Memoria del proyecto

Aplicación para facilitar las tareas de personas mayores

Trabajo de Fin de Grado de Ingeniería Informática

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Septiembre de 2022

Autora:

Paula Soria Ullán

Tutores:

Gabriel Villarrubia González

Álvaro Lozano Murciego

André Filipe Sales Mendes

Certificado de los tutores

D. Álvaro Lozano Murciego, D. André Filipe Sales Mendes y D. Gabriel Villarrubia González, profesores del Departamento de Informática y Automática de la Universidad de Salamanca.

HACEN CONSTAR:

Que el trabajo titulado “Aplicación para facilitar las tareas de personas mayores“ ha sido realizado por Dña. Paula Soria Ullán, con DNI 70921970K y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo Fin de Grado de la Titulación Grado en Ingeniería Informática de esta Universidad.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a 5 de septiembre de 2022

LOZANO MURCIEGO ALVARO - 80098605R
Firmado digitalmente por LOZANO MURCIEGO ALVARO - 80098605R
Fecha: 2022.09.06 11:15:31 +02'00'

D. Álvaro Lozano Murciego

SALES MENDES ANDRE FILIPE - Y5601546W
Firmado digitalmente por SALES MENDES ANDRE FILIPE - Y5601546W
Fecha: 2022.09.06 09:46:39 +02'00'

D. André Filipe Sales Mendes

VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Firmado digitalmente por VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-71932845T, givenName=GABRIEL, sn=VILLARRUBIA GONZALEZ, cn=VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Fecha: 2022.09.06 10:44:33 +02'00'

D. Gabriel Villarrubia González

Resumen

Según los datos del Instituto Nacional de Estadística (INE), en el año 2018 se registró que más de 850.000 personas que viven solas en España tienen 80 años o más [1]. Estos datos muestran también que la tendencia va en aumento junto con el envejecimiento de la población.

Además, más del 76% de las personas mayores de 80 años muestran una gran preocupación por la brecha digital que existe actualmente [2], porque consideran que no existe un acompañamiento en su proceso de adaptación, al tenerse que enfrentar día a día a un mundo inmerso en las nuevas tecnologías.

Con el auge de Internet, podemos realizar tareas que nunca pensamos, y cuidar de nuestros familiares es algo que se ha hecho y se seguirá haciendo con el paso del tiempo. Es por esto que múltiples empresas buscan desarrollar sistemas orientados a facilitar la vida de las personas mayores y a acercarles de una manera sencilla e intuitiva al mundo digital.

En este trabajo se busca diseñar, desarrollar e implementar una aplicación con la que podamos seguir ocupándonos de nuestra familia estando lejos de casa, estar pendientes de sus citas médicas, de que se tomen la medicación todos los días, e incluso de hablar con ellos a diario por videollamada para que no les falte de nada, además de ofrecerles a los mayores una forma fácil de utilizar las nuevas tecnologías.

Es por esto por lo que el objetivo de la aplicación es que todos los usuarios, sobre todo lo que están menos familiarizados con las tecnologías, sean capaces de tener autonomía para desplazarse por el entorno y navegar por las diferentes pantallas fácilmente para conseguir realizar una tarea determinada de forma eficaz.

Tras la realización de este Trabajo de Fin de Grado se ha conseguido llevar a cabo una aplicación Android ejecutable tanto en dispositivos móviles como en la televisión, empleando Kotlin, con la que poder ayudar a nuestros familiares a realizar tareas del día a día mediante el uso de alertas y poder comunicarnos con ellos de una forma sencilla.

Palabras clave: Brecha digital, Android, Kotlin, videollamadas, alertas, personas mayores.

Abstract

According to data from the National Statistics Institute (INE), in the year 2018 it was recorded that more than 850,000 people living alone in Spain are aged 80 or older [1]. These data also show that the trend is increasing along with the ageing of the population.

In addition, more than 76% of people over the age of 80 show great concern about the digital divide that currently exists [2], because they consider that there is no support in their adaptation process, as they have to face a world immersed in new technologies on a daily basis.

With the rise of the Internet, we can perform tasks that we never thought possible, and caring for our relatives is something that has been done and will continue to be done over time. This is why many companies are looking to develop systems aimed at making life easier for the elderly and bringing them closer to the digital world in a simple and intuitive way.

The aim of this work is to design, develop and implement an application with which we can continue to take care of our family while we are away from home, keep an eye on their medical appointments, make sure they take their medication every day, and even talk to them every day by videocall so that they don't miss anything, as well as offering the elderly an easy way to use new technologies.

This is why the aim of the application is that all users, especially those who are less familiar with technologies, are able to have the autonomy to move around the environment and navigate through the different screens easily in order to carry out a given task efficiently.

After completing this Final Degree Project, we have managed to create an Android application that can be run both on mobile devices and on television, using Kotlin, with which we can help our relatives to carry out day-to-day tasks through the use of alerts and communicate with them in a simple way.

Key words: Digital divide, Android, Kotlin, videocalls, alerts, elderly.

Índice

Índice de figuras

Índice de tablas

1. Introducción

En la actualidad, la tecnología ha tomado un peso importante en la sociedad y facilita enormemente las tareas del día a día. Sin embargo, las personas mayores tienen que enfrentarse a las barreras tecnológicas que supone un mundo que se está digitalizando cada vez más.

Existe un porcentaje muy alto de la población mayor de 80 años en España que vive sola [1], que son los más afectados por esta brecha digital, y es una tendencia cada vez mayor. Este grupo de personas se pueden aprovechar de los beneficios que nos aporta el uso de Internet en sus tareas diarias, al igual que su entorno, como mediante el uso de tecnologías como altavoces inteligentes que permiten una comunicación eficiente, pulseras de actividad con las que prevenir accidentes y realizar un diagnóstico rápido del estado de la salud, robots aspiradores para el mantenimiento del hogar, relojes con detector de caídas diseñados para personas con dependencia, etc.

