Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

NoMailbox: Aplicación Android para el envío seguro de ficheros punto a punto

NoMailbox: Android app for secure sending files point-to-point

Javier Antonio González Hernández

La Laguna, 13 de marzo de 2023

D. **Francisco Javier Rodríguez González**, con N.I.F. 43.618.712-V profesor asociado de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Alejandro Pérez Nava**, con N.I.F. 43.821.179-S profesor asociado de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"NoMailbox: Aplicación Android para el envío seguro de ficheros punto a punto"

ha sido realizada bajo su dirección por D. **Javier Antonio González Hernández**,

con N.I.F. 78.634.565-A.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 13 de marzo de 2023

Agradecimientos

A mis padres y a mi abuela materna, por todo el apoyo que me han dado a lo largo de los años para que pudiera formarme y tener un futuro con muchas oportunidades.

A mi pareja, por el apoyo incondicional durante el desarrollo del presente proyecto.

A mis amigos, que nunca han dejado de creer en mí.

A mi tutor Francisco por todo el apoyo, las recomendaciones y orientación en las distintas etapas del desarrollo del presente proyecto.

A todas esas personas que comparten su conocimiento a través de internet, abriendo un mundo de posibilidades para todo aquel que quiera aprender.

Licencia

Resumen

La privacidad es un tema recurrente cuando se habla de internet, generando muchos debates sobre como las empresas utilizan los datos personales de sus usuarios y con quien los comparten. Hoy en día se utiliza internet prácticamente para cualquier trámite telemático gracias a la inmediatez y comodidad que proporciona. Hay ocasiones donde se requiere enviar documentación con información sensible, como por ejemplo el DNI o un documento legal. En estas ocasiones se suelen utilizar servicios como el correo o una aplicación de mensajería, siendo estas las opciones menos adecuadas para el envío de este tipo de documentación, ya que no se sabe a ciencia cierta si se obtiene información de los ficheros o de la comunicación entre los usuarios implicados. Del envío de un fichero se puede obtener información, sin la necesidad de mirar el propio fichero. Es por ello por lo que este trabajo de fin de grado tiene como objetivo implementar una aplicación Android dirigida para este tipo de escenarios.

La aplicación está destinada a profesionales y sus clientes. Los clientes pueden enviar ficheros garantizándose un envío seguro, donde sus archivos no se almacenan en ningún servidor ni la transacción genera ningún tipo de información de la que se pueda sacar conclusiones de las partes implicadas. Para lograr esto se ha estudiado como conseguir que dos terminales Android puedan conectarse de forma directa y segura para enviar ficheros sin que se almacenen en un servidor, consiguiendo así la conexión punto a punto que plantea el presente proyecto.

Palabras clave: punto a punto, Spring, Android, envío seguro de ficheros, clientes, profesionales.

Abstract

Privacy is a recurring theme when talking about the internet, generating many debates about how companies use their users' personal data and with whom they share it. Nowadays the internet is used for practically any telematic procedure thanks to the immediacy and convenience it provides. There are occasions when it is necessary to send documentation with sensitive information, such as an DNI or a legal document. Services such as mail or a messaging application are usually used to send this kind of documents, being the least suitable options for sending this type of documentation, as it is not known for sure whether information is obtained from the files or from the communication between the users involved. This is the reason why this final degree project aims to implement an Android application aimed at this kind of scenarios.

The application is aimed at professionals and their clients. Clients can send files guaranteeing a secure sending, where their files are not stored on any server and the transaction does not generate any type of information to infer information about the parties involved. To achieve this, we have studied how two Android terminals can be connected directly and securely to send files without having to go through a server, achieving the point-to-point connection proposed in this project.

***Keywords**:* point-to-point, Spring, Android, secure sending files, clients, professionals.

Índice general

Capítulo 1. Introducción 1
1.1 Definición del problema 1
1.2 Justificación 2
1.3 Objetivos del proyecto 3
1.4 Estado del arte 4
1.4.1 Intercambio de información de carácter personal o profesional .. 4
1.4.2 Comparativa con aplicaciones similares 5
1.4.3 Reglamento general de protección de datos 7
1.5 Fases del proyecto 8
1.5.1 Estudio previo 9
1.5.2 Diseño 9
1.5.3 Implementación 10
1.5.4 Pruebas 10
1.5.5 Estudio de la viabilidad económica del proyecto 10
Capítulo 2. Estudio previo 11
2.1 Análisis y definición de requisitos 11
2.2 Arquitectura del sistema 12
2.3 Selección de tecnologías 13
2.3.1 Aplicación Android 13
Lenguaje de programación 14
Nivel de API 15
Librerías 16
Entorno de desarrollo 17
2.3.2 API REST 17
Framework y lenguaje de programación 18
Entorno de desarrollo 18
Servidor donde se ejecuta la API 19
2.3.3 Base de datos 19
Sistema gestor de base de datos y lenguaje de consulta 19
Herramienta de administración de bases de datos 20
2.3.4 Otras tecnologías 20

Sistema de control de versiones 20
Cliente git 20
Sistema de gestión del proyecto 21
Pruebas de la API REST 21
Capítulo 3. Aplicación Android 22
3.1 Arquitectura del sistema 22
3.2 Funcionalidades 25
3.2.1 Pantalla de bienvenida 25
3.2.2 Registro 25
3.2.3 Inicio de sesión 26
3.2.4 Navegación del profesional 27
3.2.5 Navegación del cliente 28
Capítulo 4. API REST 30
4.1 Arquitectura 30
4.2 Endpoints de la API REST 32
Capítulo 5. Base de datos 34
5.1 Diseño 34
5.2 Creación de la base de datos 34
Capítulo 6. Conexión punto a punto 36
Capítulo 7. Estudio de viabilidad económica 42
7.1 Estudio de mercado 42
7.2 Estimación de costes 43
7.3 Estimación de ingresos 46
Capítulo 8. Conclusiones y Líneas futuras 49
Capítulo 9. Summary and Conclusions 50
Capítulo 10. Presupuesto 51
Apéndice A 52
Repositorio de código 52

Índice de figuras

Figura 1: Envío de información confidencial de los españoles encuetados 4
Figura 2: Confianza de los encuestado en internet para enviar documentación 5
Figura 3: Logo de bitwarden 5
Figura 4: Logo de Sync 6
Figura 5: Logo de toffeeshare 6
Figura 6: Logo de tresorit send 6
Figura 7: Logo de DocToDoctor 7
Figura 8: Fases del proyecto 9
Figura 9: Diagrama de casos de uso del proyecto 12
Figura 10: Arquitectura general del sistema 13
Figura 11: Ventas totales de móviles por sistema operativo en 2020 14
Figura 12: Evolución de la popularidad de lenguajes de programación según el índice TIOBE 14
Figura 13: Logo de Kotlin 15
Figura 14: Logo de Java 15
Figura 15: Porcentaje de distribución de las distintas versiones de Android 16
Figura 16: Logo de Android Studio 17
Figura 17: Logo de Spring 18
Figura 18: Logo de IntelliJ IDEA 19
Figura 19: Logo de Apache Tomcat 19
Figura 20: Logo de MariaDB 20
Figura 21: Logo de DBeaver 20
Figura 22: Logo de git 20
Figura 23: Logo de Sourcetree 21
Figura 24: Logo de Github 21
Figura 25: Logo de postman 21
Figura 26: Diagrama de la arquitectura CLEAN 22

Figura 27: Estructura del primer prototipo 23
Figura 28: Estructura del segundo prototipo 23
Figura 29: Detalle de la estructura del primer prototipo..... 24
Figura 30: Detalle de la estructura del segundo prototipo 24
Figura 31: Pantalla de bienvenida 25
Figura 32: Pantalla de registro del cliente y profesional 25
Figura 33: Pantalla de inicio de sesión cliente y profesional 26
Figura 34: Lista de contactos del profesional..... 27
Figura 35: Código generado por el profesional 27
Figura 36: Pantalla de detalle del cliente..... 28
Figura 37: Pantalla de contactos del cliente 28
Figura 38: Pantalla de detalle contacto del cliente 29
Figura 39: Envío de código para añadir profesional 29
Figura 40: Estructura de ficheros de la API REST..... 30
Figura 41: Clases que componen la API REST..... 31
Figura 42: Diagrama Entidad/Relación de la base de datos 34
Figura 43: Estructura de la base de datos 34
Figura 44:Estructura de la tabla cliente 35
Figura 45: Estructura de la tabla profesional 35
Figura 46: Estructura de la tabla cliente_profesional..... 35
Figura 47: Estructura de la tabla codigo_contacto 35
Figura 48: Estrutura de la tabla profesión..... 35
Figura 49: Activación del envío de archivos..... 36
Figura 50: Solicitud de la IP y el puerto del profesional 37
Figura 51: Envío de archivo 37
Figura 52: Ruta del profesional donde se almacenan los archivos 38
Figura 53: Borrado de la IP y puerto del profesional de la base de datos..... 38
Figura 54: Implementación de envío de ficheros mediante sockets..... 39
Figura 55: Implementación de recepción de ficheros mediante sockets 40
Figura 56: Primera parte del diagrama de Gantt..... 44
Figura 57: Segunda parte del diagrama de Gantt..... 45
Figura 58: Gastos acumulados del proyecto 45
Figura 59: Planes disponibles para la aplicación 46
Figura 60: Retorno de la inversión..... 48

Índice de tablas

Tabla 1: ***Comparativa de aplicaciones*** **..... 7**
Tabla 2: ***Librerías utilizadas en la aplicación Android*** **..... 16**
Tabla 3: ***endpoints de la API REST*** **..... 32**
Tabla 4: ***número de descargas de aplicaciones similares a NoMailbox*** **..... 42**
Tabla 5: ***Sueldo medio en horas de los puestos en España en 2023*** **..... 43**
Tabla 6: ***Estimación de ingresos*** **..... 46**
Tabla 7: ***Presupuesto del proyecto*** **..... 51**

Capítulo 1. Introducción

Para contextualizar el proyecto se presentan las partes fundamentales que han permitido fijar desde el primer momento la motivación, los objetivos y el alcance del proyecto. Primero se establece cual ha sido la necesidad tecnológica que busca satisfacer el presente proyecto, definiendo de forma detallada el problema. En segundo lugar, se justifica la razón de ser del proyecto y por qué es una propuesta adecuada para el problema descrito en el apartado anterior.

Una vez aclarado el valor del proyecto, se marcan objetivos y se delimita el alcance del proyecto, marcando el camino a seguir para comenzar a diseñar y desarrollar el sistema.

Habiendo definido los objetivos y el alcance del proyecto, se hace un estudio de la situación actual en materia de envío seguro de ficheros y como influyen las leyes en materia de protección de datos al desarrollo de un proyecto software en el apartado del estado del arte.

Por último, se describen las distintas partes de las que ha constado el proyecto.

1.1 Definición del problema

Existen numerosas aplicaciones y servicios para el intercambio de archivos a través de internet de forma sencilla e inmediata. Esto ha provocado la normalización del envío de archivos de distinta naturaleza, sin tener un criterio de elección más allá de usar el servicio más popular o el que el usuario esté más acostumbrado a utilizar en su día a día.

No todos los ficheros son iguales, y por lo tanto la elección para su envío tampoco debería ser la misma. El envío de archivos con información confidencial, como pueden ser el DNI, documentación médica o legal, es un caso particular que requiere de una solución tecnológica específica que cumpla con una serie de requisitos de seguridad y privacidad que garanticen la confidencialidad de la información.

Existiendo tantas alternativas, surge una pregunta a la hora de compartir archivos confidenciales: ***¿Qué aplicación o servicio se debe utilizar?*** Esta pregunta no tiene una respuesta sencilla, ya que hay

que tener en cuenta muchos factores, como la política de privacidad y gestión de datos personales, qué mecanismos de seguridad utiliza para mantener segura la información, donde se almacena el archivo, etc.

Para contextualizar el problema, se puede pensar en un médico que solicita una fotografía a su paciente para una evaluación en una consulta telefónica. El médico no tiene una aplicación concreta para esta tarea, por lo que tendrá que recomendar usar el correo o una aplicación de mensajería, teniendo que dar datos de contacto. El paciente no está seguro si es la mejor opción para enviar información que expone su estado de salud.

Este tipo de situaciones pueden suponer un problema porque no se puede saber con seguridad que ocurre con los ficheros que se envían y que información de los implicados se registra durante la transacción. El propio fichero podría quedarse almacenado en los servidores de la empresa que ofrece el servicio, podría ser analizado [1] o, aunque eso no fuera posible por mecanismos de seguridad como el cifrado extremo a extremo[1], la empresa podría obtener información de los implicados por el mero hecho de establecer la comunicación.

1.2 Justificación

Atendiendo a la necesidad tecnológica que genera el problema descrito en el apartado anterior, el presente proyecto propone la implementación de una aplicación Android [2] nativa dirigida a clientes y profesionales que necesiten intercambiar ficheros confidenciales. El objetivo principal de la aplicación es ser una opción a tener en cuenta, proporcionando las siguientes características que pueden resultar de interés tanto para los clientes como para los profesionales que necesiten intercambiar ficheros confidenciales:

- Los clientes pueden enviar sus ficheros de manera que no se almacenen en ningún servidor y sin registro de transacciones de las que se puedan obtener datos. Esto puede incentivar que la aplicación se asocie a este tipo de escenarios donde un profesional solicite documentación con información sensible a un cliente.
- Los profesionales tienen el control de la relación profesional, asegurando que la gestión de los clientes está completamente bajo su control, requiriendo que tanto para ser agregado como

[1] **Cifrado extremo a extremo:** sistema que cifra las comunicaciones de manera que solo los usuarios que se comunican pueden tener acceso a la información compartida.

contacto por los clientes como para recibir ficheros tengan que autorizarlo previamente. Para los profesionales es un punto a favor poder controlar cuando los clientes pueden comunicarse con ellos.

Además, ambas partes se ven beneficiadas porque se facilita el procedimiento de intercambio de ficheros tanto para el cliente como para el profesional, ya que tienen a disposición una solución tecnológica especializada para este tipo de situaciones, no teniendo que buscar alternativas que pueden generar problemas y dificultar el intercambio de ficheros.

1.3 Objetivos del proyecto

La definición de los objetivos del proyecto pretende dar respuesta a la siguiente pregunta: **¿Es posible ofrecer una aplicación móvil especializada que garantice el envío seguro punto a punto de ficheros entre dos terminales?**

La parte que más valor aporta y diferencia al proyecto es la conexión directa a través de internet entre los terminales. La tendencia de las aplicaciones de este estilo es utilizar un servidor intermedio que sirva como almacenamiento del fichero enviado por el emisor, para que el receptor pueda descargarlo cuando se conecte a la red. Esto supone una comodidad tanto para los usuarios como para los programadores. Por parte de los usuarios, no requiere que ambos participantes estén conectados a la red al mismo tiempo para intercambiar ficheros. Para los desarrolladores tener un servidor donde almacenar el archivo es más sencillo desde el punto de vista técnico. Un sistema que elimina el papel del servidor como almacenamiento temporal introduce una serie de complejidades y factores que hay que tener en cuenta durante el diseño e implementación del sistema.

Este es el principal reto del proyecto, que define los dos grandes objetivos de éste:

- Diseñar e implementar un prototipo en forma de aplicación Android que permita una conexión directa y segura entre dos terminales a través de internet.
- Una vez establecida la conexión directa y segura, enviar el fichero desde el cliente hasta el profesional.

Además de los objetivos principales, también se han definido una serie de objetivos necesarios para el desarrollo del proyecto:

- Definir los requisitos de la aplicación.

- Estudiar y seleccionar las librerías, *frameworks*[2], lenguajes de programación y entornos de desarrollo más adecuados para el proyecto.
- Diseñar e implementar una API REST [3] para darle soporte a la aplicación Android en operaciones que requieran de peticiones al servidor.
- Diseñar y crear una base de datos para proporcionar al sistema persistencia de datos.
- Corregir los errores detectados en los distintos componentes del sistema.
- Estudiar la viabilidad económica del proyecto.

