el sistema de ficheros del teléfono.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.40: Requisito RS13.

RS14			
Descripción	El software deberá dar la opción de elegir varios ficheros para la subida.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.41: Requisito RS14.

RS15			
Descripción	El software deberá dar la opción de elegir entre distintos niveles de seguridad: • 0 • 1 • 2		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.42: Requisito RS15.

RS16

Descripción	El software deberá replicar un fichero si el nivel elegido es 1 o 2.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.43: Requisito RS16.

RS17			
Descripción	El software deberá cifrar un fichero si el nivel elegido es 2.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.44: Requisito RS17.

RS18			
Descripción	El software deberá dar la opción de eliminar algún fichero seleccionado para la subida.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.45: Requisito RS18.

RS19			
Descripción	El software deberá poder subir uno o varios ficheros seleccionados por el usuario.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.46: Requisito RS19.

RS20			
Descripción	El software no permitirá que el usuario sobrepase el tamaño máximo que tiene asignado.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.47: Requisito RS20.

RS21			
Descripción	El software por defecto introducirá un tamaño máximo de 1024 MB por usuario.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.48: Requisito RS21.

RS22			
Descripción	El software deberá poner un fichero a inactivo cuando llegue una petición de eliminar fichero.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.49: Requisito RS22.

RS23			
Descripción	El software deberá eliminar un fichero del disco duro pasados 15 días de estar inactivo.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.50: Requisito RS23.

RS24			
Descripción	El software deberá eliminar un fichero de la BBDD pasados 15 días de estar inactivo.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.51: Requisito RS24.

RS25			
Descripción	El software deberá almacenar el fichero en el disco duro.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.52: Requisito RS25.

RS26			
Descripción	El software deberá mostrar la lista de versiones anteriores de un fichero solicitado y la fecha en la que fue remplazado por otro.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.53: Requisito RS26.

RS27			
Descripción	El software deberá mostrar la lista de ficheros eliminados y la fecha en la que fueron borrados.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.54: Requisito RS27.

RS28			
Descripción	El software deberá restaurar un fichero poniéndolo activo, borrando su fecha de borrado y si hay una versión activa del mismo fichero, ponerla inactiva y asignarla al nuevo fichero.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.55: Requisito RS28.

RS29			
Descripción	El software deberá permitir crear un nuevo directorio.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.56: Requisito RS29.

RS30			
Descripción	El software deberá incluir un campo para poder introducir el nombre del nuevo directorio creado.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.57: Requisito RS30.

RS31			
Descripción	El software deberá poder eliminar el directorio seleccionado y a su vez eliminar los directorios internos y los ficheros se pondrán inactivos.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.58: Requisito RS31.

RS32			
Descripción	El software no permitirá eliminar el directorio raíz.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.59: Requisito RS32.

RS33			
Descripción	El sistema no podrá restaurar un directorio eliminado.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.60: Requisito RS33.

RS34			
Descripción	El sistema podrá restaurar en el directorio raíz los ficheros eliminados por la eliminación de un directorio.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.61: Requisito RS34.

RS35			
Descripción	El sistema podrá restaurar en el directorio raíz los ficheros eliminados por la eliminación de un directorio.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.62: Requisito RS35.

RS36			
Descripción	El sistema permitirá enviar una invitación para compartir un directorio con un usuario.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.63: Requisito RS36.

RS37			
Descripción	El sistema deberá incluir un campo para que el usuario pueda introducir el email del usuario con el que quiera compartir.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.64: Requisito RS37.

RS38			
Descripción	El sistema sólo mandará la invitación en caso de que el email proporcionado este relacionado con un usuario.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.65: Requisito RS38.

RS39			
Descripción	El sistema mostrará las invitaciones que tiene pendientes el usuario.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.66: Requisito RS39.

RS40			
Descripción	El sistema permitirá aceptar la invitación.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.67: Requisito RS40.

RS41			
Descripción	El sistema mostrará el nuevo directorio con los directorios y ficheros que contiene.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.68: Requisito RS41.

RS42			
Descripción	El sistema permitirá denegar la invitación.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.69: Requisito RS42.

RS43			
Descripción	El sistema eliminará de la lista de invitaciones la invitación denegada y aceptada.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.70: Requisito RS43.

RS44			
Descripción	El sistema no permitirá recuperar una invitación que haya sido denegada.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.71: Requisito RS44.

RS45			
Descripción	El sistema deberá cambiar el nombre del fichero nuevo almacenado por uno aleatorio.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.72: Requisito RS45.

RS46			
Descripción	El sistema deberá agregar el directorio nuevo en la BBDD.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.73: Requisito RS46.

RS47			
Descripción	El sistema deberá eliminar el directorio de la BBDD.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.74: Requisito RS47.

RS48			
Descripción	El sistema deberá agregar la invitación nueva en la BBDD.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.75: Requisito RS48.

RS49			
Descripción	El sistema deberá eliminar la invitación solicitada de la BBDD.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.76: Requisito RS49.

RS50			
Descripción	El sistema permitirá descargarse un fichero que haya subido al sistema.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.77: Requisito RS50.

RS51			
Descripción	El sistema no permitirá descargarse un fichero de otro usuario si este no esta compartido.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.78: Requisito RS51.

RS52			
Descripción	El sistema no permitirá registrarse a un usuario que no introduzca todos los datos correctamente.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.79: Requisito RS52.

RS53			
Descripción	El sistema sólo permitirá acceder a la aplicación a los usuarios correctamente logueados.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.80: Requisito RS53.

RS54			
Descripción	El sistema no permitirá a un usuario borrar un fichero que no sea suyo.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.81: Requisito RS54.

RS55			
Descripción	El sistema no permitirá a un usuario borrar un directorio que no sea suyo, a no ser que este compartido.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.82: Requisito RS55.

RS56			
Descripción	El sistema no permitirá a un usuario navegar por un directorio que no sea suyo, a no ser que este compartido.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.83: Requisito RS56.

RS57			
Descripción	El sistema no permitirá a un usuario aceptar una invitación que no sea suya.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.84: Requisito RS57

Tabla 4.82: Requisito RS57.

RS58			
Descripción	El sistema no permitirá a un usuario denegar una invitación que no sea suya.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.85: Requisito RS58.

RS59			
Descripción	El sistema no permitirá a un usuario restaurar un fichero que no sea suyo.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.86: Requisito RS59.

RS60			
Descripción	El sistema no permitirá a un usuario listar las versiones anteriores de un fichero que no sea suyo.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.87: Requisito RS60.

RS61			
Descripción	El sistema no permitirá a un usuario listar los ficheros borrados que no sean suyos.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.88: Requisito RS61.

RS62			
Descripción	El sistema permitirá navegar por el sistema de ficheros del teléfono.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.89: Requisito RS62.

RS63			
Descripción	El sistema no mostrará los ficheros que no sean del usuario.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.90: Requisito RS63.

RS64			
Descripción	El sistema no mostrará los directorios que no sean del usuario.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.91: Requisito RS64.

RS65			
Descripción	El sistema deberá renovar la sesión automáticamente.		
Tipo	Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.92: Requisito RS65.

RS66			
Descripción	El sistema deberá incluir una interfaz gráfica con los colores corporativos y con el logo corporativo.		
Tipo	No Funcional	**Necesidad**	Esencial
Prioridad	Alta	**Verificabilidad**	Alta

Tabla 5.93: Requisito RS66.

Matriz de trazabilidad, Requisitos de Usuario-Requisitos de Software.

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
1	X																									
2	X																									
3		X																								
4		X																								
5	X																									
6								X																		
7				X											X											
8				X											X											
9			X		X	X	X									X										
10			X		X	X	X									X										
11																				X						
12																				X						
13					X	X	X																			
14					X	X	X																			
15					X	X	X																			
16						X	X																			

17							X																			
18					X	X	X																			
19					X	X	X																			
20								X																		
21								X																		
22										X																
23										X			X													
24										X			X													
25					X	X	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	X		
44																									X	
45					X	X	X																			
46																X										
47																	X									
48																					X					
49																							X	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	X	X																			
63				X																						
64															X											
65		X																								
66																										

Tabla 5.94: Matriz de trazabilidad.

5.2 Diseño

En este apartado se expondrá el diseño del sistema a desarrollar. Se definirán las opciones del entorno de construcción y cuales son las elegidas. A continuación se definirá el diseño de la arquitectura del sistema, el diseño del sistema de ficheros distribuido y por último el diseño de la replicación.

5.2.1 Entorno de Construcción

Al realizar el estudio y el análisis del sistema de almacenamiento en la nube, se contemplaron distintas opciones para realizar el diseño. A continuación se muestra el análisis de estas opciones, así como la opción escogida en cada caso, y las razones de dicha elección.

5.2.1.1 Plataforma Móvil

Dado que el sistema a desarrollar consta de un API y de una aplicación móvil; es necesario que el sistema operativo donde se implante la aplicación sea popular y proporcione un SDK fácil de implantar y de utilizar. Los sistemas operativos más populares ahora mismo y con más crecimiento son Android y iOS.

Para evaluar entre ambos se va a comparar, primero el Hardware que se necesita para empezar a trabajar, el lenguaje de programación que se usa y por último si es libre.

5.2.1.1.1 Hardware.

Para Android no se necesita ningún tipo de hardware ya que su SDK se puede usar en cualquier computadora y no se necesita ningún tipo de licencia. No es necesario ningún dispositivo móvil ya que el SDK incorpora una máquina virtual que simula el SO y poder realizar pruebas, pero como necesitaremos probar con Wi-Fi y con 3G para la toma de tiempo, necesitaremos un dispositivo Android.

En iOS si se necesita un computador con Mac OS X 10.6 o posterior e incluso para programar en iOS 5 se necesita la versión X Lion. Esto es obligatorio, sin Mac no se puede programar. Aunque hay otras tecnologías que permiten desarrollar para iOS como Adobe Flex que permiten su instalación en Windows. Como con Andorid no se necesita dispositivo móvil para pruebas, pero para las futuras pruebas necesitaremos una.

Como no se dispone de un Mac ni de un iPhone, pero de un dispositivo Android sí, en concreto de un HTC Desire.

Por estos motivos Android gana en este punto.

5.2.1.1.2 Lenguaje de Programación.

Para el desarrollo de aplicaciones en el sistema operativo Android se utiliza el lenguaje de programación Java.

Para el desarrollo de aplicaciones en el sistema operativo iOS se utiliza el lenguaje de programación Objetic-C.

Como el equipo que desarrolla la aplicación esta familiarizado con Java, no hace falta que los trabajadores aprendan. Es otro punto a favor de realizar la aplicación en Android.

5.2.1.1.3 Sistema Libre.

En Android no se cobra nada por desarrollar las aplicaciones, es totalmente libre. Por lo único que se cobra es por registrarse en la consola de desarrolladores dónde el desarrollador se tiene que crear un perfil y pagar una cuota de 25 $. Una vez registrado se podrán subir aplicaciones a Google Play. Pero no hace falta subir las aplicaciones a Google Play para distribuirlas, ya que Android permite instalar aplicaciones autofirmadas.

En iOS para poder acceder a todos los SDK y entornos de programación hay que pagar 99 $ anuales. Con este pago, Apple proporciona el certificado con el que se pueden firmar las aplicaciones al AppStore. No hace falta tener necesariamente un Mac para programar ya que hay otras plataformas como Addobe Flex que permite desarrollar en otros sistemas operativos como Windows o Linux, pero para subir la aplicación a AppStore se tienen que abonar los 99$ anuales. Apple no permite que en iOS se instalen las aplicaciones que son autofirmadas a no ser que se le practique el Jailbreak.

Como la distribución de la aplicación de la aplicación no nos interesa y ambos se pueden utilizar gratuitamente, ambos ganan en este punto. Aunque para instalar en el iPhone se necesita que se le haya practicado el Jailbreak o disponer del certificado de Apple.

5.2.1.1.4 Conclusión

	Hardware	Lenguaje de Programación	Sistema Libre
Android	√	√	√
iOS			√

Tabla 5.95: Comparativa de plataforma móvil.

Como se puede ver en la tabla 5.95 se elegirá el sistema operativo Android y el lenguaje de programación Java.

5.2.1.2 Comunicaciones

Para las comunicaciones se elegirá utilizar Web Services debido a la interoperabilidad entre plataformas que ofrece, son sencillos de realizar y es lo más practico para la aplicación que nos proponemos a realizar.

Se tienen dos opciones usar SOAP o usar REST, entonces ¿cuándo es útil SOAP y cuándo es útil REST?:

Es más útil utilizar SOAP cuando la arquitectura necesite procesar asíncronamente, ya que REST no permite asincronías. Es útil cuando la arquitectura tenga que tratar con requerimientos

complejos no funcionales y cuando se establece un contrato formal para la descripción de la interfaz que el servicio ofrece, gracias a WSDL.

Es más útil utilizar REST cuando el servicio Web no tiene estado, cuando se utiliza una infraestructura de *catching* que mejora el rendimiento, cuando el ancho de banda necesita ser limitado, REST es más fácil de diseñar que SOAP al utilizar tecnologías como AJAX y toolkits como DWR.

Por estas características, lo más práctico es utilizar REST, ya que las cabeceras y capas adicionales de los elementos SOAP producen una sobrecarga adicional en los dispositivos con escasos recursos como PDAs o Smartphones. Otro punto a favor de REST es el uso de la caché, que como se verá en el diseño del token de sesión es un factor muy importante.

5.2.1.3 Lenguaje de Programación en el Servidor

El lenguaje de programación utilizado por el servidor es PHP con el framework Zend. Como mi compañero se va a encargar de realizar una aplicación Web en el servidor, es conveniente utilizar el mismo lenguaje para que este todo estructurado. Para poder observar las razones mirar su TFG. [31]

5.2.2 Arquitectura del Sistema

El sistema a desarrollar se compone de tres partes diferenciadas:

- La aplicación Android.
- El Api del servidor.
- El sistema de comunicación empleado.

La aplicación Android a través de las comunicaciones establecidas, se pone en contacto con el servidor, que le proporciona un API con unas determinadas funciones. Estas funciones se las proporciona una capa de servicio que contiene el servidor. Todas las acciones de las comunicaciones irán a través de HTTPS sean completamente seguras. La arquitectura del sistema puede ser observada en la figura 5.4.

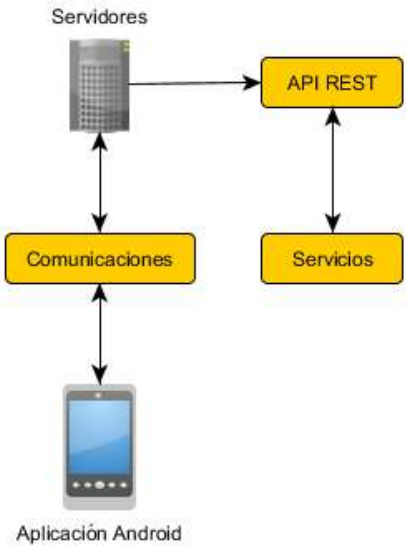


Figura 5.4: Arquitectura del sistema.

Los componentes del sistema que se observan en la figura 5.2 son:

Aplicación Android.
Comunicaciones empleadas.
API REST.
Servicios.

El API REST y los servicios proporcionan las siguientes funciones:

- Subir Fichero
- Logueo/Autenticación
- Registro
- CargaInicio
- Download/Descarga
- Delete/Eliminar fichero
- FicherosBorrados
- VersionesAnteriores
- Restaurar
- Renovar
- CrearDirectorio
- CargarDirectorio
- EliminarDirectorio
- CompartirDirectorio
- AceptarInvitación
- CancelarInvitación

La aplicación Android consta de los siguientes paquetes con las siguientes clases:

- Paquete com.android.vocs
    - VoCSActivity
    - VoCSAplication
    - VoCSBroadcast
    - VoCSDatosActivity
    - VoCSFicherosSeleccionadosActivity

    - VoCSFileExplorerActivity
    - VoCSRegistroActivity
    - VoCSService

- Paquete com.android.rest

    - VoCSRest

- Paquete com.android.auxiliar

    - VoCSArbol
    - VoCSListAuxiliar
    - VoCSNodo

5.2.2.1 Métodos del API

En este apartado se detalla la interfaz propuesta para interactuar con el servidor. La interfaz se compone de las siguientes funciones:

- Subir Fichero
- Logueo/Autenticación
- Registro
- CargaInicio
- Download/Descarga
- Delete/Eliminar fichero
- FicherosBorrados
- VersionesAnteriores
- Restaurar
- Renovar
- CrearDirectorio
- CargarDirectorio
- EliminarDirectorio
- CompartirDirectorio
- AceptarInvitación
- CancelarInvitación

Subir Fichero (nick, seguridad, directorio)

- Nick: nick del usuario que sirve a su vez como identificador del usuario. Esta variable es una cadena de caracteres.
- Seguridad: Nivel de seguridad con la que se quiere subir un fichero, esta variable sólo puede tomar tres valores 0,1 y 2.
- Directorio: Identificador del directorio. Esta variable es un número entero no negativo.

La función Subir Fichero se encarga de almacenar el fichero proporcionado por el usuario en su carpeta correspondiente. Si el directorio esta compartido, almacenar el fichero en las carpetas de los usuarios que comparten la carpeta. Por último aplicar las acciones oportunas al nivel de seguridad que se envía por parte del usuario.

Login(usuario, contraseña)

- Usuario: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.
- Contraseña: clave secreta del usuario que sirve para validar el nick del usuario. Esta variable es una cadena de caracteres.

La función login se encarga de autenticar un usuario permitiendo el acceso a la aplicación.

Registro(nick, nombre, apellidos, email, dni, contraseña)

- Nick: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.
- Nombre: nombre del usuario que se esta registrando en la aplicación. Es una cadena de caracteres.
- Apellidos: apellidos del usuario que se esta registrando en la aplicación. Es una cadena de caracteres.
- Email: email de usuario que se esta registrando en la aplicación. Es una cadena de caracteres.
- DNI: DNI del usuario que se esta registrando en la aplicación. Es una cadena de caracteres.
- Contraseña: clave secreta del usuario que sirve para validar el nick del usuario. Esta variable es una cadena de caracteres.

Esta función se encarga de registrar al nuevo usuario.

Carga Inicio(nick)

- Nick: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.

Esta función se encarga de buscar los metadatos del nivel 1 y el nivel 2 del sistema de ficheros del sistema y proporcionárselo al usuario.

Download(nick,id)

- Nick: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.
- Id: identificador del usuario que se quiere descargar. Es un entero que no puede ser negativo.

Esta función se encarga de buscar el fichero solicitado por el usuario y mandárselo

Delete(nick, id)

- Nick: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.
- Id: identificador del usuario que se quiere descargar. Es un entero que no puede ser negativo.

Esta función se encarga de eliminar un fichero seleccionado por el usuario, pasando a estar inactivo.

FicherosBorrados(nick)

- Nick: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.

Esta función se encarga de buscar todos los ficheros borrados que tiene el usuario y mandar sus metadatos.

Versiones Anteriores(nick, id)

- Nick: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.
- Id: identificador del usuario que se quiere descargar. Es un entero que no puede ser negativo.

Esta función se encarga de buscar todas las versiones anteriores del fichero proporcionado y mandar sus metadatos.

Restaurar(nick,id)

- Nick: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.
- Id: identificador del usuario que se quiere descargar. Es un entero que no puede ser negativo.

Esta función se encarga de restaurar el fichero proporcionado por el usuario, pasando a estar activo.

Renovar()

Renueva la sesión del usuario, reiniciando el tiempo de sesión.

CrearDirectorio(nombre, padre, nick)

- Nick: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.
- Padre: identificador del directorio padre del nuevo directorio que se va a crear. Es un entero que no puede ser negativo.
- Nombre: nombre del directorio. Esta variable es una cadena de caracteres.

Esta función se encarga de crear un nuevo directorio en el sistema de ficheros.

CargarDirectorio(nv, nick)

- Nick: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.
- nv: array que contiene los metadatos del fichero que se quiere cargar.

Esta función proporciona los metadatos del fichero proporcionado y los metadatos de su nivel inferior.

EliminarDirectorio(id, nick)

- Nick: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.
- Id: identificador del directorio que se quiere eliminar. Es un entero que no puede ser negativo.

Esta función se encarga de eliminar el directorio proporcionado por el usuario y a su vez eliminar todos los directorios y ficheros que tenga el directorio proporcionado asociado.

CompartirDirectorio(id, nick, email)

- Nick: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.
- Id: identificador del directorio que se quiere compartir. Es un entero que no puede ser negativo.
- Email: email del usuario al que se quiere enviar una petición.

Esta función se encarga de crear la invitación para compartir un directorio en la base de datos para su posterior envío al usuario.

AceptarInvitación(id, nick, anfitrion)

- Nick: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.
- Id: identificador del directorio que se quiere aceptar la invitación. Es un entero que no puede ser negativo.
- Anfitrion: nick de la persona que mandó la invitación del directorio. Esta variable es una cadena de caracteres.

Esta función se encarga de aceptar la invitación, creando el nuevo directorio en la raíz del sistema de ficheros y mandando los metadatos del nuevo directorio.

CancelarInvitación(id, nick, anfitrion)

- Nick: nick del usuario que sirve como identificador del usuario. Esta variable es una cadena de caracteres.
- Id: identificador del directorio que se quiere aceptar la invitación. Es un entero que no puede ser negativo.
- Anfitrion: nick de la persona que mandó la invitación del directorio. Esta variable es una cadena de caracteres.

Esta función se encarga de cancelar la invitación, eliminándola del sistema.

5.2.2.2 Mensajes de Petición

En esta sección se van a exponer los mensajes que se envían al servidor. Todos los mensajes tienen en común que usan la función POST de HTTP. A continuación se describen los mensajes.

Mensaje de Autenticación

El mensaje de autenticación consta de dos objetos:

- El primero tiene de nombre clave “entity” y tiene dos objetos que son el usuario y la contraseña. Su nombre clave será “param1” y “param2”. Las cadenas usuario y contraseña, se envían dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “logueo”.

Mensaje Registro

El mensaje de registro consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de seis objetos almacenados en un objeto json. El primer objeto es el Nick del usuario y tiene como nombre clave “param1”. El segundo objeto es el Nombre del usuario y tiene como nombre clave “param2”. El tercer objeto son los Apellidos del usuario y tiene como nombre clave “param3”. El cuarto objeto es el Email del usuario y tiene como nombre clave “param4”. El quinto objeto es el DNI del usuario y tiene como nombre clave “param5”. El sexto objeto es la contraseña y tiene como nombre clave “param6”. Todos son cadenas de texto que se envían dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “registro”.

Mensaje CargaInicial

El mensaje de cargaInicial consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de dos objetos almacenados en un objeto json. El primer objeto es el Nick del usuario y tiene como nombre clave “param1”. El segundo objeto es el Token de sesión y tiene como nombre clave “token”. Todos son cadenas de texto que se envían dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “cargaInicio”.

Mensaje descargarFichero

El mensaje de descargarFichero consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de tres objetos almacenados en un objeto json. El primer objeto es el Nick del usuario y tiene como nombre clave “param1”. El segundo objeto es el Id del fichero y tiene como nombre clave “param2”. El tercer objeto es el Token de sesión y tiene como nombre clave “token”. Todos son cadenas de texto que se envían dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “download”.

Mensaje eliminarFichero

El mensaje de eliminarFichero consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de tres objetos almacenados en un objeto json. El primer objeto es el Nick del usuario y tiene como nombre clave “param1”. El segundo objeto es el Id del fichero y tiene como nombre clave “param2”. El tercer objeto es el Token de sesión y tiene como nombre clave “token”. Todos son cadenas de texto que se envían dentro de un objeto json.

- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “delete”.

Mensaje ficherosBorrados

El mensaje de ficherosBorrados consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de dos objetos almacenados en un objeto json. El primer objeto es el Nick del usuario y tiene como nombre clave “param1”. El segundo objeto es el Token de sesión y tiene como nombre clave “token”. Todos son cadenas de texto que se almacenan dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “ficherosBorrados”.

Mensaje versionesAnteriores

El mensaje de versionesAnteriores consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de tres objetos almacenados en un objeto json. El primer objeto es el Nick del usuario y tiene como nombre clave “param1”. El segundo objeto es el Id del fichero y tiene como nombre clave “param2”. El tercer objeto es el Token de sesión y tiene como nombre clave “token”. Todos son cadenas de texto que se envían dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “versionesAnteriores”.

Mensaje restaurarFichero

El mensaje de restaurarFichero consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de tres objetos almacenados en un objeto json. El primer objeto es el id del fichero y tiene como nombre clave “param1”. El segundo objeto es el Nick del usuario y tiene como nombre clave “param2”. El tercer objeto es el Token de sesión y tiene como nombre clave “token”. Todos son cadenas de texto que se envían dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “restaurar”.

Mensaje subirFichero

El mensaje de subirFichero consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de cuatro objetos almacenados en un objeto json. El primer objeto es el Nick del usuario y tiene como nombre clave “param1”. El segundo objeto es el id del directorio donde se va a guardar el fichero y tiene como nombre clave “param2”. El tercer objeto es el nivel de seguridad que se quiere asociar al fichero y tiene como nombre clave “param3”. El cuarto objeto es el Token de sesión y tiene como nombre clave “token”. Todos son cadenas de texto que se envían dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “subir”.
- El tercero se llama “uploaded”, que contiene el fichero que se mandará al usuario.

Mensaje renovarSesion

El mensaje de renovarSesion consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de un objetos almacenado en un objeto json. El objeto es el Token de sesión y tiene como nombre clave “token”. Es una cadena de texto que se envía dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “renovar”.

Mensaje crearDirectorio

El mensaje de crearDirectorio consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de cuatro objetos almacenados en un objeto json. El primer objeto es el nombre de la carpeta que se va a crear y tiene como nombre clave “param1”. El segundo objeto es el id del directorio donde se va a crear el directorio y tiene como nombre clave “param2”. El tercer objeto es el Nick del usuario y tiene como nombre clave “param3”. El cuarto objeto es el Token de sesión y tiene como nombre clave “token”. Todos son cadenas de texto que se envían dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “crearDirectorio”.

Mensaje cargarDirectorio

El mensaje de cargarDirectorio consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de cuatro objetos almacenados en un objeto json. El primer objeto es un jsonArray compuesto por el id del directorio, por el id de los ficheros del directorio, un array por el nombre de los ficheros del directorio, por el nombre de los directorios hijos y por el id de los directorios hijos; y tiene como nombre clave “param1”. El segundo objeto es el Nick del usuario y tiene como nombre clave “param2”. El tercer objeto es el Token de sesión y tiene como nombre clave “token”. Todos son cadenas de texto que se envían dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “cargarDirectorio”.

Mensaje borrarDirectorio

El mensaje de borrarDirectorio consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de cuatro objetos almacenados en un objeto json. El primer objeto el id del directorio y tiene como nombre clave “param1”. El segundo objeto es el Nick del usuario y tiene como nombre clave “param2”. El tercer objeto es el Token de sesión y tiene como nombre clave “token”. Todos son cadenas de texto que se envían dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “eliminarDirectorio”.

Mensaje compartirDirectorio

El mensaje de compartirDirectorio consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de cuatro objetos almacenados en un objeto json. El primer objeto el id del directorio y tiene como nombre clave “param1”. El segundo objeto es el Nick del usuario y tiene como nombre clave “param2”. El tercer objeto es el email del usuario con el que se quiere compartir y tiene como nombre clave “param3”. El cuarto objeto es el Token de sesión y tiene como nombre clave “token”. Todos son cadenas de texto que se envían dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “compartirDirectorio”.

Mensaje aceptarInvitacion

El mensaje de aceptarInvitacion consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de cuatro objetos almacenados en un objeto json. El primer objeto el id del directorio y tiene como nombre clave “param1”. El segundo objeto es el Nick del usuario y tiene como nombre clave “param2”. El tercer objeto es el Nick del usuario que ha invitado y tiene como nombre clave “param3”. El cuarto objeto es el Token de sesión y tiene como nombre clave “token”. Todos son cadenas de texto que se envían dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “aceptarInvitacion”.

Mensaje cancelarInvitacion

El mensaje de cancelarInvitacion consta de dos objetos:

- El primero tiene de nombre clave “entity” y consta de cuatro objetos almacenados en un objeto json. El primer objeto el id del directorio y tiene como nombre clave “param1”. El segundo objeto es el Nick del usuario y tiene como nombre clave “param2”. El tercer objeto es el Nick del usuario que ha invitado y tiene como nombre clave “param3”. El cuarto objeto es el Token de sesión y tiene como nombre clave “token”. Todos son cadenas de texto que se envían dentro de un objeto json.
- El segundo se llama “método” de nombre clave y sólo tiene un objeto que es el método. Método es una cadena de texto que debe ser “cancelarInvitacion”.

5.2.2.3 Mensajes de Respuesta

Mensaje de respuesta de Autenticación

El mensaje de respuesta de autenticación es un mensaje que se compone de los siguientes elementos:

- Un jsonArray con el token de sesión en la posición 0 y las demás posiciones están ocupadas por arrays que contienen los elementos necesarios para representar las invitaciones. Estos arrays se componen de: el email anfitrión en la posición 0, el nombre del directorio en la posición 1, el id del fichero en la posición 2 y el Nick del usuario en la posición 3.
- El código del mensaje es el 201.

Mensaje de respuesta de Registro

El mensaje de respuesta de registro es un mensaje que se compone de los siguientes elementos:

- Una cadena en la que pone "Registrado" si todo ha ido bien, y una cadena que pone "No Registrado" si algo ha ido mal. El mensaje de retorno va en formato json.
- El código del mensaje es el 201.

Mensaje de respuesta de cargaInicial

El mensaje de respuesta de cargaInicial es un mensaje que se compone de los siguientes elementos:

- Un jsonArray con los siguientes elementos en orden:
    - Espacio utilizado por usuario
    - Espacio total del usuario
    - Un array de datos que contienen los metadatos del directorio raíz. Los metadatos se componen de una cadena de texto y de cuatro arrays:
        - Id del directorio raíz
        - Array que contiene los id de los ficheros que tiene asociados.
        - Array que contiene los nombres de los ficheros asociados.
        - Array que contiene los nombres de los directorios hijos.
        - Array que contiene el id de los directorios hijos.
    - En caso de haber directorios hijos del directorio raíz, se mandaría un array por cada directorio con los metadatos de los directorios hijos, tal y como con el directorio raíz.
- El código del mensaje es el 201.

Mensaje de respuesta de descargarFichero

El mensaje de respuesta de descargarFichero es un mensaje que se compone del código de mensaje 201 para informar que la descarga se ha realizado correctamente.

Mensaje de respuesta borrarFichero

El mensaje de respuesta de borrarFichero es un mensaje que se compone de los siguientes elementos:

- La cadena de texto "Eliminado" en caso de que la operación se haya ejecutado correctamente y en caso contrario la cadena "No Eliminado".
- El código del mensaje es 201.

Mensaje de respuesta ficherosBorrados

El mensaje de respuesta de ficherosBorrados es un mensaje que se compone de los siguientes elementos:

- Un elemento json formado por los siguientes elementos:
    - Un array con nombre clave "ficheros" que contiene el id de los ficheros borrados
    - Un array con nombre clave "nombres" que contiene el nombre de los ficheros borrados.
    - Un array con nombre clave "fecha" que contiene las fechas de borrado de los ficheros.
- El código del mensaje es 201.

Mensaje de respuesta versionesAnteriores

El mensaje de respuesta de versionesAnteriores es un mensaje que se compone de los siguientes elementos:

- Un elemento json formado por los siguientes elementos:
    - Un array con nombre clave “ficheros” que contiene el id de los ficheros borrados
    - Un array con nombre clave “nombres” que contiene el nombre de los ficheros borrados.
    - Un array con nombre clave “fecha” que contiene las fechas de borrado de los ficheros.
- El código del mensaje es 201.

Mensaje de respuesta Restaurar

El mensaje de respuesta de restaurar es un mensaje que se compone de los siguientes elementos:

- Un elemento json formado por un elemento con nombre clave “directorio” que contiene el id del directorio que se restaura.
- El código del mensaje es 201.