Este proyecto tiene como objetivo permitir a los usuarios ayudar a sus familiares mayores a realizar tareas para las que normalmente necesitarían ayuda, pudiendo realizar un seguimiento de ellas, además de acercar a las personas mayores al uso de las nuevas tecnologías de una forma sencilla e intuitiva.

Este documento tiene como finalidad explicar los aspectos más importantes del desarrollo del sistema y está estructurado en los siguientes apartados:

- **Análisis del mercado:** Se muestran aplicaciones disponibles actualmente en el mercado que pertenecen al mismo camp de desarrollo.
- **Objetivos:** Se especifican los objetivos que se pretender lograr con la realización del proyecto.
- **Conceptos teóricos:** Se desarrollan los conceptos teóricos necesarios para entender de forma correcta el proyecto.
- **Técnicas y herramientas:** Se documentan las técnicas y herramientas utilizadas durante el desarrollo del proyecto.
- **Aspectos relevantes del desarrollo:** Se recogen los aspectos más importantes del desarrollo y despliegue del sistema.
- **Conclusiones y líneas de trabajo futuras:** Se comentan conclusiones obtenidas tras finalizar el proyecto y las líneas de trabajo futuras o ampliaciones que existen sobre la aplicación.
- **Referencias:** Se enumeran las citas empleadas durante el desarrollo del proyecto.

Además, en este documento se incluye también el resto de la documentación técnica que está compuesta por los siguientes anexos:

- **Anexo I - Plan de Proyecto**: Presenta una estimación de la duración del proyecto en función de las tareas a realizar.
- **Anexo II - Especificación de Requisitos Software**: Documenta la captura y especificación de requisitos del sistema.
- **Anexo III - Análisis del Requisitos Software**: Recoge el análisis, refinamiento y estructuración de los requisitos del sistema.
- **Anexo IV - Diseño del Sistema Software**: Documenta el proceso de diseño del sistema.
- **Anexo V - Documentación Técnica**: Recoge la documentación técnica para facilitarle al lector la comprensión del código desarrollado.
- **Anexo VI - Manual de Usuario**: Realiza una guía de la funcionalidad del sistema para facilitarle al usuario la interacción con el sistema.

2. Análisis del mercado

Antes de comenzar a desarrollar la aplicación se realizó un estudio o análisis del mercado, donde se consultaron diversas aplicaciones que trabajan en el mismo campo. El proyecto cuenta con funcionalidades comunes a aplicaciones de control de medicamentos y a aplicaciones de comunicación, por lo que no existen apenas aquellas que implementen los dos campos en uno.

2.1. DepenCare

Se trata de una aplicación [3] que permite dar un mejor servicio a las personas mayores, permitiendo tener una agenda y lista de tareas, así como un chat para la comunicación. La principal desventaja es que la comunicación se produce mediante mensajes de texto en vez de mediante videollamadas, que resulta más complejo para las personas de avanzada edad.

En la Figura 1 se muestra el logo de la aplicación:

Figura 1: Aplicación DepenCare

2.2. Cuidum

Se trata de otra aplicación [4] con la que gestionar el cuidado de personas dependientes en la que se pueden realizar consultas médicas mediante chat de texto y también permite añadir tareas.

En la Figura 2 se muestra el logo de la aplicación:

Figura 2: Aplicación Cuidum

2.3. MedControl

Se trata de una aplicación [5] diseñada para recordar la toma de medicamentos y también permite realizar un seguimiento de la medicación. La desventaja de este sistema es que no es posible manejar los medicamentos de un familiar desde otra cuenta, sólo es posible crear, editar y eliminar alertas desde el mismo usuario. Tampoco es posible comunicarse con familiares.

En la Figura 3 se muestra el logo de la aplicación:

Figura 3: Aplicación MedControl

2.4. MediSafe

Al igual que la anterior, se trata de una aplicación [6] en la que se pueden gestionar los medicamentos y controlar nuestra salud. Tiene la misma desventaja que la anterior, aunque en este caso la aplicación permite crear tomas de medicamentos más personalizadas. Tampoco es posible comunicarse con nuestros familiares.

En la Figura 4 se muestra el logo de la aplicación:

Figura 4: Aplicación MediSafe

3. Objetivos

A continuación, se detallan los objetivos del sistema y personales que se pretenden cumplir al desarrollar el proyecto:

3.1. Objetivos del sistema

El objetivo principal del sistema es el diseño, desarrollo e implementación de un sistema que permita facilitar tareas a personas mayores.

- **Gestión de usuarios:** El sistema deberá permitir dar de alta y baja a los usuarios.
- **Gestión de llamadas:** El sistema deberá permitir realizar y recibir llamadas entre familiares, y visualizar el historial de llamadas.
- **Gestión de alertas:** El sistema deberá permitir configurar y recibir alertas asociadas a un familiar.
- **Gestión de solicitudes:** El sistema deberá permitir enviar y recibir solicitudes para añadir familiares.
- **Exploración y búsqueda de usuarios:** El sistema debe permitir a los usuarios realizar búsquedas y encontrar a otros usuarios registrados, filtrando por email en la base de datos.
- **Almacenamiento y recuperación de datos:** El sistema debe contar con una base de datos que permita el almacenamiento de información acerca de los usuarios, las alertas y las videollamadas.
- **Presentar información sobre usuarios, alertas y videollamadas:** El sistema debe ser capaz de recuperar información de la base de datos y mostrarla de forma detallada.
- **Compatibilidad del sistema:** El sistema debe ser compatible con la mayor cantidad de dispositivos posibles.
- **Escalabilidad:** El sistema deberá ser escalable, es decir, que se adapte correctamente al crecimiento de la aplicación.
- **Usabilidad:** El sistema deberá proporcionar una interacción sencilla e intuitiva para todos los usuarios, independientemente de su edad.

3.2. Objetivos personales

La finalidad de este apartado es expresar los objetivos personales que me han motivado a realizar este proyecto. Desde el principio he sentido interés por el desarrollo de aplicaciones Android, ya que es algo que utilizo a diario.