1.4 Estado del arte

Para identificar las tendencias actuales que influyen directamente en el proyecto se ha realizado un estudio en tres áreas fundamentales: El intercambio de información de carácter personal o profesional, aplicaciones similares a la propuesta en el proyecto y como afecta el reglamento general de protección de datos (RGPD) [4] al desarrollo de la aplicación.

1.4.1 Intercambio de información de carácter personal o profesional

Se realizó un estudio en el año 2013 sobre el intercambio de archivos de carácter personal o profesional a 716 españoles mayores de 18 años. El 63,4% de los encuestados afirmó que compartía información de carácter personal o profesional por uno de estos medios. El 51% mediante el correo electrónico, el 31% WhatsApp, y el resto mediante otro tipo de servicios [5].

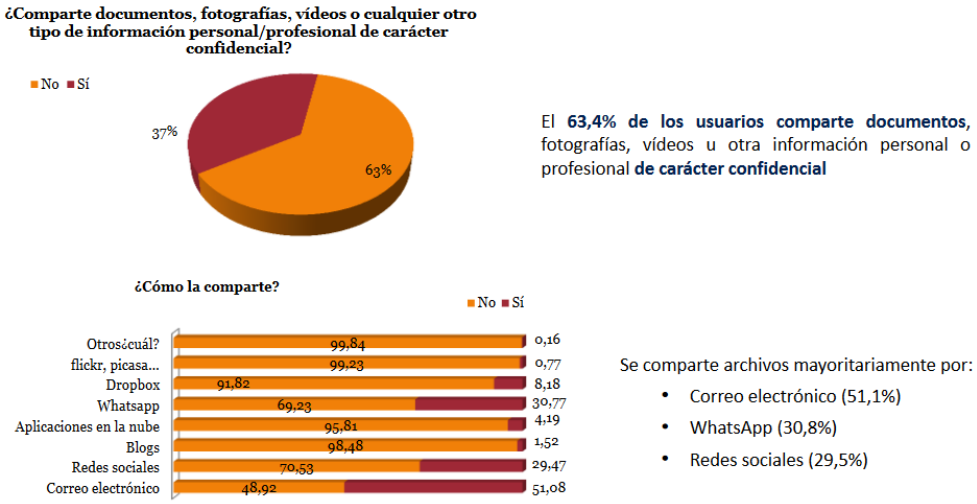


Figura 1: *Envío de información confidencial de los españoles encuetados*

[2]**Framework:** conjunto de herramientas y metodologías para resolver un problema en particular.

En cuanto al grado de confianza en internet para enviar documentación necesaria el 60% de los encuestados no confía, el 38% sí lo hace y el resto no contesta [5].

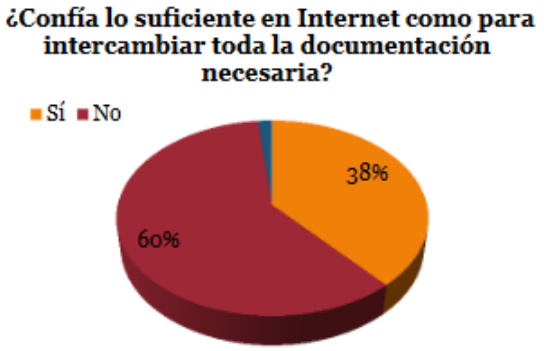


Figura 2*: Confianza de los encuestado en internet para enviar documentación*

Aunque la encuesta dista de ser actual, refleja la tendencia que había por aquel entonces en cuanto al intercambio de información confidencial. En la actualidad, donde muchos trámites son electrónicos, probablemente se sigan utilizando servicios similares a los que se usaban hace una década para intercambiar archivos de carácter personal o profesional. También, se percibe la preocupación por la privacidad online, una tendencia que ha ido aumentando a lo largo de los años [6].

1.4.2 Comparativa con aplicaciones similares

A continuación, se describen las distintas aplicaciones y sus principales características que de una forma u otra ofrecen una solución para el problema definido en este proyecto.

Bitwarden send

Es un gestor de contraseñas que ofrece la opción de enviar ficheros de hasta 100 MB desde el móvil de forma segura mediante cifrado punto a punto. Cada envío tiene asociado un tiempo de vida, eliminando su contenido una vez pasa la fecha fijada por el usuario [7].

Figura 3*: Logo de bitwarden*

Sync

Es una plataforma para almacenar y compartir documentos de forma segura, mediante cifrado punto a punto, en el ámbito profesional. Los documentos son accesibles desde el ordenador y el móvil [8].

Figura 4: *Logo de Sync*

Toffeeshare

Es un servicio *Peer-to-Peer (P2P[3])* que permite enviar ficheros directamente entre dispositivos, sin almacenar el fichero en un servidor y con cifrado extremo a extremo [9]. Al no almacenar ficheros en un servidor, es la aplicación más similar a la desarrollada en el proyecto.

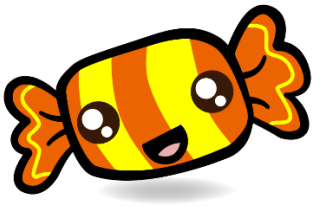


Figura 5: *Logo de toffeeshare*

Tresorit send

Es un servicio de almacenamiento en la nube que ofrece la posibilidad de compartir ficheros mediante enlaces. Utiliza el cifrado punto a punto, además de ofrecer un control de los documentos una vez compartidos, mediante registros de accesos y permitiendo revocar permisos [10].

Figura 6: *Logo de tresorit send*

DocToDoctor

Es una aplicación para compartir información clínica de forma segura entre médicos, de forma que puedan compartir opiniones y consejos sobre casos clínicos. Toda la información se almacena de forma cifrada en la nube [11].

[3] **P2P:** arquitectura de red en la que los dispositivos que la conforman pueden ser tanto clientes como servidores.

Figura 7: *Logo de DocToDoctor*

Las aplicaciones seleccionadas tienen varios aspectos en común con la aplicación desarrollada. En la siguiente tabla se reflejan las similitudes y diferencias entre las aplicaciones:

Tabla 1: *Comparativa de aplicaciones*

Aplicación	Ámbito	Cifrado extremo a extremo	Ficheros almacenados en servidores	Tamaño máximo de archivo
Bitwarden	General	Sí	Sí	100 MB
Sync	Profesional	Sí	Sí	Sin límite
Toffeeshare	General	Sí	No	Sin límite
Tresorit send	General	Sí	Sí	5 GB
Doctodoctor	Médico	No especificado	Sí	Sin límite
Nomailbox	Profesional	Sí	No	Sin límite

1.4.3 Reglamento general de protección de datos

El reglamento general de protección de datos tiene como objetivo proteger los derechos fundamentales de las personas en la era digital, facilitando la actividad económica, ya que aclara las normas aplicables a las empresas y los organismos públicos en el mercado único digital. [12]

La aplicación requiere de datos personales de los usuarios. Por una parte, los datos que usan para identificarse, y por otro, los archivos con información sensible que envían los clientes. Toda esta información debe protegerse y gestionarse acorde a las la RGPD para proteger a los ciudadanos de la Unión Europea. Por ello, a continuación, se enumeran los principales artículos que afectan directamente al desarrollo de una aplicación que requiera el uso de datos personales.

- **Licitud del tratamiento** (*Artículo 6*). Se tiene que obtener el consentimiento explícito del usuario para el tratamiento de sus datos. [13]

- **Condiciones para el consentimiento** (*Artículo 7*). Se definen los requisitos para obtener el

consentimiento del usuario, incluyendo la necesidad de que sea claro y fácil de retirar. [14]

- **Transparencia de la información, comunicación y modalidades de ejercicio de los derechos del interesado** (*Artículo 12*). Proporcionar información clara y concisa sobre el tratamiento de los datos personales del usuario. [15]

- **Información que deberá facilitarse cuando los datos personales se obtengan del interesado** (*Artículo 13*). Las empresas tienen que informar a los usuarios sobre los datos personales que se recopilan, como se utilizarán y a quien se proporcionarán. [16]

- **Derecho de acceso del interesado** (*Artículo 15*). Los usuarios tienen el derecho de acceder a sus datos personales y solicitar su eliminación. [17]

- **Seguridad del tratamiento** (*Artículo 32*). Establece las medidas de seguridad, técnicas y organizativas, para proteger los datos personales de los usuarios. [18]

Es importante cumplir con la normativa vigente en materia de protección de datos, adaptando la aplicación a los requisitos que establece la RGPD. En este apartado se han visto algunos de los más importantes para reflejar como afectan las regulaciones al desarrollo *software*. En un entorno de producción real, esto sería insuficiente, requiriéndose de asesoramiento legal por parte de expertos en la materia.

1.5 Fases del proyecto

Para facilitar el desarrollo y tener un mayor control sobre el proyecto, se ha dividido en cinco fases bien diferenciadas. Las fases y los resultados obtenidos al finalizar cada una de ellas se muestran en la siguiente figura:

Figura 8: *Fases del proyecto*

A continuación, se describen las distintas fases de forma detallada, definiendo los objetivos a llevar a cabo en cada una de ellas.

1.5.1 Estudio previo

Esta fase está dedicada a definir los puntos de interés de los que va a partir el proyecto, definiendo el alcance de este y las tecnologías para desarrollarlo.

- Definición de requisitos.
- Arquitectura general del sistema.
- Herramientas, librerías y *frameworks* más adecuados para cada componente del sistema.
- Arquitectura y patrones de diseño de la aplicación Android.
- Como establecer una conexión directa y segura entre dos terminales Android para enviar un archivo.
- Arquitectura de la API REST.
- Sistema gestor de base de datos a utilizar.

1.5.2 Diseño

Habiendo ya establecido las herramientas a utilizar y los objetivos principales del proyecto, se

procede a realizar un diseño de los componentes del sistema que se han identificado en el punto anterior:

- Arquitectura y patrones de diseño de la aplicación Android.
- Arquitectura de la API del servidor.
- Modelo de datos.

1.5.3 Implementación

Se implementan los diseños obtenidos utilizando las herramientas y librerías seleccionadas en la primera fase:

- Implementación de la aplicación Android.
- Implementación de la API REST.
- Creación de la base de datos.

1.5.4 Pruebas

En esta etapa se analiza si cada parte del sistema funciona correctamente mediante los siguientes puntos:

- La aplicación Android funciona correctamente.
- La API del servidor responde adecuadamente a las solicitudes de la aplicación Android en todo momento, pudiendo gestionar los posibles errores que surjan.
- La base de datos es capaz de responder a las necesidades de almacenamiento del proyecto.

1.5.5 Estudio de la viabilidad económica del proyecto

En esta última fase, se realiza una estimación de la viabilidad económica del proyecto siguiendo los siguientes puntos:

- Análisis de mercado
- Estimación de costes
- Estimación de ingresos

Capítulo 2. Estudio previo

El capítulo 2 está centrado en el punto de partida del proyecto. Antes de comenzar a escribir código, es fundamental saber el producto que se quiere obtener, que herramientas existen para lograrlo y cuáles son las más adecuadas. Para ello dar respuesta a estas cuestiones, se definen los requisitos del proyecto, la arquitectura general del sistema y las tecnologías seleccionadas para implementarlo.

2.1 Análisis y definición de requisitos

El análisis de requisitos establece los objetivos y el alcance del proyecto de manera que se pueda saber cuál es el resultado final esperado. Para el desarrollo de la aplicación se han definido los siguientes requisitos:

1. La aplicación debe permitir dos tipos de rol de usuario: cliente y profesional.
    a. Ambos roles pueden registrarse e iniciar sesión.
    b. Un mismo usuario puede iniciar sesión tanto como cliente como profesional.
2. Relación bajo el consentimiento del profesional. El profesional debe proporcionar un código para añadir al cliente como contacto en la aplicación.
3. Los archivos que envíe el cliente tienen que enviarse directamente al móvil del profesional. Los archivos no pueden almacenarse en ningún servidor.
4. El envío de archivos se hará previo consentimiento del profesional. La aplicación debe permitir que el profesional active y desactive el envío de fichero al cliente de su elección.
5. Los archivos recibidos por el profesional tienen que estar confinados en la propia aplicación, solo pudiendo ser accesibles desde la misma.

A partir de los requisitos se ha generado un diagrama de casos de uso que representa de forma gráfica como interactúan los usuarios con el sistema.

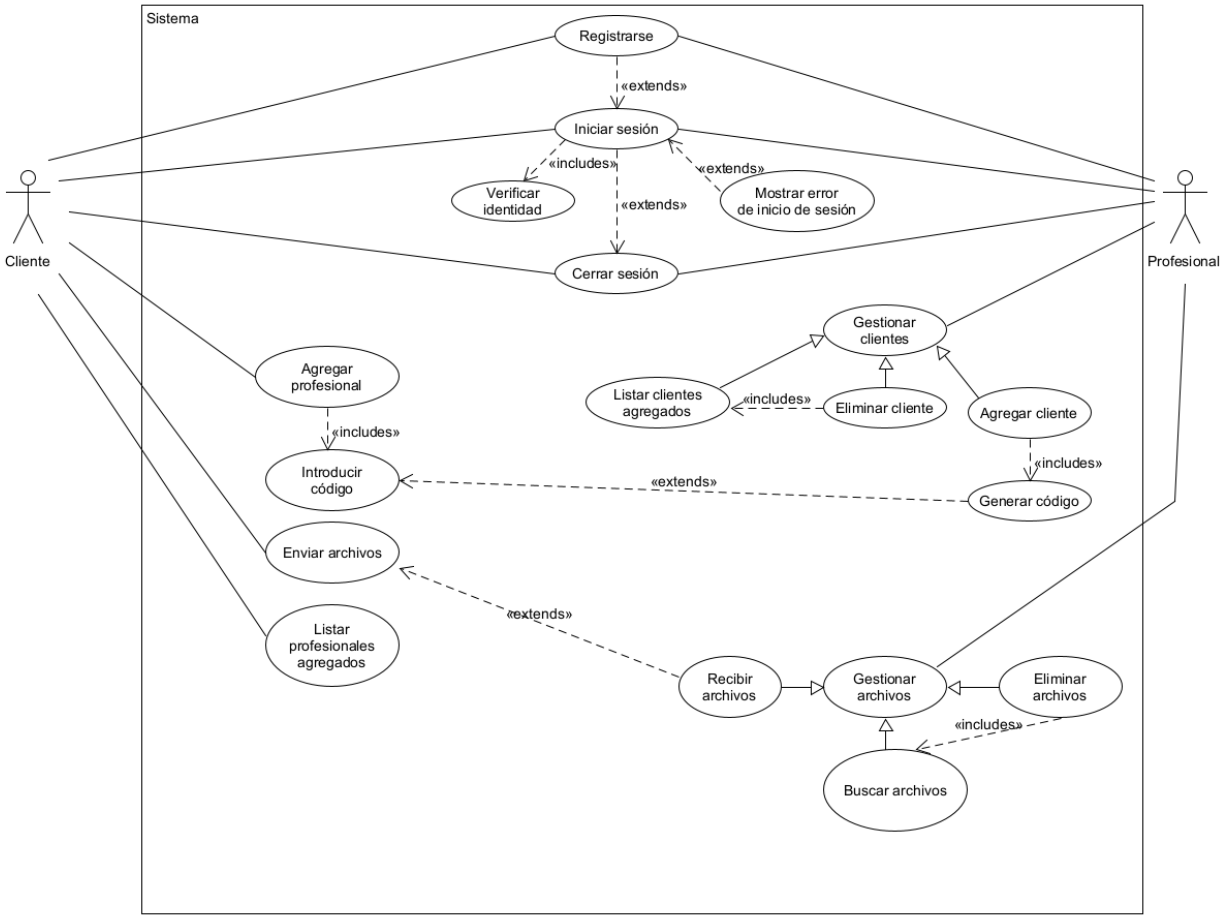


Figura 9: *Diagrama de casos de uso del proyecto*

2.2 Arquitectura del sistema

La arquitectura del sistema está formada por los siguientes componentes:

- **Dispositivos Android**. Ejecutan la aplicación Android.
- **Servidor.** Proporciona las operaciones de soporte, es decir, todas aquellas en las que no intervenga el envío del archivo. El servidor ejecuta la API REST y proporciona almacenamiento en base de datos.