Mensaje de respuesta renovar

El mensaje de respuesta de renovar es un mensaje que se compone de los siguientes elementos:

- El mensaje de respuesta que puede ser “Renovado” o “No Renovado” dependiendo del resultado de la función.
- El código del mensaje es 201.

Mensaje de respuesta crearDirectorio

El mensaje de respuesta de crearDirectorio es un mensaje que se compone de los siguientes elementos:

- Un elemento json formado por un elemento con nombre clave “directorio” que contiene el id del directorio que se restaura.
- El código del mensaje es 201.

Mensaje de respuesta cargarDirectorio

El mensaje de respuesta de crearDirectorio es un mensaje que se compone de los siguientes elementos:

- Un jsonArray con los siguientes elementos en orden:
    - Un array de datos que contienen los metadatos del directorio que se envía. Los metadatos se componen de una cadena de texto y de cuatro arrays:
        - Id del directorio raíz
        - Array que contiene los id de los ficheros que tiene asociados.
        - Array que contiene los nombres de los ficheros asociados.
        - Array que contiene los nombres de los directorios hijos.
        - Array que contiene el id de los directorios hijos.
    - En caso de haber directorios hijos del directorio raíz, se mandaría un array por cada directorio con los metadatos de los directorios hijos, tal y como con el directorio raíz.
- El código del mensaje es 201.

Mensaje de respuesta eliminarDirectorio

El mensaje de respuesta de eliminarDirectorio es un mensaje que se compone de los siguientes elementos:

- La cadena de texto "Eliminado" en caso de que la operación se haya ejecutado correctamente y en caso contrario la cadena "No Eliminado".
- El código del mensaje es 201.

Mensaje de respuesta compartirDirectorio

El mensaje de respuesta de compartirDirectorio es un mensaje que se compone de los siguientes elementos:

- La cadena de texto "Compartido" en caso de que la operación se haya ejecutado correctamente y en caso contrario la cadena "No Compartido".
- El código del mensaje es 201.

Mensaje de respuesta aceptarInvitación

El mensaje de respuesta de aceptarInvitacion es un mensaje que se compone de los siguientes elementos:

- Un jsonArray con los siguientes elementos en orden:
    - El nombre del directorio que se va a compartir.
    - Un array de datos que contienen los metadatos del directorio que se envía. Los metadatos se componen de una cadena de texto y de cuatro arrays:
        - Id del directorio raíz
        - Array que contiene los id de los ficheros que tiene asociados.
        - Array que contiene los nombres de los ficheros asociados.
        - Array que contiene los nombres de los directorios hijos.
        - Array que contiene el id de los directorios hijos.
- El código del mensaje es 201.

Mensaje de respuesta cancelarInvitacion

El mensaje de respuesta de cancelarInvitacion es un mensaje que se compone de los siguientes elementos:

- La cadena de texto "Cancelada" en caso de que la operación se haya ejecutado correctamente y en caso contrario la cadena "No Cancelada".
- El código del mensaje es 201.

5.2.3 Arquitectura del Sistema de Ficheros de la Aplicación

El sistema de ficheros de la aplicación tiene que optimizar los recursos del teléfono y del servidor; y tiene que ser escalable y rápido.

Para explicar la arquitectura del sistema de ficheros primero hay que explicar que son los niveles del sistema de ficheros. En la figura 5.5 se pueden observar los niveles del sistema de ficheros. El nivel uno esta representado por el color verde más claro el cual proporciona los ficheros y directorios del directorio raíz. El nivel dos se representa en la figura por el color verde más

oscuro, que es el nivel al cual pertenecen los ficheros y directorios de un directorio de nivel uno. Por consiguiente un tercer nivel serían los ficheros y directorios que pertenecen a un directorio de nivel dos.

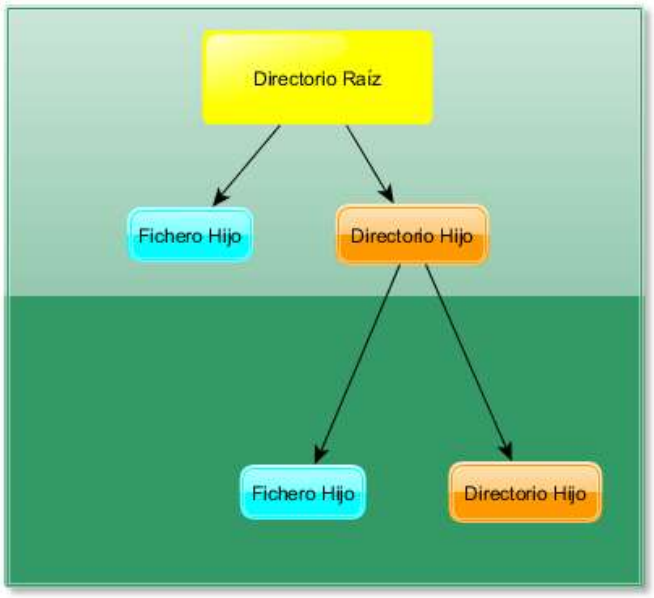


Figura 5.5: Representación de los niveles del sistema de ficheros.

Una vez que queda claro que son los niveles del sistema de ficheros, procederemos a explicar como se diseñará.

Para ello cuando la aplicación se ponga en contacto con el servidor éste le proporcionará los directorios y ficheros que contiene el directorio raíz, el nivel uno; y los ficheros y directorios de los directorios hijos del directorio raíz, el nivel dos.

La aplicación cuando recibe estos niveles los almacenará en un árbol, para proporcionar al cliente la vista de estos ficheros y estar preparado para mostrar la vista de los directorios hijos.

Se han elegido los dos primeros niveles para poder ir siempre un paso por delante del usuario. Querer ir un paso por delante del usuario nos facilita cargar el siguiente nivel, ya que se pretende que cuando el usuario acceda al directorio hijo, por detrás y sin que el usuario sea consciente de ello se solicitará sólo el nivel inferior del directorio al que se accede. Figura 5.6 muestra un ejemplo de lo que se pretende realizar.

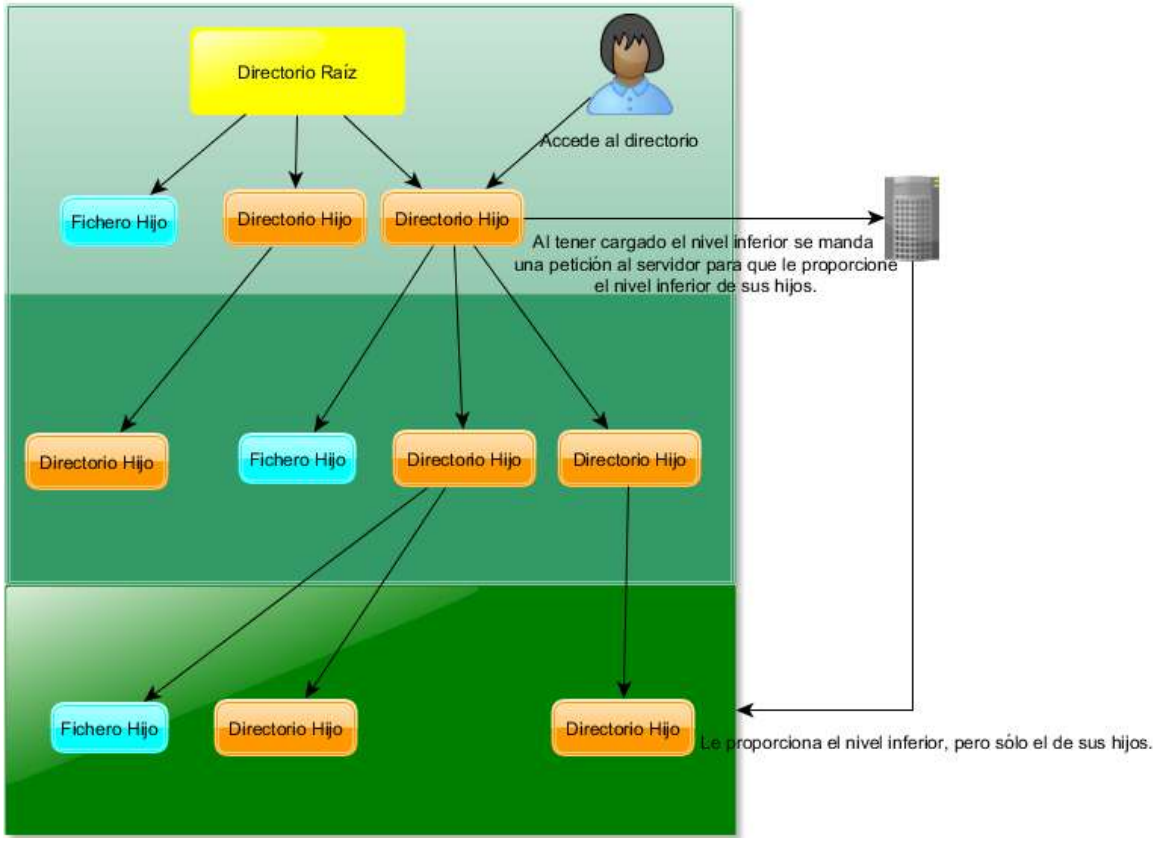


Figura 5.6: Ejemplo de comportamiento del árbol.

Como se puede observar en la figura 5.6 el directorio raíz tiene dos directorios y el usuario puede navegar por ellos, entonces accede al primer directorio hijo y la aplicación por detrás del usuario y sin que se dé cuenta de nada, solicitará el nivel inferior de los hijos, ya que el suyo lo tenemos cargado en el árbol y puede visualizarlo y realizar operaciones sin problemas. Cuando recibe la respuesta del servidor añade al árbol el nuevo nivel del directorio, para que el usuario pueda navegar por los directorios.

Esta acción sólo se realizará si los niveles inferiores no están cargados en el árbol.

Los nodos cuentan con los siguientes datos dentro de ellos, un id que los identifique, un elemento json que contiene todos los datos del directorio recopilados, el nodo padre, una lista con los nombres de los ficheros, una lista con los id de los ficheros, una lista con los nombres de los directorios, una lista con los id de los directorios y una lista de nodos hijos.

Esta estructura de ficheros es buena porque el servidor no tiene que proporcionar todo el sistema de ficheros cada vez que un usuario se conecta con él. El servidor solo tiene que proporcionar lo que el usuario necesita en cada momento, además una vez cargado no es necesario volver a solicitarlo por lo que acceder a ello es rápido y los usuarios no suelen tener niveles muy profundos en este tipo de servicios por lo que habría que solicitar pocas peticiones.

Esta arquitectura cumple con los objetivos marcados, optimiza los recursos del servidor al no tener que cargar todo el sistema de directorios, optimiza los recursos del teléfono al tener cargado solo lo que el usuario necesita, es escalable ya que permite construir nuevos directorios y generar niveles infinitos. Puede que no sea tan rápido que tener cargado todo desde el principio, pero no se suele acceder siempre a todo el sistema de directorios en un dispositivo móvil el cual se usa para realizar acciones muy puntuales.

5.2.4 Arquitectura del Modelo de Sesión

El modelo de sesión de la aplicación es un factor a tener en cuenta debido a que se necesita acreditar los mensajes que llegan del usuario. Es importante crear una sesión al usuario o un mecanismo que autentique los mensajes que llegan y se corresponden con el éste.

Para realizar este modelo, se proporcionará un token de sesión al usuario, que deberá incluir en cada uno de sus mensajes. El token de sesión no se guardará en base de datos, se almacenará en la memoria caché del servidor utilizando Memcached.

Memcached definido por la empresa que lo desarrollo lo define como un "*sistema distribuido de alto rendimiento para el cacheo de objetos en memoria, genérico por naturaleza, pero pensado para incrementar la velocidad de aplicaciones web dinámicas, aliviando la carga de las bases de datos*". [32]

El uso de la caché permite un acceso más rápido que las bases de datos y los datos se borran automáticamente sin necesidad de realizar ninguna operación. Para la implementación de nuestro modelo de sesión son las características más prácticas que ofrece, ya que los tokens que implantaremos no ocuparán mucho espacio y se borrarán automáticamente pasado cierto tiempo si estos no se renuevan.

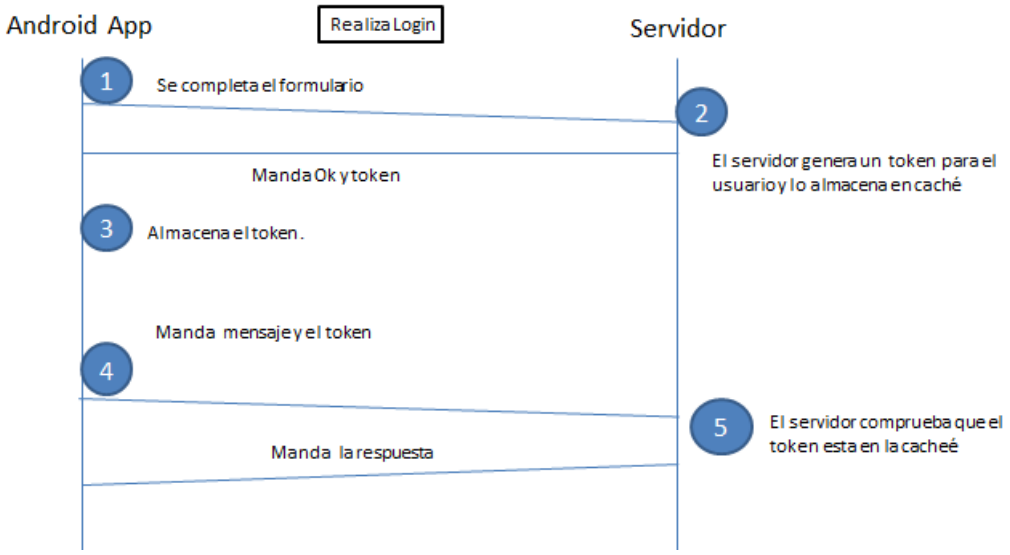


Figura 5.7: Intercambio de mensajes del Modelo de Sesión.

En la Figura 5.7 se puede observar como se gestiona el modelo de sesión. Primero el usuario rellena el formulario de autenticación en la aplicación y envía el mensaje de autenticación al servidor. El servidor recibe el mensaje, comprueba que la autenticación se corresponde y realiza el token de sesión.

El token de sesión se realiza con la operación sha256, a la que se le pasa un mensaje que es la concatenación del Nick de usuario, una cadena aleatoria de tamaño 10 y la fecha en la que realizó la autenticación, recogiendo segundos, minutos, horas, día, mes y año. El sha256 recibe también el password del usuario como semilla de la operación. La cadena de texto resultante es el token de sesión del usuario, que se almacenará en la caché del servidor con un tiempo de 20 minutos, pasados estos el token se eliminará de la caché automáticamente.

El servidor manda al usuario la respuesta de la autenticación y el token de sesión del usuario.

El usuario trata la respuesta y almacena el token de sesión en una variable global de la aplicación. Esa variable puede ser accedida desde cualquier parte de la aplicación. Cuando el usuario envía otro mensaje como el que se necesita para crear un directorio, el usuario tiene que enviar en el mensaje el token guardado.

El servidor recibe el mensaje y lo primero que comprueba es si el token esta alojado en la memoria caché, si esta alojado prosigue con la operación y le envía la respuesta al usuario. En caso de no estar alojado en la caché del servidor el servidor manda una respuesta negativa al usuario.

El usuario recibe la respuesta, si es positiva prosigue con su ejecución. Si es negativa la aplicación se cierra, para que el usuario vuelva a autenticarse.

El modelo de sesión como se ha mencionado antes tiene un sistema que permite renovar la sesión del token, ya que la sesión no va a durar sólo 20 minutos. Para ello la aplicación manda un mensaje de renovar sesión. Este mensaje de renovarSesión lo tiene que realizar a espaldas del usuario, para que no note nada. El mensaje es enviado por la aplicación cada 9 minutos. Se ha elegido 9 minutos para poder enviar dos mensajes antes de que caduque el token, ya que si la sesión dura 20 minutos es recomendable que si por alguna razón no se pudo enviar o se perdió el mensaje, de tiempo a enviar otro antes de que caduque. La Figura 5.8 muestra cómo se efectuará susodicha operación.

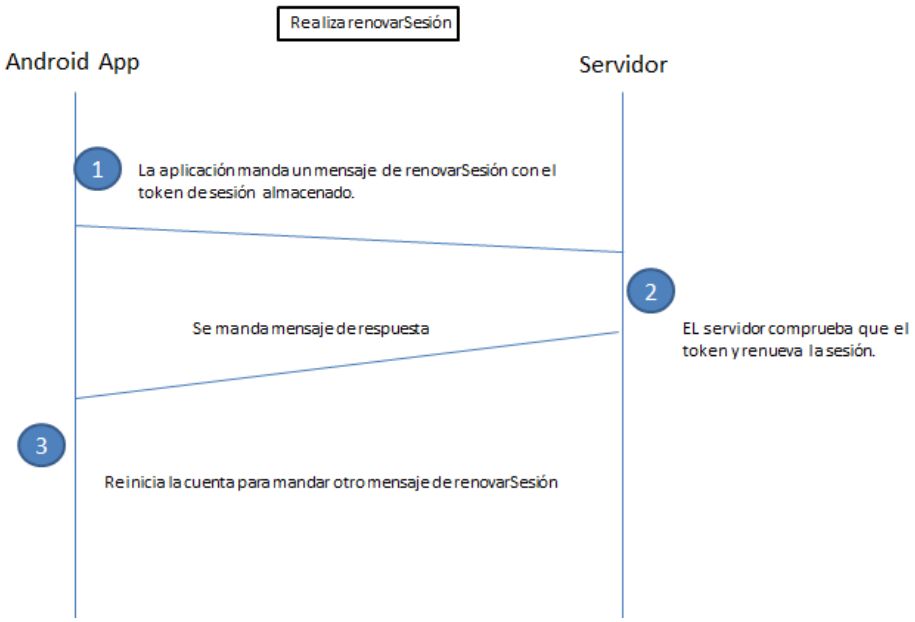


Figura 5.8: Intercambio de mensajes al renovar sesión.

Como se puede ver en la figura 5.8 la aplicación automáticamente manda un mensaje de renovarSesión al servidor con el token almacenado.

El servidor comprueba que el token de sesión está en la memoria caché. Si está, entonces el servidor renueva el token de sesión renovando su tiempo en 20 minutos y mandando un mensaje al usuario informando que se ha renovado correctamente. Si no está en la memoria caché el servidor manda un mensaje que indica que no se ha autenticado.

La aplicación comprueba el mensaje, si es negativo se sale de la aplicación. Si es positivo la aplicación inicia la cuenta atrás para volver a enviar el mensaje al servidor de renovar sesión.

5.2.5 Diagrama de Clases

En la figura 5.9 se puede observar el diagrama de clases. Que contiene las actividades y clases auxiliares que forman parte de la aplicación.

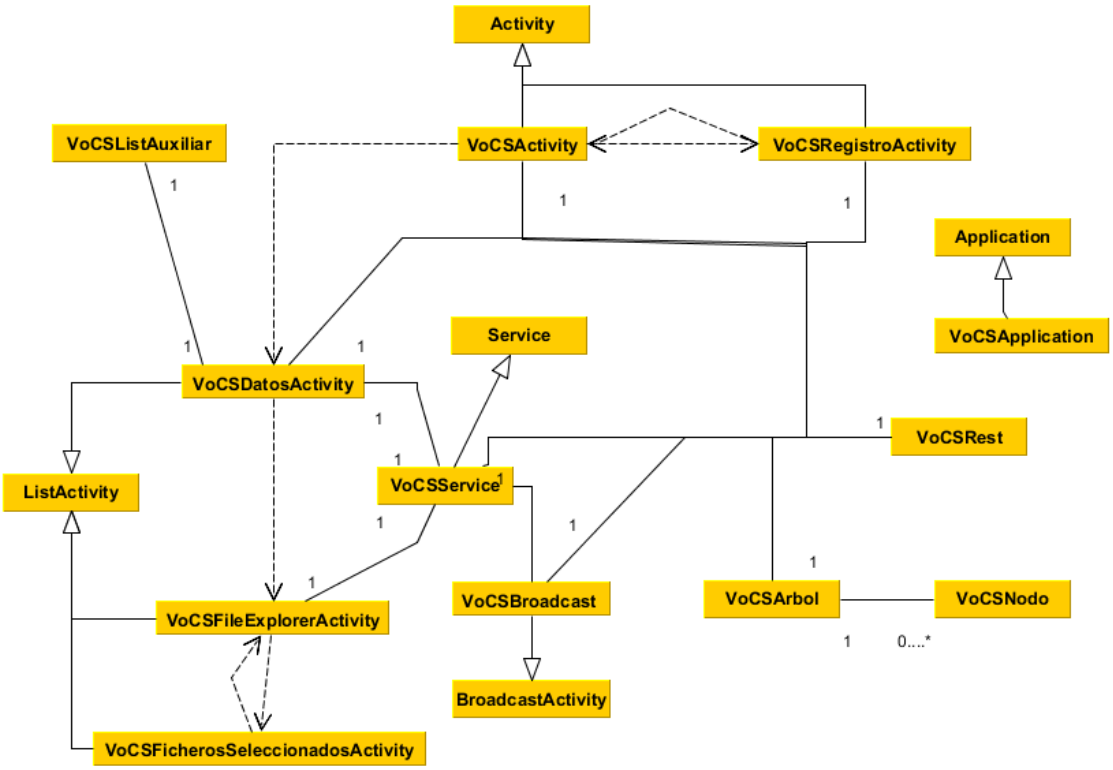


Figura 5.9: Diagrama de Clases.

A continuación se explicará una por una su funcionamiento y sus parámetros y funciones. Primero se empezarán a definir las actividades y sus clases asociadas. Después proseguiremos con las clases que las denominaremos auxiliares como son VoCSRest, VoCSAplication, VoCSArbol…

5.2.5.1 VoCSActivity

Esta clase es la que se inicia al arrancar la aplicación, desde ella se puede acceder a la clase VoCSRegistro y a la clase VoCSDatosActivity.

Figura 5.10: Clase VoCSActivity

Desde el botón entrar se puede acceder a VoCSDatosActivity y desde el botón registrar se accede a la clase VoCSRegistroActivity.

5.2.5.2 VoCSRegistroActivity

A esta clase se accede desde la clase principal VoCSActivity, esta clase consta de un formulario que se enviará al servidor para realizar el registro. Figura 5.11.

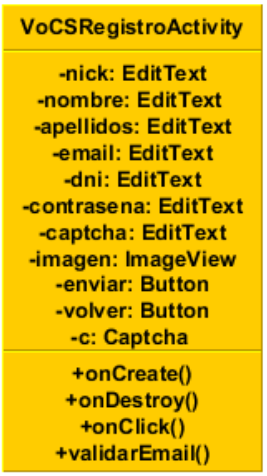


Figura 5.11: Clase VoCSRegistroActivity.

La clase VoCSRegistroActivity permite volver a la clase principal con el botón volver. Con el botón enviar manda al servidor el formulario de registro y espera la respuesta. Siendo la respuesta positiva o negativa, esta se muestra por pantalla al usuario.

5.2.5.3 VoCSDatosActivity

VoCSDatosActivity no es el Activity principal, pero es el más importante ya que es la clase en la que recae la mayoría de las funcionalidades y la que da inicio a la mayoría de ellas. Figura 5.12

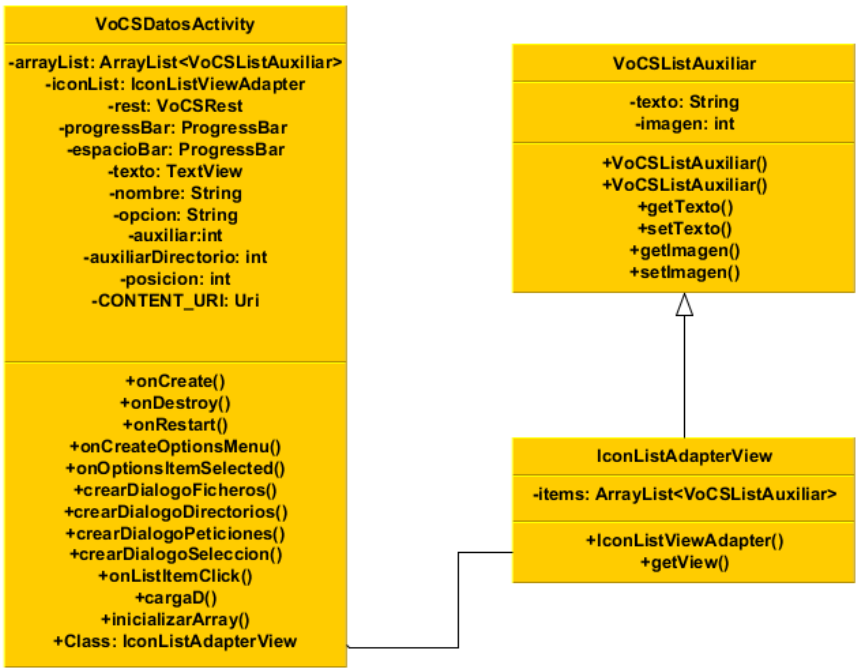


Figura 5.12: Clase VoCSDatosActivity.

Se puede observar en la figura 5.12 que la clase VoCSRegistroActivity contiene en ella la clase IconListAdapterView que extiende de VoCSListAuxiliar. Esta clase muestra el sistema de ficheros y cuenta con varios menús que permiten visualizar todas las funcionalidades.

5.2.5.4 VoCSFileExplorerActivity

La clase VoCSFileExplorerActivity es la clase que se encarga de mostrar el sistema de ficheros interno del sistema para que el usuario pueda elegir un fichero para subir al servidor. A esta clase se accede desde la clase VoCS. Figura 5.13

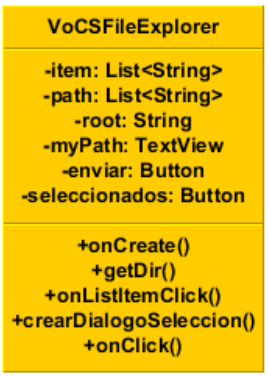


Figura 5.13: VoCSFileExplorer.

Desde esta clase no se puede volver a la clase VoCSDatosActivity, hay que destruirla pulsando el botón de retroceso.

5.2.5.5 VoCSFicherosSeleccionadosActivity

Esta clase se encarga de listar y eliminar los ficheros que se han seleccionado para la subida, ya que como se comentó se pueden subir varios ficheros al mismo tiempo. A esta clase se accede desde VoCSFileExplorerActivity.

Figura 5.14: VoCSFicherosSeleccionadosActivity.

Cuenta con los botones volver que permite volver a la clase VoCSFileExplorerActivity y borrar que permite borrar un elemento de la lista seleccionado.

5.2.5.6 VoCSRest

La clase VoCSRest es la que se encarga de conectarse con la API REST que proporciona el servidor. Todas las funciones que contactan con el servidor están recopiladas en esta clase. Ninguna otra se conecta con el servidor. Figura 5.15.

Figura 5.15: Clase VoCSRest.

Todas las clases que necesitan contactar con el servidor lo hacen a través de esta clase. Esta les proporciona la respuesta del servidor, pero es la que se encarga de inicializar y meter los datos en el árbol de directorios.

5.2.5.7 VoCSService

La clase VoCSService hereda de la clase Service de Android. Esta clase puede proporcionar servicios cuando la aplicación se encuentra en segundo plano. Figura 5.16.

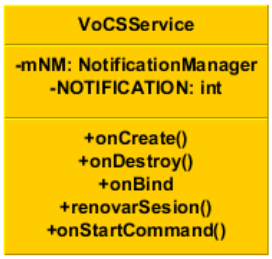


Figura 5.16: Clase VoCSService.

Esta clase es muy útil en este tipo de sistemas, ya que si la aplicación esta subiendo o descargando un archivo pesado, esta clase permite que la aplicación se quede en reposo o segundo plano y la descarga o la subida siga su curso.

5.2.5.8 VoCSAplication

La clase VoCSAplication hereda de la clase Application. Esta clase permite conservar variables globales mientras la aplicación esta activa y puede ser accedida por cualquier clase en cualquier momento. Como se puede observar en la Figura 5.17, esta clase contiene todo tipo de variables importantes para la aplicación como el árbol que contiene el sistema de ficheros, el nodo actual en el que se esta, el nodo raíz etc.

Figura 5.17: Clase VoCSApplication

5.2.5.9 VoCSBroadcast

La clase VoCSBroadcast hereda de la clase BroadcastReceiver. Esta clase se encarga de realizar la renovación de sesión cuando el service lo activa. Este realiza una llamada al sistema Android que envía el mensaje renovarSesión cada nueve minutos y se vuelve a llamar a los siguientes 9 minutos y así sucesivamente.

Figura 5.18: Clase VoCSBroadcast.

5.2.5.10 VoCSArbol

La clase VoCSArbol compone la estructura del sistema de ficheros de la aplicación. Esta compuesta por elementos de la clase VoCSNodo.

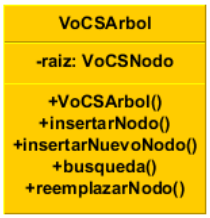


Figura 5.19: Clase VoCSArbol.

5.2.5.11 VoCSNodo

La clase VoCSNodo compone la estructura del árbol de directorios que se representa con la clase VoCSArbol.

Figura 5.20: Clase VoCSNodo.

5.2.6 Diseño de la Interfaz

En este apartado se mostrará el diseño de la interfaz de la aplicación, se mostrará una por una sus acciones y sus posibles variantes.

5.2.6.1 Interfaz VoCSActivity

En la figura 5.21 se puede observar la interfaz de la clase VoCSActivity. Se encarga de mostrar el formulario de autenticación.

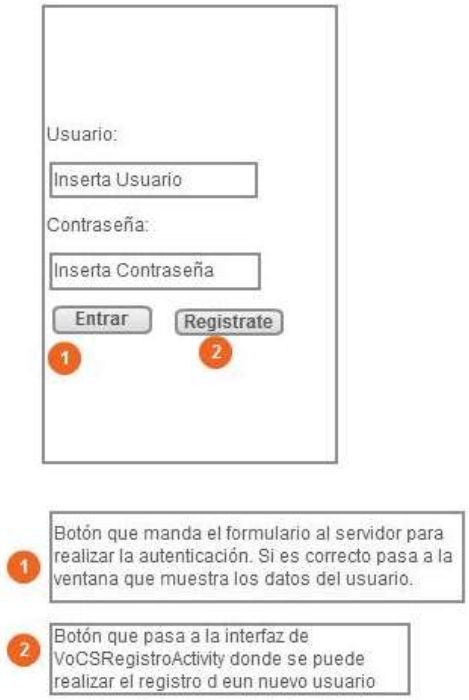


Figura 5.21: Interfaz VoCSActivity.

5.2.6.2 Interfaz VoCSRegistroActivity

Esta interfaz es la que se encarga de realizar el registro de un nuevo usuario. La figura 5.22 muestra la interfaz de VoCSRegistroActivity.

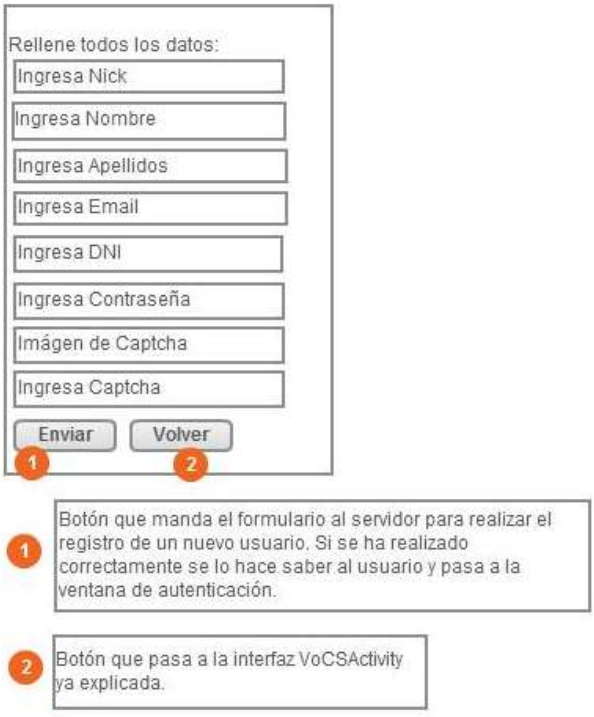


Figura 5.22: Interfaz VoCSRegistroActivity.