Por tanto, los objetivos personales que me propuse fueron:

- Familiarizarme con el desarrollo de aplicaciones Android y las distintas arquitecturas que se emplean. Conocer el SDK de Android, el entorno de desarrollo de Android Studio y aprender el lenguaje de programación Kotlin, que es un lenguaje cada vez más utilizado y con mucha potencia. También aprender el lenguaje de programación JavaScript empleado en el servidor.
- Emplear las técnicas de ingeniería aprendidas a lo largo del grado de Ingeniería Informática en las asignaturas de Ingeniería de Software I, Ingeniería de Software II y Gestión de Proyectos. Además, emplear los conocimientos de asignaturas como Programación, Interacción Persona-Ordenador, Bases de Datos, etc.
- Aprender a utilizar la plataforma de Firebase, ya que es un entorno que facilita mucho la creación de aplicaciones, en concreto los servicios de autenticación, base de datos, funciones en la nube o mensajería.
- Realizar una interfaz gráfica amigable e intuitiva que haga que la experiencia del usuario sea agradable.
- Solucionar por mí misma los problemas que hayan surgido durante el desarrollo de la aplicación.
- Mejorar mi capacidad de buscar los recursos necesarios para elaborar el proyecto.

4. Conceptos teóricos

4.1. Aplicación móvil nativa

Una aplicación nativa [7] es aquella que ha sido desarrollada para un sistema operativo específico con en el lenguaje de programación específico de dicho sistema. En este caso, se trata de una aplicación de Android, y se ha desarrollado utilizando el lenguaje de programación Kotlin.

Algunas ventajas de las aplicaciones nativas son:

- Tienen un gran nivel de personalización y optimización, y por ello se puede ofrecer una experiencia de usuario óptima.
- Se pueden utilizar sin conexión a Internet.
- Son más seguras porque que tienen que pasar por controles de seguridad exhaustivos para ser publicadas.

Algunas desventajas serían los costes, el tiempo y los recursos que hay que invertir.

En la Figura 5 se muestra un ejemplo de aplicaciones nativas, en comparación con aplicaciones híbridas y aplicaciones web.

Figura 5: Aplicación móvil nativa

4.2. Android TV

Es una plataforma de TV inteligente de Google [8] donde los usuarios pueden adquirir una televisión con una nueva plataforma integrada o mediante la compra de un dispositivo receptor independiente que se puede añadir a la televisión.

Algunas de las prácticas que se deben aplicar a la hora de desarrollar una aplicación que se verá en la televisión son las siguientes:

- Dejar suficiente margen a los lados, crear el diseño con orientación horizontal y colocar controles de navegación sobre el lado izquierdo o derecho de la pantalla para ahorrar espacio vertical.
- Evitar fuentes ilegibles, evitar fuentes finas o con trazos muy angostos, utilizar los tamaños de fuente estándar de Android, utilizar texto claro sobre texto oscuro que facilita la lectura en la televisión.
- Minimizar y simplificar la interacción al máximo, diseñar una navegación sencilla con grupos de celdas que orienten al usuario de cómo moverse con un pad direccional del mando.

Figura 6: Android TV

4.3. Jitsi Meet

Se trata de una aplicación [9] de videoconferencias de código abierto. Es un sistema P2P, es decir, el vídeo y el audio de cada participante se comparte por cada nodo, y, por lo tanto, al no estar centralizado en un servidor VPS, se evitan cortes en las videoconferencias. Es escalable, ya que con el tiempo se pueden asignar más recursos de forma puntual. Es compatible con WebRTC y permite realizar videollamadas encriptadas.

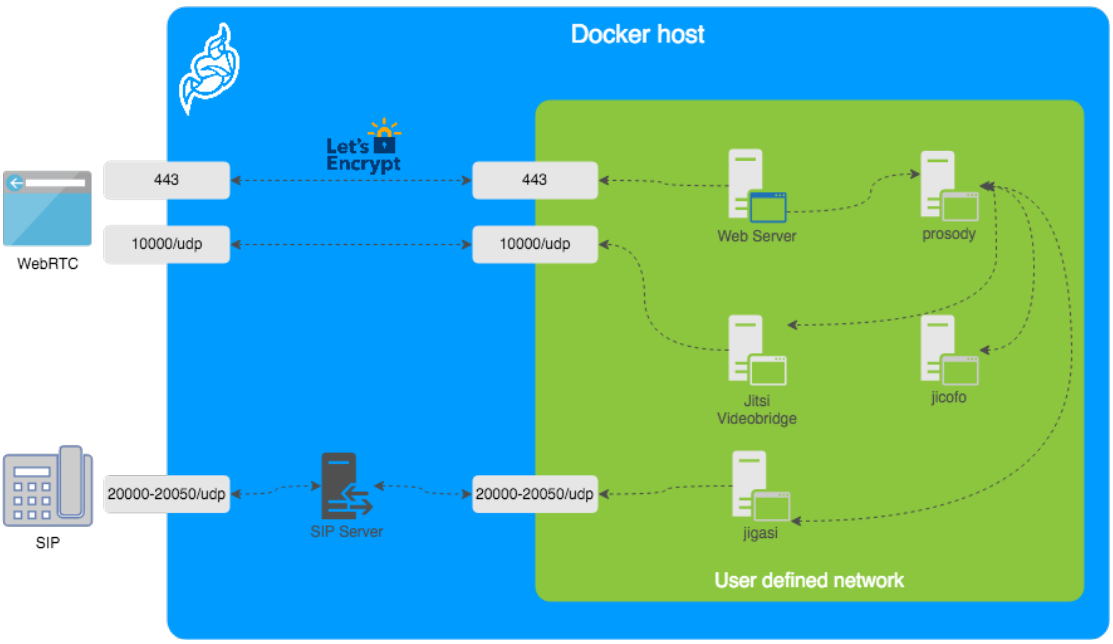


Figura 7: Arquitectura de Jitsi Meet

4.4. WebRTC

WebRTC o *Web Real-Time Communication* [10], es un proyecto libre y de código abierto que proporciona a los navegadores web y a las aplicaciones móviles comunicación en tiempo real a través de APIs. Permite que la comunicación de audio y vídeo funcione dentro de las páginas web al permitir la comunicación entre pares, eliminando la necesidad de instalar plugins o descargar aplicaciones. Como se ha mencionado antes, se emplea en la comunicación mediante Jitsi Meet.