Para interconectar los componentes se utiliza la arquitectura de cliente-servidor, de manera que los dispositivos Android actúan como clientes que solicitan operaciones al servidor, aunque con una particularidad. En el momento del envío del fichero, el profesional activa la conexión y actuaría como servidor al cual el dispositivo del cliente se conecta para enviar el fichero.

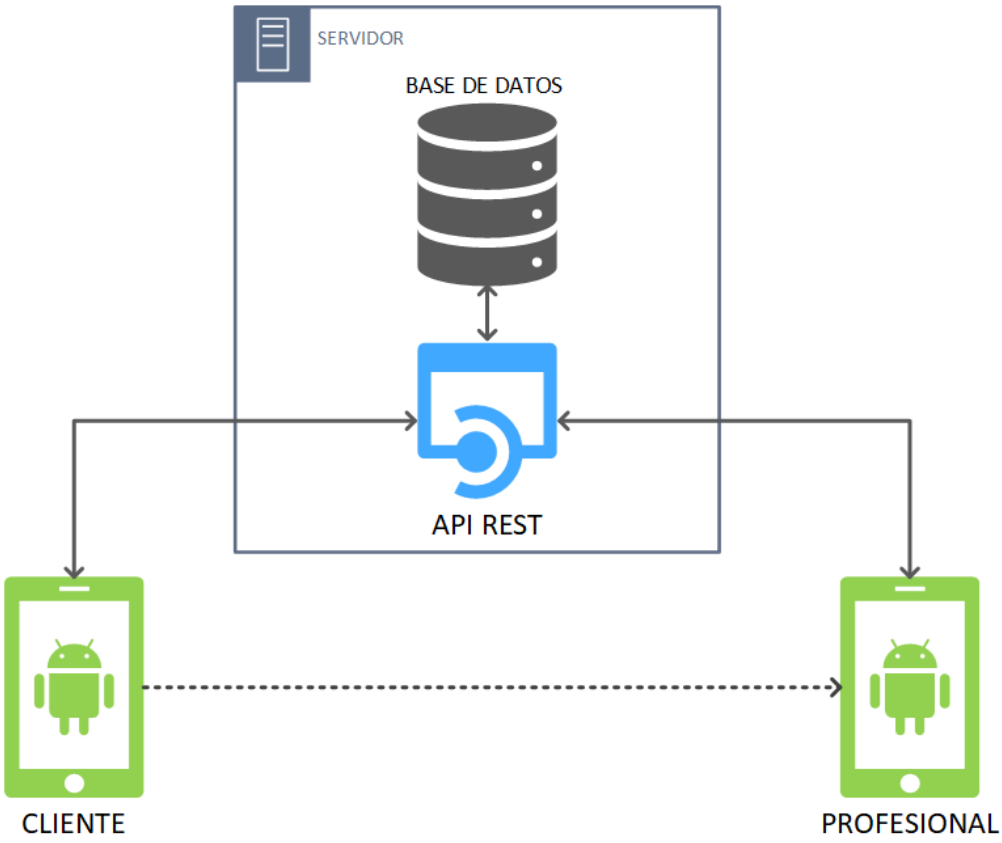


Figura 10: *Arquitectura general del sistema*

2.3 Selección de tecnologías

Una vez identificada la arquitectura general del sistema y cada uno de sus componentes, se describen las tecnologías utilizadas en cada componente del sistema.

2.3.1 Aplicación Android

El proyecto es una aplicación Android nativa. Android es un sistema operativo dirigido a móviles basado en el núcleo de Linux desarrollado por Android Inc[4]., que posteriormente fue adquirido por Google[5] [19].

La razón principal de su elección como plataforma de desarrollo es su amplio uso y su gran cuota de mercado, como se muestra en la figura 11.

[4] **Android Inc.:** Compañía de software que desarrolló Android.

[5] **Google:** Compañía de software, conocida principalmente por desarrollar el buscador Google.

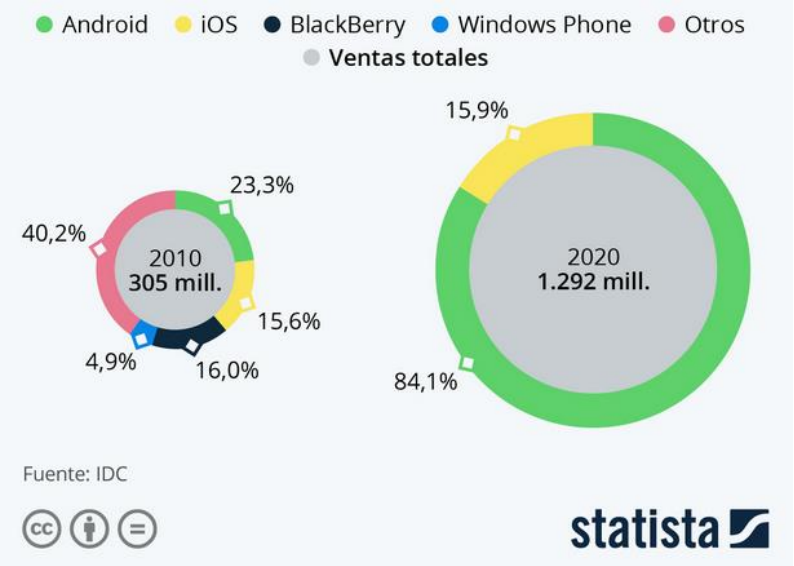


Figura 11: *Ventas totales de móviles por sistema operativo en 2020*

A la hora de afrontar el desarrollo de una aplicación Android, existen numerosas opciones para implementar cada una de las partes de la aplicación, requiriendo un estudio pormenorizado de las tecnologías disponibles. A continuación, se listan las tecnologías utilizadas para cada parte del desarrollo.

Lenguaje de programación

En el desarrollo de Android existen dos opciones principales: **Java** [20] y **Kotlin** [21]. Java es un lenguaje desarrollado por Sun Microsystems [22] en 1995. Se caracteriza por ser multiplataforma y orientado a objetos, lo que le hizo ganarse una gran popularidad que aún mantiene en la actualidad. Es uno de los lenguajes más utilizados según el índice TIOBE, el cual mide la popularidad de los lenguajes de programación [23].

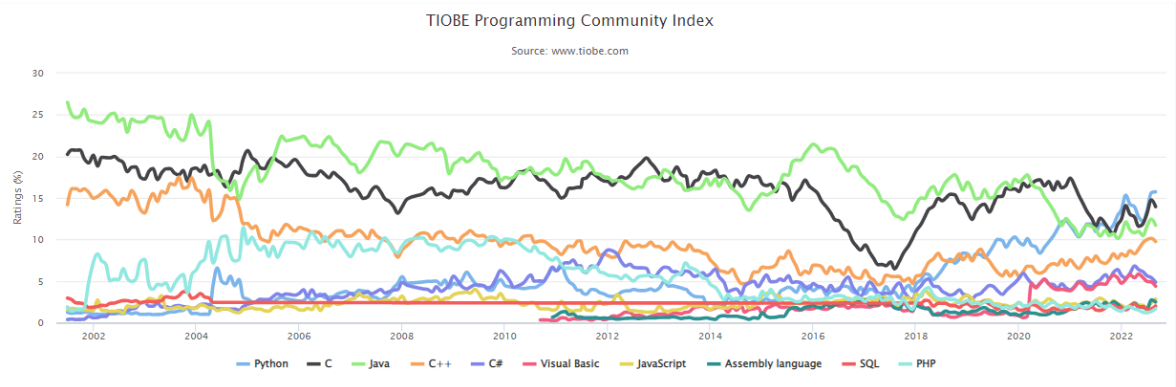


Figura 12*: Evolución de la popularidad de lenguajes de programación según el índice TIOBE*

Kotlin es un lenguaje de programación desarrollado por la compañía de software JetBrains [24] en 2011. Es un lenguaje de tipado estático que permite generar código más seguro de una manera más concisa. Es interoperable con Java y facilita trabajar con código asíncrono gracias a las *corrutinas* [6] [25]. Google recomienda el uso de Kotlin para el desarrollo de aplicaciones Android [26].

Figura 13: *Logo de Kotlin*

Figura 14: *Logo de Java*

La aplicación Android se comenzó desarrollando en Java, pero el resultado obtenido no fue el esperado debido a una serie de complicaciones derivados de la incorrecta implementación de la arquitectura seleccionada. Aunque el primer prototipo era plenamente funcional, era complicado corregir errores y ampliar funcionalidad. Por lo tanto, se decidió reiniciar el desarrollo y se comenzó a implementar un segundo prototipo, esta vez usando Kotlin como lenguaje principal.

La elección de Kotlin se debe a dos factores principales: es el lenguaje que prioriza Google en el desarrollo Android, y las distintas características del lenguaje facilitan y agilizan el desarrollo, permitiendo además implementar tanto la lógica como la interfaz gráfica de la aplicación en Kotlin.

Nivel de API

Para el desarrollo de la aplicación se ha elegido el nivel de **API 21**, que corresponde a la versión de **Android 5.0** [27]. La razón principal es poder ofrecer la aplicación al máximo número de usuarios posibles, ya que puede abarcar muchas áreas y distintos tipos de usuarios.

Se han tenido en cuenta las compatibilidades de las librerías utilizadas en el proyecto, las cuales no requieren un nivel de API superior para funcionar correctamente. Esto es posible gracias a la librería **AndroidX** [28], ya que incluye librerías de compatibilidad con versiones anteriores de Android.

[6] **Corrutinas:** Mecanismo de Kotlin para la administración de tareas en segundo plano.

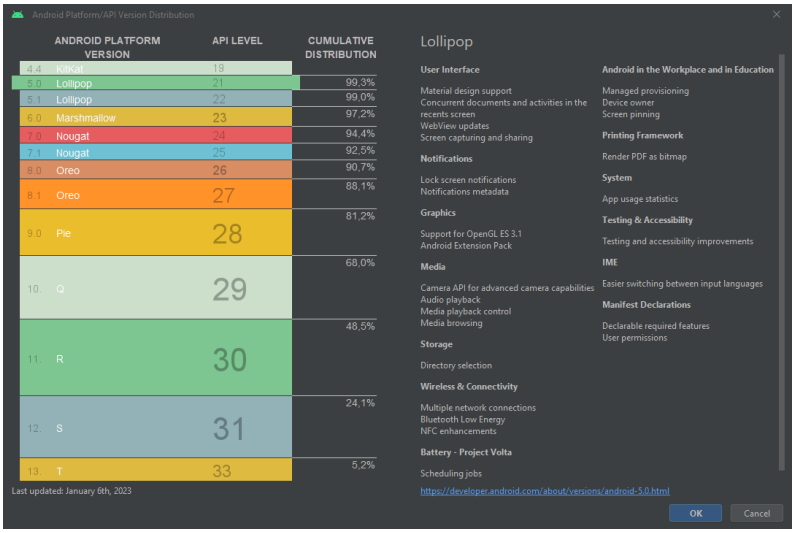


Figura 15: *Porcentaje de distribución de las distintas versiones de Android*

Librerías

A continuación, se listan las principales librerías que se han utilizado para desarrollar la aplicación Android.

Tabla 2: *Librerías utilizadas en la aplicación Android*

Librería	Descripción
AndroidX [28]	Incluye librerías de compatibilidad, componentes de arquitectura y herramientas que facilitan el desarrollo de aplicaciones Android.
ViewModel [29]	Librería incluida en AndroidX. Permite exponer el estado de la interfaz de usuario y encapsular la lógica de negocio [29].
LiveData [30]	Librería incluida en AndroidX. Permite actualizar la interfaz de usuario en base a los cambios en los datos que está representando.
JetPack Compose [31]	Librería incluida en AndroidX. Permite crear la interfaz gráfica de forma declarativa haciendo uso de Kotlin.
Corrutinas [32]	Librería incluida en AndroidX. Facilita la ejecución de código asíncrono.
Retrofit [33]	Simplifica y facilita las peticiones HTTP, gestionando las respuestas de forma asíncrona.
Gson [34]	Permite convertir objetos en representaciones en JSON.

java.net [35]	Librería de Java para las comunicaciones en red. Incluye los sockets que se han utilizado para establecer la comunicación directa entre los móviles.

Entorno de desarrollo

Android Studio [36] es un IDE[7] dirigido al desarrollo de aplicaciones para Android, basado en IntelliJ IDEA[8]. Ofrece un potente editor de código y todas las herramientas necesarias para poner a disposición de los desarrolladores un entorno unificado donde desarrollar para todos los dispositivos Android.

La elección de este IDE es debido a que es el más indicado para desarrollar aplicaciones Android. Ofrece una serie de herramientas que han sido muy útiles durante el desarrollo del prototipo de la aplicación, como:

- Emuladores de dispositivos Android para probar el funcionamiento de la aplicación.
- Integración con programas de gestión de versiones.
- Herramientas que facilitan la creación de la interfaz gráfica de la aplicación.
- Automatización de la construcción del proyecto.

Figura 16: *Logo de Android Studio*

2.3.2 API REST

Para conectar los terminales Android con el servidor para que puedan intercambiar información se ha implementado una **API REST**.

Una API REST es una interfaz de programación de aplicaciones que utiliza el protocolo HTTP para solicitar datos y operaciones sobre los mismos, en diferentes formatos [37].

Existen dos formatos principales para intercambiar información entre el cliente y el servidor: **XML**[9] y **JSON**[10]. XML es un lenguaje de marcas similar a HTML, cuyo propósito principal es compartir datos a

[7] **IDE:** Entorno de desarrollo integrado. Ofrece un entorno que facilita la programación y la construcción de proyectos.
[8] **IntelliJ IDEA:** IDE de Java y Kotlin desarrollado por JetBrains.
[9] **XML:** Extensible Markup Language (Lenguaje de Marcado Extensible)
[10] **JSON:** JavaScript Object Notation (Notación de Objetos de JavaScirpt)

través de distintos sistemas, como internet [38]. JSON es un formato basado en texto para representar datos estructurados haciendo uso de la sintaxis de objetos de JavaScript [11]. Se suele utilizar para transmitir datos en aplicaciones web [39].

El formato elegido fue JSON, ya que el proceso de **serialización**[12] y **deserialización**[13] es más sencillo por tener una sintaxis simplificada y una mayor facilidad para usarlo en Android, teniendo librerías como gson desarrolladas por la propia Google.

La elección de la API REST para la integración entre el cliente y servidor estaba clara desde el principio, ya que, además de ser la opción más usada [40], es la tecnología con la que más experiencia se tenía.

Framework y lenguaje de programación

Para el desarrollo de la API REST se ha elegido Spring Boot, que se basa en Spring framework [41], un *framework* de Java para el desarrollo de aplicaciones web. Spring Boot [42] es la manera más popular y rápida de comenzar con un proyecto Spring, proporcionando la configuración automática del proyecto, incluyendo un servidor web embebido, entre otras herramientas.

Se eligió desde un principio por ser el *framework* con el que más experiencia se tenía para desarrollar aplicaciones web en el lado servidor y por compartir el mismo lenguaje en el lado cliente, ya que el primer prototipo de la aplicación Android se hizo en Java.

Figura 17: *Logo de Spring*

Entorno de desarrollo

El IDE seleccionado ha sido IntelliJ IDEA, ya que es el mismo en el que se basa Android Studio, agilizando la forma de trabajar al conocer las herramientas, los atajos y la interfaz del IDE. Además, es el IDE líder para el desarrollo en los lenguajes Java y Kotlin [24].

[11] **JavaScript:** Lenguaje de programación usado principalmente en programación web.
[12] **Serialización (JSON):** Proceso por el cual se convierte un objeto en una cadena de texto.
[13] **Deserialización (JSON):** Es el proceso inverso a la serialización. Se convierte la cadena de texto en un objeto.