5.2.6.3 Interfaz VoCSDatosActivity

Esta interfaz es la más compleja de todas, por ello se explicará detalladamente. Empezaremos con la vista básica de la interfaz de VoCSDatosActivity.

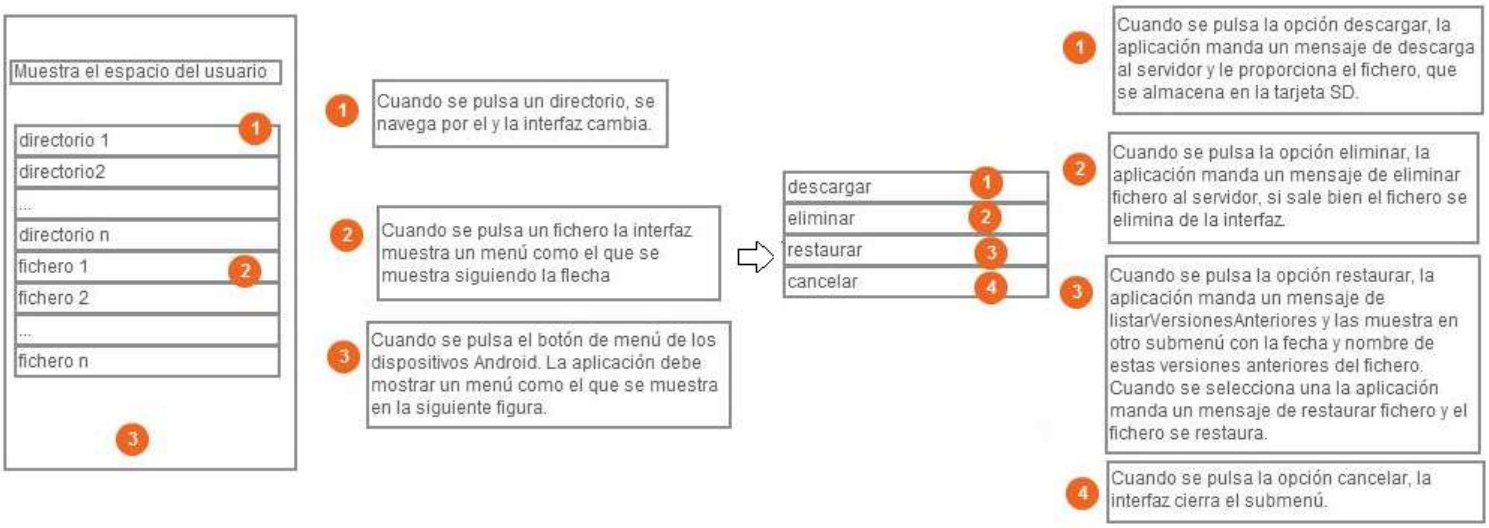


Figura 5.23: Interfaz principal, VoCSDatosActivity.

Como se puede observar en esta figura, si se sigue la opción 1, la interfaz cambia. La figura 5.24 muestra el resultado se seguir ese camino. Cuando se sigue la opción 2, se muestra un submenú como el que se muestra siguiendo la flecha con esas opciones, descargar, eliminar etc. Cuando se sigue el paso 3, la figura 5.23 muestra el cambio que se produce en la interfaz VoCSDatosActivity mostrando el menú cuando se pulsa el botón del menú que incluyen los dispositivos Android.

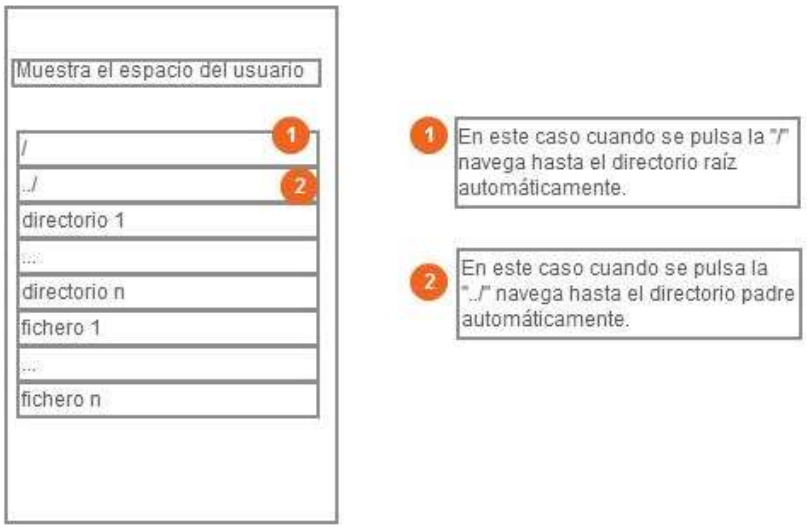


Figura 5.24: Interfaz de navegación VoCSDatosActivity.

Esta interfaz tiene las mismas funcionalidades que la anterior, salvo que se muestran los iconos "/" y "../" y sus funcionalidades.

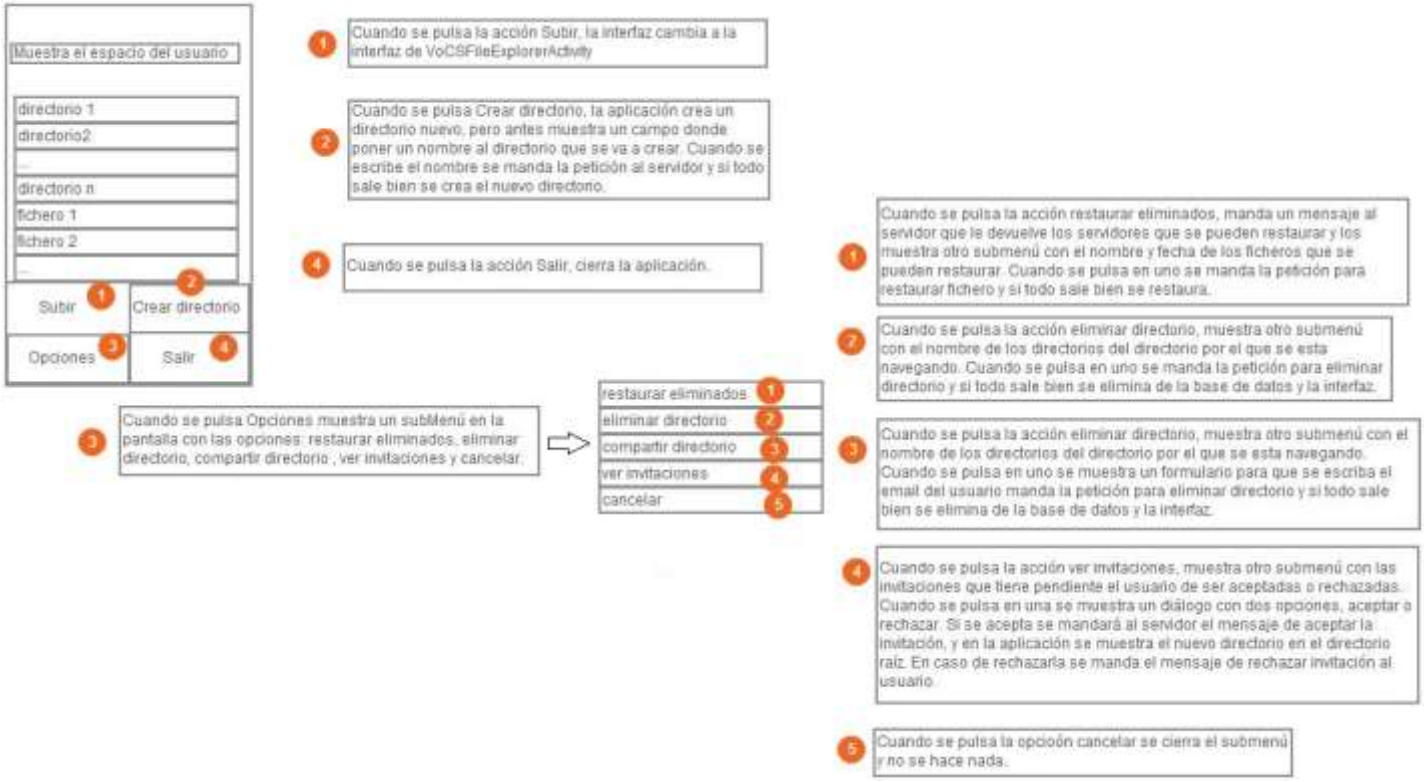


Figura 5.25: Interfaz que muestra la funcionalidad del menú.

Esta interfaz muestra el menú, este menú tiene cuatro opciones:

- Subir: Cambia la interfaz a VoCSFileExplorer.
- Crear Directorio: la aplicación muestra un diálogo con un campo en el cual se puede escribir el nombre del directorio nuevo que se quiere crear. Cuando se crea, el nuevo directorio debe aparecer en la interfaz.
- Opciones: muestra un submenú con varias acciones como son: restaurar eliminados, eliminar directorio, compartir directorio y ver invitaciones. Estas funciones se definen en la figura 5.25.
- Salir: cierra la aplicación.

5.2.6.4 Interfaz VoCSFileExplorerActivity

Esta interfaz es la que se encarga de mostrar el sistema de ficheros interno del teléfono, para seleccionar los ficheros que se quieren subir al servidor. La figura 5.26 muestra esta interfaz.

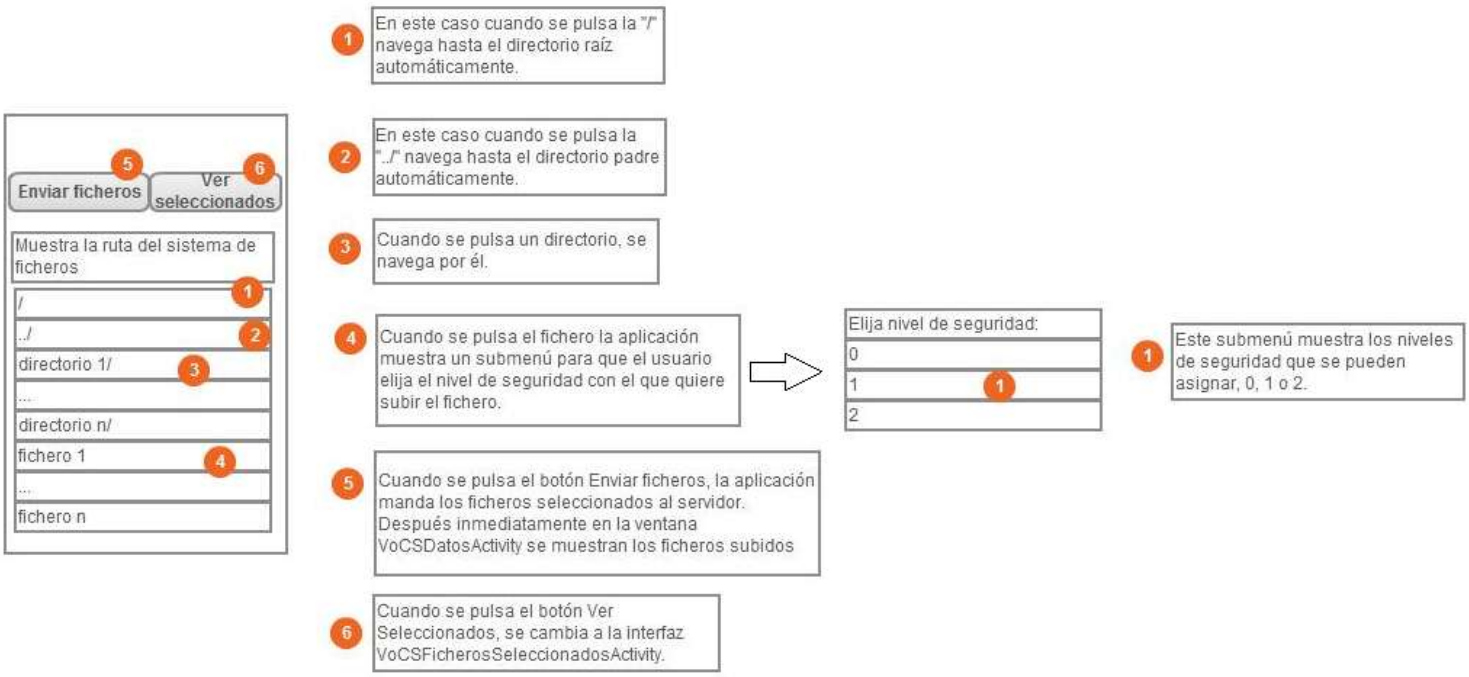


Figura 5.26: Interfaz de la clase VoCSFileExplorer.

Esta interfaz es muy básica, cuando se pulsa un directorio se navega por él con las mismas reglas que la interfaz VoCSDatosActivity. Cuando se pulsa un fichero muestra un submenú para que se elija el nivel de seguridad con el que se quiere subir la aplicación. Cuando se pulsa el botón Enviar ficheros se envían todos los ficheros seleccionados al servidor. Por último cuando se pulsa el botón Ver seleccionados, cambia la interfaz a VoCSFicherosSeleccionadosActivity.

5.2.6.5 Interfaz VoCSFicherosSeleccionadosActivity

Esta interfaz se encarga de listar y borrar, si el usuario lo desea, algún fichero que haya sido seleccionado en la clase VoCSFileExplorerActivity. La figura 5.27 muestra su interfaz.

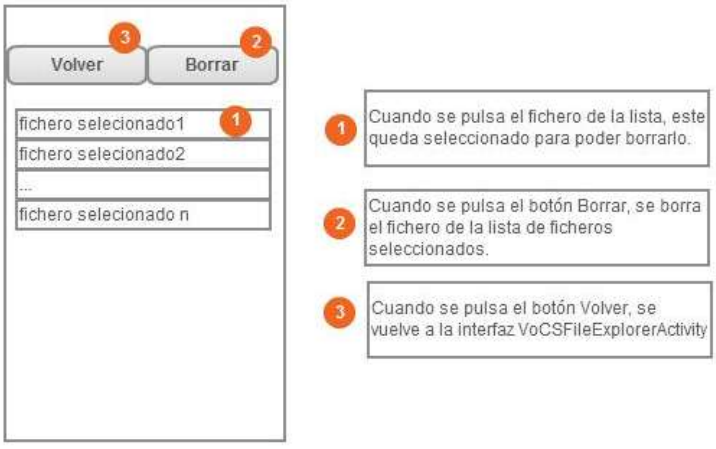


Figura 5.27: Interfaz VoCSFicherosSeleccionadosActivity.

Cuando se pulsa un fichero de la lista de ficheros seleccionados, éste queda marcado y cuando se pulsa el botón Borrar se elimina de la lista de ficheros seleccionados. Cuando se pulsa el botón Volver la aplicación retorna ala interfaz VoCSFileExplorerActivity.

5.2.7 Diagramas de Secuencia

Los diagramas de secuencia en UML muestran la forma en la que los objetos se comunican entre sí al transcurrir el tiempo. Los diagramas constan de objetos que se representan como cuadrados, estímulos más conocidos como mensajes que se representan con una flecha y el tiempo, que será representado por una línea discontinua.

Los diagramas se suelen representar como el visto en la figura 5.28

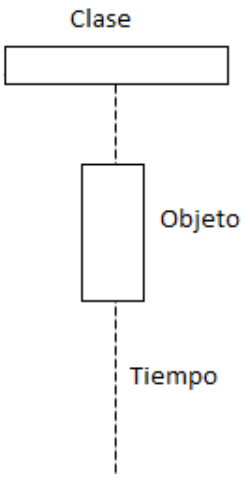
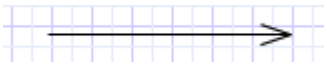
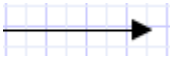


Figura 5.28: Ejemplo de diagrama.

Hay tres tipos de mensajes, mensaje simple que transfiere el control de un objeto a otro, mensaje síncrono que el objeto se queda esperando la respuesta a ese mensaje antes de continuar su trabajo y por último los mensajes asíncronos son aquellos en los que los objetos no se quedan esperando por la respuesta y continúan con su trabajo.

El mensaje simple es como el que se muestra a continuación: 

El mensaje síncrono es como el que se muestra en la comunicación: 

El mensaje asíncrono es como el que se muestra a continuación: 

5.2.7.1 Autenticarse

En este apartado se muestra el diagrama de secuencia que sigue la aplicación para autenticar a un usuario.

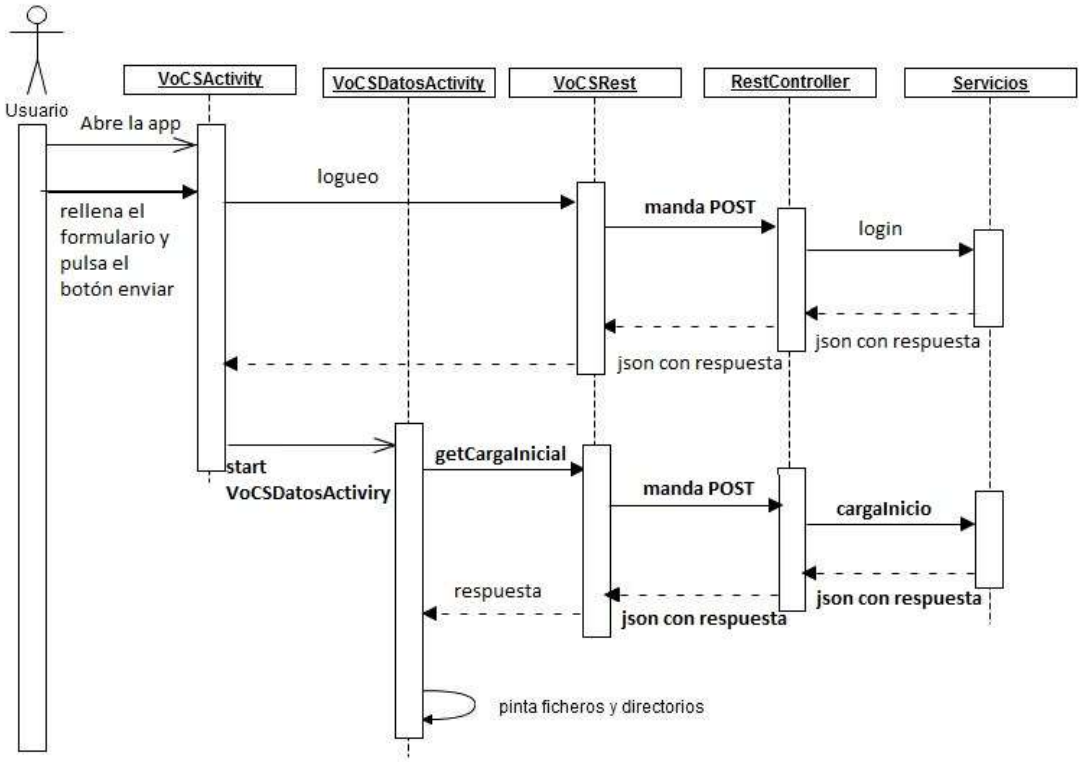


Figura 5.29: Diagrama de secuencia de autenticación.

5.2.7.2 Registrar

En este apartado se muestra el diagrama de secuencia de la función de registro.

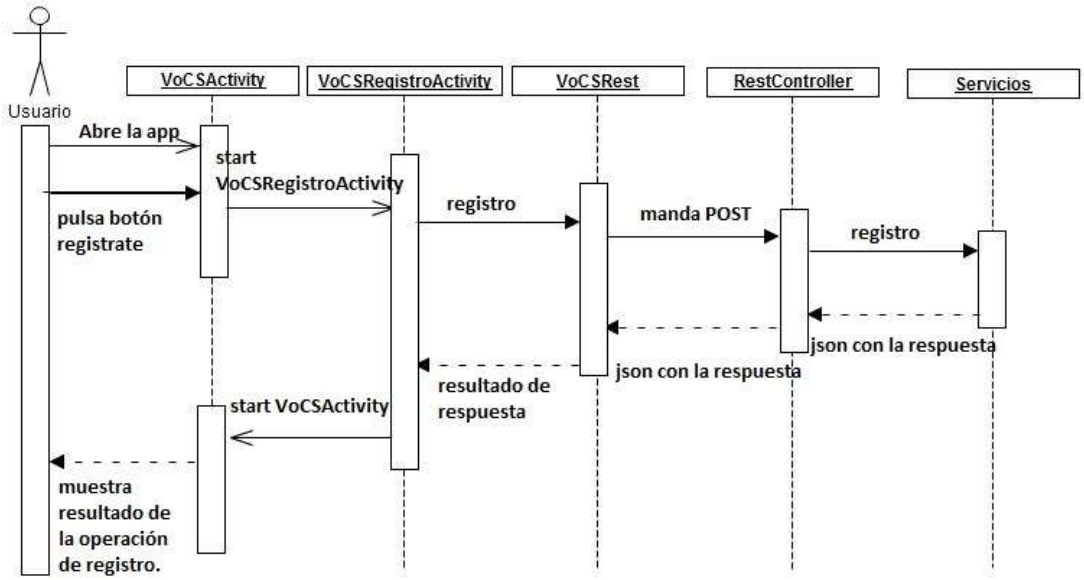


Figura 5.30: Diagrama de secuencia de la operación registrar.

5.2.7.3 Descargar Fichero

En este apartado se muestra el diagrama de secuencia de la función de descargar fichero.

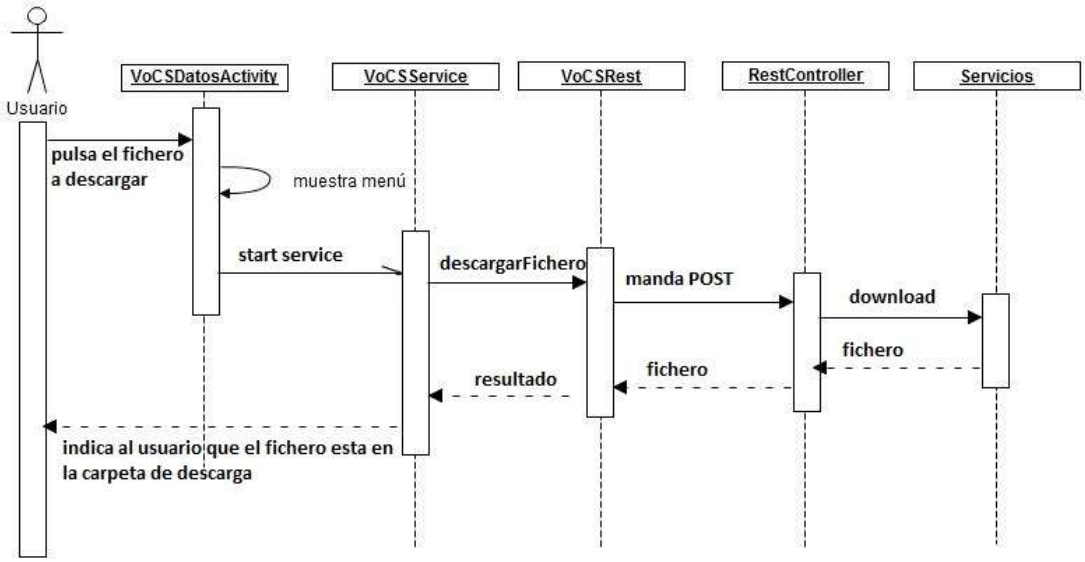


Figura 5.31: Diagrama de secuencia de la opción descargar.

5.2.7.4 Eliminar Fichero

En este apartado se muestra el diagrama de secuencia de la función eliminar fichero.

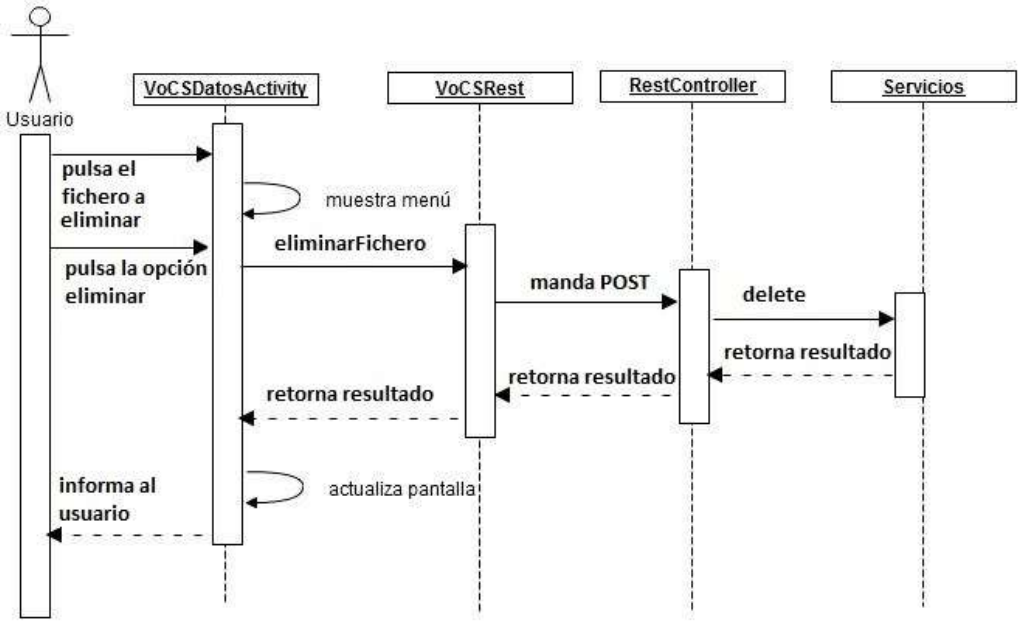


Figura 5.32: Diagrama de secuencia eliminar fichero.

5.2.7.5 Restaurar Fichero

En este apartado se muestran los diagramas de secuencia que se pueden seguir para restaurar un fichero. Las dos formas para realizar la función de restaurar un fichero son las siguientes: desde la lista una versiones anteriores o desde la lista de ficheros borrados.

Versiones Anteriores

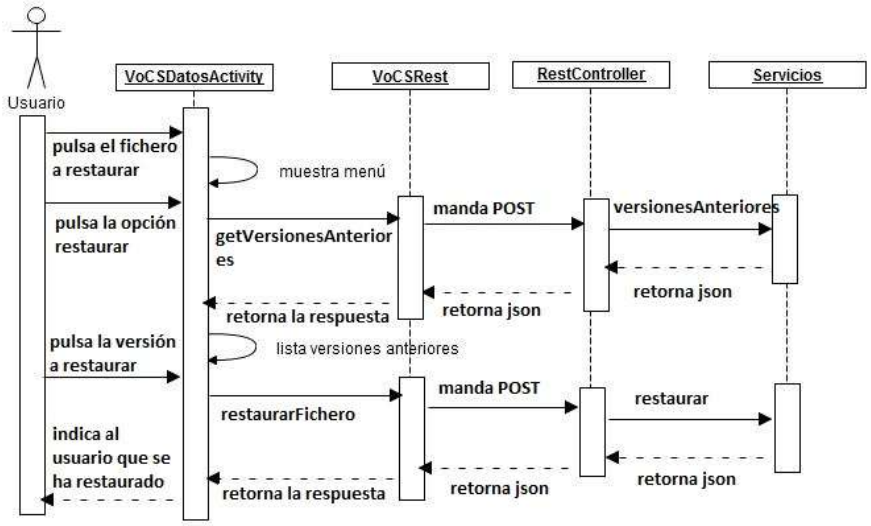


Figura 5.33: Diagrama de secuencia versiones anteriores

Ficheros Borrados

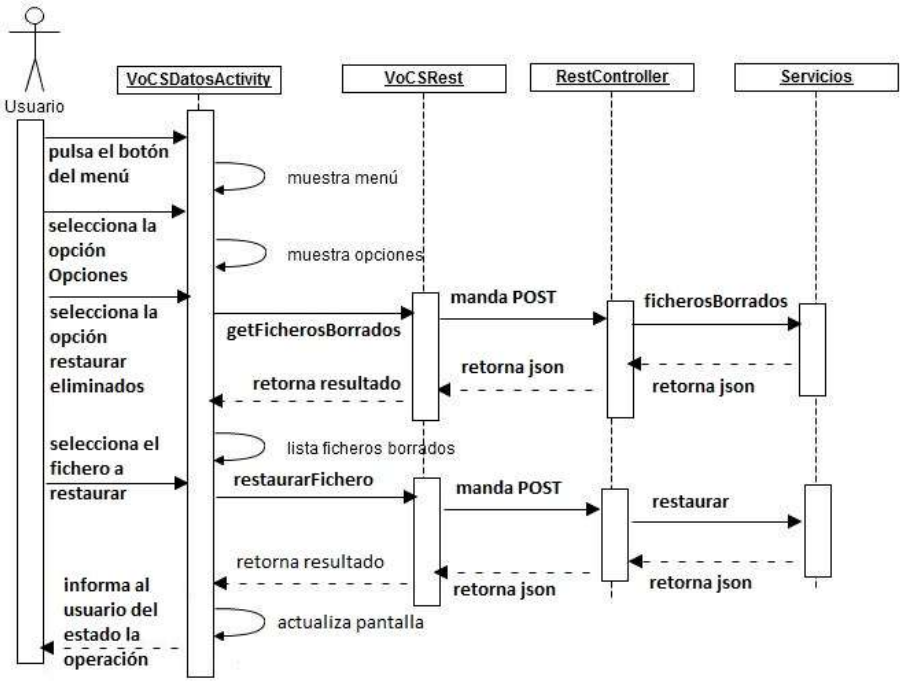


Figura 5.34: Diagrama de secuencia ficheros borrados.

5.2.7.6 Subir Fichero

En este apartado se muestra el diagrama de secuencia de la función subir fichero.

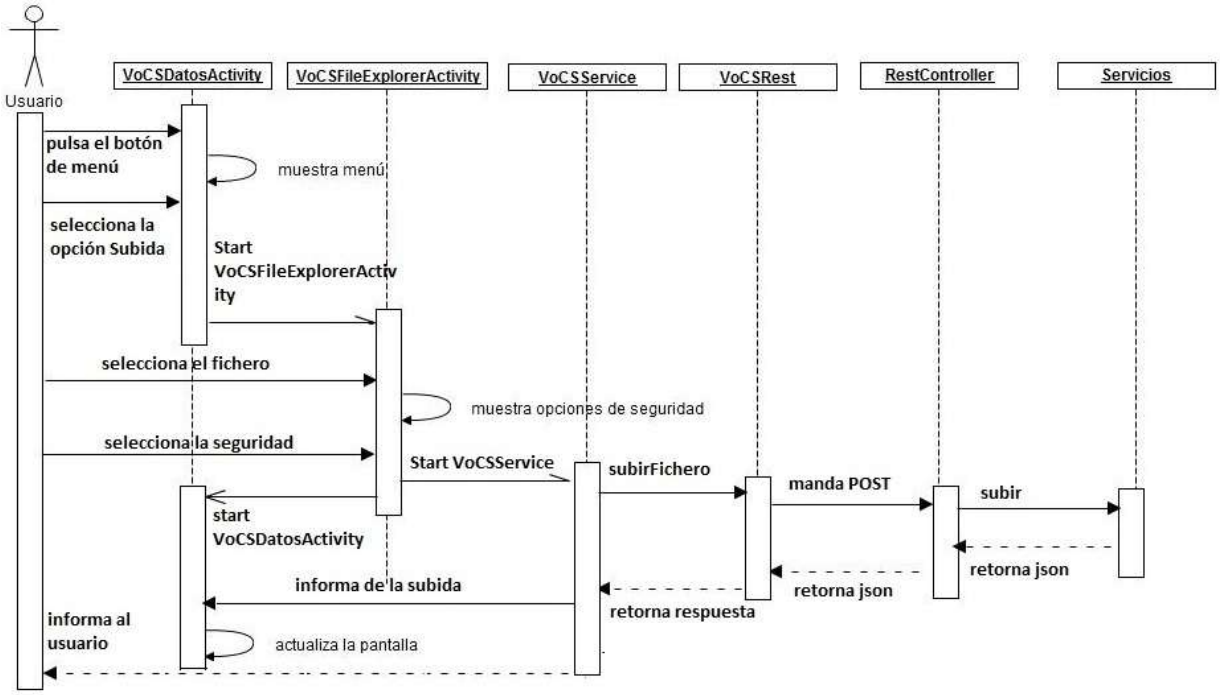


Figura 5.35: Diagrama de secuencia de subir fichero.