Figura 8: WebRTC

4.5. Material Design

Se trata de un sistema [11] adaptable de componentes y herramientas que respaldan las mejores prácticas de diseño de interfaz de usuario. Tiene un código fuente abierto, agiliza la colaboración entre diseñadores y desarrolladores, y ayuda a los equipos a crear productos atractivos. Utiliza un lenguaje visual propio inspirado en el mundo real, en cómo se comportan las luces y las sombras.

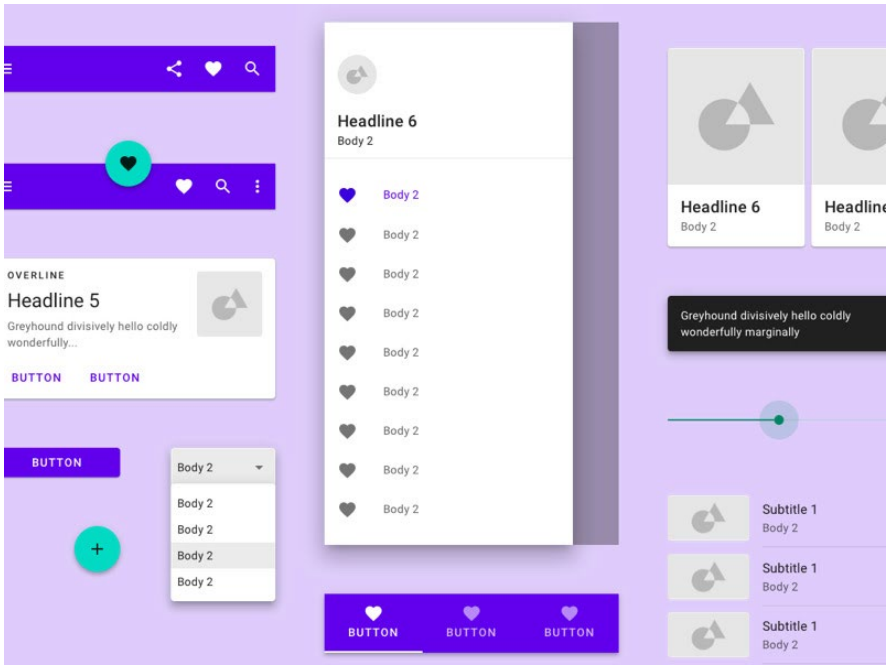


Figura 9: Material Design

4.6. MVP - Modelo Vista Presentador

El MVP [12] se trata de un patrón de diseño derivado del conocido MVC (Modelo Vista Controlador) y que se emplea principalmente en la construcción de interfaces de usuario. El uso de este patrón ayuda a separar la capa de vista de la lógica, a escribir un código más limpio, facilita la realización de pruebas, etc.

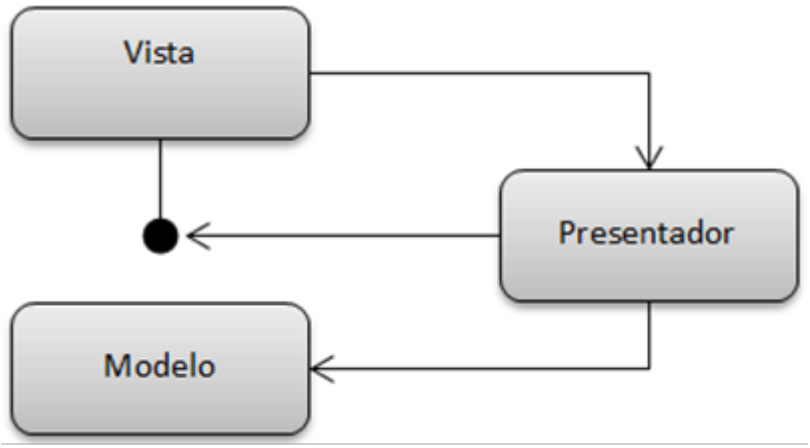


Figura 10: Modelo MVP

Modelo

Es una interfaz que define los datos que se mostrarán o sobre los que actuará la interfaz de usuario. Es la capa encargada del acceso a la base de datos, API Rest, memoria cache. etc.

Vista

Es una interfaz pasiva que exhibe datos (el modelo) y órdenes de usuario de las rutas (eventos) al presentador para actuar sobre los datos. Es la capa encargada de diseñar la UI, realizar peticiones y mostrar resultados. No debe existir lógica de negocio, y están las Actividades, Fragmentos, etc.

Presentador

Asume la funcionalidad de “intermediario” entre las otras dos partes del patrón, y tiene toda la lógica de presentación. Recupera datos de los repositorios, es decir, del modelo, y los formatea para mostrarlos en la vista.

4.7. Arquitectura Cliente-Servidor

Se trata de un modelo de diseño de software [13] en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. La comunicación se produce mediante el conjunto de protocolos TCP/IP. En el caso de este proyecto, la aplicación móvil tiene el rol de cliente mientras que el conjunto de las funciones de Firebase y el servidor VPS tienen el rol de servidor.

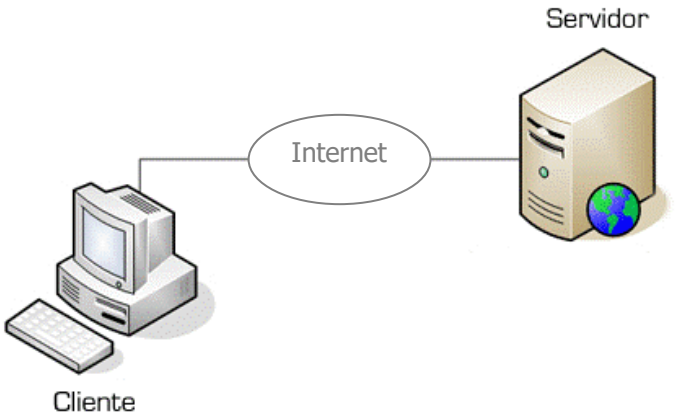


Figura 11: Arquitectura Cliente-Servidor

Cliente

Se encarga de interactuar directamente con los usuarios finales mediante una interfaz gráfica de usuario y realiza solicitudes o peticiones al servidor, así que tiene por tanto un papel activo en la comunicación (dispositivo maestro). Pueden ser dispositivos de diferentes tipos como ordenadores, *smartphones*, *smart tv's*, *tablets*, etc. Se habla de cliente como *frontend*.