Figura 18: *Logo de IntelliJ IDEA*

Servidor donde se ejecuta la API

Para la ejecución de la API REST se utiliza el servidor Apache Tomcat [43] que viene con Spring Boot. Apache Tomcat es un contenedor de aplicaciones web de Java, que se utiliza como servidor web [44].

Figura 19: *Logo de Apache Tomcat*

2.3.3 Base de datos

En este apartado se describen las tecnologías utilizadas para la persistencia de datos del sistema.

Sistema gestor de base de datos y lenguaje de consulta

Para la elección del sistema gestor de base de datos se ha elegido uno de tipo relacional, debido a su popularidad y escalabilidad. Concretamente se ha elegido el sistema gestor **MariaDB**, el cual es de código abierto y se puede utilizar para datos de transacciones de alta disponibilidad [45].

El lenguaje de consulta que se utiliza es **SQL**[14], ya que es el que utiliza MariaDB. Es un lenguaje utilizado para el almacenamiento, la manipulación y la recuperación de información en bases de datos relacionales. SQL se basa en el algebra y cálculo relacional, ofreciendo una gran versatilidad a la hora de realizar las consultas [46].

[14] **SQL:** *Structured Query Language* (Lenguaje de consulta estructurada).

Figura 20: *Logo de MariaDB*

Herramienta de administración de bases de datos

Para conectarse y administrar la base de datos existen múltiples clientes SQL. Se ha seleccionado DBeaver, el cual es multiplataforma y soporta múltiples gestores de bases de datos como MySQL, PostgreSQL, Oracle, etc. [47]

Figura 21*: Logo de DBeaver*

2.3.4 Otras tecnologías

En este apartado se listan las tecnologías que no están directamente relacionadas con la implementación del sistema, pero que han ayudado a gestionar y seguir un control del proyecto.

Sistema de control de versiones

Para mantener un control del proceso del desarrollo se ha utilizado **git** [48] como sistema de control de versiones. Git es un sistema de control de versiones distribuido y de software libre, diseñado para registrar los cambios y mantener un histórico de los proyectos.

Figura 22: *Logo de git*

Cliente git

El cliente Git utilizado es **Source Tree** [49], que ofrece numerosas herramientas y una interfaz gráfica para la gestión de repositorios git. Una opción muy útil que ofrece es la automatización del flujo de trabajo **Gitflow** [50], una manera de trabajar con varias ramas destinadas a corregir errores,

implementar funcionalidad y hacer despliegues en producción.

Figura 23*: Logo de Sourcetree*

Sistema de gestión del proyecto

Para hacer un seguimiento de las distintas funcionalidades a implementar y errores a corregir durante la implementación de la aplicación, se ha utilizado las herramientas de ***Issues***[15] y ***Projects***[16] de **Github**, plataforma donde también se aloja el proyecto.

Figura 24: *Logo de Github*

Pruebas de la API REST

Para comprobar que la API REST funcionaba correctamente a medida que se implementaba, se han utilizado las herramientas para hacer pruebas de **Postman** [51].

Figura 25: *Logo de postman*

[15] ***Issues*** **de Github:** Son tareas por realizar en el proyecto.

[16] ***Projects*** **de Github:** Permite organizar las tareas del proyecto.

Capítulo 3. Aplicación Android

En este capítulo se ofrece una descripción detalla de la arquitectura de la aplicación, enumerando y definiendo las partes más importantes que la componen, para luego ilustrar la interfaz de usuario y cuáles son las operaciones que tienen a su disposición según el rol cliente y profesional.

3.1 Arquitectura del sistema

A la hora de comenzar un proyecto es importante elegir una arquitectura que defina como se organiza y relaciona el código, de manera que sea robusto, mantenible, extensible y reusable. La arquitectura elegida para la aplicación Android es la **arquitectura CLEAN** [52].

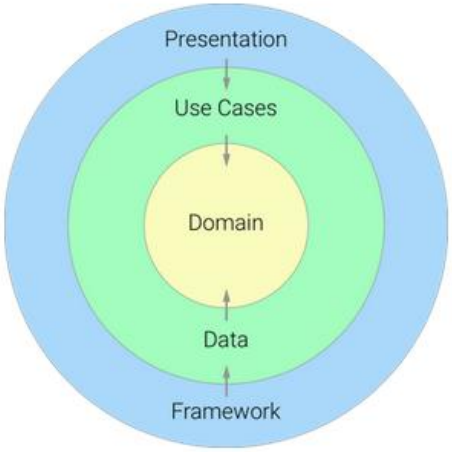


Figura 26: Diagrama de la arquitectura CLEAN *[53]*

Por ello, basándose en la arquitectura CLEAN se ha dividido la aplicación en tres paquetes principales:

- ***Data*.** Define las fuentes de datos remotas y locales.
- ***Domain*.** Define los objetos de negocio e implementa la lógica de negocio.
- ***Presentation*.** Implementa la interfaz gráfica con la que interactúa el usuario.

Al separarse la interfaz gráfica y los datos, hay que comunicarlos de alguna forma, para que la interfaz gráfica pueda representar los cambios en los datos correctamente. Para ello se ha utilizado el patrón de diseño **Model-View-ViewModel** (MVVM) [54] para conectar la vista y el modelo.

En la capa de datos se ha utilizado el patrón repositorio [55], que facilita utilizar distintas fuentes de datos, tanto locales como remotas.

Como se comentó en la introducción en el apartado de selección de tecnologías, se hizo un primer prototipo donde se intentó implementar sin mucho éxito la arquitectura. Aunque se dividió el código en paquetes y había cierta organización, a la hora de conectar las distintas capas surgían los problemas, sobre todo por no usar patrones de diseño.

En el segundo prototipo se puede apreciar una estructura más lógica, pasando de cuatro paquetes que representan las capas de la arquitectura a tres. En el primer prototipo la lógica de negocio estaba en el paquete *usecases,* en cambio, en el nuevo está dentro del paquete *domain*, siendo más lógico ya que esta capa es la que contiene el dominio del negocio.

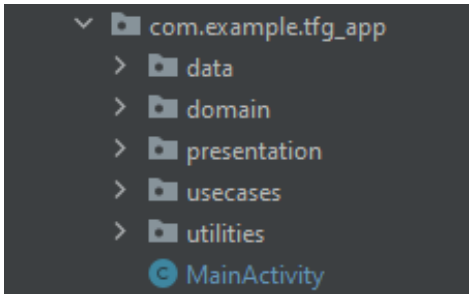


Figura 27: *Estructura del primer prototipo*

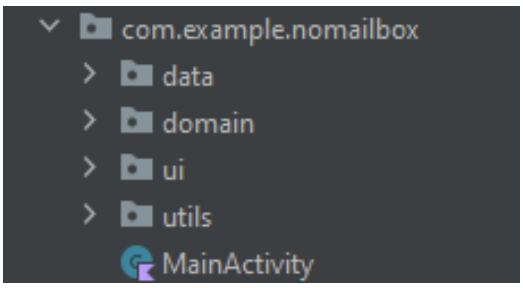


Figura 28: *Estructura del segundo prototipo*

Existe otro paquete denominado *utilities* que contiene utilidades, funcionalidades que no entran en ninguna capa de la arquitectura, pero que son imprescindibles para el correcto funcionamiento de la aplicación.

A continuación, se muestra como se ha pasado de una estructura menos organizada a una donde cada capa se organiza en paquetes cada uno con una responsabilidad específica. Esto ha ayudado a desarrollar nuevas funcionalidades y corregir errores de manera más rápida, ya que en cada momento se sabe dónde está y cómo se relaciona el código de la aplicación.

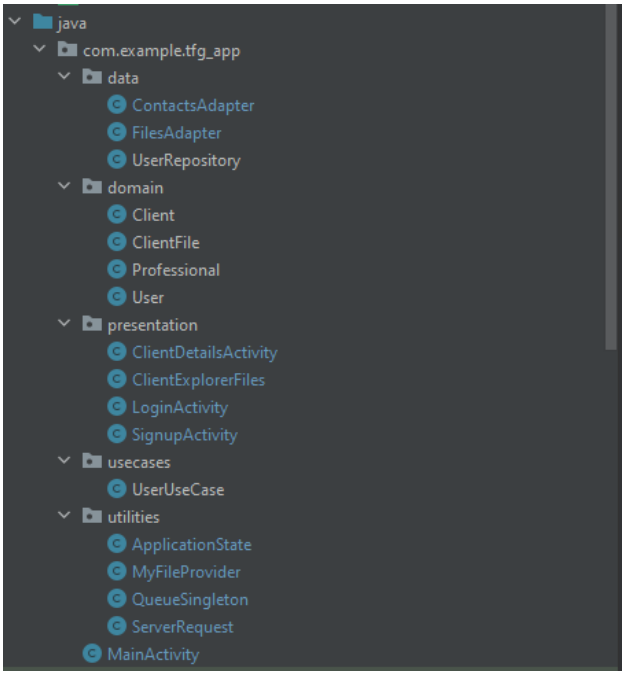


Figura 29: *Detalle de la estructura del primer prototipo*

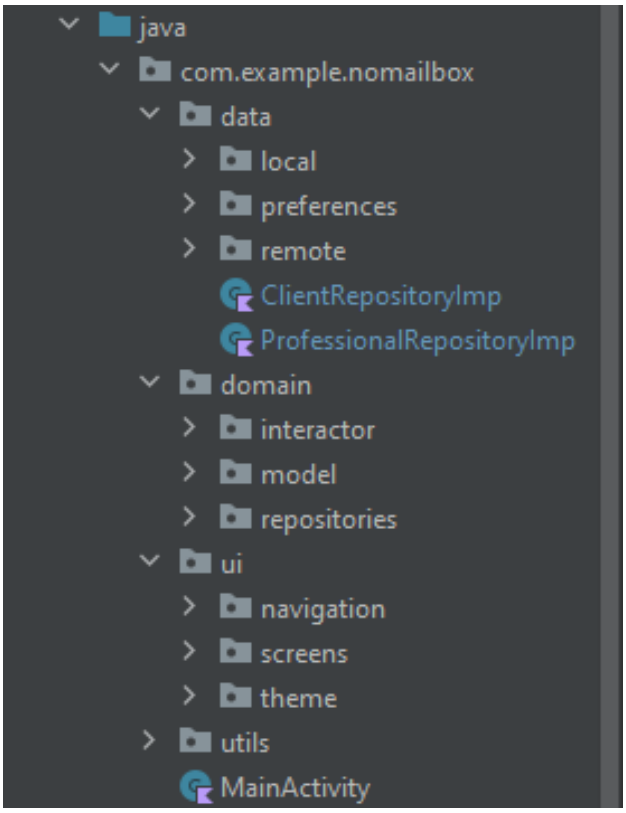


Figura 30: *Detalle de la estructura del segundo prototipo*

3.2 Funcionalidades

3.2.1 Pantalla de bienvenida

La pantalla de bienvenida es la primera pantalla que se muestra al iniciar la aplicación. Desde esta pantalla se puede seleccionar el rol con el que se desea continuar: cliente o profesional.

Figura 31: *Pantalla de bienvenida*

3.2.2 Registro

El usuario puede registrarse con el rol seleccionado en la pantalla de registro.

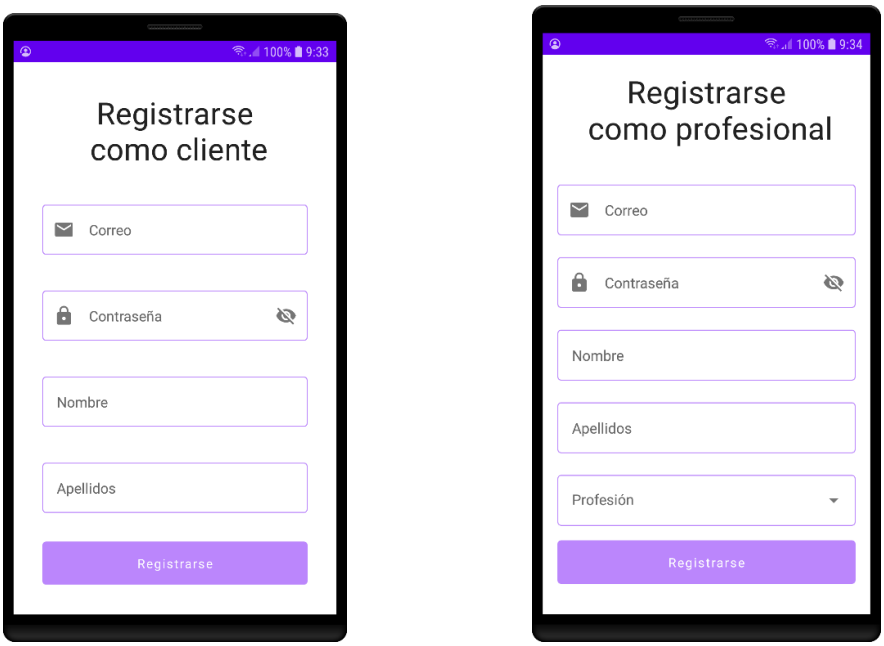


Figura 32: *Pantalla de registro del cliente y profesional*

3.2.3 Inicio de sesión

Una vez registrado el cliente o profesional, puede introducir su dirección de correo y su contraseña para iniciar sesión.

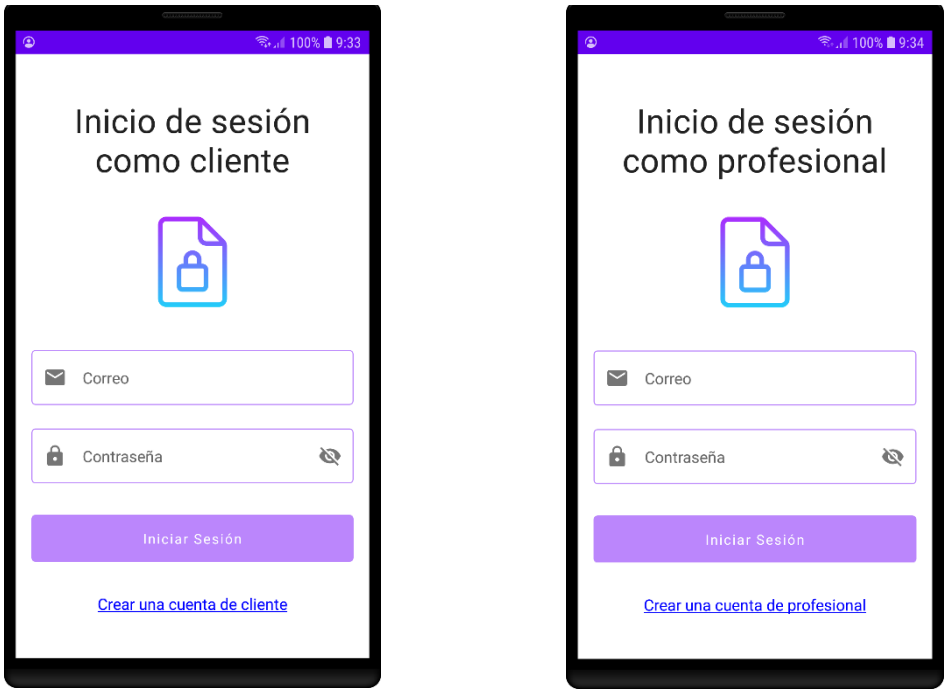


Figura 33: *Pantalla de inicio de sesión cliente y profesional*

3.2.4 Navegación del profesional

Lo primero que ve el profesional al iniciar sesión es la lista de contactos.