5.2.7.7 Renovar Token

Este apartado muestra el diagrama de secuencia de la función renovar token.

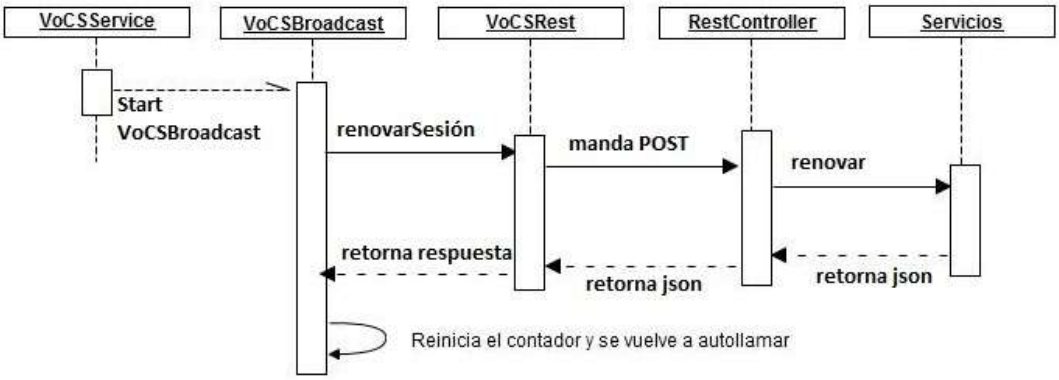


Figura 5.36: Diagrama de secuencia renovar sesión.

5.2.7.8 Crear Directorio

Este apartado muestra el diagrama de secuencia de la función crear directorio.

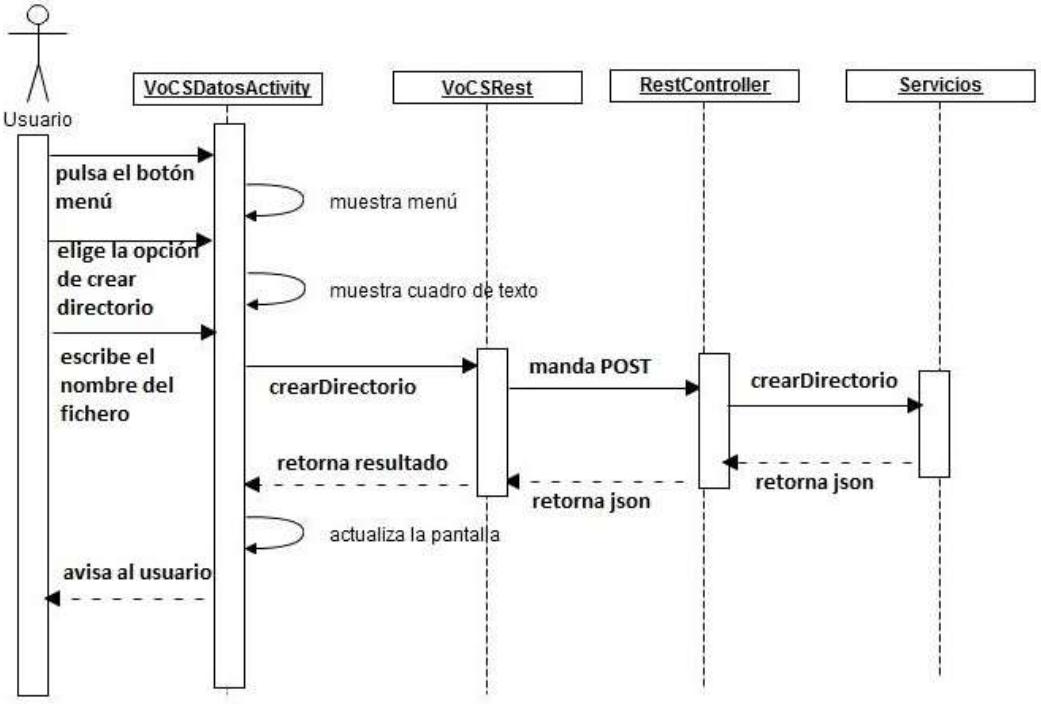


Figura 5.37: Diagrama de secuencia crear directorio.

5.2.7.9 Carga Directorio

Este apartado muestra el diagrama de secuencia de la función crear directorio. En la que la aplicación carga un directorio cuando el usuario navega por un directorio.

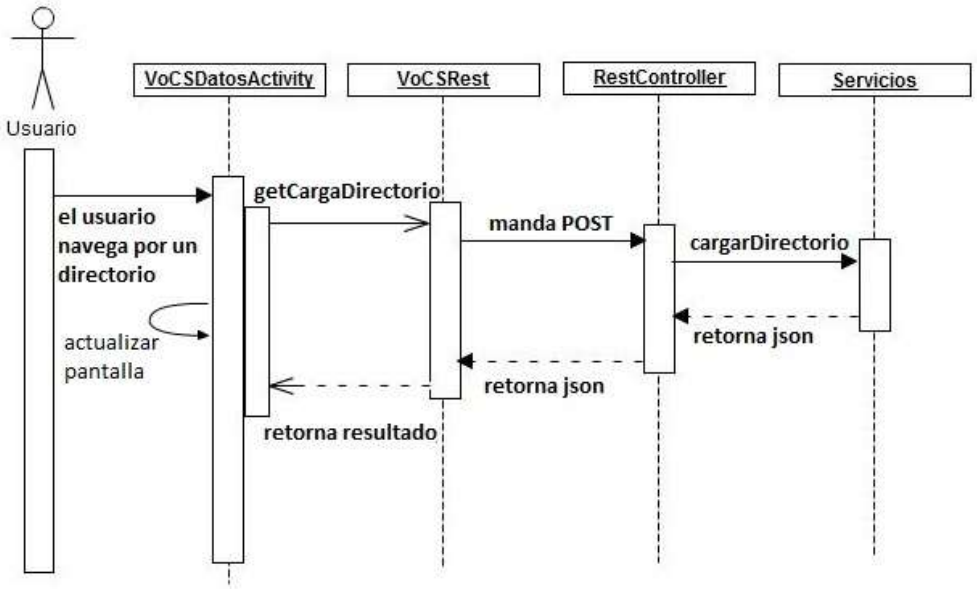


Figura 5.38: Diagrama de secuencia de carga de directorio.

5.2.7.10 Borrar Directorio

Este apartado muestra el diagrama de secuencia de la función borrar directorio. En la que un usuario borra un directorio.

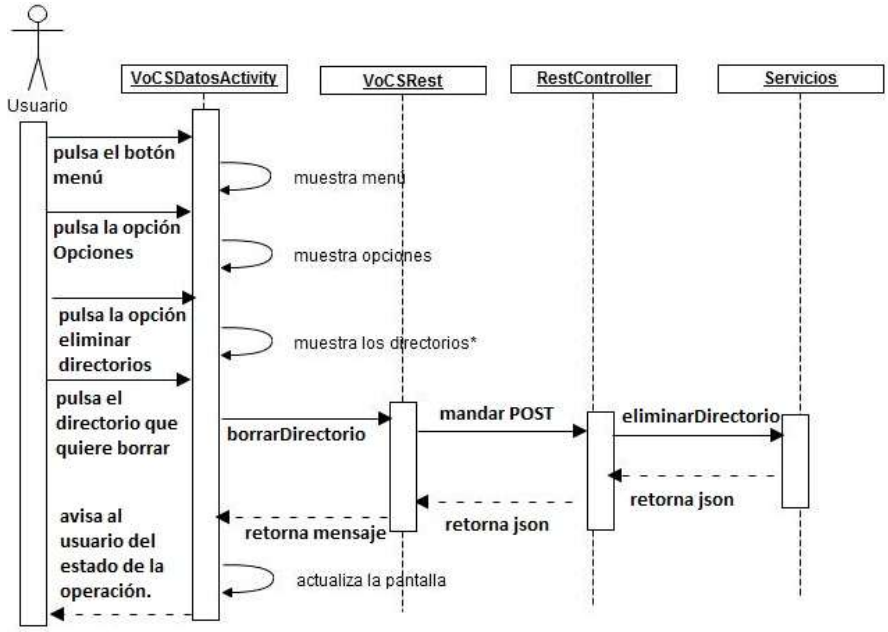


Figura 5.39: Diagrama de secuencia de borrar directorio.

La acción muestra los directorios lleva * debido a que no muestra todos los directorios, sólo muestra los directorios del directorio por el cual el usuario esta navegando. Si el usuario esta navegando por el directorio raíz, sólo mostraría los directorios hijos directos de este.

5.2.7.11 Compartir Directorio

En este apartado se muestra el diagrama de secuencia de la función compartir directorio. En la que un usuario comparte o envía una invitación de compartición de un directorio.

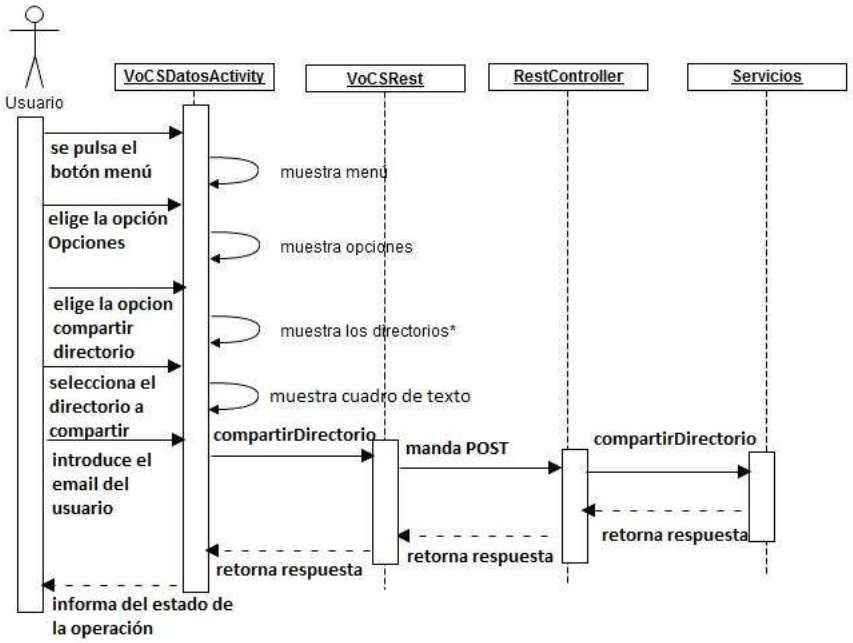


Figura 5.40: Diagrama de secuencia de compartir directorio.

La acción muestra los directorios lleva * debido a que no muestra todos los directorios, sólo muestra los directorios del directorio por el cual el usuario esta navegando. Si el usuario esta navegando por el directorio raíz, sólo mostraría los directorios hijos directos de este.

5.2.7.12 Aceptar Invitación

En este apartado se muestra el diagrama de secuencia de la operación aceptar invitación. En la que un usuario acepta la invitación de un directorio a compartir.

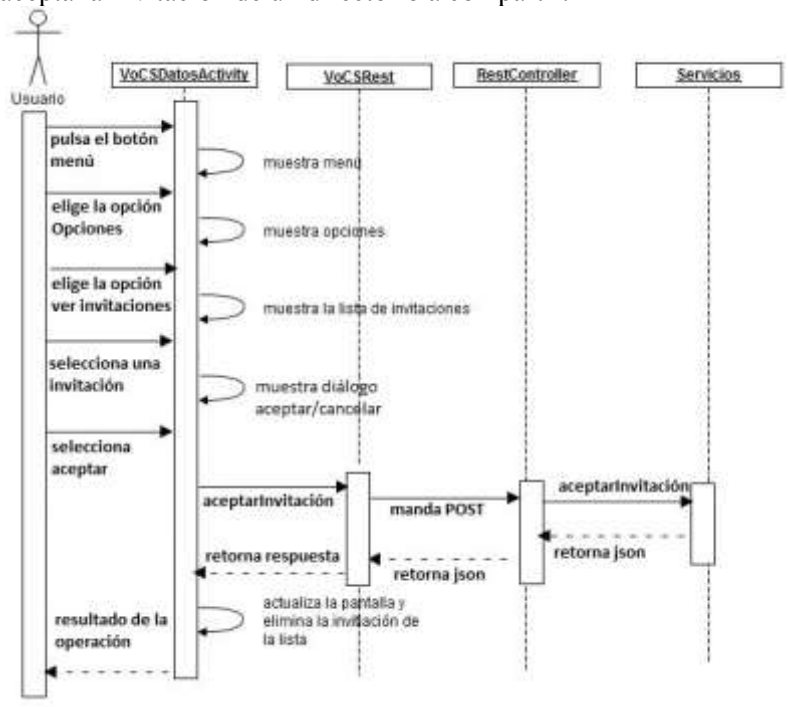


Figura 5.41: Diagrama de secuencia de aceptar invitación.

5.2.7.13 Cancelar Invitación

En este apartado se muestra el diagrama de secuencia de la operación cancelar invitación. En la que un usuario cancela una invitación de compartición de un directorio.

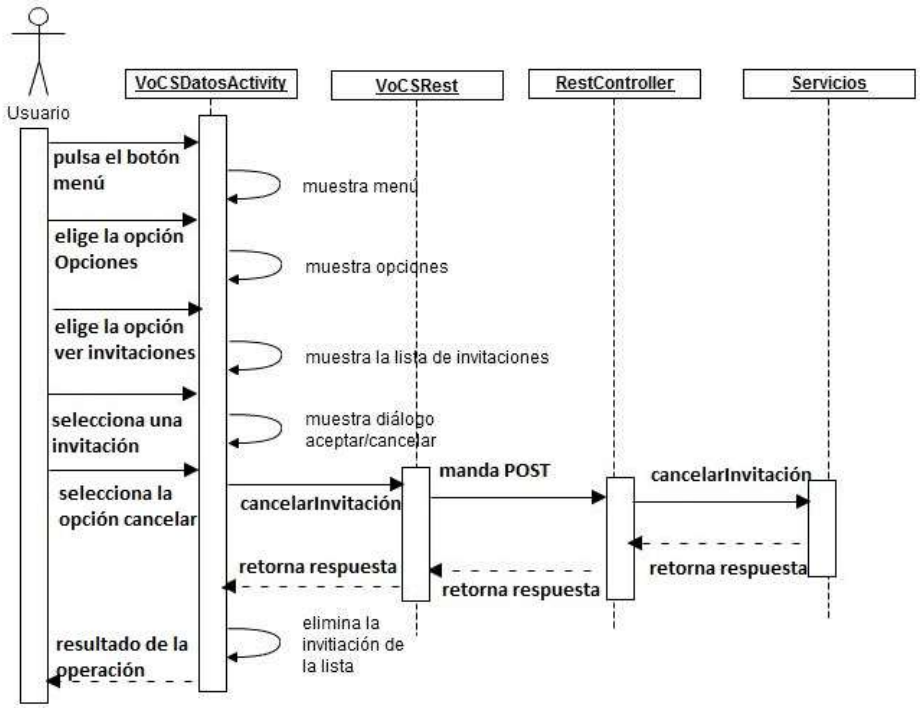


Figura 5.42: Diagrama de secuencia de cancelar invitación.

6 Implementación

La implementación de la aplicación se ha realizado utilizando la API de Google 7 que es compatible con la versión de Android Eclair 2.1 y superior. Para la implementación de las clases que proporcionan el servicio web REST se ha utilizado Zend Framework versión 1.11.10 y el lenguaje de programación PHP 5.3.6.

6.1 Clases Específicas de Android

En este apartado se describirán las clases utilizadas en el desarrollo de la aplicación, que proporciona la API de Google.

6.1.1 Clase Activity

La clase Activity es la base para cualquier aplicación Android con interfaz de usuario. Todas las clases Activitiy interactúan con los usuarios y se encarga de crear la interfaz de usuario.

La clase Activity tiene otras funcionalidades como ventanas flotantes o pueden estar incrustadas en otras actividades.

Las actividades se manejan por el sistema con una pila. El sistema crea una pila de actividades en la cual la pila que esta en cabeza de la pila es la que se esta mostrando. Cuando se inicia otra actividad deja de estar en la cabeza de la pila.

Las actividades tienen cuatro estados fundamentales. El primero de ellos es cuando esta activa. El segundo de ellos es cuando esta pausada cuando el foco de actividad se lo lleva otra actividad, pero sigue viva y visible para el usuario, manteniendo su estado. Sólo es eliminada en casos extremos en los que se necesite memoria. El tercero de ellos es cuando están detenidas, estas actividades mantienen su estado pero no esta visible por el usuario y cuando el sistema necesita memoria la elimina. Su cuarto estado se da cuando están pausadas o detenidas que el sistema puede echarla de memoria, finalizarla o matarla. Pero cuando se tienen que mostrar al usuario, éstas deben de restaurarse y volverse a crear. La figura 6.1 muestra el ciclo de vida de una actividad.

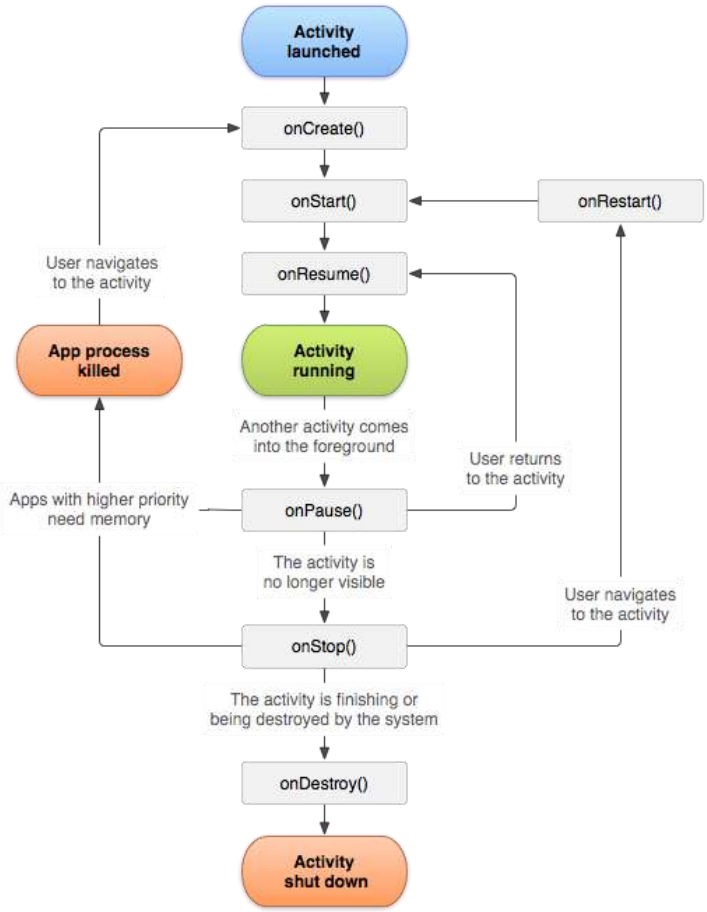


Figura 6.1: Ciclo de vida de una actividad.

La figura 6.1 detalla el ciclo de actividad y consta de los siguientes métodos:

- onCreate: es llamado cuando la actividad se crea. En este método es donde se realizan las inicializaciones estáticas como crear las vistas.
- onStart: se llama cuando la actividad se hace visible al usuario.
- onResume: se llama cuando la actividad empieza a interactuar con el usuario. Cuando está en este punto, la actividad esta en la cabeza de la pila de actividades.
- onPause: se llama cuando el sistema está a punto de comenzar la reanudación de la actividad anterior. Se suele utilizar para confirmar los cambios no guardados en los datos persistentes, parando animaciones y operaciones que pueden hacer que la CPU valla lenta. Hasta que este método no finaliza, no prosigue con la siguiente actividad.
- onStop: se le llama cuando la actividad ya no es visible para el usuario, porque otra actividad se ha reanudado y esta en la cabeza de la pila. Esto ocurre cuando una nueva actividad se esta iniciando o una actividad esta en la cabeza de la pila o está siendo destruida.
- onRestart: se llama siempre que la actividad haya sido detenida y antes de que haya sido iniciada de nuevo.
- onDestroy: La última llamada antes de que la actividad sea destruida. Esto puede suceder porque la actividad haya terminado o porque este siendo destruida temporalmente por el sistema para ahorrar espacio. Hay métodos que permiten distinguir entre estos escenarios.

6.1.2 Clase Service

Un servicio es un componente de aplicación que puede realizar operaciones de larga duración en *background* y no proporciona una interfaz de usuario. Otro método o componente de la aplicación puede iniciar un servicio y continuará funcionando en segundo plano, incluso si el usuario cambia a otra aplicación. Además, un componente puede unirse a un servicio para interactuar con él e incluso realizar la comunicación entre procesos.

Pueden inicializarse dos tipos de servicios:

Started: Un servicio es "*Started*" cuando un componente de la aplicación (por ejemplo, una actividad) lo inicia llamando al método `StartService ()`. Una vez iniciado, un servicio puede ejecutarse en background indefinidamente, incluso si el componente que lo comenzó se destruye.

Bound: Un servicio es "*Bound*" cuando un componente de aplicación se une a ella llamando al método `bindService ()`. Un servicio "*Bound*" ofrece una interfaz de cliente-servidor que permite que los componentes interactúen con el servicio, enviando solicitudes, obteniendo resultados, e incluso hacerlo a través de los procesos de comunicación entre procesos.

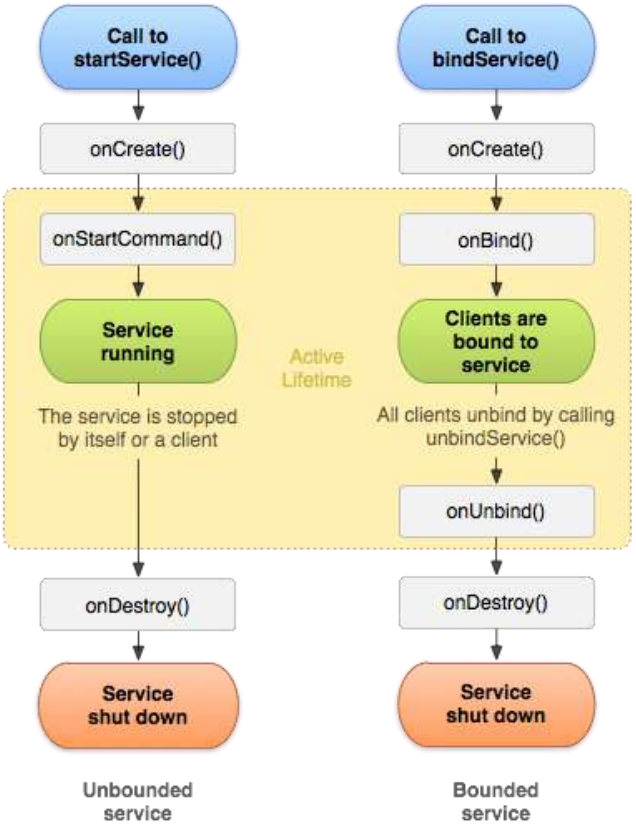


Figura 6.2: Ciclo de vida de un servicio.

La figura 6.2 define el ciclo de vida de los servicios "*Started*" y "*Bound*".

Los métodos de devolución de llamada más importantes son los siguientes:

- onStartCommand: el sistema llama a este método cuando otro componente, como una actividad, pide que el servicio se inicie, llamando a `StartService()`. Una vez que éste método se ejecuta, el servicio se inicia y puede ejecutarse en segundo plano de forma

indefinida. Si se decide implementar esta forma, es su responsabilidad detener el servicio cuando su trabajo está hecho, llamando `stopSelf ()` o `StopService ()`.

- onBind: el sistema llama a este método cuando otro componente pretende enlazar con el servicio (por ejemplo, para realizar RPC), llamando `bindService ()`. En la implementación de este método, se debe proporcionar una interfaz que los clientes utilizan para comunicarse con el servicio, devolviendo un *IBinder*. Siempre se debe implementar este método, pero si no se desea permitir la unión, entonces se debe devolver *null*.

- onCreate: el sistema llama a este método cuando el servicio se creó por primera vez, para realizar de una sola vez los procedimientos de instalación (antes de llamar a cualquiera de los dos `onStartCommand ()` o `onBind ()`. Si el servicio se está ejecutando, este método no se llama.

- onDestroy: el sistema llama a este método cuando el servicio ya no se utiliza y se está destruyendo. El servicio debe implementar este método para limpiar los recursos, tales como hilos, oyentes registrados, receptores, etc. Esta es la última llamada del servicio.

6.1.3 Clase Aplication

Clase base para aquellas aplicaciones que necesiten mantener el estado global de la aplicación. Esta clase es una instancia única disponible y visible para todos los módulos de la aplicación.

Con esta clase se pueden generar variables, las cuales pueden ser accedidas desde cualquier punto de la aplicación.

6.1.4 Clase View

Esta clase representa el bloque de construcción básico para los componentes de interfaz de usuario. Una vista ocupa un área rectangular en la pantalla y es responsable de crear el dibujo y el manejo de eventos. *View* es la clase base para los widgets, que se utilizan para crear componentes interactivos en la interfaz de usuario (botones, campos de texto, etc.). La subclase "*ViewGroup*" es la clase base de los "*layouts*", que son contenedores invisibles que soportan otras vistas y definen sus propiedades de diseño.

Todas las vistas de una ventana están dispuestas en un solo árbol. Se pueden añadir vistas desde el código o desde uno o varios ficheros XML de diseño.

6.1.5 Clase Intent

La clase Intent es una descripción abstracta de una operación que va a ser realizada. Se puede utilizar con startActivity para lanzar una actividad, con broadcastIntent para enviarlo a cualquiera de los componentes BroadcastReceiver y `startService` o `bindService`, para comunicarse con el servidor.

Un intent proporciona facilidad para realizar enlaces en tiempo de ejecución entre código en diferentes aplicaciones. Se utiliza en la puesta en marcha de las actividades y como método de unión de actividades.

6.1.6 Clase BroadcastReceiver

Es la clase base para el código que recibirá Intents enviados por sendBroadcast. Hay dos clases principales de emisiones que se pueden recibir:

Broadcast normales son completamente asíncronos. Todos los receptores de la transmisión se ejecutan en un orden definido.
Broadcast ordenados se entregan cada vez a un receptor. Como cada receptor ejecuta a su vez, se puede propagar un resultado al siguiente receptor o puede ser abortada sin necesidad de que se pasen a los siguientes receptores. El orden de ejecución de los receptores se puede asignar con el atributo de prioridad, si varios receptores tienen el mismo atributo de prioridad el orden de ejecución es arbitrario.

Un objeto BroadcastReceiver sólo es válido durante la duración de la llamada a `OnReceive`. Una vez que la función finalice, el sistema considera que el objeto puede ser terminado y ya no esta activo.

6.1.7 Clase Toast

Toast es una vista que contiene un mensaje corto y rápido para el usuario. La clase Toast ayuda a mostrar y crear estos mensajes.

La vista que se muestra al usuario aparece como una vista flotante sobre la aplicación, nunca recibe el foco. La idea es ser lo más discreto posible sin dejar de mostrar la información que se desea ver.

6.1.8 Dialog

Un dialog es una pequeña ventana que aparece delante de la actividad actual. La actividad subyacente pierde el foco y el Dialog o diálogo asume toda la interacción del usuario. Los diálogos se utilizan normalmente para notificaciones que tienen que interrumpir al usuario y para realizar tareas cortas que se relacionan con la actividad inicial directamente.

La clase Dialog es la clase base para la creación de cuadros de diálogo. Para su creación se deben utilizar alguna de las siguientes subclases:

- AlertDialog: Es un cuadro de diálogo que permite crear hasta tres botones, listas de elementos seleccionables que pueden incluir casillas de verificación u botones de radio.
- ProgressDialog: Es un diálogo que muestra una rueda de progreso o barra de progreso. Es una extensión de AlertDialog y soporta botones.
- DatePickerDialog: Es un diálogo que permite seleccionar una fecha.
- TimePckerDialog: Es un diálogo que permite seleccionar un tiempo.

6.1.9 AsyncTask

La clase AsyncTask permite un fácil y apropiado uso de los hilos al usuario. Permite realizar operaciones en segundo plano y publicar los resultados en el subproceso de la interfaz de usuario sin tener que manipular los hilos.

Los AsyncTask están diseñados para operaciones de corta duración.

Una tarea asíncrona se define mediante una ejecución de una tarea en segundo plano y cuyo resultado se publica en la interfaz de usuario. Una tarea es definida por tres tipos llamados Parámetros, Progreso y Resultado y por cuatro pasos onPreExecute, doInBackground, onProgressUpdate y onPostExecute.

- OnPreExecute, invoca en el subproceso una interfaz de usuario después de que se ejecute la tarea como por ejemplo una barra de estado.

- DoInBackground, se invoca después de onPreExecute. Este paso se utiliza para ejecutar procesos que pueden llevar mucho tiempo.

- OnProgressUpdate, se invoca después de que se llame al método publishProgress. Éste método se utiliza para mostrar cualquier información en la interfaz de usuario mientras el calculo en doInBackground se esta ejecutando. Por ejemplo actualizar la barra de progreso.

- OnPostExecute se invoca una vez que haya terminado el método doInBackground ya que el resultado de la operación doInBackground pasa a este método.

6.2 Desarrollo de la Aplicación

En esta sección se detallarán las funciones más importantes de las clases que componen la aplicación Android.

6.2.1 Clase VoCSActivity

Los métodos más importantes de la clase VoCSActivity son: `onCreate(Bundle savedInstanceState)` que se encarga de inicializar su vista y el método `onClick(View v)` que consta de un switch con dos casos, el primero se da si se pulsa el botón "Entrar" y la segunda se da en caso de que se pulse el botón "Regístrese aquí". La figura 5.3 muestra la interfaz de autenticación de la aplicación.

Figura 6.3: Interfaz de Autenticación.

Cuando se pulsa el botón “Entrar” se llama a la función *logueo* de la clase VoCSRest pasándole el usuario, la contraseña y app que es una instancia de la clase VoCSAplication.

En caso de retornar “1” la función *logueo* (mirar Logueo) entonces se introduce el nick de usuario en una variable global de la aplicación y se llama al activity VoCSDatosActivity. En caso de recibir cualquier otra cosa se muestra un mensaje de error.

```
        case R.id.btRegistrar:
            Intent in = new Intent();
            in.setClass(this, VoCSRegistroActivity.class);
            startActivity(in);
            break;
        }
    }
```

Figura 6.4: Código “case registrar”

La figura 6.4 muestra el fragmento de código en caso de que se pulse el botón “Regístrese aquí”. Este fragmento de código realiza la llamada al activity VoCSRegistroActivity.

6.2.2 Clase VoCSRegistroActivity

Los métodos más importantes de la clase VoCSRegistroActivity son: `onCreate(Bundle savedInstanceState)` que se encarga de inicializar su vista y el método `onClick(View v)` que consta de un switch con dos casos, el primero se da si se pulsa el botón “Enviar” y la segunda se da en caso de que se pulse el botón “Regístrese aquí”. La figura 5.4 muestra la interfaz de registro de la aplicación.

Figura 6.5: Interfaz de registro de la aplicación.