Servidor

Se encarga de recibir las solicitudes de los clientes, procesarlas y responder acorde a la petición, así que tiene por tanto un papel pasivo en la comunicación (dispositivo esclavo). Se habla de servidor como *backend*.

Middleware

Es una capa o nivel de software situada entre la red y las aplicaciones, que ofrece una interfaz de programación de aplicaciones (API) común para el intercambio de mensajes entre diversas secciones de aplicaciones de proceso distribuido. Protege a los usuarios de la complejidad de tener que manejar múltiples entornos de proceso.

4.8. Arquitectura Serverless

Se denominan arquitecturas *serverless* [14] o sin servidor a aquellas en las que los desarrolladores de las aplicaciones no administran los servidores, sino que se delegan a otras empresas como Amazon (AWS Lambda), Microsoft (Azure) o Google (Firebase), es decir, no significa que no se utilicen servidores, sino que se abstrae la tecnología, que puede ser funcional a pesar de estar lejos del usuario.

Algunos beneficios de esta arquitectura son:

- **Costes:** Proporciona servicios por lo que se utiliza, es decir, solo se paga por lo que se usa.
- **Mantenimiento:** No hay que administrar ningún servidor ni hay que instalar ningún sistema operativo o software.
- **Escalamiento fácil y eficiente:** Se puede escalar fácilmente para elegir la capacidad deseada.
- **Alta disponibilidad:** Tienen disponibilidad incorporada, por lo tanto, no se necesita tener una infraestructura especializada.

5. Técnicas y herramientas

En este apartado se describirán las técnicas y herramientas utilizadas a lo largo del desarrollo del sistema tanto en la parte de *frontend* que será la aplicación de Android Studio en sí, como en la parte de *backend* que será el servidor, además de otras herramientas empleadas.

Todas ellas facilitan el desarrollo del sistema, aportando herramientas que facilitan la programación y depuración, o aportando bibliotecas con las que poder reutilizar código de terceros ahorrando tiempo de desarrollo, asegurando un correcto funcionamiento y mantenimiento, y sin problemas de seguridad.

5.1. Cliente

En la parte de *frontend* se ha empleado Android Studio [15], el IDE o entorno de desarrollo integrado oficial de Android. Para desarrollar aplicaciones en Android se emplea Java o Kotlin para la parte lógica y XML [16] para la interfaz gráfica de usuario.

En este caso se utiliza Kotlin [17] al ser el lenguaje de programación nativo para Android y que nos permite una interoperabilidad con Java. Es sencillo de aprender, ayuda a escribir aplicaciones más rápido, aumentar la productividad y la seguridad del código, etc.

Figura 12: Android Studio

Incluye un conjunto de funciones que facilitan la programación de la aplicación, y por lo que se ha escogido este entorno de desarrollo, entre las que tenemos:

- Un editor de código inteligente que te sugiere opciones de autocompletado y soluciones coherentes a errores o avisos.
- Un emulador rápido y cargado de funciones
- Un sistema de compilación flexible y compatible con Gradle [18], que se ejecuta en una herramienta de depuración integrada y lo hace de forma independiente de la línea de comandos.

- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android
- Aplicación de cambios para insertar cambios de código y recursos a la app en ejecución sin reiniciarla
- Integración con GitHub y plantillas de código como ayuda para compilar funciones de otras apps e importar códigos de muestra.
- Variedad de marcos de trabajo y herramientas de prueba
- Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones, entre otros
- Compatibilidad con NDK (C++) para aplicaciones en las que necesitemos utilizar demasiados recursos del dispositivo móvil y, por ende, optimizar de la mejor manera posible los mismos.
- Compatibilidad integrada con servicios en la nube como Google Cloud Platform.

Para desarrollar la aplicación se han empleado algunas herramientas de código abierto que han facilitado la programación.

Android SDK

El Android SDK, también conocido como *Android Software Development Kit*, es el kit de desarrollo de software oficial de Android creado por Google. Viene incluido con Android Studio, que es el IDE o entorno de desarrollo integrado oficial de Android. Incluye un conjunto de herramientas y bibliotecas de desarrollo de software necesarias para crear aplicaciones Android, algunas de ellas son:

- **Android Emulator:** Emulador de dispositivos Android basado en QEMU.
- **Android SDK Build-Tools:** Herramientas de compilación.
- **GPU Debugging Tools:** Depurador de código.
- **Documentation for Android SDK**

Jitsi Meet SDK

Se trata de una librería de software libre y de código abierto [19] de Android que incorpora toda la experiencia de Jitsi Meet y la hace reutilizable por aplicaciones de terceros. Se utiliza para realizar la comunicación mediante videollamadas entre los usuarios de la aplicación.

Figura 13: Jitsi Meet

Para integrar el SDK en el proyecto es necesario añadir el repositorio de Maven y la dependencia de la librería en los archivos de Gradle. Para utilizarla en el código bastará con crear una sala de Jitsi Meet y establecer los parámetros necesarios como el dominio del servidor, el JWT del usuario y el identificador de la sala.

Es posible permitir el acceso con autenticación a través de JWT o Json Web Token [20]. Se crea uno para cada usuario que quiere acceder a una videollamada concreta, y es un token que se genera empleando sus datos personales y los datos de la llamada, y posteriormente, se firma con una clave privada.

Glide

Se trata de una biblioteca [21] de código abierto de administración y carga de imágenes, que junta la decodificación de medios, el almacenamiento en caché y la agrupación de recursos, en una sola interfaz simple y fácil de usar. Facilita mucho la obtención, decodificación, visualización y tratamiento de imágenes y videos.