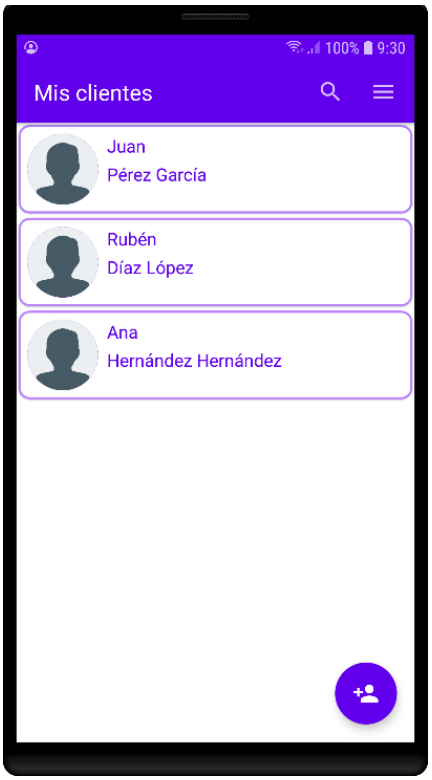


Figura 34: *Lista de contactos del profesional*

Figura 35: *Código generado por el profesional*

El profesional puede añadir un cliente nuevo pulsando el botón de añadir contacto ubicado en la parte inferior derecha. Luego aparecerá un modal con el código generado, el cual le hará llegar al cliente. Una vez el cliente introduzca el código, se agregará a la lista de contactos.

Por último, el profesional puede seleccionar un cliente para navegar a una pantalla donde se muestran detalles del cliente junto con los ficheros enviados por el mismo. Desde la lista de ficheros el profesional puede seleccionar uno y abrirlo. También puede activar o desactivar la recepción de ficheros.

Figura 36: *Pantalla de detalle del cliente*

3.2.5 Navegación del cliente

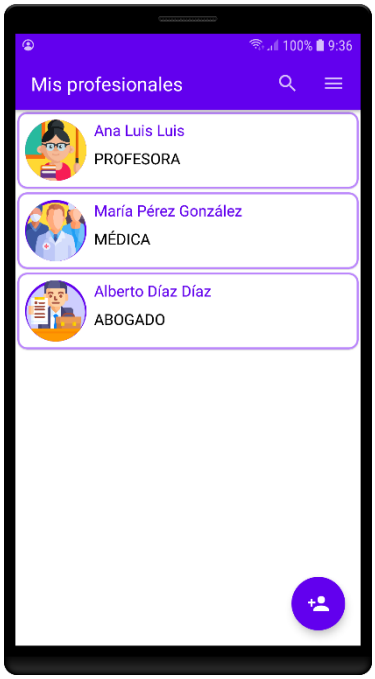


Figura 37: *Pantalla de contactos del cliente*

Al igual que los profesionales, lo primero que ven los clientes al iniciar sesión son sus contactos. Los contactos se muestran como una lista de profesionales pudiendo seleccionar uno para mostrar la pantalla de detalle del profesional, desde donde podrá enviarle un archivo.

En la pantalla de detalle del profesional el cliente puede ver los archivos que le ha enviado al profesional, si el envío está disponible y puede enviarle un archivo. Al pulsar el botón “Enviar archivo” se abre el explorador de archivos del dispositivo para seleccionar el archivo que se quiere enviar.

Figura 38: Pantalla de detalle contacto del cliente

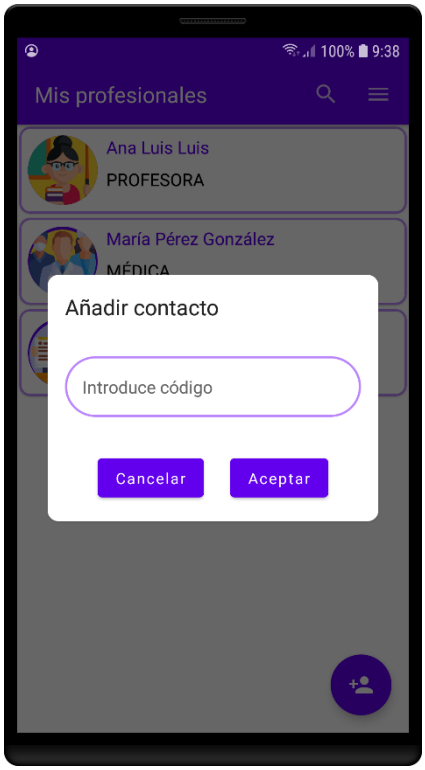


Figura 39: *Envío de código para añadir profesional*

Por último, el cliente puede añadir un profesional siempre y cuando éste le proporcione el código para añadir clientes. Para ello se utilizar el botón de añadir contacto que se encuentra en la esquina inferior derecha en la pantalla de lista de contactos.

Capítulo 4. API REST

La API REST es el componente que permite comunicar el cliente con el servidor. A continuación, se describirá la arquitectura y los *endpoints*[17] creados para la API REST.

4.1 Arquitectura

La arquitectura de la API REST está dividida en las siguientes cuatro capas:

- **Modelo.** Clases que representan las entidades. Mapean tablas de la base de datos.
- **Repositorio.** Clases encargadas de realizar peticiones a las bases de datos y mapear los resultados a entidades.
- **Service.** Clases que implementan la lógica de negocio.
- **Controlador**. Clases que definen los *endpoints* que gestionan las peticiones HTTP que se realizan al servidor.

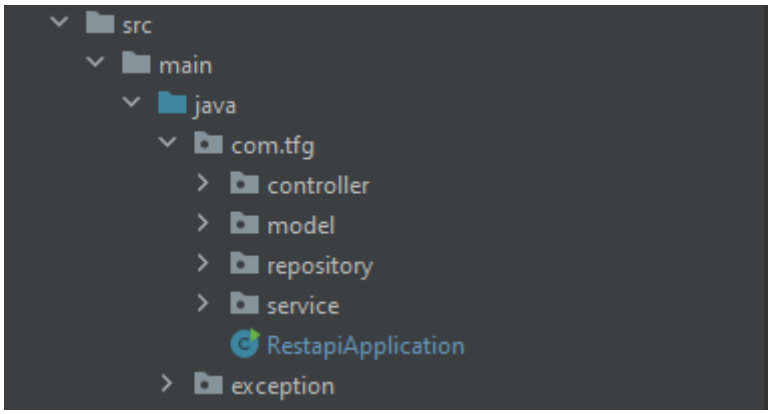


Figura 40: *Estructura de ficheros de la API REST*

En la figura 40 también se puede observar cómo hay un paquete que no se ha comentado hasta ahora: *exception*. Este paquete está dedicado a controlar los errores que surgen durante la ejecución de las peticiones HTTP, para poder informar al usuario en la aplicación que pudo haber salido mal si ocurriera un error durante la petición.

[17] **Endpoint:** dirección a la que se envían las peticiones del cliente.

Cada uno de estos paquetes contienen las clases que se muestran a continuación:

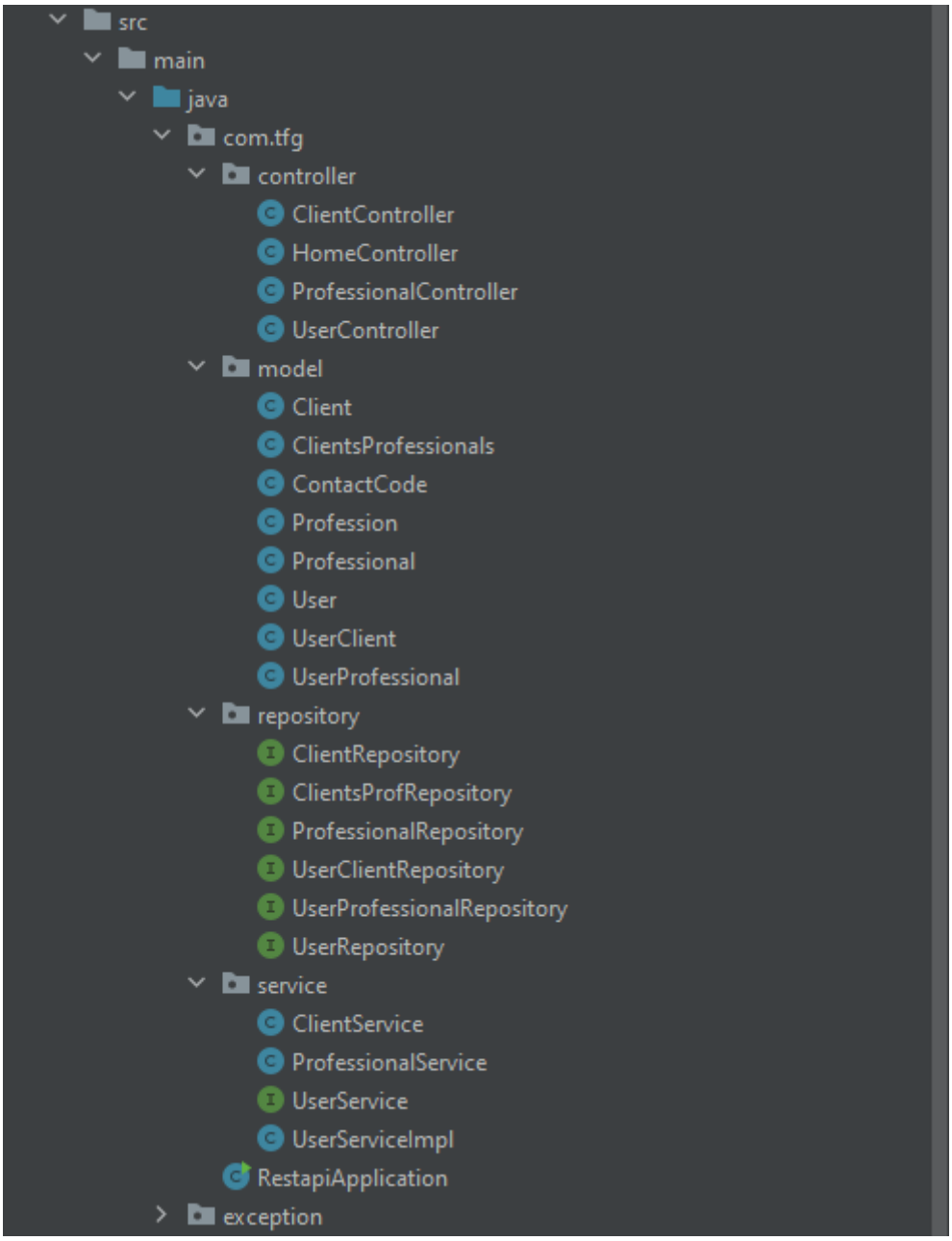


Figura 41: *Clases que componen la API REST*

4.2 Endpoints de la API REST

En este subapartado se definen las distintas direcciones que gestionan las solicitudes HTTP que se realizan al servidor:

Tabla 3: *endpoints de la API REST*

Endpoint	**Método**	**Descripción**
/user/signup/client	POST	Crea un usuario con rol cliente
/user/signup/profesional	POST	Crea un usuario con rol profesional
/user/login/client	POST	Autentica al usuario como cliente.
/user/login/profesional	POST	Autentica al usuario como profesional.
/professional/clients?professionalId={id-profesional}	POST	Devuelve los clientes de un profesional.
/client/professionals? clientId={id-cliente}	GET	Devuelve los profesionales de un cliente.
/professional/generatecode?profid={id-profesional}	GET	Devuelve el código para añadir un cliente.
/client/getcode?code={código}& clientid={id-cliente}	GET	Devuelve si el código del cliente se ha introducido correctamente o no.

/professional/enableconnection	POST	Habilita la conexión del profesional para poder recibir ficheros de un cliente en concreto. Requiere la dirección IP y el puerto como parámetros.
/professional/closeconnection? *ClientId={clientId}&ProfId={profId}*	DELETE	Cierra la conexión para deshabilitar recibir ficheros de un cliente determinado.
/client/profconnectioninfo? *ClientId={clientId}&ProfId={profId}*	GET	Devuelve la IP y el puerto de un profesional en concreto.

Capítulo 5. Base de datos

Para la base de datos se ha realizado el diseño y se ha creado la base de datos y las tablas. En los siguientes apartados se muestra el diseño en forma de diagrama entidad/relación y las tablas creadas.

5.1 Diseño

El esquema de la base de datos consta de cinco tablas:

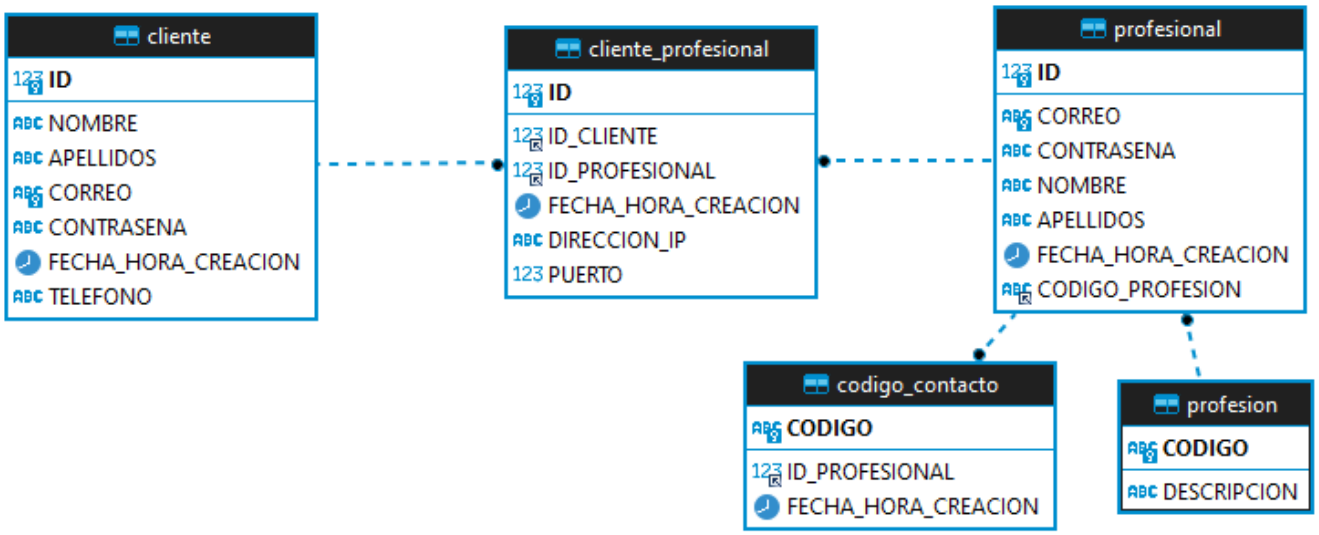


Figura 42: *Diagrama Entidad/Relación de la base de datos*

5.2 Creación de la base de datos

A continuación, se muestran las tablas creadas para el proyecto.

Table Name	Engine	Auto Increment	Data Length	Description
cliente	InnoDB	12	32K	
cliente_profesio...	InnoDB	22	16K	
codigo_contacto	InnoDB	0	16K	
profesion	InnoDB	0	16K	
profesional	InnoDB	12	32K	

Figura 43: *Estructura de la base de datos*

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
ID	1	bigint(20)	[v]	[v]	PRI		auto_increment
NOMBRE	2	varchar(100)	[v]	[ ]			
APELLIDOS	3	varchar(100)	[v]	[ ]			
CORREO	4	varchar(100)	[v]	[ ]	UNI		
CONTRASENA	5	varchar(100)	[v]	[ ]			
FECHA_HORA_CREACION	6	datetime	[v]	[ ]		current_timest...	
TELEFONO	7	varchar(15)	[ ]	[ ]		NULL	

Figura 44:*Estructura de la tabla cliente*

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
ID	1	bigint(20)	[v]	[v]	PRI		auto_increment
CORREO	2	varchar(100)	[v]	[ ]	UNI		
CONTRASENA	3	varchar(100)	[v]	[ ]			
NOMBRE	4	varchar(100)	[v]	[ ]			
APELLIDOS	5	varchar(100)	[v]	[ ]			
FECHA_HORA_CREACION	6	datetime	[v]	[ ]		current_timest...	
CODIGO_PROFESION	7	varchar(5)	[v]	[ ]	MUL		

Figura 45: *Estructura de la tabla profesional*

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default	Extra
ID	1	bigint(20)	[v]	[v]	PRI		auto_increment
ID_CLIENTE	2	bigint(20)	[v]	[ ]	MUL		
ID_PROFESIONAL	3	bigint(20)	[v]	[ ]	MUL		
FECHA_HORA_CREACION	4	datetime	[ ]	[ ]		current_timest...	
DIRECCION_IP	5	varchar(15)	[ ]	[ ]		NULL	
PUERTO	6	int(10) unsigned	[ ]	[ ]		NULL	

Figura 46: *Estructura de la tabla cliente_profesional*

Column Name	#	Data Type	Not Null	Auto Increment	Key	Default
ID_PROFESIONAL	1	bigint(20)	[v]	[ ]	MUL	
CODIGO	2	varchar(10)	[v]	[ ]	PRI	
FECHA_HORA_CREACION	3	datetime	[v]	[ ]		current_timest...