El método `onClick` primero realiza una serie de comprobaciones ya que todos los campos son obligatorios y el email tiene que tener el formato siguiente: una cadena de caracteres, el siguiente carácter “@” seguido de una cadena caracteres y el carácter “.” seguido del dominio. Si el email no sigue este patrón o no se rellena algún campo, no se realiza el registro.

```
VoCSAplication app = (VoCSAplication) getApplicationContext();
VoCSRest rest=new VoCSRest();
Toast.makeText(this, "Enviando los datos", Toast.LENGTH_SHORT).show();
int res=rest.registro(app,nick.getText().toString(),nombre.getText().toString(),apellidos.getText().toString(),
                      email.getText().toString(),dni.getText().toString(),contrasena.getText().toString());
if(res==1){
    Toast.makeText(this, "El registro se ha completado introduzca los datos de logueo.", Toast.LENGTH_SHORT).show();
    Intent intent = new Intent();
    intent.setClass(this, VoCSActivity.class);
    startActivity(intent);
}else if(res==0){
    Toast.makeText(this, "Ese nick ya esta cogido. Elija otro!.", Toast.LENGTH_LONG).show();
}else{
    Toast.makeText(this, "El registro no se ha completado introduzca los datos de nuevo.", Toast.LENGTH_LONG).show();
}
break;
```

Figura 6.6: Código "case enviar".

En caso de pulsar el botón "Enviar", realiza la llamada a la clase VoCSRest, al método `registro` (mirar Registro) pasándole los parámetros del formulario como el Nick, el nombre etc. y app que es una instancia de la clase VoCSAplication. Si la función devuelve 1 entonces se completa el registro y activa el activity VoCSActivity y muestra el mensaje de que todo ha salido bien al usuario. Si el usuario introducido ya existe, informa al usuario de que ese Nick esta cogido y que elija otro. Si no se ha completado bien también se informa al mensaje con otro Toast.

En caso de pulsar el botón "Volver" activa el activity VoCSAtivity.

6.2.3 Clase VoCSDatosActivity

La clase VoCSDatosActivity es la clase más extensa y con más funcionalidades, por lo que se dividirá en partes para explicar todas sus funcionalidades.

6.2.3.1 Mostrar Datos Iniciales

Mostrar los datos iniciales se compone de varias tareas:

El método principal es `onCreate`, es el que se encarga de inicializar el servicio con la llamada `starService` (mirar Clase VoCSService) después llama a la función cargaInicial de la clase VoCSRest (mirar getCargaInicial), llama a la función `inicializarArray` y por último actualiza el tamaño del usuario.

La función `inicializarArray` es la función que crea la lista que muestra los datos con las carpetas y con los ficheros. Esta función realiza las siguientes acciones:

1º Obtiene los datos del nodo actual.
2º Si el nodo actual no es el directorio raíz, en el primer campo de la lista introduce la imagen asociada para acceder al directorio raíz y en el segundo campo la imagen asociada para acceder al directorio anterior.
3º Si tiene directorios, introduce en la lista los directorios con sus respectivos nombres además de la imagen asociada.
4º Si tiene ficheros, comprueba por la extensión que tipo de fichero es y dependiendo de si es pdf o txt o jpg incluirá en la lista un tipo de imagen u otra. En total hay imágenes para pdf, archivos comprimidos como zip, tar, jar y sit; archivos rar, xls y xlsx(Microsoft Excel); html, avi, divx, mp3 y mp4 (archivos multimedia); ppt y pptx (Microsoft Power Point); jpg y png (imágenes); doc y docx (Microsoft Word) y txt; si el fichero no tiene alguna extensión como la anterior pone una imagen estándar.

5º Una vez introducidos los datos en la lista hay que inicializarla. Añadiendo los elementos de la lista en `iconList` que es un elemento de la clase `IconListViewAdapter`.

La figura 6.7 muestra el resultado de las operaciones descritas.

Figura 6.7: Resultado de vista de directorio raíz y directorio hijo.

La carga de los directorios se realiza con la función que permite pulsar en un elemento de la lista. El método es el siguiente `onListItemClick(ListView l, View v, int position, long id)`. Este método se utiliza para mostrar el menú que permite realizar algunas acciones con los ficheros y para navegar por los directorios. Éste método no es tan simple como parece, ya que realiza la llamada a espaldas del usuario para cargar, si lo hubiese, un tercer nivel.

Éste método parte comprobando si esta en el directorio raíz u en otro directorio, ya que si esta en cualquier otro directorio las dos primeras posiciones están ocupadas con "/" y "../" que si se pulsa la primera carga el nodo raíz que esta almacenado en una variable global de la aplicación y si se pulsa la segunda entonces se carga el nodo padre que esta almacenado como uno de los datos del nodo por el cual se esta navegando. La figura 6.8 muestra la acción que se acaba de describir.

```
if(!app.getNodoActual().equals(app.getNodoRaiz())){
    if(posicion==0){
        VoCSNodo raiz= app.getNodoRaiz();
        app.setNodoActual(raiz);
        iconList.clear();
        inicializarArray();
        entro=true;
    }else if(posicion==1){
        VoCSNodo actual= app.getNodoActual();

        app.setNodoActual(actual.nodoPadre);
        iconList.clear();
        inicializarArray();
        entro=true;
    }else{
        posicion=posicion-2;
    }
}
```

Figura 6.8: Código de "/" y "../"

Si no ha entrado en lo descrito anteriormente, se mira si se ha pulsado en un directorio por la posición, ya que en la lista primero se ponen los directorios y después los ficheros, y como se tienen los directorios y los ficheros en un `ArrayList` en el nodo por el que se navega se puede saber cuantos hay. Por lo que si la posición es menor que el tamaño del `ArrayList` de directorios es un directorio, en caso contrario es un fichero y se muestra un menú que se explicará en la sección siguiente.

Si ha pulsado el directorio, el método comprueba si el nodo esta en la lista de nodos pedidos. Si esta entonteces busca el nodo en el árbol, lo inserta como nodo actual e inicializa la lista.

Si no está en la lista de nodos pedidos, comprueba si el nodo se encuentra en el árbol, si esta en el árbol, comprueba si tiene directorios hijos, si no tiene lo asigna como nodo actual y actualiza la vista y la lista con los ficheros del nuevo directorio. Si tiene hijos entonces añade el nodo a la lista de directorios pedidos, el id del directorio al que se quiere acceder a la cola de directorios que se están pidiendo y llama al método que pide la carga.

Si el nodo no se encuentra en el árbol, si el nodo al que se va a acceder esta siendo solicitado debido a que se encuentra en la cola de pedidos, realiza una espera hasta que salga el elemento que se esta solicitando no este en la cola. Después se inicializa un árbol nuevo, debido a que es probable que el árbol haya sido modificado durante el proceso de carga del directorio anterior (mirar getCargaDirectorio) se busca el directorio al que se quiere acceder, si no esta en el árbol ha ocurrido un error y actualiza al directorio raíz. Si esta en el árbol, se inserta en la cola de pedidos el directorio al que se quiere acceder, el nodo en la lista de nodos pedidos y realiza la función `cargaD` en caso de que el nodo tenga directorios hijos. Acto seguido inserta el nodo como actual e inicializa la lista.

La función `cargaD` que se menciona varias veces en los procesos anteriores realiza las siguientes acciones:

Primero crea un `AsyncTask` (mirar AsyncTask) después llama a la función `getCargaNodo` de la clase `VocsRest` y si el resultado es “Ok” saca el primer elemento de la cola de pedidos. Si el resultado no es “Ok” muestra mensaje de error y saca el elemento de la cola de pedidos. Figura 5.14.

```
public void cargaD(final VoCSAplication app,final VoCSNodo nodo){
    AsyncTask<Void, Void, Void> task=new AsyncTask<Void, Void, Void>() {

        @Override
        protected Void doInBackground(Void... arg0) {
            // TODO Auto-generated method stub
            VoCSRest rest=new VoCSRest();
            String resultado=rest.getCargaDirectorio(app,nodo);
            Queue<Integer> cola=app.getCola();
            if(!resultado.equals("Ok")){
                cola.poll();
                app.setCola(cola);
                Toast.makeText(app.getApplicationContext(), "Ha ocurrido un error",Toast.LENGTH_LONG).show();
            }else{
                cola.poll();
                app.setCola(cola);
            }
            return null;
        }
    };
    task.execute();
}
```

Figura 6.9: Método cargad, AsyncTask.

6.2.3.2 Menús

En esta sección se exponen los diferentes menús que dispone la aplicación. La clase VoCSDatosActivity dispone de dos tipos de menú.

El primer tipo de menú se puede apreciar cuando un usuario pulsa un fichero de la lista, entonces se llama a la función `crearDialogoSeleccion` que muestra un submenú que permite seleccionar entre distintas opciones como "descargar", "eliminar", "restaurar" y "cancelar". La figura 6.10 muestra el submenú.

Figura 6.10: Submenú ficheros.

La opción restaurar es la única que muestra un submenú con las versiones anteriores del fichero y la fecha en la que fue sustituida por una versión posterior. La figura 6.11 muestra el submenú.

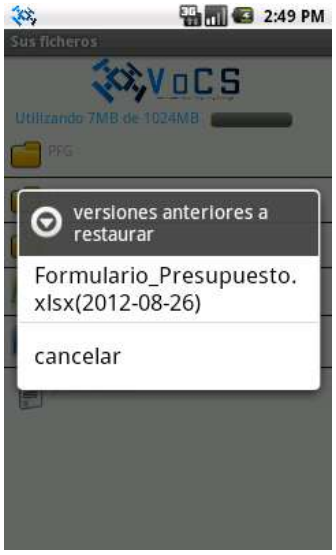


Figura 6.11: Submenú versiones anteriores.

Ahora se va a describir brevemente como que es lo que hacen las acciones descritas en la figura 6.10.

La opción “descargar” realiza lo siguiente, muestra mensaje al usuario que le indica que se esta descargando el fichero seleccionado y se inician unas variables globales de la aplicación que se utilizarán en otra parte de la aplicación, las variables son el nombre y el identificador del fichero de descarga y si se pulsa la opción de descarga. Acto seguido crea un hilo para mejorar el rendimiento con la clase AsyncTask en la cual llama al servicio que como ya fue inicializado entrará en la función `onStartCommand` que ejecutará la función de descargar (mirar función VoCSService).

La opción de “eliminar” lo primero que realiza es llamar a la función eliminarfichero de la clase VoCSRest (mirar eliminarfichero). Si la respuesta es “Ok” muestra al usuario que el fichero se ha eliminado correctamente, acto seguido elimina de la lista el fichero seleccionado y elimina en los datos del nodo el nombre del fichero y el id del fichero. En caso de no recibir “Ok” se muestra al usuario un mensaje indicando que no se ha eliminado el fichero.

La opción “restaurar” llama a la función `getVersionesAnteriores` mirar clase VoCSRest, y si retorna true, se llama a la función crearDialogoFicheros, si no indica al usuario que no hay ficheros. La función crearDialogoFicheros funciona igual tanto para la opción restaurar como para la opción del menú opciones, restaurar borrados. Aunque depende de donde se llame si desde restaurar versiones o desde ficheros borrados, entonces realiza algunas acciones diferentes. Esta función muestra los ficheros como se muestra en la figura 6.11 y cuando se selecciona un fichero se llama a la función VoCSRest restaurarFichero y si retorna “Ok” se informa al usuario que se ha restaurado correctamente y al ser una versión anterior, no se refresca la pantalla si no que sólo se cambia la información del nodo en el que está el fichero, insertando el id del fichero restaurado y eliminando el id anterior. No hace falta cambiar nada más ya que el nombre sigue siendo el mismo.

Si se elige la opción “cancelar” entonces se cierra el submenú quitándole el foco de ejecución al diálogo que contiene el submenú.

El segundo tipo de menú es el que se muestra cuando se pulsa el botón menú del dispositivo móvil como se puede ver en la figura 6.10 el menú de VoCSDatosActivity dispone de cuatro opciones “Subir”, “Crear directorio”, “Opciones” y “Exit”.

Figura 6.12: Menú de VoCSDatosActivity.

La opción “Subir” llama a la actividad VoCSFileExplorerActivity.

La opción “Crear directorio” muestra un diálogo el cual informa al usuario de que tiene que escribir el nombre del directorio, muestra un campo que permite escribir y dos botones “Ok” y “cancel”. Si se pulsa “cancel” se quita diálogo, pero si se escribe algo y se pulsa aceptar, se llama a la función crearDirectorio de la clase VoCSRest. Si la función devuelve “OK” entonces se actualiza la pantalla para mostrar el nuevo directorio creado.

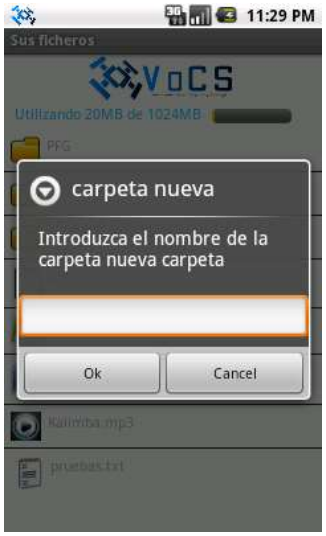


Figura 6.13: Interfaz que muestra el diálogo para crear directorio.

Si se pulsa la opción “Opciones”, se muestra un submenú con las siguientes opciones “restaurar eliminados”, “eliminar directorio”, “compartir directorio”, “ver invitaciones” y “cancelar”. La figura 6.14 muestra la interfaz que muestra el submenú.

Figura 6.14: Interfaz que muestra submenú de Opciones.

La opción “restaurar eliminados” realiza la llamada a getFicherosBorrados de la clase VoCSRest que obtiene todos los ficheros inactivos del usuario. Si la respuesta es “true” y hay ficheros a restaurar entonces se llama a la función `crearDiálogoFicheros` que como se explicó en la funcionalidad de restaurar versiones anteriores, comparte método, salvo que si se llama desde esta opción hay cierta diferencia. El método muestra un submenú como el que se ve en la figura 6.15 en el cual están todos los ficheros inactivos del usuario. Cuando se selecciona un fichero, primero se comprueba si el directorio al que corresponde el usuario esta en el árbol y por lo tanto cargado en la aplicación, si no esta cargado no se hace nada ya que en el servidor si se habrá restaurado y cuando se solicite reaparecerá. Si esta en el árbol, entonces obtiene el nodo con los datos.

Comprueba si el nodo contiene el fichero, si lo tiene lo trata como una versión anterior y sólo modifica su id. En caso de no tener el fichero introduce en el nodo los datos del fichero: el nombre y su id. Por último actualizará la pantalla.

Figura 6.15: Ficheros inactivos del usuario.

La opción "eliminar directorio", muestra los directorios del directorio por el que se esta navegando en ese momento, como muestra la figura 6.16. Esto se muestra con llamando a la función `crearDialogoDirectorios` que comparte método con la opción compartir directorio, aunque con sutiles diferencias.

Figura 6.16: Directorios a borrar.

Cuando el usuario selecciona un directorio, llama a la función borrarDirectorio de la clase VoCSRest, si devuelve "Ok" entonces elimina el directorio del árbol y actualiza la pantalla.

La opción "compartir directorio", muestra los directorios del directorio por el que se esta navegando en ese momento como muestra la figura 6.17. Esto se muestra con llamando a la función `crearDialogoDirectorios` que comparte método con la opción borrar directorios.

Figura 6.17: Directorios a compartir.

El método `crearDialogoDirectorios` para la opción "compartir directorio", muestra un diálogo para que el usuario pueda escribir un email al cual se enviará la invitación de compartición de directorio. La figura 6.18 muestra el diálogo de petición de email.

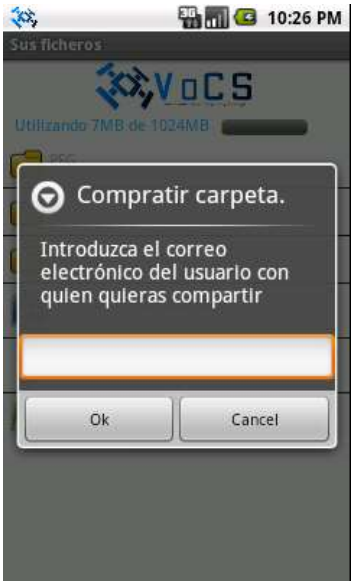


Figura 6.18: Diálogo que pide email de usuario.

Cuando el usuario introduce el email correspondiente y pulsa el botón "Ok", entonces se llama a la función compartirDirectorio de la clase VoCSRest. En caso de recibir "Ok" de la función se muestra un mensaje informativo al usuario informándole de que cuando el usuario acepte la invitación compartirán la carpeta.

La opción "ver invitaciones" muestra las invitaciones pendientes de aceptar o rechazar que tiene el usuario y permite aceptarla o rechazarla. Esta opción llama a la función `crearDialogoPeticiones` que es la que se encarga de crear y mostrar el diálogo con las invitaciones que cada una esta compuesta del nombre de la carpeta, del nick del usuario y del email del usuario. La figura 6.19 muestra las invitaciones del usuario.

Figura 6.19: Invitaciones del usuario.

Cuando el usuario selecciona la invitación se muestra el diálogo de la figura 6.20.

Figura 6.20: Aceptar invitación.

Este diálogo consta de dos botones "Ok" y "Cancelar". Si el usuario selecciona "Ok", se llama a la función aceptarInvitación de la clase VoCSRest. Si recibe un "Ok" entonces se muestra un mensaje indicándole al usuario que todo ha salido bien, se actualiza la pantalla para mostrar al usuario el nuevo directorio y se elimina la petición.

Si el usuario selecciona "Cancelar", se llama a la función cancelarInvitación de la clase VoCSRest. Si recibe un "Ok" entonces se muestra un mensaje indicándole al usuario que todo ha salido bien y se elimina la petición.

6.2.4 Clase FileExplorerActivity

Esta clase se encarga de mostrar el sistema de ficheros interno del teléfono para poder seleccionar un fichero y mandarlo al servidor. Consta de cinco métodos, `onCreate`, `getDir`, `onListItemClick`, `crearDialogoSeleccion` y `onClick`.

El método `onCreate` se encarga de inicializar todos los elementos de la vista llamando a los métodos como `getDir`.

El método `getDir` se encarga de crear la lista de ficheros y directorios que se muestra en la pantalla. Lo primero que realiza es mostrar la ruta a la que ha accedido el fichero. Si la ruta no es la del directorio principal, entonces introduce en el primer elemento de la lista "/" y en el segundo "../". A continuación obtiene todos los archivos y los archivos que son directorios les añade una "/" al final de su nombre. Los archivos los añade a la lista.

Figura 6.21: Sistema de ficheros interno del teléfono

Cuando se pulsa en un directorio o en un fichero se llama a la función `onListItemClick`. Este método cuando se pulsa un directorio, comprueba si se puede acceder a él o no. Si se puede acceder a él, llama al método `getDir` pasándole por parámetro la ruta del directorio. Si no puede acceder muestra un diálogo indicando que no se puede acceder a la carpeta.

Si se pulsa un fichero, se añade el fichero en la lista de ficheros a subir y se llama a la función `crearDialogoSeleccion` que muestra un submenú con los niveles de seguridad con los que puede subir el fichero. La figura 6.22 muestra el diálogo de los niveles de seguridad. Que cuando se selecciona uno se añade en la lista de seguridad el nivel de seguridad del fichero a subir. Si se pulsa cancelar se elimina el fichero de la lista de ficheros a subir.

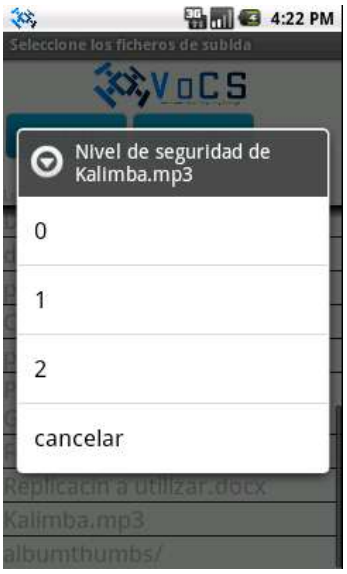


Figura 6.22: Niveles de seguridad.

En la figura 6.21 se pueden observar dos botones "Enviar Ficheros" y "Seleccionados". Si se pulsa "Enviar Ficheros" el método `onClick` activa la variable global que indica que se van a subir ficheros y llama al servicio a su clase `onStartCommand` y finaliza el activity con el método finish.

Si se pulsa el botón "Seleccionados" entonces llama al activity VoCSFicherosSeleccionadosActivity.

6.2.5 Clase VoCSFicherosSeleccionadosActivity

La clase FicherosSeleccionadosActivity es una clase sencilla que solo realiza una función concreta, mostrar los ficheros que se han seleccionado y eliminarlos de la lista.

Figura 6.23: VoCSFicherosSeleccionadosActivity.

Esta clase consta de dos botones y una lista que muestra los ficheros que se han seleccionado para subir al servidor. Si el usuario pulsa el fichero, este se selecciona para poder borrarlo en caso de que se pulse el botón borrar.

El método `onClick` que se activa cuando se pulsa alguno botones, tiene dos caminos. Cuando se pulsa el botón "Volver" finaliza la actividad con el método finish. Si se pulsa el botón "Borrar" elimina el fichero que ha sido seleccionado de la lista, el fichero de la lista de ficheros a subir y el nivel de seguridad de la lista del nivel de seguridad de los ficheros a subir.

Si no se selecciona ningún fichero y se muestra un mensaje al usuario que indica que el usuario no ha seleccionado ningún fichero.

6.2.6 Clase VoCSService

La clase VoCSService es la clase que se encarga de generar el servicio de la aplicación que permite ejecutar las tareas subir fichero y descargar fichero en segundo plano que son las tareas que más tiempo necesitan.

El servicio consta de 4 métodos pero los más importantes son el método `onStarComand` y el método `renovarSesion`. Los otros dos métodos son `onCreate` que inicializa el servicio y `onDestroy` que destruye el servicio.

El método `renovarSesion` se encarga de llamar por primera vez a la clase VoCSBroadcast que es la que realiza la función de renovar sesión. Este método no llama directamente al broadcast si no que realiza una llamada al servicio del sistema de despertador y le fija un tiempo de 9 minutos, cuando este tiempo termina llama automáticamente a la clase VoCSBroadcast.

```
public void renovarSesion(){
    Intent inten = new Intent();
    inten.setClass(getApplicationContext(), VoCSBroadcast.class);
    PendingIntent pIntent=PendingIntent.getBroadcast(getApplicationContext(), 234324243, inten,0);
    AlarmManager aManager=(AlarmManager) getSystemService(ALARM_SERVICE);
    aManager.set(AlarmManager.RTC_WAKEUP, System.currentTimeMillis()+(9*1000*60), pIntent);
}
```

Figura 6.24: Código RenovarSesión.

El método `onStartComand` es el método más extenso ya que es el que se encarga de realizar las funciones de descarga y de subida. Cuando se llama a este método desde una subida, primero se muestra crea una notificación como la que se muestra en la figura 6.25.

Figura 6.25: Notificación.

Una vez creada la notificación se recorre la lista de ficheros a subir. Antes de mandar el primer fichero, se calcula el tamaño que ocupa y antes de enviarlo se comprueba si supera el tamaño máximo. Si supera el tamaño máximo no lo envía y muestra mensaje informativo al usuario. Si no lo supera llama a la función subirFichero de VoCSRest. Si el resultado es positivo y hay otro fichero en la lista de ficheros a subir entonces actualiza la notificación mostrando el nombre del siguiente fichero a subir y aumentando la barra de progreso de la notificación. Después muestra un mensaje al usuario indicando que la subida se ha realizado correctamente y aumenta el espacio utilizado por el usuario. Si el último fichero de la lista de ficheros subidos se sube correctamente, limpia las listas de niveles de seguridad y ficheros a subir; y actualiza la notificación indicando que la subida se ha realizado correctamente.

Si se pulsa en la notificación creada se puede acceder a la actividad VoCSDatosActivity.

Cuando se llama a este método desde una descarga, se crea la notificación que indica que se inicia la descarga y llama a la función descargarFichero de la clase VoCSRest. Si retorna el resultado correcto, actualiza el estado de la notificación e informa de que la descarga se ha producido con éxito e informa al usuario con un mensaje Toast que la función se ha realizado correctamente.

6.2.7 Clase VoCSBroadcast

La clase VoCSBroadcast extiende de la clase `BroadcastReceivier` es una clase sencilla que consta de un solo método. El método es `onReceive`, este método crea un hilo con la clase `AsyncTask` y en su método `doInBackground` llama a la función renovarSesión de la clase VoCSRest. Si el resultado es "Nok" entonces pone el token de sesión a `null` y termina la ejecución.

Si la función devuelve "Ok" entonces realiza una llamada a un servicio del sistema. El servicio de sistema es el de alarmas de la clase `AlarmManager`, y se introduce una petición de que cuando pasen 9 minutos se vuelva a llamar a la clase VoCSBroadcast automáticamente.

6.2.8 Clase VoCSRest

La clase VoCSRest es la que se encarga de realizar las funciones que conectan con el servidor. Estas funciones envían los mensajes correspondientes siguiendo el patrón de la API y recibe y trata las respuestas del servidor.

Esta clase tiene dos métodos estándar que son utilizados por varios métodos que son los siguientes:

- El método conexión que se encarga de crear la conexión HTTP.
- El método `newSslSocketFactory` que es le método que permite establecer la conexión SSL con el servidor que tiene un certificado autofirmado. Este método es necesario ya que Android no permite conexiones SSL con certificados que no estén en el dispositivo. El método se encarga de introducir el certificado en el dispositivo y proporcionárselo a la conexión para evitar el rechazo de la conexión.

Los métodos que realizan las llamadas al servicio REST son las siguientes:

6.2.8.1 Logueo

La función de logueo es la función que se encarga de autenticar al usuario y lo primero que hace es llamar a la función conexión que le proporciona la conexión creada, esta la guarda en una variable global de la aplicación para que las demás funciones la puedan coger.

Después crea el mensaje que se envía al servidor, para ello crea un objeto json e introduce el usuario y la contraseña y lo introduce en un objeto llamado "entity". Acto seguido introduce la cadena "logueo" en el objeto con nombre clave "entity". Mirar Mensajes que se Envían al Servidor.

```
HttpPost httpPost = new HttpPost(url);

MultipartEntity reqEntity = new MultipartEntity(HttpMultipartMode.BROWSER_COMPATIBLE);

try
{

    JSONObject jsonObj = new JSONObject();

    //Add some values
    jsonObj.put("param1", usuario);
    jsonObj.put("param2", password);

    //Add the JSON "part"
    reqEntity.addPart("entity", new StringBody(jsonObj.toString()));
    reqEntity.addPart("metodo",new StringBody("logueo"));
    httpPost.setEntity(reqEntity);
    //Execute the request "POST"
    HttpResponse httpResp = httpClient.execute(httpPost);
```

Figura 6.26: Código que muestra como se genera un mensaje.

La figura 6.29 muestra el código de ejemplo para esta función que explica como introducir los datos en el mensaje de autenticación que se envía al usuario. Este código es muy similar en el resto de funciones, salvo que en algunas se enviarán más o menos parámetros.

A continuación, la función comprueba si el mensaje de respuesta contiene el código "201" que es el código de aceptación base para todos los mensajes, si no lo tiene retorna el entero "2". Si tiene el código, entonces trata el mensaje de respuesta que retorna el servidor para el caso de autenticación. Mirar Mensajes que se Envían desde el Servidor.

Después de comprobar el código del mensaje, recupera el mensaje y trata la respuesta que esta en formato json. La respuesta viene en un array de elementos json y el primer elemento es el token de sesión que almacena en una variable de sesión de la aplicación. Los siguientes elementos son la cantidad de invitaciones que tiene el usuario por aceptar o rechazar. Estos elementos son arrays de elementos en los cuales en la posición 0 esta el email del anfitrión de la invitación, el nombre del directorio a compartir, el id del directorio a compartir y el Nick del anfitrión del directorio. Estos datos se guardan en listas en forma de variable global de aplicación.

Para terminar retorna 1 y si se produce alguna excepción un 2.

6.2.8.2 Registro

El método registro es se encarga de solicitar la operación de registro al servidor. Esta es una de las funciones que no reutiliza la conexión creada por la función logueo ya que el usuario no se puede autenticar no esta registrado.

Lo primero que realiza es llamar a la función conexión y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

Después comprueba que el código del mensaje se corresponde con el 201, en caso contrario retorna 2; y recoge la respuesta del mensaje que esta en formato json. Convierte el mensaje en una cadena y comprueba si en la cadena pone “Registrado”, si es así retorna un 1, si no es así retorna un 0.

6.2.8.3 GetCargaInicial

El método getCargaInicial es el que se encarga de solicitar la carga inicial del sistema de ficheros de la aplicación y tratar la respuesta para inicializar el árbol.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

A continuación compara el código del mensaje y si no tiene el código 201, retorna `null`. Después trata el mensaje de respuesta del servidor que es un elemento json, en concreto un array que contiene en la posición 0 el espacio utilizado por el usuario, en la posición 1 el espacio total del usuario y los siguientes elementos son los metadatos del árbol por lo que en la posición 2 van los metadatos de la raíz, pero las siguientes posiciones están ocupadas en función a la cantidad de directorios o nodos hijos que tenga. Por último inicializa el árbol, le pasa todos los nodos que tenga e introduce el árbol, el nodo raíz y el nodo actual como variables globales.

6.2.8.4 DescargarFichero

La función descargarFichero se encarga de enviar la petición de descarga al servidor y de almacenar en el directorio “descargas VoCS” el fichero descargado.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

A continuación compara si el código del mensaje es el 201, si no lo es retorna “error”. Después comprueba si la tarjeta SD esta montada en el dispositivo, en caso contrario retorna “SDno”. Acto

seguido crea, en caso de no estar creada, la carpeta de descarga de la aplicación donde se descargarán los ficheros. El fichero transferido se guarda en un array de bytes que será reconstruido en un fichero con el mismo nombre en la carpeta de descargas creadas.

6.2.8.5 EliminarFichero

Esta función se encarga de solicitar al servidor la operación de eliminar un fichero existente del usuario.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

A continuación compara si el código del mensaje es el 201, si no lo es retorna "Nok". Para finalizar comprueba si el mensaje recibido por el usuario contiene la cadena "Eliminado", si es así retorna "Ok", en caso contrario retorna "Nok".

6.2.8.6 GetFicherosBorrados

El método getFicherosBorrados se encarga de solicitar al servidor todos los ficheros borrados de los que dispone el usuario para poder restaurarlos.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

A continuación compara si el código del mensaje es el 201, si no lo es retorna `false`. Acto seguido comprueba que la respuesta no sea nula por no tener ningún fichero eliminado, de ser así retorna false. Si la respuesta no es nula, obtiene del mensaje en formato json los elementos con nombre clave "nombres", que es un array que contiene los nombres de los ficheros, "ficheros", que es un array que contiene los id de los ficheros; y "fecha", que es un array que contiene la fecha en la que fueron borrados los ficheros. Estos arrays se pasan a listas y se guardan como variables globales de la aplicación.

6.2.8.7 GetVersionesAnteriores

El método `getVersionesAnteriores` se encarga de solicitar al servidor todas las versiones anteriores de un fichero para poder restaurarlas.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

A continuación compara si el código del mensaje es el 201, si no lo es retorna `false`. El tratamiento del mensaje es igual que la función anterior salvo que las variables globales en las que se guardan las listas creadas son diferentes para no superponerlas con las listas creadas en ficherosBorrados.

6.2.8.8 RestaurarFichero

El método `restaurarFichero` se encarga de solicitar al servidor que restaure el fichero deseado.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

A continuación compara si el código del mensaje es el 201, si no lo es retorna "NoK". Si la respuesta enviada es distinta de "NoRestaurado", entonces se obtiene del mensaje en formato json el elemento con nombre clave "directorio" que contiene el id del directorio del fichero donde se tiene que restaurar. Este id se guarda en una variable global de la aplicación y se retorna "Ok".

6.2.8.9 SubirFichero

El método `subirFichero` se encarga de enviar el fichero al servidor y de tratar la respuesta del servidor.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

A continuación se compara si el código del mensaje es el 201, si no lo es retorna "NoK". A continuación obtiene el mensaje en formato json que contiene los elementos con nombre clave "espacioUtilizado" y "id". El elemento espacioUtilizado es el espacio utilizado por el usuario y el elemento id es el id del fichero que se ha subido.