Figura 14: Glide

Al igual que en el caso anterior, para integrar Glide en el proyecto tendremos que añadir el repositorio Maven y las dependencias de la biblioteca a los archivos de Gradle. Se utiliza en el código para adaptar el tamaño de las imágenes y para colocar imágenes a partir de sus direcciones URL de Firebase Storage.

5.2. Servidor

Firebase

Se trata de una plataforma digital [22] de Google diseñada para facilitar el desarrollo de aplicaciones de calidad de una forma rápida y eficiente, con el objetivo de mejorar el rendimiento de estas a través de la implementación de sus distintos servicios como backend (BaaS o Backend as a Service) que harán que la aplicación sea mucho más manejable, segura y fácil de utilizar para los usuarios. Se encuentra alojada en la nube, y está disponible para Android, iOS y web entre otras tecnologías.

Figura 15: Firebase

Los servicios utilizados en la aplicación son:

Authentication [23]

Permite la autenticación mediante diversos mecanismos como contraseñas, números de teléfono o proveedores de terceros como Facebook, Google y Twitter. Utiliza los estándares OAuth 2.0 y OpenID Connect para facilitar la integración con nuestro propio *backend*.

Firebase Firestore [24]

Se trata una base de datos NoSQL o no relacional, flexible, escalable y en la nube con el objetivo de almacenar y sincronizar datos mediante el uso de agentes de escucha en tiempo real. Los datos están estructurados como documentos organizados en colecciones.

Firebase Storage [25]

Se trata de un módulo de almacenamiento para contenido pesado generado por los usuarios de la aplicación donde se pueden guardar imágenes, videos o audios. Emplea una estructura en forma de árbol de directorios.

Cloud Functions [26]

Se trata de un framework sin servidores que permite escribir e implementar código en los servidores de Firebase de forma que responda de forma automática a un evento en nuestra aplicación.

Cloud Messaging [27]

Permite enviar mensajes o notificaciones push a los usuarios de la aplicación de forma segura y sencilla.

Visual Studio Code

Es un editor de código fuente [28] desarrollado por Microsoft. Es software libre, gratuito y multiplataforma. Se basa en Electron, un framework que se utiliza para implementar Chromium y Node.js como aplicaciones para escritorio, que se ejecuta en el motor de diseño Blink.

Figura 16: Visual Studio Code

Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es compatible con múltiples lenguajes de programación como, por ejemplo, C, Java, Python, Node.js, JavaScript, etc.

Node.js

Es un entorno en tiempo de ejecución [29] de JavaScript [30] de código abierto y multiplataforma. Se caracteriza por ser asíncrono, con E/S de datos en una arquitectura orientada a eventos y estar basado en el motor V8 de Google.

Figura 17: Node.js

Algunas de las características más importantes de este entorno serían:

- **Asíncrono y controlado por eventos:** Todas las API son asíncronas, es decir, sin producir bloqueos. Esto significa que nunca espera a que una API devuelva datos.
- **Muy rápido:** Al estar construido en el motor JavaScript V8 de Google Chrome, es muy rápido en la ejecución de código.
- **Sin almacenamiento en búfer:** Nunca almacenan en búfer ningún dato, simplemente generan los datos en fragmentos.
- **Procesos en un solo hilo, pero altamente escalable:** Utiliza un modelo de un solo hilo con bucle de eventos. Utiliza un solo programa de subprocesos y el mismo programa puede proporcionar servicio a un número mucho mayor de solicitudes que los servidores tradicionales.

5.3. Herramientas CASE

Las herramientas CASE [31] o Ingeniería de Software Asistida por Computación, son un conjunto de herramientas o aplicaciones informáticas que se utilizan para facilitar el desarrollo de software, incrementando la calidad, la productividad y el mantenimiento durante el proceso, para reducir los costes en tiempo y dinero.

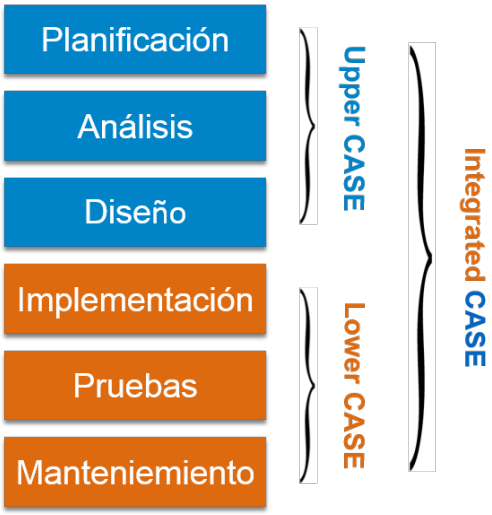


Figura 18: Herramientas CASE

UML

La herramienta UML [32] o Unified Modeling Language, es el lenguaje de modelado de sistema de software más conocido y utilizado actualmente. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Se utiliza para especificar métodos o procesos, detallar los artefactos en el sistema y para documentar y construir.

Figura 19: UML

Visual Paradigm

Es una herramienta CASE [33] que ayuda a los equipos de desarrollo de software a capturar los requisitos correctos y transformarlos en diseños precisos, lo que ayuda a los desarrolladores a crear el software adecuado según los requisitos. Nos permite realizar distintos tipos de diagramas, como diagramas de clases, diagramas de despliegue, diagramas de paquetes, diagramas de secuencia, etc.

Figura 20: Visual Paradigm

Microsoft Office Project

Se trata de un software [34] de administración de proyectos para asistir en la planificación, asignación de recursos a tareas, dar seguimiento al progreso, administrar presupuesto y analizar cargas de trabajo.

Figura 21: Microsoft Office Project

EZEstimate

Es una herramienta que permite estimar el tiempo y el esfuerzo que habrá que emplear en el desarrollo de un sistema completo a partir de la métrica de UCP o Puntos de caso de uso.