Figura 47: *Estructura de la tabla codigo_contacto*

Column Name	#	Data Type	Not Null	Auto Increment	Key
CODIGO	1	varchar(5)	[v]	[ ]	PRI
DESCRIPCION	2	varchar(100)	[v]	[ ]	

Figura 48: *Estrutura de la tabla profesión*

Capítulo 6. Conexión punto a punto

A continuación, se explicará cómo los tres componentes del sistema, la aplicación Android, el servidor y la base de datos, interactúan entre sí para conseguir enviar un fichero de forma directa entre los terminales móviles. Para ello se siguen los pasos que se describen a continuación:

1. El profesional activa la conexión y solicita a la API REST almacenar su dirección IP y puerto en la base de datos (1). El servidor confirma al profesional que su dirección IP y puerto se han guardado en la base de datos (2). Ahora el profesional actúa como servidor, ejecutando un socket servidor con la dirección IP y puerto anteriormente especificado.

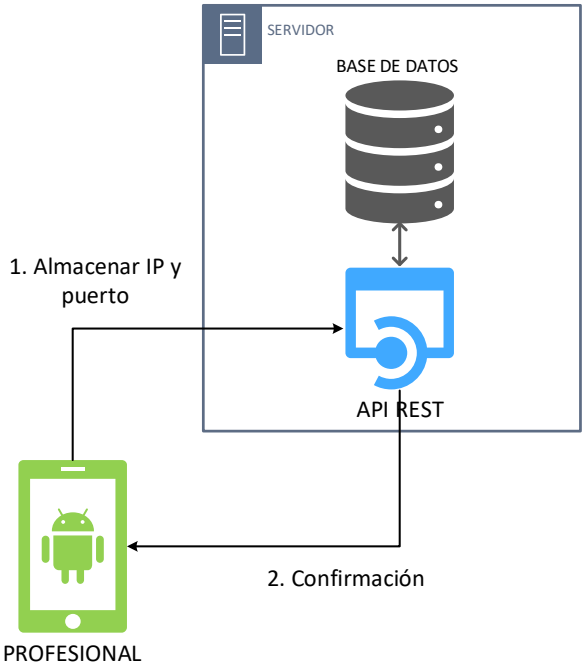


Figura 49: *Activación del envío de archivos*

2. El cliente solicita a la API REST la dirección IP y el puerto del profesional al que quiere enviar el fichero (1). El servidor le contesta con la dirección IP y el puerto del profesional solicitado (2).

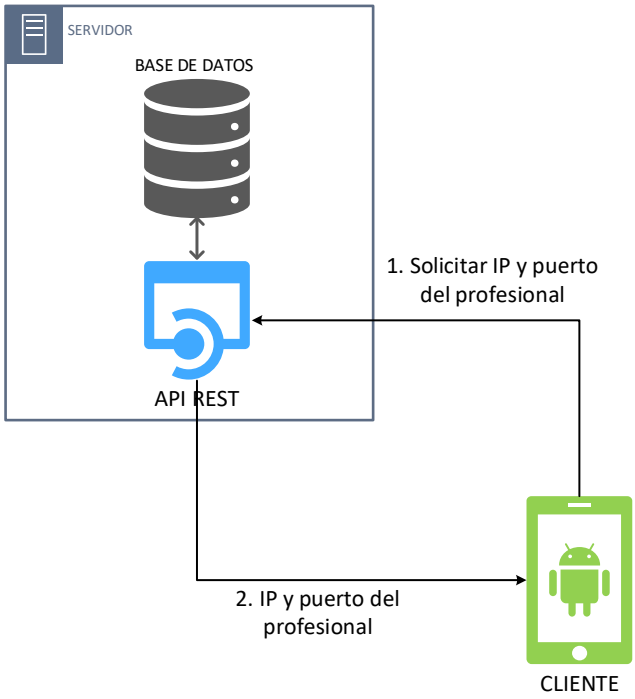


Figura 50: *Solicitud de la IP y el puerto del profesional*

3. El cliente se conecta a través de un socket con el profesional y le envía el fichero. Una vez recibido el fichero, el profesional cierra la conexión.

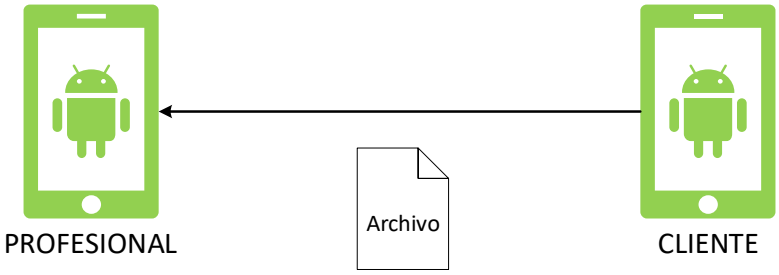


Figura 51: *Envío de archivo*

4. El archivo se almacena en el profesional en la propia jerarquía de ficheros de la aplicación, en una carpeta llamada *files*. La carpeta *files* contiene, a su vez, carpetas cuyo nombre es el identificador del cliente. En estas carpetas es donde se almacenan los archivos de cada cliente para saber a quién pertenecen.

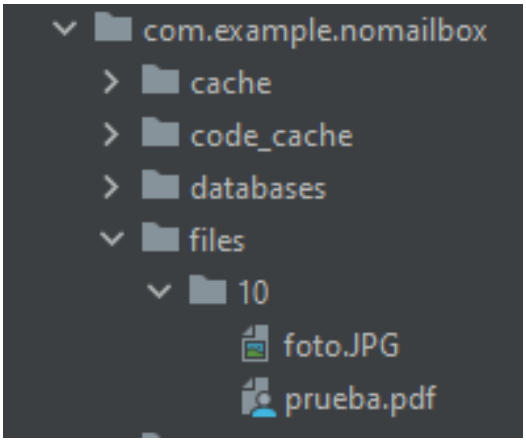


Figura 52: *Ruta del profesional donde se almacenan los archivos*

5. El profesional solicita a la API REST eliminar su IP y puerto de la base de datos (1), para evitar que el cliente pueda enviarle más archivos.

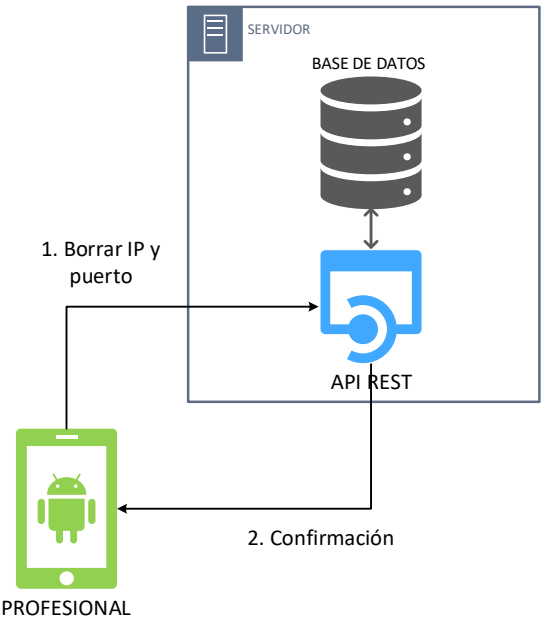


Figura 53: *Borrado de la IP y puerto del profesional de la base de datos*

A continuación, se muestran las implementaciones para enviar y recibir ficheros haciendo uso de los sockets del cliente y del servidor.

```
fun send(clientId: Int, file: File, profIp: String, profPort: Int) {
    GlobalScope.launch { this: CoroutineScope
        withContext(Dispatchers.IO) { this: CoroutineScope
            val fileInputStream = FileInputStream(file)

            val fileBuffer = ByteArray( size: 4 * 1024)
            var bytesRead: Int

            val clientSocket = Socket(profIp, profPort)
            val dataOutputStream = DataOutputStream(clientSocket.outputStream)

            dataOutputStream.writeInt(clientId)
            dataOutputStream.writeLong(file.length())
            dataOutputStream.writeUTF(file.name)


            while(fileInputStream.read(fileBuffer).also { bytesRead = it} != -1) {
                dataOutputStream.write(fileBuffer,  off: 0, bytesRead)
                dataOutputStream.flush()
            }

            fileInputStream.close()
            dataOutputStream.close()
            clientSocket.close()
        }
    }
```

Figura 54: *Implementación de envío de ficheros mediante sockets*

Como se puede observar en el código, antes de enviar el fichero se envía quien es el cliente propietario de éste a través de su identificador en la base de datos, luego el tamaño del fichero para que el socket del servidor pueda saber cuántos bytes tiene que leer. Después se envía el nombre del fichero para que el profesional pueda identificarlo correctamente. Por último, se envía el fichero.

```
fun receive() {
    while (true) {
        val client: Socket = serverSocket.accept()
        var bytesRead = 0

        val dataInputStream = DataInputStream(client.getInputStream())
        val clientId = dataInputStream.readInt().toLong()
        val clientDir: File = File( pathname: context.applicationInfo.dataDir + "/files/" + clientId)
        if (!clientDir.exists()) {
            clientDir.mkdir()
        }
        var size = dataInputStream.readLong()
        val fileName = dataInputStream.readUTF()
        val file =
            File( pathname: context.applicationInfo.dataDir + "/files/" + clientId + "/" + fileName)
        val fileOutputStream = FileOutputStream(file)
        val buffer = ByteArray( size: 4 * 1024)
        while (size > 0 && dataInputStream.read(
                buffer, off: 0,
                Math.min(buffer.size.toLong(), size).toInt()
            ).also { it: Int
                bytesRead = it
            } != -1
        ) {
            fileOutputStream.write(buffer, off: 0, bytesRead)
            size -= bytesRead.toLong()
        }
        fileOutputStream.close()
        client.close()
        serverSocket.close()
    }
}
```

Figura 55: *Implementación de recepción de ficheros mediante sockets*

En este caso el código implementa el socket del lado servidor para la recepción de ficheros. Primero espera a que se conecte el cliente. Una vez conectado empieza a leer los bytes que le llegan desde el socket del cliente. Primero lee el identificador del cliente, para saber a quién pertenece el fichero. Esto es importante ya que según a quien pertenezca el fichero se guarda en la carpeta que corresponda con el id del cliente. Una vez lee el identificador, hace lo propio con el nombre del fichero para crearlo con el nombre correcto. Por último, lee el fichero y lo almacena.

El prototipo diseñado e implementado permite enviar el fichero directamente al profesional, pero tiene una importante limitación: **solo funciona si ambos están conectados a la misma red.** A la hora de intentar conectar cliente y profesional desde distintas redes se encontraron numerosos desafíos, ya que

exponer el dispositivo profesional a internet para que pueda ser localizado por el cliente es una tarea compleja debido a la dependencia de configuraciones externas, como el *firewall* [18]del *router* o la tecnología NAT[19].

Aun siendo posible hacer funcionar este sistema fuera de la red local, no es para nada recomendable que un dispositivo Android actúe como servidor expuesto a internet, principalmente por cuestiones de seguridad y rendimiento.

Para hacer funcionar un sistema de este tipo sin que uno de los dispositivos móviles actúe como un servidor, se requeriría estudiar una solución en la cual ambos dispositivos se conecten directamente a través de una VPN. Esta solución presenta una serie de complejidades como la configuración de la VPN y de certificados digitales para que la conexión sea privada.

Otra limitación del prototipo es la **ausencia de cifrado del fichero**. Se estudiaron distintas opciones como los **SSLSocket** [56], pero requerían de configuración e instalación de certificados, complicando su implementación. Por último, se valoraron opciones como **cifrado híbrido**, haciendo uso tanto de cifrado simétrico como asimétrico, pero no se consiguió que los ficheros se enviaran correctamente, quedando el apartado de la seguridad en el terreno de la investigación y las pruebas.

[18] **Firewall:** Cortafuegos. Mecanismo de seguridad del router para filtrar los paquetes entrantes y salientes.
[19] **NAT:** Network Address Translation (Traducción de direcciones de red). Permite la traducción de direcciones IP públicas a privadas y viceversa.

Capítulo 7. Estudio de viabilidad económica

En este capítulo se hará un estudio desde un punto de vista profesional sobre la viabilidad del proyecto, para hacer una estimación sobre si su capacidad de generar beneficios es mayor a la inversión económica necesaria para llevarlo a cabo.

7.1 Estudio de mercado

Para saber si el tipo de aplicación que se desarrolla tiene una demanda suficiente en la actualidad, se hace un estudio de mercado, listando las aplicaciones similares disponibles, la popularidad que tienen y el valor que aporta la nuestra aplicación respecto al resto.

En el capítulo dos: estudio previo, en el subapartado estado del arte, se listaron una serie de aplicaciones similares a las del proyecto. De estas aplicaciones se han seleccionados las más descargadas en la tienda digital **Google Play**[20] y se muestra el número de descargas que tienen en la actualidad para tener una referencia a la hora de estimar la popularidad de NoMailbox.

Tabla 4: *número de descargas de aplicaciones similares a NoMailbox*

Aplicación	Descargas
Sync – Secure cloud storage [57]	Mas de 1.000.000
ToffeShare: File Sharing [58]	Más de 100.000
Tresorit [59]	Más de 100.000

En este apartado se hace un estudio de mercado muy simplificado y superficial, pero suficiente para hacer las estimaciones requeridas para el estudio de viabilidad económica. En un entorno profesional se requiere de un equipo de expertos que realicen un estudio completo de mercado, teniendo en cuenta tendencias, identificación de la competencia, realización de encuestas y análisis de los resultados obtenidos.

[20] Tienda digital del sistema operativo Android de distribución de aplicaciones móviles

7.2 Estimación de costes

Para estimar los costes se hace uso de un **diagrama de Gantt**, que es una herramienta de gestión de proyecto para la planificación y programación de tareas. Para su creación se ha utilizado el programa **Microsoft Project** [60].

Las tareas del proyecto se han agrupado en cuatro fases. La fase de análisis, para saber que se quiere implementar; la fase de diseño, para saber cómo implementarlo; la fase de desarrollo, donde comienza la creación de cada uno de los componentes del sistema; y, por último, la fase de paso a producción, donde se realizan las pruebas y el despliegue de la aplicación final.

A continuación, se muestra una tabla a modo de resumen con los recursos humanos necesarios para el proyecto, con su función y sueldo en horas. Para los sueldos se ha consultado la página talent.com[21] [61], obteniéndose la media de sueldo en horas para cada puesto.