Actualiza el espacio utilizado e inserta en el directorio con el id igual al id del mensaje, el id del fichero y en caso de ser un fichero borrado y no una versión anterior el nombre del fichero.

6.2.8.10 RenovarSesion

Es el método que se encarga de solicitar al servidor la renovación del token de sesión.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

A continuación compara si el código del mensaje es el 201, si no lo es retorna "Nok". Para finalizar comprueba si el mensaje recibido por el usuario contiene la cadena "Renovado", si es así retorna "Ok", en caso contrario retorna "Nok".

6.2.8.11 CrearDirectorio

Es el método que se encarga de solicitar la operación de crear directorio al servidor proporcionándole los datos necesarios y de interpretar el mensaje de respuesta del servidor.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

A continuación compara si el código del mensaje es el 201, si no lo es retorna "error". A continuación obtiene el mensaje en formato json y obtiene el elemento con nombre clave "directorio" que contiene el id del directorio creado por el usuario.

Después se crea un `JSONArray` con los metadatos del fichero, aunque sólo tenga el id y los demás elementos estén vacíos.

Para finalizar introduce nuevo directorio en el árbol de directorios y retorna "OK".

6.2.8.12 GetCargaDirectorio

El método `getCargaDirectorio` se encarga de solicitar al servidor el nivel inferior de un directorio proporcionado e interpretar el mensaje de retorno.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

A continuación compara si el código del mensaje es el 201, si no lo es retorna "error". A continuación obtiene el mensaje en formato json que es un array de elementos. En la posición 0 van los metadatos del directorio que se proporciona al servidor, pero las siguientes posiciones están ocupadas en función a la cantidad de directorios o nodos hijos que tenga ya que irán los metadatos de los directorios hijos. Como en la función getCargaInicial se introducen los nuevos nodos en el árbol, siendo su padre el directorio que se proporciona al servidor.

```
if(httpResp.getStatusLine().getStatusCode() != HttpStatus.SC_CREATED){
    return "error";
}
String respStr=EntityUtils.toString(httpResp.getEntity());
JSONArray respJSON=new JSONArray(respStr);
VoCSArbol arbol=app.getArbol();
VoCSNodo padre=null;
System.out.println(respStr);
for(int i=1;i<respJSON.length();i++){
    padre=arbol.InsertarNodo(Integer.parseInt
            (respJSON.getJSONArray(i).getString(0)),respJSON.getJSONArray(i), nodo);

}

app.setArbol(arbol);
return "Ok";
```

Figura 6.27: Tratamiento del mensaje cargaDirectorio

6.2.8.13 BorrarDirectorio

El método `borrarDirectorio` se encarga de solicitar al servidor el borrado de un directorio que proporciona el usuario.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

A continuación compara si el código del mensaje es el 201, si no lo es retorna "error". Para finalizar comprueba si el mensaje recibido por el usuario contiene la cadena "Eliminado", si es así retorna "Ok", en caso contrario retorna "error".

6.2.8.14 CompartirDirectorio

El método `compartirDirectorio` se encarga de solicitar al servidor que envíe una invitación para compartir un directorio, a un usuario con un email determinado.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

A continuación compara si el código del mensaje es el 201, si no lo es retorna "error". Para finalizar comprueba si el mensaje recibido por el usuario contiene la cadena "Compartido", si es así retorna "Ok", en caso contrario retorna "error".

6.2.8.15 AceptarInvitacion

El método `aceptarInvitación` se encarga de solicitar al servidor el directorio que ha recibido en la invitación e interpretar el mensaje de respuesta del servidor.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

A continuación compara si el código del mensaje es el 201, si no lo es retorna "error". A continuación obtiene el mensaje de respuesta en formato json, que contiene el nombre del directorio y los metadatos del directorio. Después inserta en el nodo raíz el nuevo directorio e inserta en el árbol el nuevo directorio.

6.2.8.16 CancelarInvitacion

El método `cancelarInvitación` se encarga solicitar al servidor que rechace una invitación que ha sido enviada al usuario.

Para ello obtiene la conexión de la variable global y después crea el mensaje para el servidor (mirar Mensajes que se Envían al Servidor) como se hace en la figura 6.29.

Compara si el código del mensaje es el 201, si no lo es retorna "error". Y si lo es "Ok".

6.2.9 Clase VoCSArbol

La clase Árbol es la que se encarga de realizar el árbol que se explica en el apartado 4.2.3.

Esta clase consta de cuatro métodos: `insertarNodo`, `InsertarNodoNuevo`, `búsqueda y reemplazarNodo.`

El método `insertarNodo` se encarga de insertar un nodo, siendo un nodo la información relevante del directorio, en el árbol de directorios. Si no hay raíz, entonces inserta en la raíz el nodo nuevo nodo que se crea. Si hay raíz, entonces llama a la función `InsertaDatos` del nodo padre. Retorna el nodo padre.

`InsertarNodoNuevo` es igual que la función `InsertarNodo`, solo que cuando se llama a esta función se esta añadiendo un nuevo nodo y hay que añadir al nodo padre el id y el nombre del directorio.

El método búsqueda realiza una búsqueda por el id del nodo y a partir de un nodo proporcionado. La búsqueda se realiza de una manera recursiva y descendiente del árbol hasta que se encuentre el nodo. Si se encuentra se retorna el nodo solicitado y si no se encuentra se retorna null.

El método `reemplazarNodo` se encarga de remplazar un nodo existente por otro. Este método es generalmente usado cuando los datos de un nodo han sido modificados.

Si el nodo que se manda es nulo se retorna el nodo.

Si la variable "datos" coincide con el id del nodo se llama a la función `recargarNodo` pasándole el nodo por el que se quiere remplazar.

Si la variable dato no coincide con el id del nodo, entonces se comprueba si el nodo tiene hijos y si los tiene, se busca en los hijos por el valor de la variable "datos". Si la búsqueda es positiva y se encuentra un nodo con id igual al valor de la variable "datos", se llama a la función `recargarNodo` pasándole el nodo por el que se quiere remplazar.

6.2.10 Clase VoCSNodo

La clase VoCSNodo es la que se encarga de crear un nodo y ofrecer los métodos necesarios para su buena inicialización y mantenimiento. Consta de tres métodos: `InsertaDatos, cargaInicial y recargarNodo.`

El método `InsertaDatos` se encarga de insertar un nuevo nodo hijo en la lista de nodos hijos del nodo padre.

El método `cargaInicial` es llamado por el constructor y es el que se encarga de dividir el elemento json que contiene todos los datos útiles del directorio, enviados en el orden correcto. Estas divisiones las almacena en distintos objetos que serán recorridos y almacenados en listas, como la de nombre de los ficheros, la de id de los ficheros, la de nombre de los directorios y la de id de los directorios.

El método `recargarNodo` se encarga de introducir los datos del nodo que se le pasa por parámetro en los datos del nodo.

6.3 Desarrollo del Servidor

En éste apartado se definen los métodos más importantes del servidor. Los ficheros utilizados en el servidor son `RestController.php` y `Servicios.php` que proporcionan los servicios de la API.

6.3.1 RestController.php

La clase RestController contiene 6 funciones, que son las siguientes la función `init, la función indexAction, la función getAction, la función postAction, la función putAction y la función deleteAction.`

La función más importante de todas es la función `postAction` que es en la función en la que recae toda la funcionalidad, ya que como se explicó en análisis todos los mensajes son POST.

La función `postAction` se compone de un switch que compara el parámetro recibido en el mensaje con nombre clave "método" con los siguientes casos:

- Los cuatro primeros `getParamsServidorService`, `recibirReplicaService`, `recibirAlmacenarReplicaService` y `peticionFicheroService` son métodos dedicados a la replicación por lo que no se van a explicar en este TFG.[31].

- El caso `subir` llama a la función "subir fichero" de la clase `My_Utils_Servicios` y en caso de recibir un resultado nulo envía una contestación con código 404 y con el mensaje "NOk". Si no es nulo, se envía el mensaje de respuesta con el resultado de la función y el código HTML 201.

- El caso `logueo`, llama a la función "login" de la clase `My_Utils_Servicios` y en caso de recibir un resultado nulo envía un mensaje con código 503 y con el mensaje "No Autenticado". Si no es nulo, se envía el mensaje de respuesta con el resultado de la función y el código HTML 201.

- El caso `registro`, llama a la función "registro" de la clase `My_Utils_Servicios` y en caso de recibir un resultado nulo envía una contestación con código 404 y con el mensaje "NOk". Si no es nulo, se envía el mensaje de respuesta con el resultado de la función y el código HTML 201.

- El caso `cargaInicio`, llama a la función "cargaInicio" de la clase `My_Utils_Servicios` y en caso de recibir un resultado nulo envía una contestación con código 404 y con el mensaje "NOk". Si no es nulo, se envía el mensaje de respuesta con el resultado de la función y el código HTML 201.

- El caso `download`, llama a la función "download" de la clase `My_Utils_Servicios` y en caso de recibir un resultado falso envía una contestación con código 404 y con el mensaje "NOk". Si no es falso, se envía el mensaje de respuesta con el código HTML 201.

- El caso `delete`, llama a la función "delete" de la clase `My_Utils_Servicios` y en caso de recibir un resultado falso envía una contestación con código 404 y con el mensaje "NoEliminado". Si no es falso, se envía el mensaje de respuesta con el código HTML 201 y el mensaje "Eliminado".

- El caso `ficherosBorrados`, llama a la función "ficherosBorrados" de la clase `My_Utils_Servicios` y en caso de recibir un resultado nulo envía una contestación con el código 404 y el mensaje "NOk". En caso de no ser nulo envía un mensaje de respuesta con el resultado y el código 201.

- El caso `versionesAnteriores`, llama a la función "versionesAnteriores" de la clase `My_Utils_Servicios` y en caso de recibir un resultado nulo envía una contestación con el código 404 y el mensaje "No Restaura". En caso de no ser nulo envía un mensaje de respuesta con el resultado y el código 201.

- El caso `restaurar`, llama a la función "restaurar" de la clase `My_Utils_Servicios` y en caso de recibir un resultado nulo envía una contestación con el código 404 y el mensaje "NoRestaurado". En caso de no ser nulo envía un mensaje de respuesta con el resultado y el código 201.

- El caso `renovar`, llama a la función "renovar" de la clase `My_Utils_Servicios` y en caso de recibir un resultado nulo envía una contestación con el código 404 y el

mensaje “NoRenovado”. En caso de no ser nulo envía un mensaje de respuesta con el mensaje “Renovado” y el código 201.

- El caso `crearDirectorio`, llama a la función “crearDirectorio” de la clase `My_Utils_Servicios` y en caso de recibir un resultado nulo envía una contestación con el código 404 y el mensaje “NOk”. En caso de no ser nulo envía un mensaje de respuesta con el resultado y el código 201.

- El caso `cargarDirectorio`, llama a la función “cargarDirectorio” de la clase `My_Utils_Servicios` y en caso de recibir un resultado nulo envía una contestación con el código 404 y el mensaje “NOk”. En caso de no ser nulo envía un mensaje de respuesta con el resultado y el código 201.

- El caso `eliminarDirectorio`, llama a la función “eliminarDirectorio” de la clase `My_Utils_Servicios` y en caso de recibir un resultado falso envía una contestación con el código 404 y el mensaje “No Eliminado”. En caso de no ser nulo envía un mensaje de respuesta con el mensaje “Eliminado” y el código 201.

- El caso `compartirDirectorio`, llama a la función “compartirDirectorio” y en caso de recibir un resultado falso envía una contestación con el código 404 y el mensaje “No Compartido”. En caso de no ser falso envía un mensaje de respuesta con el mensaje “Compartido” y el código 201.

- El caso `aceptarInvitacion`, llama a la función “aceptarInvitacion” y en caso de recibir un resultado falso envía una contestación con el código 404 y el mensaje “NoAceptada”. En caso de no ser falso envía un mensaje de respuesta con el resultado y el código 201.

- El caso `cancelarInvitacion`, llama a la función “cancelarInvitacion” y en caso de recibir un resultado falso envía una contestación con el código 404 y el mensaje “No Cancelada”. En caso de no ser falso envía un mensaje de respuesta con el mensaje “Cancelada” y el código 201.

- En caso por defecto responde con un mensaje que pone “Problema” y con el código “200”.

6.3.2 Servicios.php

El fichero `Servicios.php` esta compuesto por la clase `My_Utils_Servicios`, que implementa los servicios de la API REST. A continuación se expondrán los métodos del servicio.

6.3.2.1 Subir Fichero

El método `Subir Fichero` se encarga de recibir un fichero y almacenarlo en el servidor.

La función subir se encarga de realizar las siguientes acciones:

1. Si el token de sesión del usuario no existe, se sale de la función y envía un mensaje indicando que no se ha podido realizar la operación.
2. Si la carpeta del usuario no existe, la crea.

3. Obtiene el fichero, en caso de ser válido prosigue y lo guarda en la carpeta del usuario.
4. Comprueba que no supera el espacio que tiene asignado el usuario. Si lo supera manda un mensaje indicando que no se ha podido realizar la operación.
5. Se proporciona al fichero un nombre aleatorio que no genere colisiones con los que ya hay introducidos en la carpeta de usuario.
6. Se introduce el fichero en la base de datos.
7. Se busca si hay ficheros que estén en el mismo directorio del usuario con el mismo nombre y extensión; y que estén activos. Si hay ficheros, hay que poner el fichero activo a borrado, insertar relación de versión anterior y si el fichero que se esta poniendo a borrado tiene versiones anteriores, reasignarlas al nuevo fichero.
8. Se inserta en la base de datos, en la tabla que relaciona al fichero con el directorio.
9. Se inserta en la base de datos, en la tabla que relaciona al fichero con el usuario.
10. Se inserta en la base de datos, en la tabla que relaciona al fichero con el servidor.
11. Comprobar si el directorio en el que se esta subiendo el fichero esta compartido. Si esta compartido, compartirlo con los demás usuarios, insertando el fichero en las tablas de relación correspondientes (usuario_fichero,…).
12. Se realiza el tratamiento de la seguridad del fichero. Si el fichero sube con seguridad 2, se tiene que realizar una replica del fichero y se tiene que cifrar el fichero. Si el fichero sube con seguridad 1 se tiene que realizar 1 replica del fichero. Si el fichero sube con seguridad 0 no se realiza ninguna replica.
13. Se actualiza el espacio del usuario y se manda como respuesta.

6.3.2.2 Logueo

La función `logueo` es la que se encarga de autenticar al usuario, pero además de autenticarle se encarga de generar el token de sesión y de mandar las invitaciones pendientes de aceptar o rechazar que tiene el usuario.

La función sigue los siguientes pasos:

1. La función realiza el hash de comparación y las iteraciones necesarias, con la contraseña proporcionada, con la función sha256 con la contraseña y el salt que esta guardado en la base de datos cuando el usuario se registra.
2. Se realiza el token de sesión con la concatenación del Nick de usuario, una cadena aleatoria de tamaño 10 y la fecha en la que realizó la autenticación, recogiendo segundos, minutos, horas, día, mes y año. El sha256 recibe también la comparación del primer paso como semilla de la operación.
3. Se compara la contraseña y el usuario con el almacenado en la base de datos.
4. Si existe alguno, se prosigue y se buscan las invitaciones que tenga el usuario en la base de datos. Se manda al usuario las invitaciones y el token de sesión. Si no existe ninguno se manda un mensaje indicando que no se ha podido loguear.
5. El token se almacena en la memoria caché.

6.3.2.3 Registro

El método `registro` se encarga de realizar el registro de un nuevo usuario. Sigue los siguientes pasos:

1. Recibe los datos del usuario.
2. Realiza una cadena aleatoria.
3. Se realizan cinco funciones sha256 con la contraseña y la cadena aleatoria como salt, para no almacenarla en claro.

4. Se comprueba si no hay un usuario con el mismo nick en la base de datos. Si no hay se inserta en la base de datos. Si no hay se envía mensaje de que no se ha podido registrar.
5. Se crea la clave de cifrado y se almacena junto con el salt en la base de datos.
6. Se envía al usuario un mensaje de registro completado si ha ido bien. Si no ha ido bien se manda un mensaje de que no se ha podido registrar.

6.3.2.4 CargaInicio

El método `cargaInicio` es el método que se encarga de proporcionar la carga inicial de los directorios para que el usuario pueda crear su árbol de directorios. Este método necesita dos métodos que se llaman `sacarRaiz y sacar2Nivel`.

La función sigue los siguientes pasos:

1. Comprueba que el token de sesión se encuentra en la caché. De no estar se envía mensaje de que no se ha podido enviar la carga de inicio.
2. Obtiene los metadatos del directorio raíz y de su nivel inferior llamando a la función sacarRaiz.
3. La función sacarRaíz obtiene los metadatos del directorio raíz y llama a la función sacar2Nivel.
4. Sacar2Nivel obtiene los metadatos del segundo nivel y los devuelve.
5. La función sacarRaíz devuelve todos los metadatos a la función principal.
6. Obtiene el espacio que ha utilizado y el espacio total.
7. Envía al usuario el espacio utilizado, el espacio total y los metadatos.

6.3.2.5 Download

El método `download` se encarga de proporcionar el fichero solicitado al usuario, se compone de los siguientes pasos.

1. Comprueba que el token de sesión se encuentra en caché. De no estar se envía mensaje de que no se pudo realizar la operación de descarga.
2. Comprueba que el fichero esta alojado en el servidor, mirando en la tabla de la base de datos que relaciona el servidor con los ficheros alojados.
3. Si no esta, realiza la operación de descarga desde un servidor que tenga alojado el fichero.
4. Si esta alojado, comprueba la seguridad de dicho fichero y si tiene nivel 2 lo descifra.
5. Envía el fichero al usuario.

6.3.2.6 Delete

El método `delete` se encarga de borrar el fichero poniéndolo a inactivo.

Este método sigue los siguientes pasos:

1. Comprueba que el token de sesión se encuentra en caché. De no estar se envía mensaje de que no se pudo realizar la operación de borrado.
2. Pone el fichero como inactivo.
3. Envía el resultado de la operación.

6.3.2.7 FicherosBorrados

La función `ficherosBorrados` es la que se encarga de buscar todos los ficheros inactivos del usuario y proporcionárselos en el mensaje de respuesta.

Esta función se compone de los siguientes pasos:

1. Comprueba que el token de sesión se encuentra en caché. De no estar, se envía mensaje de que no se pudo realizar la operación.
2. Busca en la base de datos los ficheros que tiene inactivos.
3. Manda los metadatos de los ficheros al usuario.

6.3.2.8 VersionesAnteriores

El método `versionesAnteriores` se encarga de proporcionar las versiones anteriores de un fichero proporcionado por el usuario. Estas versiones anteriores se envían al usuario en el mensaje de respuesta.

Se compone de los siguientes pasos:

1. Comprueba que el token de sesión se encuentra en caché. De no estar, se envía mensaje de que no se pudo enviar la operación.
2. Busca en la base de datos los ficheros que tengan el mismo nombre y se encuentren en el mismo directorio.
3. Manda al usuario los metadatos de los ficheros almacenados en la base de datos.

6.3.2.9 Restaurar

El método `restaurar` se encarga de poner activo el fichero y en caso de que el fichero fuera una versión anterior de un fichero, entonces inactiva el fichero que estaba activo. Los ficheros que el fichero activo tenía como versiones anteriores se relacionan como versiones anteriores del fichero a restaurar.

Se compone de los siguientes pasos:

1. Comprueba que el token de sesión se encuentra en caché. De no estar, se envía mensaje de que no se pudo realizar la operación.
2. Busca su directorio, si el directorio no existe porque se ha eliminado, se asocia al directorio raíz.
3. Pone activo el fichero.
4. Si el fichero estaba asociado a otro fichero como una versión anterior. Pone dicho fichero a borrado y se asocia al fichero que queremos restaurar. A su vez todos los ficheros que tenía asociado el fichero que estaba activo, se asocian al fichero que se quiere restaurar.
5. Se envía al usuario el directorio en el que ha sido restaurado el fichero.

6.3.2.10 Renovar

El método `renovar` se encarga de renovar el tiempo del token de sesión de la aplicación, en caso de que el token de sesión este en la memoria caché.

Se compone de los siguientes pasos:

1. Comprueba que el token de sesión se encuentra en caché. De no estar, se envía mensaje de que no se pudo realizar la operación.
2. Remplaza el token por el mismo pero reinicia la cuenta regresiva de tiempo que puede estar almacenado el objeto en caché.
3. Envía al usuario el resultado de la operación.

6.3.2.11 CrearDirectorio

El método `crearDirectorio` se encarga de crear el nuevo directorio según los datos proporcionados por el usuario.

Sigue los siguientes pasos:

1. Comprueba que el token de sesión se encuentra en caché. De no estar, se envía mensaje de que no se pudo realizar la operación.
2. Introduce un nuevo directorio en la base de datos.
3. Manda el id del directorio al usuario.

6.3.2.12 CargarDirectorio

El método `cargarDirectorio` se encarga de proporcionar los metadatos del directorio proporcionado y del nivel inferior.

Este método sigue los siguientes pasos:

1. Comprueba que el token de sesión se encuentra en la caché. De no estar se envía mensaje de que no se ha podido enviar la carga de inicio.
2. Llama a la función `sacar2Nivel` con los metadatos del fichero que se quiere cargar.
3. `Sacar2Nivel` obtiene los metadatos del segundo nivel y los devuelve.
4. Envía al usuario los metadatos.

6.3.2.13 EliminarDirectorio

El método `eliminarFichero` se encarga de eliminar el directorio proporcionado por el usuario.

Este método sigue los siguientes pasos:

1. Comprueba que el token de sesión se encuentra en caché. De no estar, se envía mensaje de que no se pudo realizar la operación.
2. Borra el directorio de la base de datos.
3. Borra todos los directorios hijos y todos los niveles que tenga por debajo.
4. Pone como inactivo todos los ficheros que tiene asociado y todos los ficheros de los directorios hijos y de los niveles inferiores.
5. Envía al usuario el resultado de la operación.

6.3.2.14 CompartirDirectorio

El método `compartirDirectorio` se encarga de mandar la invitación para compartir el directorio que desea compartir.

Este método sigue los siguientes pasos:

1. Comprueba que el token de sesión se encuentra en caché. De no estar, se envía mensaje de que no se pudo realizar la operación.
2. Inserta la una nueva invitación en la base de datos con el nick del usuario y el nick del usuario que invita.
3. Envía un email al usuario al que se ha invitado al directorio.
4. Envía al usuario el resultado de la operación.

6.3.2.15 AceptarInvitacion

La función `aceptarInvitacion` se encarga de enviar al usuario el directorio el nombre y los metadatos del directorio que ha aceptado compartir.

Éste método sigue los siguientes pasos:

1. Comprueba que el token de sesión se encuentra en caché. De no estar, se envía mensaje de que no se pudo realizar la operación.
2. Inserta en la base de datos en la tabla directorios, en el directorio raíz del usuario el directorio que se va a compartir.
3. Elimina la invitación de la base de datos.
4. Llama a la función `getNvDirectorios`.
5. La función `getNvDirectorios` obtiene los metadatos del directorio y los devuelve.
6. Manda al usuario los metadatos del directorio que se acaba de aceptar.

6.3.2.16 CancelarInvitacion

El método `cancelarInvitación` se encarga de eliminar la invitación de la base de datos.

Éste método sigue los siguientes pasos:

1. Comprueba que el token de sesión se encuentra en caché. De no estar, se envía mensaje de que no se pudo realizar la operación.
2. Elimina la invitación de la base de datos.
3. Envía al usuario el resultado de la función.

7 Análisis de Rendimiento

En este capitulo se mostrará la evaluación del sistema diseñado. Para la evaluación se realizarán diferentes pruebas y se analizaran los resultados.

Para la realización de las pruebas se han utilizado dos PC, un móvil, una red Wi-Fi y una red 3G. Sus características se pueden apreciar en las tablas 7.1, 7.2, 7.3, 7.4 y 7.5.

Procesador	**Pentium 4 2.4 GHz**
RAM	1 GB
Disco Duro	78,7 GB
Caché	512KB

Tabla 7.1: Especificaciones del PC1.

Procesador	**AMD Athlon 64 processor 3200+ 800MHz**
RAM	1 GB
Disco Duro	265,6 GB
Caché	1GB

Tabla 7.2: Especificaciones del PC2.

Procesador	**1GHz**
RAM	512 KB
Batería	1400 mAh
Plataforma	Android Eclair 2.1 con HTC Sense
Tarjeta SD	2GB

Tabla 7.3: Especificaciones de móvil Android.

Conexión	**Wi-FI**
Velocidad de subida	10,01 Mbps
Velocidad de bajada	8,97 Mbps

Tabla 7.4: Especificaciones de la red Wi-Fi.

Conexión	**3G**
Velocidad de subida	934 Kbps
Velocidad de bajada	3,01 Mbps

Tabla 7.5: Especificaciones de la red 3G.

Conexión	**Ethernet 100-10 Mbps**

Tabla 7.6: Especificaciones de la red interna.

Antes de exponer las pruebas se debe explicar que la aplicación consta de dos modelos de replicación llamados Anillo y VoCS; y un sistema de cifrado que se puede encontrar su explicación en el trabajo de fin de grado de mi compañero. [32]

Las operaciones que se realizarán son la descarga y la subida de ficheros utilizando redes Wi-Fi y 3G.

7.1 Descargar Fichero

Primero se expondrán las gráficas de la descarga de fichero con la red Wi-Fi, después con la red 3G y finalmente se compararán los resultados de ambas redes.

Para todos los casos se realizarán tres tipos de pruebas, la primera se realiza cuando el servidor tiene alojado el fichero, la segunda se realizará cuando el fichero no esta alojado en el servidor y la replicación Anillo esta activada y la última prueba se realizará cuando el fichero no esta alojado en el servidor y la replicación VoCS está activa.

7.1.1 Descargar Fichero Wi-Fi

Para evaluar el rendimiento de la operación descargar, se han realizado pruebas con ficheros de un tamaño pequeño, mediano y grandes con respecto a las capacidades de un teléfono móvil. Los ficheros que se han probado son de tamaño 512 KB, 15MB y 100MB.

Tamaño de Fichero	Prueba	Tiempo(s)
512KB	Alojado en servidor	5,64
15MB	Alojado en servidor	29,861
100MB	Alojado en servidor	124,803
512KB	No alojado en servidor Anillo	5,723
15MB	No alojado en servidor Anillo	32,329
100MB	No alojado en servidor Anillo	142,889
512KB	No alojado en servidor VoCS	6,029
15MB	No alojado en servidor VoCS	37,705
100MB	No alojado en servidor VoCS	150,808

Tabla 7.7: Pruebas de descarga Wi-Fi.

La figura 7.1 muestra la gráfica que compara las tres pruebas. En el eje de coordenadas X se muestra el tamaño de los ficheros y en el eje de coordenadas Y se muestra el tiempo en segundos.

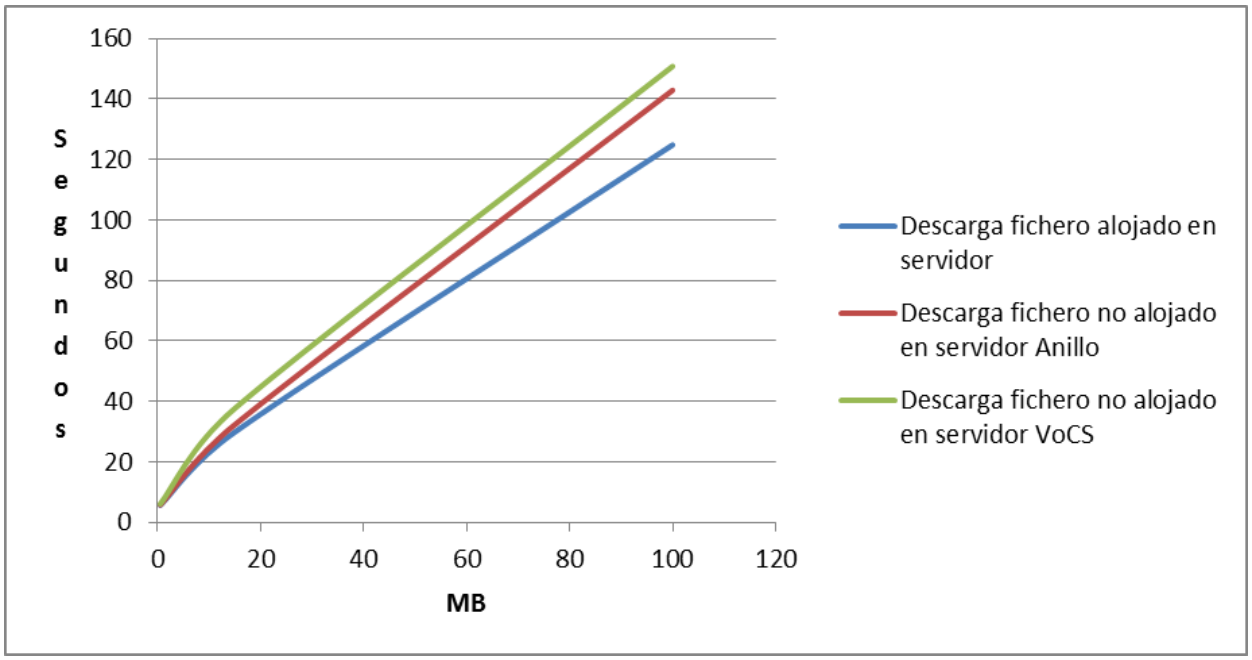


Figura 7.1: Figura que muestra el resultado de descargar fichero con Wi-Fi.

Teóricamente el resultado de descargar un fichero de 100 MB con una conexión de 10,01 Mbps y una oscilación del 20%, se obtiene que:

(100MB/10,1Mbps) x 1,2= 95,05 segundos

Como se muestra en la figura 7.1 para ficheros de 100 MB, si el fichero esta alojado en el servidor tarda una media de 120 segundos, menos el calculo teórico de la transmisión hace un total de 25 segundos dedicado a solicitar el fichero, que el servidor lo proporcione y que la aplicación los escriba en la memoria. Es un tiempo bastante bueno.
Si no esta alojado en el servidor y esta activada la replicación anillo, tarda 142 segundos debido a que el fichero no esta alojado en el servidor y se lo tiene que solicitar al servidor que lo tiene por lo que tarda más. Esta replicación envía el fichero directamente sin replicarlo en el servidor en el que se ha solicitado. Si el tiempo que ha tardado le restamos el cálculo teórico de la descarga del fichero, obtenemos un resultado de 47 segundos que es lo que tarda en solicitar el fichero, proporcionarlo y escribirlo en memoria. Es un tiempo bastante bueno.

Si no esta alojado en el servidor y la replicación VoCS es la prueba que más tarda debido a que solicita el fichero a un servidor que tiene el fichero y además de mandarlo al usuario, lo replica en el sistema. Tarda un total de 150 segundos, menos el cálculo teórico de la descarga del fichero hace un total de 55 segundos. Como se puede observar la diferencia esta en la escritura del fichero en el disco duro.

7.1.2 Descargar Fichero 3G

Para evaluar el rendimiento de la operación descargar, se han realizado pruebas con ficheros de un tamaño pequeño, mediano y grandes con respecto a las capacidades de un teléfono móvil. Los ficheros que se han probado son de tamaño 512 KB, 15MB y 100MB.

Tamño de Fichero	Prueba	Tiempo(s)
512KB	Alojado en servidor	6,501
15MB	Alojado en servidor	55,324

100MB	Alojado en servidor	320,314
512KB	No alojado en servidor Anillo	6,602
15MB	No alojado en servidor Anillo	57,89
100MB	No alojado en servidor Anillo	334,006
512KB	No alojado en servidor VoCS	6,90
15MB	No alojado en servidor VoCS	58,391
100MB	No alojado en servidor VoCS	347,875

Tabla 7.8: Prueba de descarga 3G.