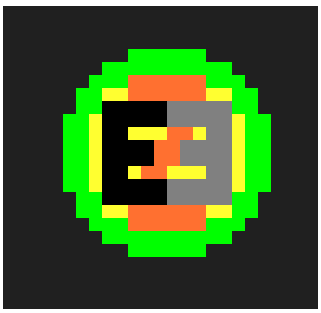


Figura 22: EZEstimate

5.4. Control de versiones

GitHub

Es un repositorio online gratuito [35] que permite gestionar proyectos, realizar backups y controlar versiones de código, de forma que se puedan administrar cambios en el proyecto a la vez que evoluciona. Está basado en Git, que es un sistema de control de versiones distribuido de código abierto.

Figura 23: GitHub

6. Aspectos relevantes del desarrollo

En este apartado se van a desarrollar los aspectos relevantes del proceso de desarrollo del proyecto.

6.1. Marco de trabajo

Para el desarrollo del proyecto se ha utilizado el marco de trabajo del Proceso Unificado [36] que es de carácter genérico y puede especializarse para una gran variedad de sistemas software. Utiliza el lenguaje UML para modelar de forma gráfica conceptos del proyecto y de su desarrollo.

Las características principales de este marco de trabajo son:

- **Iterativo e incremental:** El proceso unificado se divide en cuatro fases denominadas Inicio, Elaboración, Construcción y Transición, y cada una de estas etapas se divide en iteraciones. Al finalizar cada iteración se revisa el progreso del proyecto a través de los hitos generados por ellas. Con este proceso se consigue ir ampliando y refinando el sistema, es decir, también es un proceso incremental al ir añadiendo funcionalidad.
- **Dirigido por casos de uso:** Los casos de uso son descripciones de las acciones que debe realizar un actor sobre el sistema. Se utilizan para capturar los requisitos funcionales, por lo que constituyen la base del proceso unificado. En cada iteración se toma un conjunto de casos de uso y se pasa por las cuatro etapas del proceso.
- **Centrado en la arquitectura:** En el proceso unificado no existe un modelo único que cubra todos los aspectos del sistema. Es por esto que existen múltiples modelos que definen la arquitectura del software.
- **Enfocado en riesgos:** Requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

Como se ha comentado antes, el proceso unificado está compuesto de cuatro fases o etapas que se dividen en interacciones. Estas etapas son las siguientes:

- **Inicio:** Fase en la que se define el alcance del proyecto, el modelo de negocio, se realiza la elicitación de requisitos, se definen los objetivos y se realiza la planificación temporal.
- **Elaboración:** Fase en la que se obtiene la visión refinada del proyecto a realizar, la implementación iterativa del núcleo de la aplicación, la resolución de riesgos altos, nuevos requisitos y se ajustan las estimaciones.
- **Construcción:** Fase que abarca la evolución hasta convertirse en producto listo incluyendo requisitos mínimos. Aquí se afinan los detalles menores como los diferentes tipos de casos o los riesgos menores.
- **Transición:** En esta fase final, el programa debe estar listo para ser probado, instalado y utilizado por el cliente sin ningún problema. Una vez finalizada esta fase, se debe comenzar a pensar en futuras novedades para la misma.

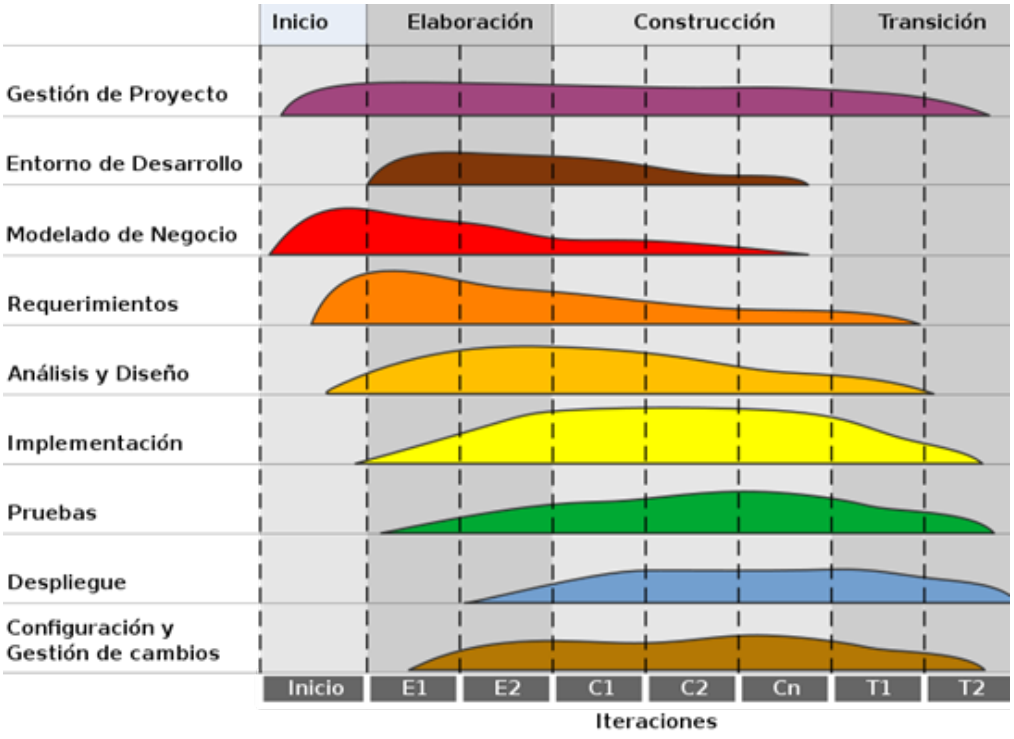


Figura 24: Proceso Unificado

6.2. Estimación del esfuerzo

Para realizar la estimación del esfuerzo y tiempo que llevará el desarrollo del proyecto recogido en el "Anexo I: Plan de proyecto" [35], se ha realizado una aproximación empleando la métrica de UCP (Use Case Points) o Puntos de caso de uso, que se emplea para evaluar la funcionalidad en forma de casos de uso. Para ello se ha estudiado la complejidad de diferentes aspectos del sistema.