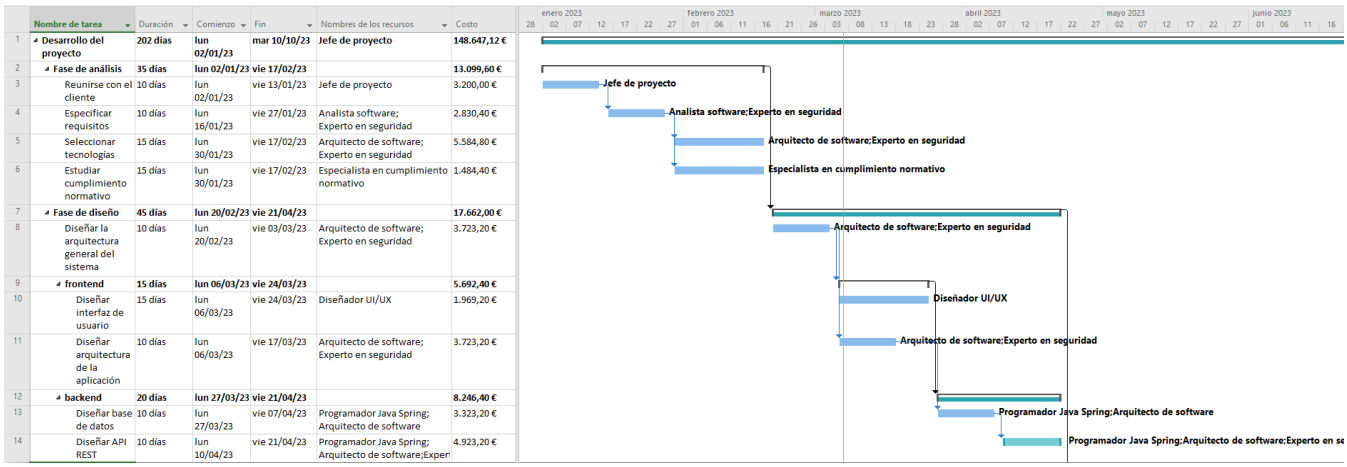
Tabla 5: *Sueldo medio en horas de los puestos en España en 2023*

Puesto	Función	Salario en horas
Jefe de proyecto	Supervisar el desarrollo del proyecto.	24,36 €/h
Analista	Definir requisitos de negocio, características del software y elaboración de documentación.	15,38 €/h
Arquitecto software	Diseñar las arquitecturas del sistema, decidir que tecnologías van a usarse, patrones de diseño y estándares de codificación.	26,54 €/h
Experto en ciberseguridad	Diseñar y garantizar la seguridad del sistema.	20,00€/h
Programador Android	Creación de la aplicación Android.	18,72 €/h
Programador Java Spring	Creación de la API REST y la base de datos.	15,38 €/h

[21] Página para la búsqueda de empleo y consulta de sueldos, entre otras características.

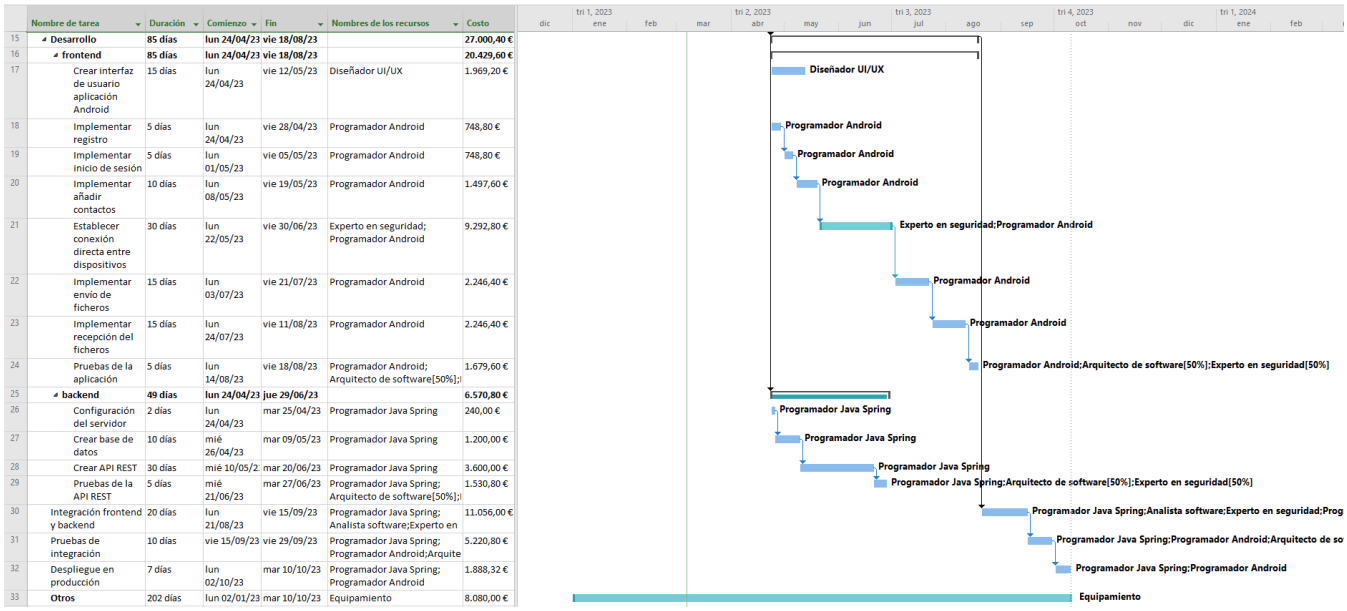
Diseñador UI/UX	Diseñar y crear la interfaz de usuario de la aplicación Android.	16,41 €/h
Especialista en cumplimiento normativo	Garantizar que el software cumpla con el reglamento general de protección de datos de la Unión Europea.	12,31 €/h

Una vez definidas las tareas y los recursos humanos necesarios se crea el diagrama de Gantt. Como se puede comprobar, el proyecto requiere de un tiempo de desarrollo de **237 días laborables,** con un gasto asociado de **148.647,12€**. La jornada laboral definida ha sido de 8 horas de lunes a viernes.



	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos	Costo
1	**Desarrollo del proyecto**	**202 días**	**lun 02/01/23**	**mar 10/10/23**	**Jefe de proyecto**	**148.647,12 €**
2	**Fase de análisis**	**35 días**	**lun 02/01/23**	**vie 17/02/23**		**13.099,60 €**
3	Reunirse con el cliente	10 días	lun 02/01/23	vie 13/01/23	Jefe de proyecto	3.200,00 €
4	Especificar requisitos	10 días	lun 16/01/23	vie 27/01/23	Analista software; Experto en seguridad	2.830,40 €
5	Seleccionar tecnologías	15 días	lun 30/01/23	vie 17/02/23	Arquitecto de software; Experto en seguridad	5.584,80 €
6	Estudiar cumplimiento normativo	15 días	lun 30/01/23	vie 17/02/23	Especialista en cumplimiento normativo	1.484,40 €
7	**Fase de diseño**	**45 días**	**lun 20/02/23**	**vie 21/04/23**		**17.662,00 €**
8	Diseñar la arquitectura general del sistema	10 días	lun 20/02/23	vie 03/03/23	Arquitecto de software; Experto en seguridad	3.723,20 €
9	**frontend**	**15 días**	**lun 06/03/23**	**vie 24/03/23**		**5.692,40 €**
10	Diseñar interfaz de usuario	15 días	lun 06/03/23	vie 24/03/23	Diseñador UI/UX	1.969,20 €
11	Diseñar arquitectura de la aplicación	10 días	lun 06/03/23	vie 17/03/23	Arquitecto de software; Experto en seguridad	3.723,20 €
12	**backend**	**20 días**	**lun 27/03/23**	**vie 21/04/23**		**8.246,40 €**
13	Diseñar base de datos	10 días	lun 27/03/23	vie 07/04/23	Programador Java Spring; Arquitecto de software	3.323,20 €
14	Diseñar API REST	10 días	lun 10/04/23	vie 21/04/23	Programador Java Spring; Arquitecto de software;Experi	4.923,20 €

Figura 56: *Primera parte del diagrama de Gantt*



	Nombre de tarea	Duración	Comienzo	Fin	Nombres de los recursos	Costo
15	**Desarrollo**	**85 días**	**lun 24/04/23**	**vie 18/08/23**		**27.000,40 €**
16	**frontend**	**85 días**	**lun 24/04/23**	**vie 18/08/23**		**20.429,60 €**
17	Crear interfaz de usuario aplicación Android	15 días	lun 24/04/23	vie 12/05/23	Diseñador UI/UX	1.969,20 €
18	Implementar registro	5 días	lun 24/04/23	vie 28/04/23	Programador Android	748,80 €
19	Implementar inicio de sesión	5 días	lun 01/05/23	vie 05/05/23	Programador Android	748,80 €
20	Implementar añadir contactos	10 días	lun 08/05/23	vie 19/05/23	Programador Android	1.497,60 €
21	Establecer conexión directa entre dispositivos	30 días	lun 22/05/23	vie 30/06/23	Experto en seguridad; Programador Android	9.292,80 €
22	Implementar envío de ficheros	15 días	lun 03/07/23	vie 21/07/23	Programador Android	2.246,40 €
23	Implementar recepción del ficheros	15 días	lun 24/07/23	vie 11/08/23	Programador Android	2.246,40 €
24	Pruebas de la aplicación	5 días	lun 14/08/23	vie 18/08/23	Programador Android; Arquitecto de software[50%];I	1.679,60 €
25	**backend**	**49 días**	**lun 24/04/23**	**jue 29/06/23**		**6.570,80 €**
26	Configuración del servidor	2 días	lun 24/04/23	mar 25/04/23	Programador Java Spring	240,00 €
27	Crear base de datos	10 días	mié 26/04/23	mar 09/05/23	Programador Java Spring	1.200,00 €
28	Crear API REST	30 días	mié 10/05/2	mar 20/06/23	Programador Java Spring	3.600,00 €
29	Pruebas de la API REST	5 días	mié 21/06/23	mar 27/06/23	Programador Java Spring; Arquitecto de software[50%];I	1.530,80 €
30	Integración frontend y backend	20 días	lun 21/08/23	vie 15/09/23	Programador Java Spring; Analista software;Experto en	11.056,00 €
31	Pruebas de integración	10 días	vie 15/09/23	vie 29/09/23	Programador Java Spring; Programador Android;Arquite	5.220,80 €
32	Despliegue en producción	7 días	lun 02/10/23	mar 10/10/23	Programador Java Spring; Programador Android	1.888,32 €
33	**Otros**	202 días	lun 02/01/23	mar 10/10/23	Equipamiento	8.080,00 €

Figura 57: *Segunda parte del diagrama de Gantt*

Por último, se muestra de manera gráfica los gastos acumulados por trimestre y en la totalidad del proyecto una vez finalizado, para ver de una manera visual la evolución de los gatos y la inversión total necesaria para crearlo.

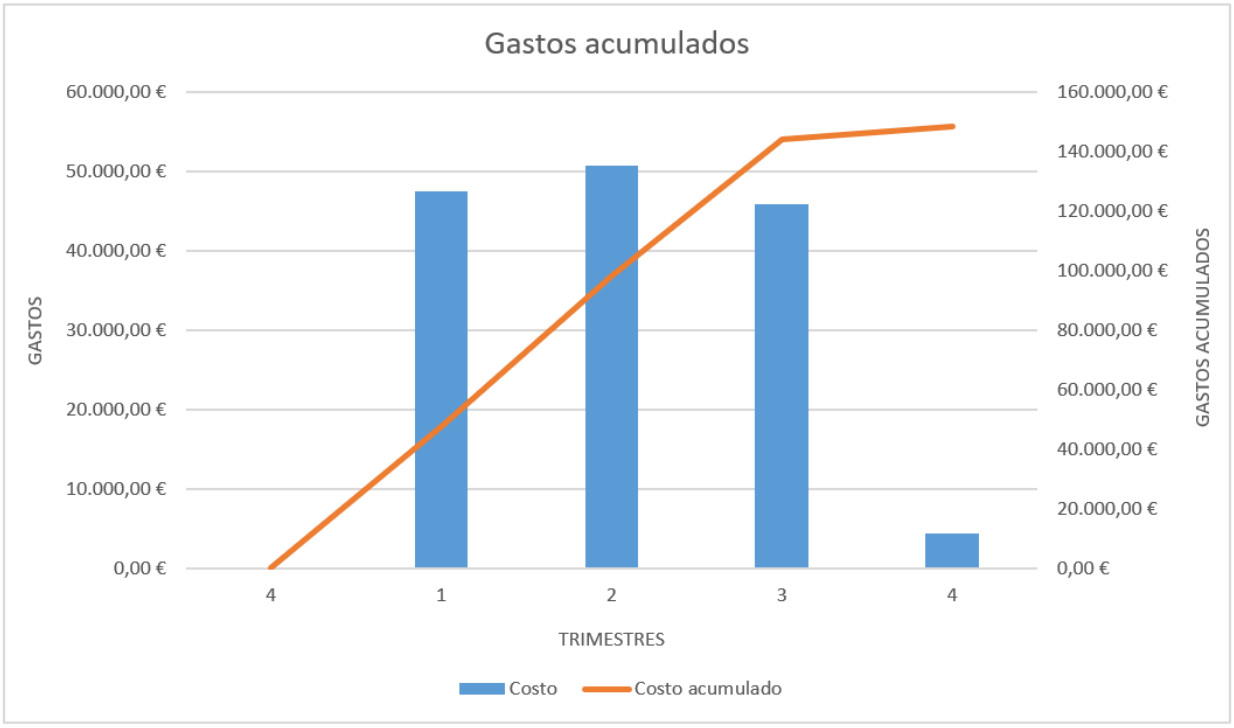


Figura 58: *Gastos acumulados del proyecto*

7.3 Estimación de ingresos

Debido a la propia naturaleza de la aplicación, donde la privacidad es el objetivo principal, hay que estudiar modelos de comercialización que no impliquen el uso de datos del usuario para generar ingresos.

Se ha decantado por el modelo de negocio *freemium*[22], basado en ofrecer una serie de funcionalidades gratuitas y limitadas, mientras que las más avanzadas y sin limitaciones requieren un pago previo.

Los clientes solo tendrán una versión de la aplicación sin limitaciones. Los profesionales tendrán dos versiones: una gratuita y otra *premium* con todas las opciones disponibles tras un pago de 10€.

Figura 59: *Planes disponibles para la aplicación*

En la siguiente tabla se muestra una estimación de las compras del plan *premium* durante un periodo de 24 meses con los ingresos y gastos acumulados. Esta estimación se ha realizado para obtener los datos que se utilizan para determinar en qué punto se espera recuperar la inversión hecha en el desarrollo y mantenimiento de la aplicación, o lo que es lo mismo se utilizan para calcular el ROI.

Tabla 6: *Estimación de ingresos*

Mes	Planes gratuitos	Planes premium	Ingresos acumulados	Gastos acumulados
1	20	0	0,00	148.647,12

[22] **Freemium:** La palabra deriva de las palabras *free* (gratis) y *premium* (de primera calidad).

2	35	6	60,00	152.647,12
3	70	30	360,00	156.647,12
4	150	100	1.360,00	160.647,12
5	300	197	3.330,00	164.647,12
6	420	100	4.330,00	168.647,12
7	500	150	5.830,00	171.647,12
8	800	500	10.830,00	175.647,12
9	900	1000	20.830,00	178.647,12
10	1000	300	23.830,00	182.147,12
11	1312	400	27.830,00	186.147,12
12	1500	560	33.430,00	186.347,12
13	1800	700	40.430,00	190.347,12
14	2002	777	48.200,00	193.847,12
15	3000	900	57.200,00	197.847,12
16	2500	1000	67.200,00	201.347,12
17	2800	1400	81.200,00	205.347,12
18	3500	2000	101.200,00	208.847,12
19	4300	2200	123.200,00	212.847,12
20	5000	3000	153.200,00	215.847,12
21	6150	3500	188.200,00	219.847,12
22	8205	2000	208.200,00	222.347,12
23	9029	1209	220.290,00	226.347,12
24	10000	1275	233.040,00	227.347,12

La tabla 5 de estimación de ingresos se ha utilizado para calcular el ROI. En la siguiente gráfica se muestra el resultado de este cálculo. Se puede comprobar que para recuperar la inversión se requiere de 32 meses desde el comienzo del desarrollo de la aplicación y de 24 desde su comercialización.