La figura 7.2 muestra la gráfica que compara las tres pruebas. En el eje de coordenadas X se muestra el tamaño de los ficheros y en el eje de coordenadas Y se muestra el tiempo en segundos.

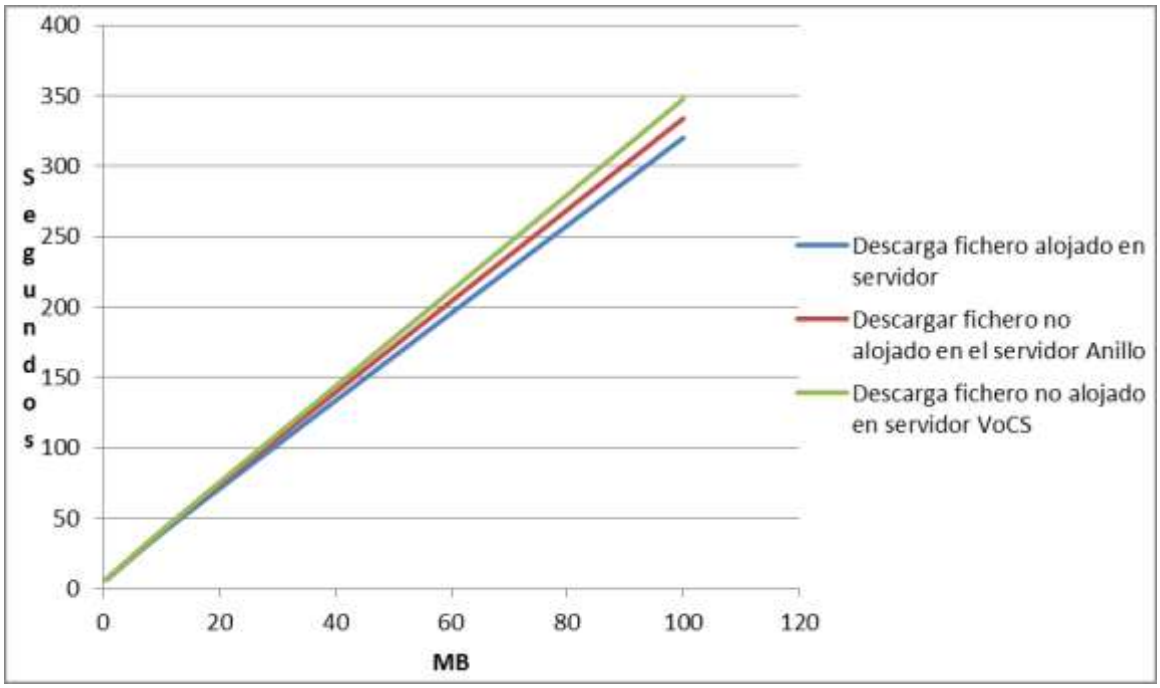


Figura 7.2: Figura que muestra el resultado de descargar fichero con 3G.

Teóricamente el resultado de descargar un fichero de 100 MB con una conexión de 3,01 Mbps y una oscilación del 20%, se obtiene que:

(100MB/3,01Mbps) x 1,1= 292,2 segundos

Como se muestra en la figura 7.2 para ficheros de 100 MB si el fichero esta alojado en el servidor tarda una media de 320 segundos, menos el calculo teórico de la transmisión hace un total de 28 segundos dedicado a solicitar el fichero, que el servidor lo proporcione y que la aplicación los escriba en la memoria. Es un tiempo bastante bueno y muy similar al producido con la red Wi-FI.

Si no esta alojado en el servidor y esta activada la replicación anillo, tarda 334 segundos debido a que el fichero no esta alojado en el servidor y se lo tiene que solicitar al servidor que lo tiene por lo que tarda más. Si el tiempo que ha tardado le restamos el cálculo teórico de la descarga del fichero, obtenemos un resultado de 42 segundos que es lo que tarda en solicitar el fichero, proporcionarlo y escribirlo en memoria. Es un tiempo bastante bueno y similar al producido con la red Wi-Fi.

Si no esta alojado en el servidor y la replicación VoCS es la prueba que más tarda debido a que solicita el fichero a un servidor que tiene el fichero y además de mandarlo al usuario, lo replica en el sistema. Tarda un total de 347 segundos, menos el cálculo teórico de la descarga del fichero hace un total de 55 segundos, este tiempo es bueno y similar al producido con la red Wi-fi. Como se puede observar la diferencia entre replicaciones esta en la escritura del fichero en el disco duro.

7.1.3 Comparativa

Como se puede apreciar por ambas gráficas, los resultados obtenidos por las redes Wi-Fi y 3G son buenos, pero el uso de Wi-Fi es recomendable antes que el uso de 3G, ya que es bastante más rápido debido a su velocidad de descarga. Los resultados para la aplicación son bastante buenos, ya que para ambas pruebas los tiempos que no dependen de la conexión son bastante similares.

7.2 Subir Fichero

Primero se expondrán las gráficas de la subida de un fichero con la red Wi-Fi, después con la red 3G y finalmente se compararán los resultados de ambas redes.

Para todos los casos se realizarán cinco tipos de pruebas, la primera se realiza cuando él se selecciona un fichero con seguridad baja, la segunda se realizará cuando el fichero se suba con seguridad media y la replicación Anillo este activada, la tercera se realizará cuando se seleccione seguridad alta y la replicación Anillo esta activada, se realizará cuando el fichero se suba con seguridad media y la replicación VoCS este activada, la quinta se realizará cuando se seleccione seguridad alta y la replicación VoCS esta activada.

7.2.1 Subir Fichero Wi-Fi

Para evaluar el rendimiento de la operación subir fichero, se han realizado pruebas con ficheros de un tamaño pequeño, mediano y grande; con respecto a las capacidades de un teléfono móvil. Los ficheros que se han probado son de tamaño: 512 KB, 15MB y 100MB.

Tamaño de Fichero	Prueba	Tiempo(s)
512KB	Seguridad Baja	3,053
15MB	Seguridad Baja	22,251
100MB	Seguridad Baja	107,909
512KB	Seguridad Media Anillo	3,23
15MB	Seguridad Media Anillo	23,675
100MB	Seguridad Media Anillo	119,832
512KB	Seguridad Media VoCS	4,839
15MB	Seguridad Media VoCS	25,665
100MB	Seguridad Media VoCS	123,767
512KB	Seguridad Alta Anillo	16,169
15MB	Seguridad Alta Anillo	498,155
100MB	Seguridad Alta Anillo	3431,343
512KB	Seguridad Alta VoCS	18,096
15MB	Seguridad Alta VoCS	512,481
100MB	Seguridad Alta VoCS	3487,329

Tabla 7.6: Pruebas de subida de un fichero Wi-Fi.

La figura 7.3 muestra la gráfica que compara las pruebas. En el eje de coordenadas X se muestra el tamaño de los ficheros y en el eje de coordenadas Y se muestra el tiempo en segundos.

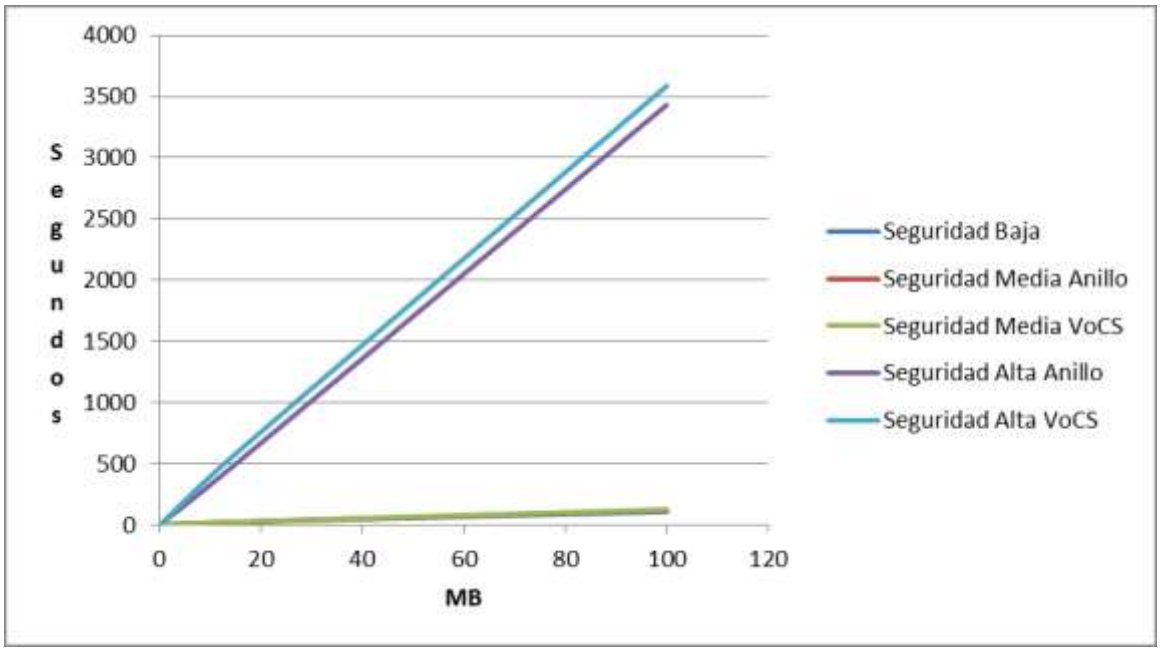


Figura 7.3: Muestra el resultado de subir fichero con Wi-Fi.

El cálculo teórico de la subida de un fichero de 100 MB es de:

(100MB/8,97Mbps) x 1,1=107,02 segundos

Como se puede apreciar en la figura, la seguridad Alta, que se encarga de cifrar y de replicar un fichero, para ficheros de más de 15 MB empieza a ser un proceso muy cargante para el servidor y para la aplicación, llegándose a prolongar demasiado tiempo. Por ejemplo para ficheros de 100 MB la seguridad alta supondría casi 1 hora de tiempo, esta operación es incompatible para un dispositivo móvil.

Para la seguridad baja tarda una media de 107,909, menos el tiempo teórico de subida tarda un total de 0,889 segundos, por lo que el rendimiento de la aplicación y del servidor es bastante bueno.

Para la seguridad media con replicación en Anillo, un fichero de 100MB tarda una media de 119,832 segundos, menos el tiempo teórico de subida tarda un total de 12,812 segundos, un tiempo bastante óptimo ya que en función a lo que tarda la transferencia no supone una gran cantidad de tiempo.

Para la seguridad media con replicación VoCS, un fichero de 100MB tarda una media de 123,767 segundos, menos el tiempo teórico de subida tarda un total de 16,747 segundos, un tiempo bastante óptimo ya que en función a lo que tarda la transferencia no supone una gran cantidad de tiempo.

El peor de los casos es la replicación VoCS, debido a que esta replicación antes de enviar una replica a un servidor, realiza un consulta a los servidores de su grupo para sabe su estado actual y el que obtenga el mejor estado es el elegido para la replicación. La replicación Anillo sin embargo

no es así, los servidores tienen un servidor que es al que van a realizar la replicación siempre que éste no este caído. Para más información mirar el PFC de mi compañero [32]

En este caso resulta muy similar el rendimiento de las replicaciones, pero esto es debido a que no se disponen de los recursos necesarios. Para poder estimar un tiempo bueno de la replicación VoCS se necesitaría completar un grupo de servidores a los que solicitar el estado y así poder ver el tiempo que se dedica a realizar esas consultas.

7.2.2 Subir Fichero 3G

Para evaluar el rendimiento de la operación subir fichero, se han realizado pruebas con ficheros de un tamaño pequeño, mediano y grande; con respecto a las capacidades de un teléfono móvil. Los ficheros que se han probado son de tamaño: 512 KB, 15MB y 100MB.

Tamaño de Fichero	Prueba	Tiempo(s)
512KB	Seguridad Baja	7,113
15MB	Seguridad Baja	80,045
100MB	Seguridad Baja	523,645
512KB	Seguridad Media Anillo	9,823
15MB	Seguridad Media Anillo	81,737
100MB	Seguridad Media Anillo	527,645
512KB	Seguridad Media VoCS	11,324
15MB	Seguridad Media VoCS	84,456
100MB	Seguridad Media VoCS	547,326
512KB	Seguridad Alta Anillo	20,865
15MB	Seguridad Alta Anillo	560,553
100MB	Seguridad Alta Anillo	3839,757
512KB	Seguridad Alta VoCS	21,321
15MB	Seguridad Alta VoCS	576,272
100MB	Seguridad Alta VoCS	3902,757

Tabla 7.7: Pruebas de subida de un fichero 3G.

La figura 7.4 muestra la gráfica que compara las pruebas. En el eje de coordenadas X se muestra el tamaño de los ficheros y en el eje de coordenadas Y se muestra el tiempo en segundos.

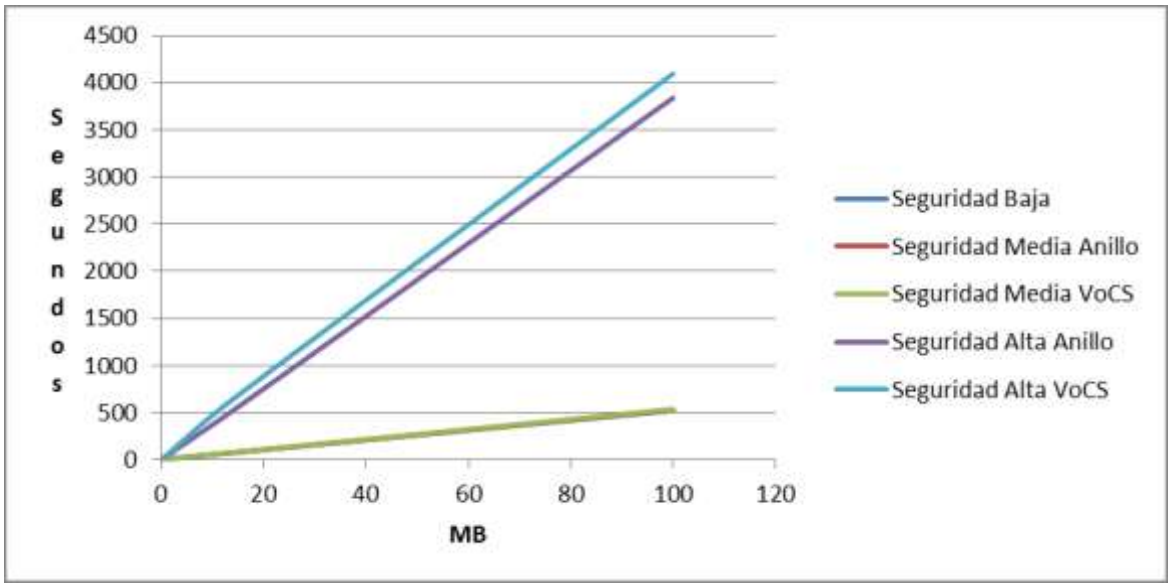


Figura 7.4: Muestra el resultado de subir fichero con 3G.

El cálculo teórico de la subida de un fichero de 100 MB es de:

(100MB/1,72Mbps) x 1,1=511,8 segundos

Como se puede apreciar en la figura 7.4, es bastante similar a lo que sucede con las pruebas realizadas con la red Wi-Fi y el caso de seguridad Alta, pero con la red 3G los ficheros de 15MB empiezan a resultar una carga pesada tardando algo menos de 10 minutos. Para la red 3G a partir de ficheros de 15 MB resulta ser un proceso bastante cargante para el servidor y la aplicación y su tiempo de ejecución no es muy asequible para una aplicación móvil.

Para la seguridad baja tarda una media de 523, menos el tiempo teórico de subida tarda un total de 11 segundos, por lo que el rendimiento de la aplicación y del servidor es bastante bueno en comparación a lo que tarda la transferencia.

Para la seguridad media con replicación en Anillo, un fichero de 100MB tarda una media de 527 segundos, menos el tiempo teórico de subida tarda un total de 16 segundos, un tiempo bastante óptimo en comparación a lo que tarda la transferencia.

Para la seguridad media con replicación VoCS, un fichero de 100MB tarda una media de 547 segundos, menos el tiempo teórico de subida tarda un total de 36 segundos, un tiempo bastante óptimo en comparación a lo que tarda la transferencia.

7.2.3 Comparativa

En este apartado se expondrán una serie de gráficas que comparan directamente las pruebas obtenidas para cada red en cada tipo de seguridad.

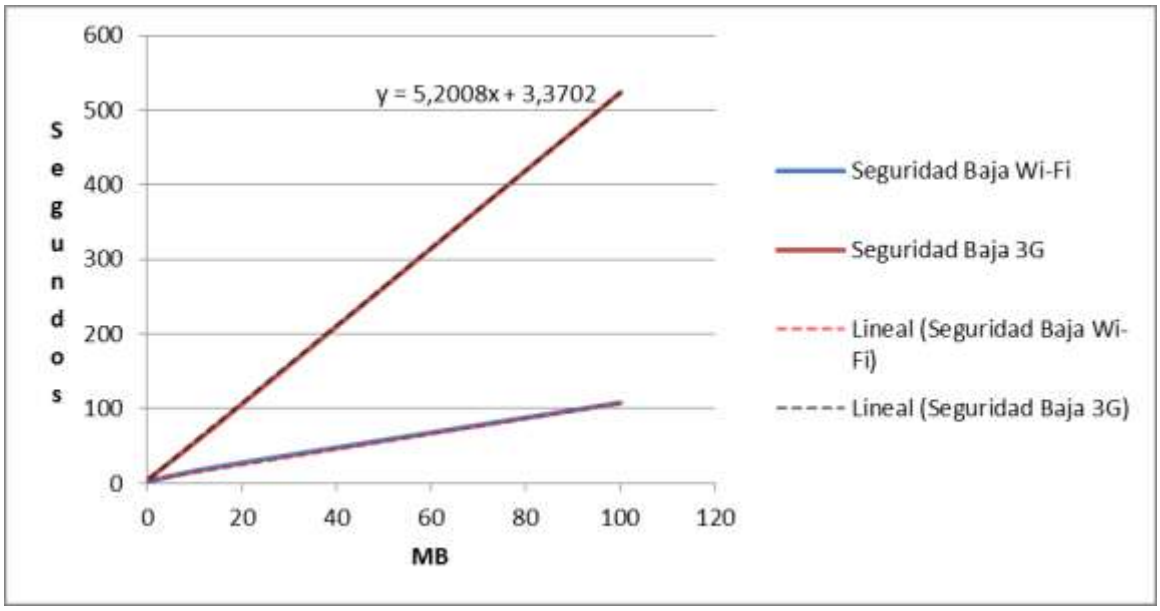


Figura 7.5: Comparativa Seguridad Baja.

Como se puede apreciar en la figura 7.5, hay una notable diferencia entre usar una red Wi-Fi y una red 3G. Se nota que con ficheros a partir de 20 MB la diferencia es sustancial, ya que con Wi-Fi apenas llega a 30 segundos pero con 3G ronda los 2 minutos. A medida que se escogen ficheros más grandes, la diferencia se hace cada vez mayor y esto se puede comprobar con el fichero de 100 MB que con 3G ronda los 9 minutos y con Wi-Fi apenas llega a los 2 minutos.

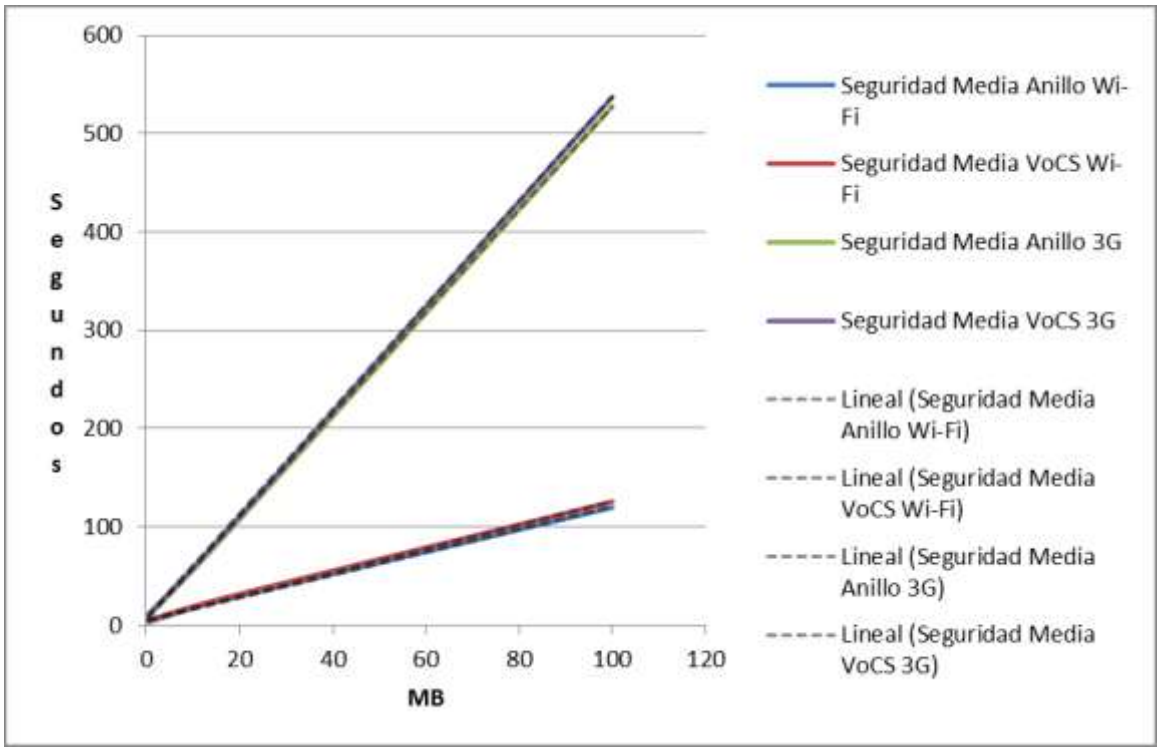


Figura 7.6: Comparativa Seguridad Media.

Con la seguridad Media pasa exactamente lo mismo que con la seguridad Baja. Hasta ficheros de 20 MB la diferencia de tiempo es pequeña, pero a medida que los ficheros crecen la diferencia de tiempos se hace cada vez mayor.

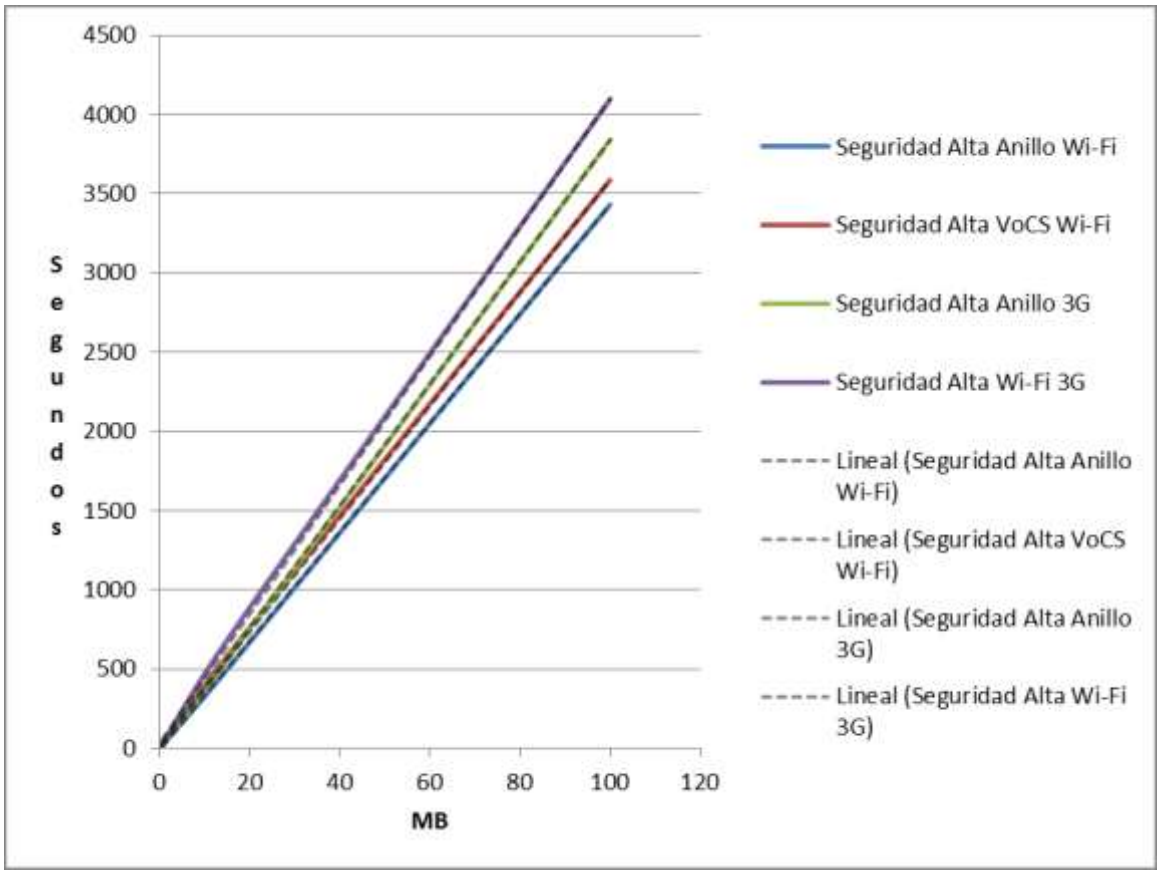


Figura 7.7: Comparación Seguridad Alta.

Con la seguridad Alta no hay tanta diferencia, ya que en este caso es el cifrado lo que esta haciendo que sea ineficiente. La diferencia es muy parecida a la obtenida con la seguridad Media, salvo que el fichero se tiene que cifrar.

8 Conclusiones

En este capitulo se expondrán las conclusiones finales del proyecto, una serie de trabajos futuros en base a este proyecto y un esbozo del esfuerzo realizado hasta terminar el proyecto.

8.1 Conclusiones Generales

Una vez finalizado el proyecto se puede indicar que se ha cumplido el objetivo marcado (1.4 Objetivo) que era el análisis, diseño e implementación de un sistema multiplataforma para el sistema de almacenamiento en la nube VoCS mediante el uso de tecnologías y estándares Web. A lo largo de este documento se ha mostrado el análisis, diseño e implementación cumpliendo con el objetivo principal.

Los subobjetivos que se han cumplido son los siguientes:

- Se ha montado y preparado el entrono de desarrollo para realizar la implementación del sistema.
- Se ha análizado un sistema multiplataforma para el sistema de almacenamiento en la nube VoCS.
- Se ha diseñado un sistema multiplataforma para el sistema de almacenamiento en la nube VoCS.
- Se ha implementado un prototipo Android que cumpla con el análisis y diseño de la aplicación.
- Se ha desarrollado una API REST en el servidor que cumpla con el análisis y diseño.
- Se ha realizado la integración de sistemas de seguridad en el sistema de comunicaciones establecido, las comunicaciones van a través de SSL y todos los mensajes tienen un token de sesión del usuario.
- Se ha implementado un sistema de ficheros escalable que optimice los recursos del dispositivo móvil.
- Se ha realizado una evaluación de las operaciones subida y descarga de ficheros desde el dispositivo móvil, utilizando las redes Wi-Fi y 3G.

Se pueden concluir, que se han cumplido todos los objetivos presentados en el presente Trabajo de Fin de Grado con éxito.

8.2 Trabajos Futuros

Dado que el presente proyecto es un prototipo para servir como base a futuros proyectos relacionados con el almacenamiento en la nube desde un sistema multiplataforma. Ofrece una gran cantidad de posibilidades futuras. A continuación se exponen algunos de ellos:

- Buscar el medio más óptimo para realizar las tareas más primordiales y las que más recursos necesitan. Activar el Wi-Fi si esta disponible con la descarga de un fichero de más de 100 MB y no realizarla con 3G ya que supondría el gasto de bastantes datos.

- Poder renombrar los directorios y ficheros que se han creado en el servidor cuando el usuario desee.

- Completa sincronización con el sistema de ficheros del sistema.

- Restaurar directorios, permitiendo restaurar directorios que se han eliminado con todos los ficheros asociados a este.

- Mover directorios y ficheros, permitiendo modificar su ubicación dentro del sistema de ficheros.

- Poder dar permisos a los ficheros y los directorios creados. Con estos permisos el usuario podrá asignar al ficheros y directorios permisos de descarga o que solo estén visibles para una persona etc.

- Realizar ficheros y directorios públicos visibles para todo tipo de usuarios. Todo tipo de usuarios podrá acceder a los directorios y descargar sus ficheros.

- Buscar un fichero o un directorio dentro del sistema de ficheros del usuario.

- Realizar un sistema de ficheros semántico, que facilite la organización del espacio de nombres y localización de los archivos del sistema, permitiendo al usuario el acceso transparente al sistema de ficheros.

- Optimizar la aplicación para las últimas versiones de Android.

- Realizar la aplicación en un sistema operativo diferente como iOS o Windows Phone.

8.3 Presupuesto y Planificación

En esta sección se presenta el presupuesto del presente proyecto, así como la planificación resultante.

El presupuesto del presente TFG se muestra en el Anexo III. El salario de los distintos roles se ha obtenido de [30].

En cuanto a la planificación final del proyecto, se contempla en el Anexo IV. Se puede observar que todas las tareas han sido completadas al 100%, acabando con éxito el trabajo.

9 Bibliografía

[1] Android developer (Junio 2012): http://developer.android.com/index.html

[2] Android developer (Junio 2012): http://developer.android.com/about/dashboards/index.html

[3] Android developer (Junio 2012): http://developer.android.com/about/dashboards/index.html

[4] Apple (Junio 2012):http://www.apple.com/au/iphone/features/notification-center.html

[5] Apple (Junio 2012): http://support.apple.com/kb/HT4211?viewlocale=es_ES)

[6] Apple (Junio 2012): http://www.apple.com/hotnews/thoughts-on-flash/.

[7] Daily iPhone Blog (Junio 2012): http://dailyiphoneblog.com/2011/10/04/iphone-evolution-iphone-vs-iphone-3g-vs-iphone-3gs-vs-iphone-4-vs-iphone-4s/iphone-vs-iphone-3g-vs-iphone-3gs-vs-iphone-4-vs-iphone-4s/

[8]Cocoa Fundamentals Guide, Apple Developer (Junio 2012): http://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/CocoaFundamentals/CocoaFundamentals.pdf

[9] Microsoft (Junio 2012): http://www.microsoft.com/windowsphone/es-es/features/default.aspx

[10] Microsoft (Junio 2012): http://msdn.microsoft.com/en-us/library/ff967549(v=vs.92)

[11] CHW (Junio 2012): http://www.chw.net/2010/04/filtran-arquitectura-de-windows-phone-7/

[12] Isabel Gómez y Eduardo Ortega. *Windows Phone: Arquitectura de la Plataforma de Desarrollo* (Junio 2012): http://www.slideshare.net/movilforum/arquitectura-de-la-plataforma-de-desarrollo-de-windows-phone-7-5518091

[13] Almudena Díaz, Pedro Merino, Javier Rivas. *Análisis de Symbian OS para desarrollar aplicaciones distribuidas sobre terminales GPRS* (Junio 2012): http://www.lcc.uma.es/~pedro/mobile/Publications/pdfs/1520.pdf

[14] Infomafia.net (Junio 2012): http://www.infomafia.net/sistemas-operativos-21/arquitectura-del-sistema-operativo-symbian-356

[15] George Couloris, Jean Dollimore y Tim Kindberg. *Sistemas Distribuidos Conceptos y Diseño* (3ª Edición). Pearson Education.

[16] James F. Jurose y Keith W.Ross. *Redes de computadoras. Un enfoque descendente*. (3ª Edición). Pearson Education.

[17] Grupo ARCOS. Asignatura Sistemas Distribuidos. Titulación: Grado en Ingeniería Informática. Universidad Carlos III de Madrid. *Tema 5: Comunicación con sockets*

[18] Grupo ARCOS. Asignatura Sistemas Distribuidos. Titulación: Grado en Ingeniería Informática. Universidad Carlos III de Madrid. *Tema 6: Llamadas a procedimientos remotos.*

[19] Grupo ARCOS. Asignatura Sistemas Distribuidos. Titulación: Grado en Ingeniería Informática. Universidad Carlos III de Madrid. *Tema 8: Java RMI.*

[20] Simon St.Lauren, Joe Johnson y Edd Dumbill. *Programming Web Services with XML-RPC* (1ª Edición). Valerie Quercia.