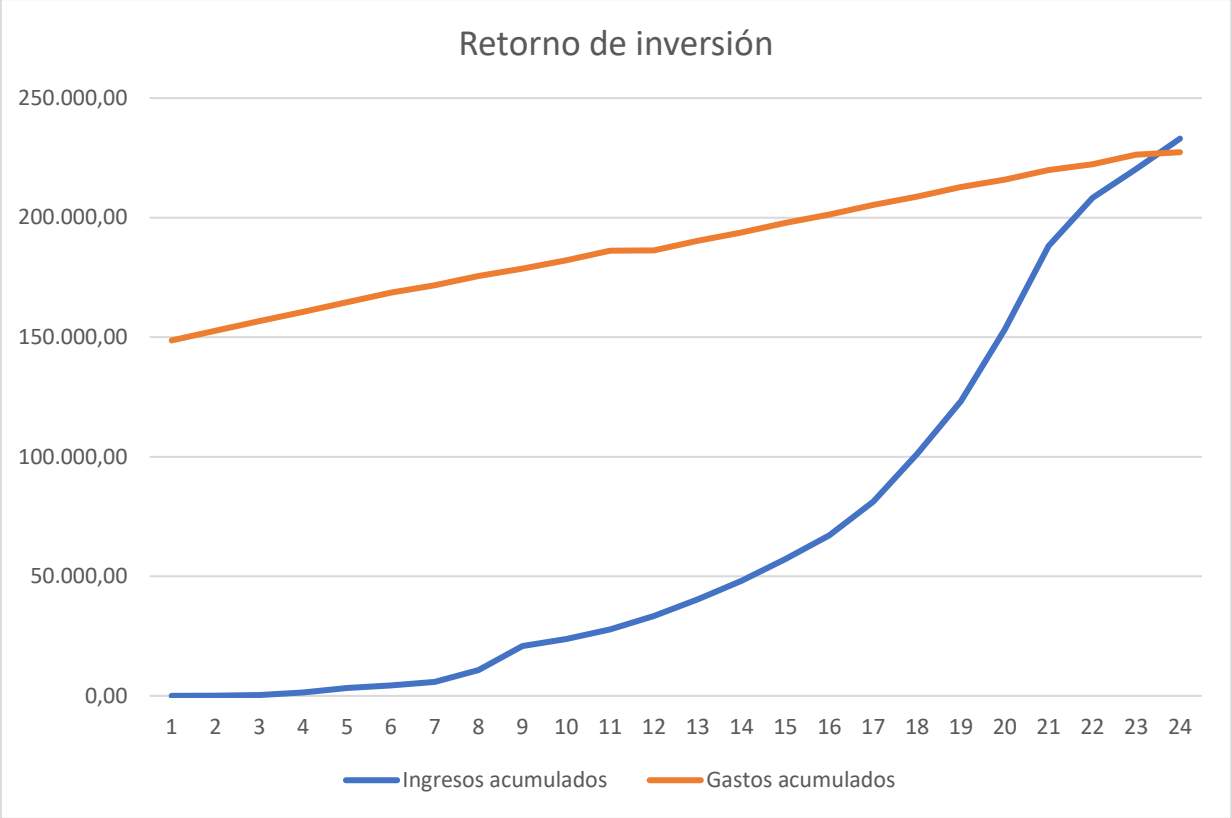


Figura 60: *Retorno de la inversión*

Capítulo 8. Conclusiones y Líneas futuras

Implementar un sistema que envíe ficheros directamente desde un dispositivo móvil a otro a través de internet, sin tener que almacenarlos en un servidor, supone un reto técnico importante. Las mayores dificultades que se han encontrado han sido intentar conectar ambos dispositivos móviles desde distintas redes e implementar la capa de seguridad necesaria para enviar los archivos de forma segura. En este aspecto el proyecto puede mejorarse aplicando medidas de seguridad como cifrado punto a punto y rediseñando el sistema para que los dispositivos Android se puedan conectar independientemente de donde se encuentren.

Aún con las limitaciones del proyecto, todo lo implementado ha servido para aprender a desarrollar una aplicación Android que se comunica con un servidor y que es capaz de enviar archivos a otro dispositivo Android en una red local. Para implementar el prototipo se ha requerido del estudio de múltiples tecnologías, arquitecturas, patrones de diseño, etc. y de la implementación de distintas aplicaciones como son la aplicación Android y la API REST, además de la base de datos, obteniendo como resultado un aprendizaje en una gran variedad de tecnologías y conceptos.

A destacar, la implementación de un segundo prototipo de la aplicación Android con un código de mejor calidad y más robusto frente a errores, reflejando la importancia de realizar un buen diseño antes de empezar a escribir una sola línea de código.

Capítulo 9. Summary and Conclusions

Implementing a system that sends files directly from one mobile device to another via the Internet, without having to store them on a server, is a significant technical challenge. The biggest difficulties that have been found were trying to connect both mobile devices from different networks and implementing the necessary security layer to send the files safely. In this point, the project can be improved by applying security measures such as selected point-to-point and redesigning the system so that Android devices can connect regardless of where they are.

Even with the limitations of the project, everything implemented has served to learn how to develop an Android application that communicates with a server and is capable of sending files to another device on a local network. To implement the prototype, the study of multiple technologies, architectures, design patterns, etc. has been required and the implementation of different applications such as the Android application and the REST API, in addition to the database, getting as a result a learning in a great variety of technologies and concepts.

It is worth noting the implementation of a second prototype of the Android application with a code of better quality and more robust against errors, reflecting the importance of carrying out a good design before starting to write a single line of code.

Capítulo 10. Presupuesto

Para finalizar, se reflejan los costes asociados al desarrollo del proyecto.

Tabla 7: *Presupuesto del proyecto*

Tareas	Horas	Coste/hora
Investigación y selección de tecnologías	70 horas	15€
Diseño de los componentes del sistema	100 horas	15€
Implementación del proyecto	120 horas	15€
Documentación	30 horas	15€
Total	320 horas	4.800€

Apéndice A

Repositorio de código

- **Aplicación Android:** https://github.com/gonherdev/TFG-android-app
- **API REST:** https://github.com/gonherdev/TFG-API-REST

Bibliografía

[1] [En línea]. Available: https://policies.google.com/privacy#infocollect. [Último acceso: 2023].

[2] «Android,» [En línea]. Available: https://www.android.com/intl/es_es/. [Último acceso: 2023].

[3] «redhat,» [En línea]. Available: https://www.redhat.com/es/topics/api/what-is-a-rest-api. [Último acceso: 2023].

[4] [En línea]. Available: https://eur-lex.europa.eu/legal-content/ES/TXT/HTML/?uri=CELEX:32016R0679&from=ES. [Último acceso: 2023].

[5] «https://www.prot-on.com/es/EstudioProt-On.pdf,» [En línea]. Available: https://www.prot-on.com/es/EstudioProt-On.pdf. [Último acceso: 2023].

[6] «precisesecurity,» [En línea]. Available: https://www.precisesecurity.com/articles/53-percent-of-web-users-more-concerned-about-online-privacy-in-2019/. [Último acceso: 2023].

[7] Bitwarden, Inc., «bitwarden,» [En línea]. Available: https://bitwarden.com/help/about-send/. [Último acceso: noviembre 2022].

[8] «sync,» [En línea]. Available: https://www.sync.com/. [Último acceso: 2023].

[9] «toffeeshare,» [En línea]. Available: https://toffeeshare.com/es/. [Último acceso: 2023].

[10] tresorit, «tresorit,» [En línea]. Available: https://send.tresorit.com/. [Último acceso: noviembre 2022].

[11] doctodoctor, «doctodoctor,» [En línea]. Available: https://doctodoctor.com/es/inicio/. [Último acceso: noviembre 2022].

[12] Comisión Europea, «commission.europa.eu,» [En línea]. Available: https://commission.europa.eu/law/law-topic/data-protection/data-protection-eu_es. [Último acceso: 2023].

[13] «eur-lex.europa.eu,» [En línea]. Available: https://eur-lex.europa.eu/legal-content/ES/TXT/HTML/?uri=CELEX:32016R0679&from=ES#d1e1954-1-1. [Último acceso: 2023].

[14] «eur-lex.europa.eu,» [En línea]. Available: https://eur-lex.europa.eu/legal-content/ES/TXT/HTML/?uri=CELEX:32016R0679&from=ES#d1e2067-1-1. [Último acceso: 2023].

[15] «eur-lex.europa.eu,» [En línea]. Available: https://eur-lex.europa.eu/legal-content/ES/TXT/HTML/?uri=CELEX:32016R0679&from=ES#d1e2245-1-1. [Último acceso: 2023].

[16] «eur-lex.europa.eu,» [En línea]. Available: https://eur-lex.europa.eu/legal-content/ES/TXT/HTML/?uri=CELEX:32016R0679&from=ES#d1e2317-1-1. [Último acceso: 2023].

[17] «eur-lex.europa.eu,» [En línea]. Available: https://eur-lex.europa.eu/legal-content/ES/TXT/HTML/?uri=CELEX:32016R0679&from=ES#d1e2576-1-1. [Último acceso: 2023].

[18] «eur-lex.europa.eu,» [En línea]. Available: https://eur-lex.europa.eu/legal-content/ES/TXT/HTML/?uri=CELEX:32016R0679&from=ES#d1e3443-1-1. [Último acceso: 2023].

[19] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Android. [Último acceso: 2023].

[20] «Java,» [En línea]. Available: https://www.java.com/en/. [Último acceso: 2023].

[21] «Kotlin,» [En línea]. Available: https://kotlinlang.org/. [Último acceso: 2023].

[22] «wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Sun_Microsystems. [Último acceso: 2023].

[23] TIOBE Software BV, «TIOBE,» TIOBE Software BV, 2022. [En línea]. Available: https://www.tiobe.com/tiobe-index/. [Último acceso: 2022].

[24] «jetbrains,» [En línea]. Available: https://www.jetbrains.com/es-es/idea/. [Último acceso: 2023].

[25] «developers.android,» [En línea]. Available: https://developer.android.com/kotlin?hl=es-419. [Último acceso: 2023].

[26] «developer.android,» [En línea]. Available: https://developer.android.com/kotlin/first?hl=es-419. [Último acceso: 2023].

[27] «android,» [En línea]. Available: https://www.android.com/intl/es_es/versions/lollipop-5-0/. [Último acceso:

2023].

[28] «android,» [En línea]. Available: https://developer.android.com/jetpack/androidx?hl=es-419. [Último acceso: 2023].

[29] «android,» [En línea]. Available: https://developer.android.com/topic/libraries/architecture/viewmodel?hl=es-419. [Último acceso: 2023].

[30] «android,» [En línea]. Available: https://developer.android.com/topic/libraries/architecture/livedata?hl=es-419. [Último acceso: 2023].

[31] «android,» [En línea]. Available: https://developer.android.com/jetpack/compose?gclid=Cj0KCQiA9YugBhCZARIsAACXxeIwCqOLYUQTfVKz1RyX3H_un7g56VjUyQOGRjRyStmgeGW363o_cs0aAqoCEALw_wcB&gclsrc=aw.ds&hl=es-419. [Último acceso: 2023].

[32] «android,» [En línea]. Available: https://developer.android.com/kotlin/coroutines?hl=es-419. [Último acceso: 2023].

[33] [En línea]. Available: https://square.github.io/retrofit/. [Último acceso: 2023].

[34] «github,» [En línea]. Available: https://github.com/google/gson. [Último acceso: 2022].

[35] «oracle,» [En línea]. Available: https://docs.oracle.com/javase/7/docs/api/java/net/package-summary.html. [Último acceso: 2023].

[36] «android,» [En línea]. Available: https://developer.android.com/studio. [Último acceso: 2023].

[37] «wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional. [Último acceso: 2023].

[38] «mozilla,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/XML/XML_introduction. [Último acceso: 2023].

[39] «mozilla,» [En línea]. Available: https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON. [Último acceso: 2023].

[40] J. Simpson, «nordicapis,» 2022. [En línea]. Available: https://nordicapis.com/20-impressive-api-economy-statistics/. [Último acceso: 2023].

[41] «spring,» [En línea]. Available: https://spring.io/. [Último acceso: 2023].

[42] spring. [En línea]. Available: https://spring.io/quickstart. [Último acceso: 2023].

[43] «apache,» [En línea]. Available: https://tomcat.apache.org/index.html. [Último acceso: 2023].

[44] «wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Tomcat. [Último acceso: 2023].

[45] «mariadb,» [En línea]. Available: https://mariadb.org/es/. [Último acceso: 2023].

[46] https://es.wikipedia.org, «Wikipedia,» 2022. [En línea]. Available: https://es.wikipedia.org. [Último acceso: 2022].

[47] «dbeaver,» [En línea]. Available: https://dbeaver.io/. [Último acceso: 2023].

[48] «git,» [En línea]. Available: https://git-scm.com/. [Último acceso: 2023].

[49] «sourcetreeapp,» [En línea]. Available: https://www.sourcetreeapp.com/. [Último acceso: 2023].

[50] «atlassian,» [En línea]. Available: https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow. [Último acceso: 2023].

[51] «postman,» [En línea]. Available: https://www.postman.com/. [Último acceso: 2023].

[52] [En línea]. Available: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html. [Último acceso: 2023].

[53] «androidcurso,» 2017. [En línea]. Available: http://www.androidcurso.com/index.php/932.

[54] [En línea]. Available: https://developer.android.com/topic/libraries/architecture/viewmodel?hl=es-419. [Último acceso: 2023].

[55] [En línea]. Available: https://developer.android.com/codelabs/basic-android-kotlin-training-repository-pattern?hl=es-419. [Último acceso: 2023].

[56] «android,» [En línea]. Available: https://developer.android.com/reference/javax/net/ssl/SSLSocket. [Último acceso: 2023].

[57] [En línea]. Available: https://play.google.com/store/apps/details?id=com.sync.mobileapp&gl=ES.

[58] [En línea]. Available: https://play.google.com/store/apps/details?id=com.toffeeshare.android&gl=ES. [Último acceso: 2023].

[59] [En línea]. Available: https://play.google.com/store/apps/details?id=com.tresorit.mobile&gl=ES. [Último acceso: 2023].

[60] [En línea]. Available: https://www.microsoft.com/es-es/microsoft-365/project/project-management-software. [Último acceso: 2023].

[61] «telant,» [En línea]. Available: https://es.talent.com/salary. [Último acceso: 2023].

[62] Google, LLC., «Google Developers,» 2022. [En línea]. Available: https://developer.android.com/kotlin/first. [Último acceso: 2022].

[63] Google, LLC., «Google Developers,» 2020. [En línea]. Available: https://developer.android.com/training/volley?hl=es-419. [Último acceso: 2022].

[64] Wikipedia, «Wikipedia,» 2022. [En línea]. Available: https://es.wikipedia.org/wiki/XAMPP. [Último acceso: 2022].

[65] «Statista,» [En línea]. Available: https://es.statista.com/estadisticas/934626/servicios-de-mensajeria-instantanea-mas-utilizados-por-los-usuarios-de-internet-en-espana/. [Último acceso: 2022].

[66] T. Klosowski, «nytimes,» [En línea]. Available: https://www.nytimes.com/wirecutter/guides/online-security-secure-messaging-apps/. [Último acceso: noviembre 2022].

[67] «eur-lex.europa.eu,» [En línea]. Available: https://eur-lex.europa.eu/legal-content/ES/TXT/HTML/?uri=CELEX:32016R0679&from=ES#d1e1954-1-1. [Último acceso: 2023].

[68] «eur-lex.europa.eu,» [En línea]. Available: https://eur-lex.europa.eu/legal-content/ES/TXT/HTML/?uri=CELEX:32016R0679&from=ES#d1e2317-1-1. [Último acceso: 2023].

[69] [En línea]. Available: https://www.android.com/intl/es_es/.

[70] M. M. Roa, «statista,» 2021. [En línea]. Available: https://es.statista.com/grafico/18920/cuota-de-mercado-mundial-de-smartphones-por-sistema-operativo/. [Último acceso: 2023].

[71] Google, LLC, «Google Developers,» 2022. [En línea]. Available: https://developer.android.com/studio/intro?hl=es. [Último acceso: 2022].