[21] XML-RPC.com (Junio 2012): http://xmlrpc.scripting.com/spec.html

[22] Grupo ARCOS. Asignatura Sistemas Distribuidos. Titulación: Grado en Ingeniería Informática. Universidad Carlos III de Madrid. *Tema 9: Servicios Web.*

[23] Ethan Cerami. *Web Services Essentials* (1ª Edición). Simon St.Laurent.

[24] Rafael Navarro Marset. *Rest vs Web Services*. Modelado, Diseño e Implementación de Servicios Web 2006-07

[25] W3C Ubiquitous Web domain (Junio 2012): http://www.w3.org/XML/

[26] InfoQ (Junio 2012): http://www.infoq.com/articles/rest-introduction

[27] JSON (Junio 2012): http://www.json.org/json-es.html

[28] William Stallings. *Fundamentos de Seguridad en Redes Aplicaciones y Estándares* (2ª Edición). Pearson Prentice Hall.

[29] Grupo de Seguridad de las T.I. Ingeniería de la seguridad. Titulación: Grado en Ingeniería Informática. Universidad Carlos III de Madrid. *Tema 3-1: Protocolos de sistemas distribuidos.*

[30] *Salario medio de un programador* (Septiembre 2012): http://www.tufuncion.com/trabajo-programador.

[31] Nicolás Schiavón Raineri, *VoCS: Sistema de almacenamiento en la nube voluntario.* Trabajo de Fin de Grado. Universidad Carlos III de Madrid.

[32] Memcached (Julio 2012): http://memcached.org/.

[33] Gobierno de España (Junio 2012): http://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750

[34] Terry.Cho's blog (Junio 2012): http://javamaster.wordpress.com/2009/07/24/scrum-based-task-management/

10 Anexos

10.1 Anexo I Manual de Usuario

El manual de usuario permitirá a un nuevo usuario manejar correctamente la aplicación. El manual de usuario se compone de los siguientes módulos:

10.1.1 Autenticar Usuario

Cuando el usuario abre la aplicación se encuentra la interfaz inicial que se muestra en la figura 10.1

Figura 10.1: Interfaz inicial.

Para que el usuario pueda autenticarse en la aplicación y acceder a ella, debe de introducir el nick de usuario y su contraseña en los campos donde pone "Ingresa Usuario" e "Introduzca contraseña". Una vez introducidos tiene que pulsar el botón azul "Entrar".

Si el usuario no está registrado, debe de registrarse pulsando el botón "Regístrese aquí". Cuando se pulsa el botón, la aplicación cambiará de interfaz y mostrará la interfaz para registrarse. Se puede ver en el siguiente punto Registrar Usuario.

Si se ha pulsado el botón "Entrar" y los datos son correctos la aplicación mostrará la siguiente interfaz. Figura 10.2: Interfaz principal.

Figura 10.2: Interfaz principal.

10.1.2 Registrar Usuario

Para registrar un usuario se debe pulsar el botón "Regístrese aquí" de la interfaz inicial se puede apreciar en la figura 10.1. Cuando se pulsa ese botón se muestra la interfaz de registro, se puede apreciar en la figura 10.3. Esta interfaz está compuesta por siete campos, los cuales son todos obligatorios. Estos campos son el nick, el nombre, los apellidos, el email y el DNI del usuario; la contraseña y el captcha.

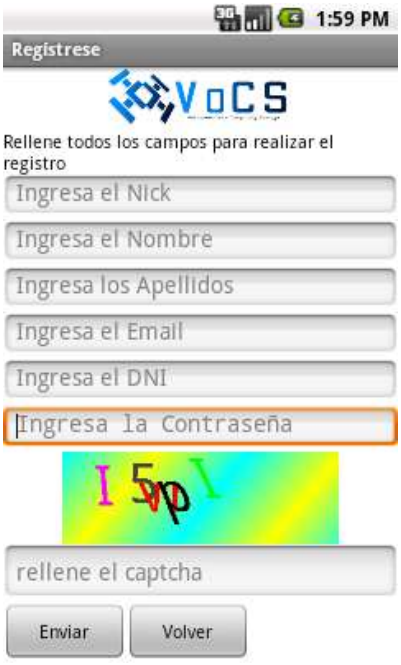


Figura 10.3: Interfaz de registro.

El captcha se tiene que rellenar introduciendo correctamente los valores que aparecen en la imagen. Estos valores pueden ser letras en mayúsculas y minúsculas y números.

10.1.3 Navegar Directorio

Para navegar por los directorios basta con pulsar en los directorios de la figura 10.2. Cuando se pulsa en el directorio PFG se muestra la interfaz como la de la figura 10.4.

Figura 10.4: Acceder a carpeta PFG.

Las dos primeras posiciones que son un "/" y un "../" y sirven para lo siguiente retroceder en el sistema de ficheros, si se pulsa "/" se retrocede al directorio principal y si se pulsa "../" se retrocede al directorio anterior.

Para explicar su funcionamiento se va a crear un directorio nuevo llamado libros. La Figura 10.5 muestra el directorio creado en el directorio PFG a la izquierda y a la derecha muestra la interfaz del directorio libros que se acaba de pulsar.

Si desde el directorio libros se pulsa "../", entonces se mostraría la interfaz de la figura 10.5 de la parte derecha. Pero si desde el directorio libros se pulsa "/", entonces se mostraría la interfaz de la figura 10.2.

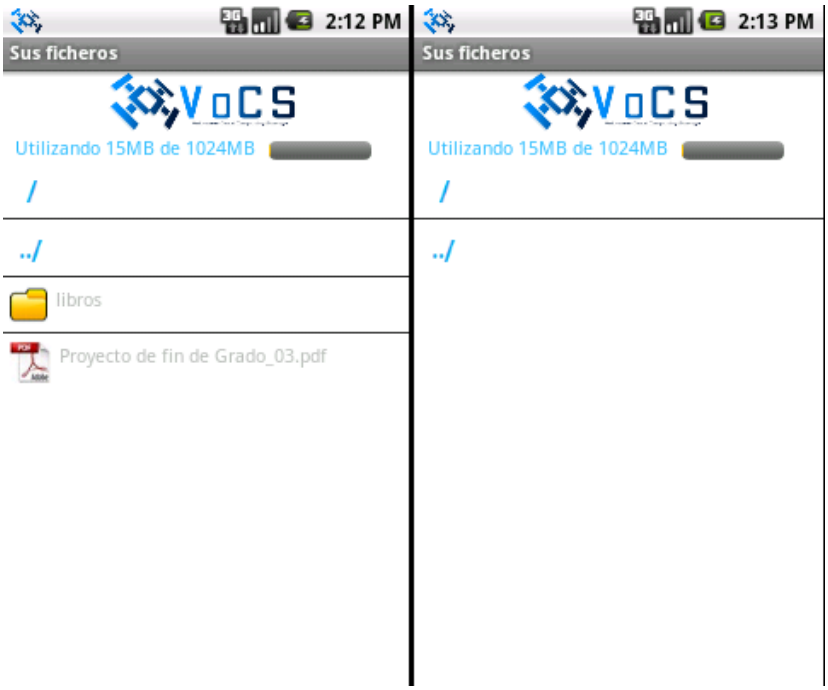


Figura 10.5: Ejemplo 1.

10.1.4 Descargar Fichero

Para descargar un fichero se tiene que pulsar el fichero que se desee descargar de la figura 9.2. Como se puede apreciar en la carpeta de descargas de VoCS, que se crea cuando se descarga un fichero por primera vez, esta vacía Figura 10.6.

Figura 10.6: Carpeta de descargas VoCS.

Si se selecciona un fichero como por ejemplo “Replicación a utilizar.docx” de la interfaz principal se muestra un menú como el que se muestra en la figura 10.7.

Figura 10.7: Submenú de selección opciones.

Como se quiere descargar el fichero se va a seleccionar la opción "descargar". Se empieza a descargar el fichero y cuando finaliza se informa del resultado mediante un mensaje y mediante una notificación por si la aplicación se ha dejado en segundo plano. En la figura 10.8 se muestra a la izquierda el mensaje informativo y a la derecha el mensaje informativo en la barra de notificaciones.

Figura 10.8: Mensajes informativos de la descarga.

Una vez descargado el fichero se puede encontrar en la carpeta de descargas de VoCS que se muestra en la figura 10.9.

Figura 10.9: muestra el fichero descargado.

10.1.5 Subir Fichero

Para subir un fichero a la nube, se tiene que pulsar el botón menú que tienen los dispositivos móviles Android. Cuando se pulsa el botón de menú aparece un menú con cuatro opciones. La figura 10.10 muestra la interfaz cuando se pulsa el botón menú desde la interfaz principal.

Figura 10.10: Interfaz de menú de opciones.

Para subir un fichero hay que pulsar la opción que tiene una flecha que apunta hacia arriba y pone "Subir". Cuando se pulsa esa opción, se muestra el sistema de ficheros interno del teléfono para

que se pueda seleccionar uno o varios ficheros. La figura 10.11 muestra el sistema interno del teléfono para seleccionar los ficheros que se quieran subir.

Figura 10.11: Sistema de ficheros del teléfono.

Cuando se selecciona un fichero como por ejemplo "Proyecto de fin de Grado_03.pdf", se muestra un submenú para seleccionar el nivel de seguridad con el que se quiere subir un fichero como el que se muestra en la figura 10.12.

Figura 10.12: Submenú de la seguridad.

Seleccionar un nivel u otro sólo tienen efectos en el rendimiento de la aplicación, ya que seleccionando el nivel 0 el fichero se guarda en el servidor, seleccionando el nivel 1 se guarda en

el servidor y se replica en otro servidor; y cuando se selecciona el nivel 2 se replica, se cifra y se guarda.

Se selecciona el nivel 0, hay dos formas de indicar que la subida de un fichero se ha realizado con éxito, a través de una notificación por si la aplicación se deja en segundo plano y a través de un mensaje que indica que se ha subido el fichero con éxito, muy similar a la figura 10.8.

10.1.6 Eliminar Fichero

Para eliminar un fichero se tiene que seleccionar un fichero. Para el ejemplo se va a usar el fichero "Kalimba.mp3" de la figura 10.2. Cuando se selecciona el fichero, se muestra el menú de la figura 10.7 y se seleccionará la opción "eliminar". La figura 10.13 muestra el resultado de seleccionar la opción eliminar fichero.

Figura 10.13: Fichero "Kalimba.mp3" ha sido borrado.

10.1.7 Restaurar Fichero Eliminado

Para restaurar un fichero que ha sido eliminado, se tiene que pulsar el botón menú que tienen los dispositivos móviles Android. Cuando se pulsa el botón de menú, se puede apreciar el menú en la figura 10.10.

Para restaurar un fichero eliminado, hay que pulsar en la opción "Opciones" en la que aparece una imagen de una tuerca. Cuando se pulsa esta opción, se muestra el submenú con las opciones "restaurar eliminados", eliminar directorio, "compartir directorio" y "ver invitaciones".

Figura 10.14: Submenú con las opciones.

Se elegirá la opción "restaurar eliminados". Esta opción muestra todos los ficheros borrados que tiene el usuario, incluidos versiones anteriores. Como se puede observar en la figura 10.15, se pueden observar todos los ficheros que tiene eliminados el usuario mostrando su nombre y su fecha de borrado.

Figura 10.15: Interfaz que muestra los ficheros inactivos del usuario.

Seleccionamos el fichero "Kalimba.mp3" y se restaura el fichero que se ha eliminado también. La figura 10.16 muestra el resultado de la operación.

Figura 10.16: Restaurar fichero "Kalimba.mp3".

10.1.8 Restaurar Versión Anterior

Para restaurar una versión anterior, se tiene que seleccionar algún fichero que tenga una versión anterior, ya que si no disponemos de ninguna versión anterior no podremos restaurarla. Para el ejemplo se va a usar el fichero "Formulario_Presupuesto.xlsx" de la figura 10.2. Cuando se selecciona el fichero, se muestra el menú de la figura 10.7 y se seleccionará la opción "restaurar".

Cuando se pulsa la opción restaurar se muestra un submenú con las versiones anteriores del fichero mostrando su nombre y la fecha en la que fueron remplazados por una versión posterior. La figura 10.17 muestra el submenú.

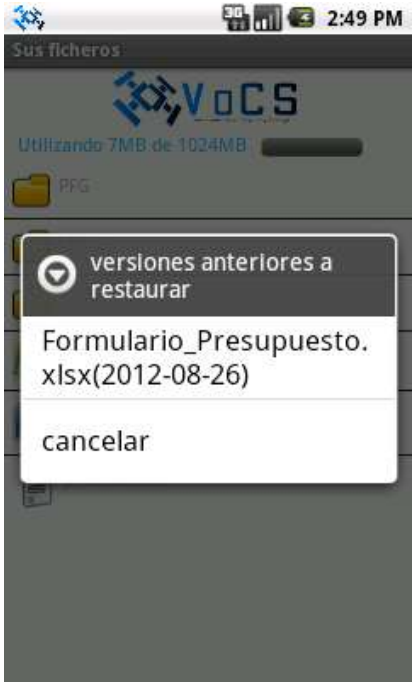


Figura 10.17: Versiones anteriores a restaurar.

Una vez seleccionado una versión anterior se restaura, pero en la interfaz no tiene ningún efecto, ya que el fichero ya esta cargado en la interfaz y no cambia ni su nombre ni su aspecto.

10.1.9 Crear Directorio

Para crear un directorio se tiene que pulsar el botón menú que tienen los dispositivos Android y se mostrará el menú de la figura 10.10. Para poder crear un directorio se tiene que seleccionar la opción en la que aparece un directorio y que tiene como título “Crear Directorio”.

Cuando se selecciona esta opción se muestra un diálogo que pide al usuario que introduzca un nombre para el directorio. La figura 10.18 muestra el diálogo.

Figura 10.18: Diálogo de nuevo directorio.

Si se introduce el nombre del directorio, como por ejemplo "VoCSDroid"; y se pulsa el botón "Ok", entonces se crea el directorio como se muestra en la figura 10.19. Pero si se pulsa el botón "cancelar", se cancela la operación.

Figura 10.19: Directorio nuevo VoCSDroid.

10.1.10 Eliminar Directorio

Para eliminar un directorio, se tiene que pulsar el botón menú que tienen los dispositivos móviles Android. Cuando se pulsa el botón de menú, se puede apreciar el menú de la figura 10.10.

Para eliminar un directorio, hay que pulsar en la opción “Opciones” como se muestra en la figura 10.14.

Se selecciona la opción “eliminar directorio” y se muestra un submenú con los directorios hijos del directorio por el que se esta navegando. Al estar en la interfaz principal, se muestran los directorios de la figura 10.20.

Figura 10.20: Interfaz que muestra los directorios.

Se selecciona el directorio que se acaba de crear “VoCSDroid” y la interfaz se actualiza eliminando dicho directorio.

10.1.11 Compartir Directorio

Para compartir un directorio, se tiene que pulsar el botón menú que tienen los dispositivos móviles Android. Cuando se pulsa el botón de menú, se puede apreciar el menú de la figura 10.10.

Para compartir un directorio, hay que pulsar en la opción “Opciones” como se muestra en la figura 10.14.

Se elige la opción “compartir directorio” y se muestra un submenú con los directorios hijos del directorio por el que se esta navegando. Al estar en la interfaz principal, se muestran los directorios de la figura 10.21.

Figura 10.21: Interfaz directorio a compartir.

Se elige un directorio como por ejemplo "PFG", que contiene varios ficheros y directorios y resultará beneficioso para el siguiente caso. Cuando se selecciona el fichero, se muestra un diálogo para introducir el email del usuario con quien se quiere compartir, como se muestra en la figura 10.22.

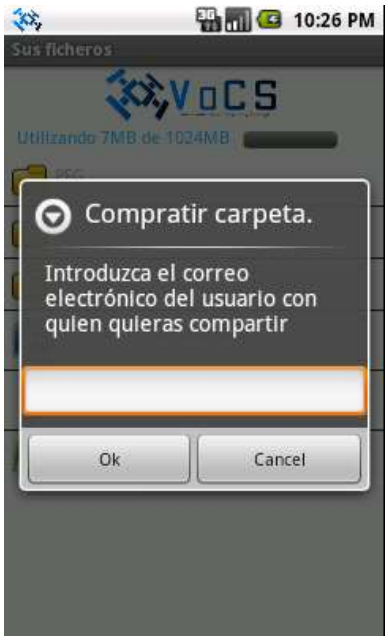


Figura 10.22: Interfaz donde se inserta el email.

Si se introduce el email y se pulsa "Ok", si todo ha ido bien se muestra un mensaje indicando que todo ha salido bien y se ha enviado la invitación. Si se pulsa "Cancel" se cancela la operación.

10.1.12 Aceptar/Cancelar Invitación

Para este caso se cambiará al usuario al cual se ha mandado la invitación. Figura 10.23.

Figura 10.23: Interfaz principal del nuevo usuario.

Para aceptar o cancelar una invitación, se tiene que pulsar el botón menú que tienen los dispositivos móviles Android. Cuando se pulsa el botón de menú, se puede apreciar el menú de la figura 10.10.

Para aceptar o cancelar una invitación, hay que pulsar en la opción "Opciones" como se muestra en la figura 10.14.

Se elige la opción "ver invitaciones" y se muestran las invitaciones que tiene pendientes de aceptar o de cancelar como se muestra en la figura 10.24. Se puede apreciar el nombre del directorio y entre paréntesis el nick del usuario que ha mandado la invitación y su email.

Figura 10.24: Interfaz de invitaciones pendientes.

Cuando Se selecciona la invitación se muestra un diálogo con dos botones “Ok” y “Cancelar” como el que se muestra en la figura 9.25.

Figura 10.25: Interfaz de aceptar/cancelar invitación.

Si se pulsa el botón “Cancelar”, entonces se cancela el proceso y a su vez se rechaza la invitación no volviendo a aparecer en la lista de invitaciones.

Si se pulsa “Ok”, se acepta la invitación del usuario y se puede comprobar que ha aparecido el directorio “PFG” y que internamente tiene los mismos datos. La figura 10.26 muestra el directorio y los directorios y ficheros internos.

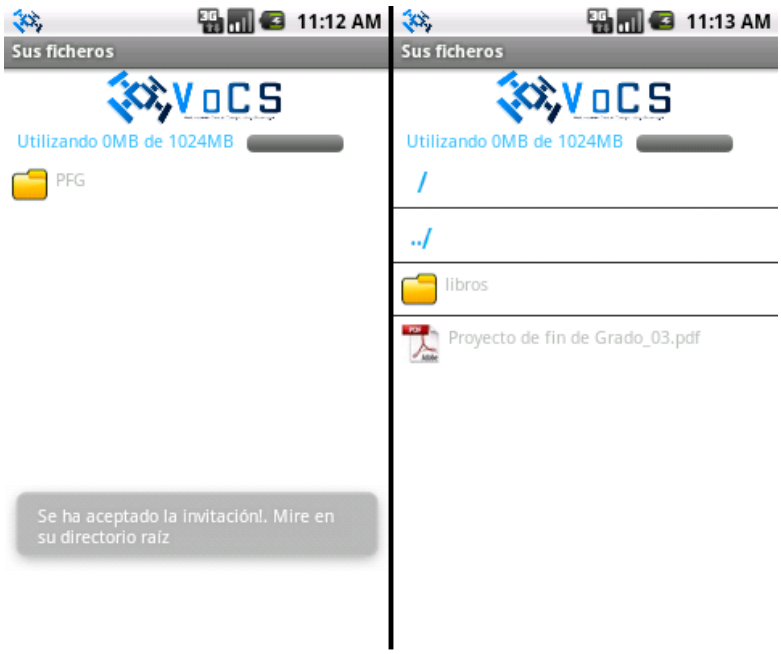


Figura 10.26: Interfaz con fichero compartido.

10.2 Anexo II Manual de Instalación

Para poder instalar la aplicación hay que obtener la aplicación en formato .apk que se puede obtener de la siguiente ruta del disco proporcionado "/AndroidVoCS/bin/AndoridVoCS.apk".

Existen varias formas para introducir la aplicación en el móvil e instalarla:

- Por USB: Se conecta el teléfono al PC y se crea una carpeta en el sistema de ficheros del teléfono. Acto seguido se abre un programa gestor de ficheros como por ejemplo "Astro", que es gratuito y se puede encontrar en Google Play, se accede a la carpeta y se instala en el dispositivo.

- Por email: Se recibe un email con la aplicación y cuando se recibe se puede observar el botón "Open" que cuando se pulsa se accede directamente al instalador de Android.

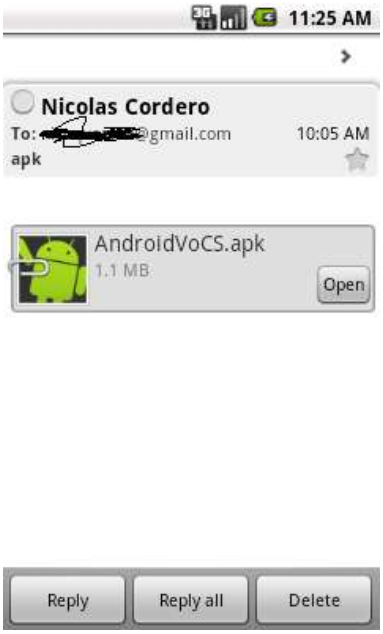


Figura 10.27: Email con la aplicación.

Cuando se pulsa el botón "Open" se inicial el instalador de Android que pregunta si se quiere instalar la aplicación. Para instalarla se pulsa el botón "Ok" y se muestra que se ha instalado la aplicación. Figura 10.28.

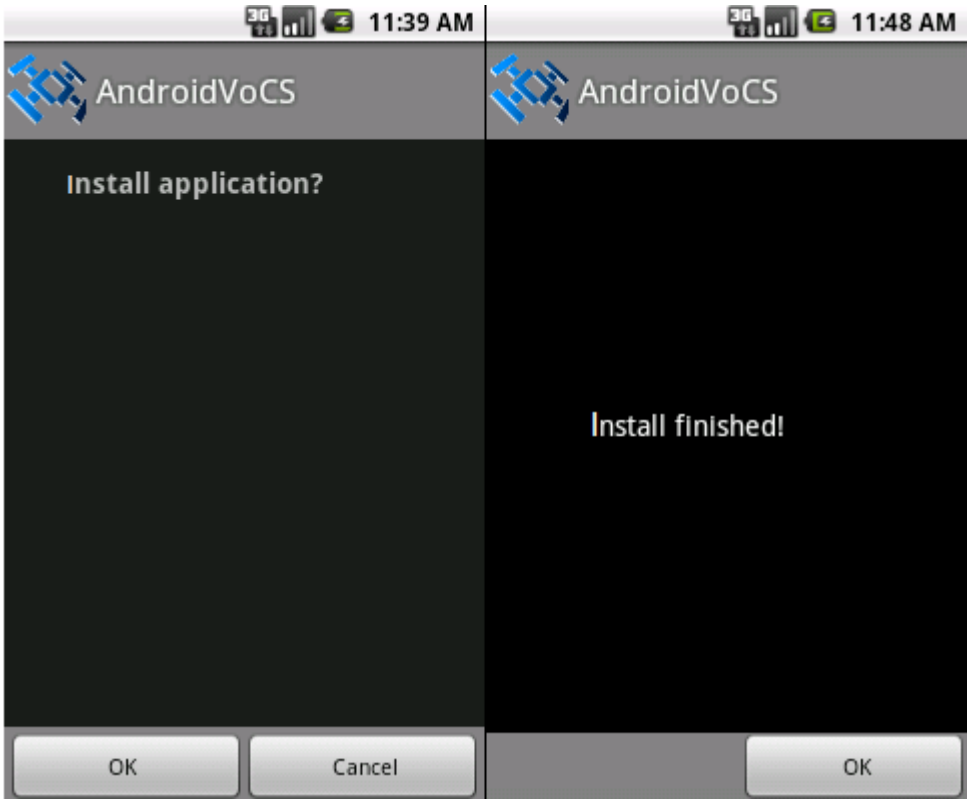


Figura 10.28: Instalación de la aplicación.

10.3 Anexo III Planificación Inicial

A continuación se muestra la planificación inicial.

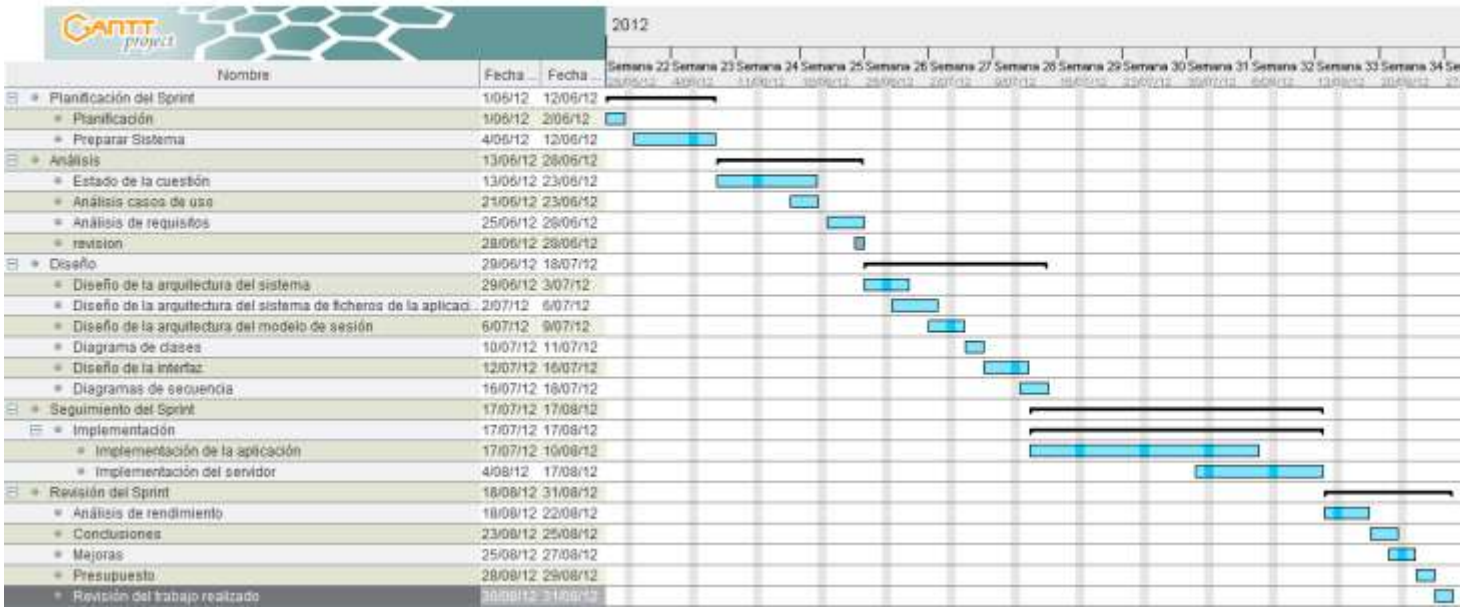


Figura 10.29: Diagrama de Gantt inicial.

10.4 Anexo IV Planificación Final

A continuación se muestra la planificación final con el diagrma de Gantt.

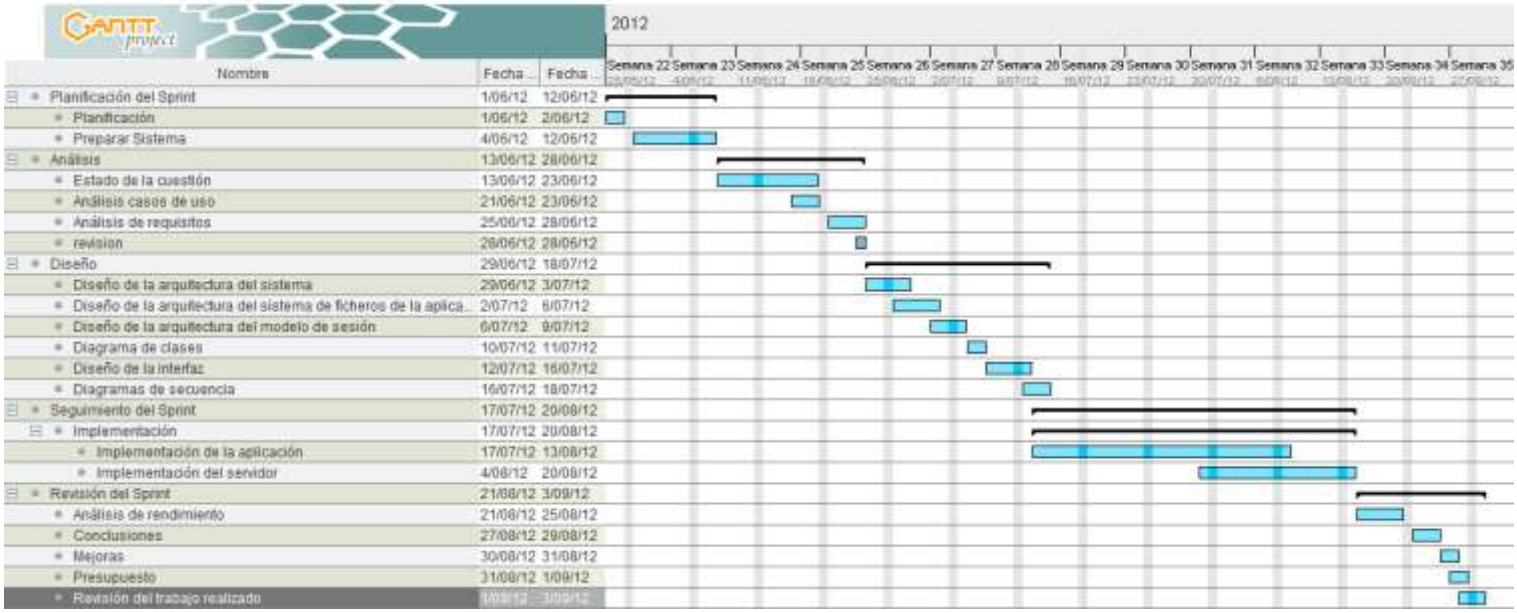


Figura 10.30: Diagrama de Gantt final.

10.5 Anexo V Presupuesto

1. Autor:
   Nicolás Cordero Fernández
2. Departamento:
   Informática.
3. Descripción del Proyecto
   - Título **VoCSDroid: Sistema de almacenamiento en la nube.**
   - Duración (meses) 3
   - Tasa de costes Indirectos: 20%
4. Presupuesto total del Proyecto (valores en Euros)
   8.820,00 Euros.
5. Desglose presupuestario (costes indirectos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a titulo informativo)	Categoría	Dedicación (hombres mes) [a)]	Coste hombre mes	Coste (Euro)
Cordero Fernández, Nicolás		Analista	1	3.000,00	3.000,00
Cordero Fernández, Nicolás		Programador	1	1.300,00	1.300,00
Cordero Fernández, Nicolás		Diseñador	1	2.200,00	2.200,00
					0,00
					0,00
		Hombres	**3**	**Total**	6.500,00

mes

a) 1 Hombre 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]
Acer ASPIRE 5942G	700	100	3	60	35,00
HTC Desire	150	100	3	60	7,50
Pentium 4 2.4 GHz	100	100	3	60	5,00
AMD Athlon 64 processor 3200+	150	100	3	60	7,50
			3	60	0,00
					0,00
				Total	55,00

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

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	Costes imputable
Internet	Moviestar	100,00
Tarifa de datos	Pepephone	45,00
Microsoft Office	Microsoft	130,00
Dietas		400,00

Luz	Iberdrola	120,00
	Total	795,00

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

6. Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	6.500
Amortización	55
Subcontratación de tareas	0
Costes de funcionamiento	795
Costes Indirectos	1.470
Total	8.820