Primero se ha medido la complejidad de los actores siguiendo una tabla de pesos, como se puede ver en la Figura 1:

Complejidad	Peso	Descripción
Simple	1	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API).
Media	2	Otro sistema que interactúa mediante un protocolo (por ejemplo, TCP/IP) o una persona que interactúa a través de una interfaz en modo texto.
Compleja	3	Una persona que interactúa con el sistema mediante una interfaz gráfica de usuario (GUI).

Tabla 1: Complejidad de los actores

Después se ha determinado la complejidad de los casos de uso siguiendo también una tabla de pesos, como se puede ver en la Figura 2:

Complejidad	Peso	Descripción
Simple	5	Menos de 4 transacciones Menos de 5 clases
Media	10	Entre 4 y 7 transacciones Entre 5 y 10 clases
Compleja	15	Más de 7 transacciones Más de 10 clases

Tabla 2: Complejidad de los casos de uso

A continuación, se han establecido los valores de los factores de complejidad técnica siguiendo la Tabla 3:

Factor técnico	Descripción	Peso
T1	Sistema distribuido	2
T2	Rendimiento o tiempo de respuesta	1
T3	Eficiencia del usuario final	1
T4	Procesamiento interno complejo	1
T5	Reusabilidad	1
T6	Facilidad de instalación	0.5
T7	Facilidad de uso	0.5
T8	Portabilidad	2
T9	Facilidad de cambio	1
T10	Concurrencia	1
T11	Características especiales de seguridad	1
T12	Provee acceso directo a terceras partes	1
T13	Se requiere entrenamiento especial de usuario	1

Tabla 3: Factores de complejidad técnica

Por último, se han medido los valores de los factores de complejidad del entorno siguiendo la Tabla 4:

Factor ambiental	Descripción	Peso
E1	Familiaridad con el modelo de proyecto utilizado y familiaridad con UML	1.5
E2	Personal tiempo parcial	-1
E3	Capacidad de los analistas	0.5
E4	Experiencia en la aplicación	0.5
E5	Experiencia en orientación a objetos	1
E6	Motivación	1
E7	Dificultad del lenguaje de programación	-1
E8	Estabilidad de los requisitos	2

Tabla 4: Factores de complejidad del entorno

Al finalizar la definición de todos los valores de complejidad de los actores, casos de uso y factores de complejidad técnica y de entorno, se ha utilizado la herramienta EZEstimate para calcular el tiempo que llevará desarrollar el proyecto, como se muestra en la Figura 25.

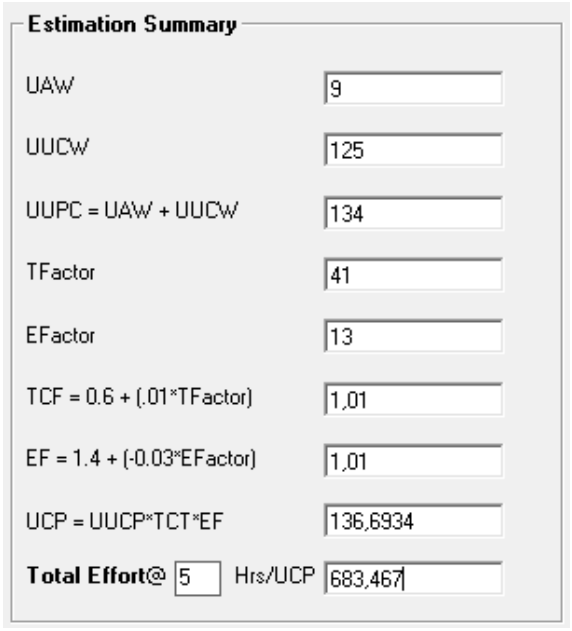


Figura 25: Resultados de EZEstimate

6.3. Planificación temporal

Para realizar la planificación temporal del sistema, recogida en el “Anexo I: Plan de proyecto” [37] de la memoria, se ha seguido el marco de trabajo del Proceso Unificado. Este anexo tiene como objetivo recoger de forma gráfica y visual las diferentes actividades a realizar para completar el proyecto, las dependencias que hay entre ellas y estimar la duración total en base al tiempo de cada tarea.

Se ha utilizado la herramienta de Microsoft Office Project para elaborar un calendario donde ver las tareas a realizar, las dependencias entre ellas y el tiempo global de desarrollo.

Como se ha comentado en el apartado anterior, el Proceso Unificado se divide en cuatro fases diferentes, y estas a su vez se dividen en seis disciplinas diferentes que dependiendo de la etapa en la que nos encontremos tendrán más o menos importancia.

Estas disciplinas son las siguientes:

- **Modelado de negocio:** Se realiza un análisis de aspectos relativos al proyecto como investigación y búsqueda de información.
- **Requisitos:** Se realiza un análisis textual y se definen objetivos, además de realizar la captura de requisitos y una especificación de casos de uso.
- **Análisis:** Se realiza un análisis en profundidad de los requisitos recogidos en la etapa anterior, creando un modelo del dominio, definiendo paquetes de análisis y realizando los casos de uso de la fase anterior.
- **Diseño:** Se definen los patrones arquitectónicos a utilizar, se elaboran clases de diseño, se realiza una representación de los casos de uso elaborados y se crea un modelo de despliegue para describir el diseño final.
- **Implementación:** Se programa el sistema diseñado.
- **Pruebas:** Se realizan pruebas unitarias de componentes y pruebas finales de integración para comprobar el funcionamiento del sistema y corregir posibles errores.

En la Figura 26 se puede ver la estructura del calendario realizado:

	Nombre	Asignado a	Duración	Depende de	Inicio	Finalización
1	**Inicio del proyecto**		**130 días**		**7/3/2022**	**2/9/2022**
2	**Inicio**		**19 días**		**7/3/2022**	**31/3/2022**
3	**Iteración 1**		**10 días**		**7/3/2022**	**18/3/2022**
4	**Modelado de negocio**		**4 días**		**7/3/2022**	**10/3/2022**
5	Planteamiento del problema	Paula Soria	2 días		7/3/2022	8/3/2022
6	Búsqueda de necesidades	Paula Soria	2 días	5	9/3/2022	10/3/2022
7	Búsqueda de actores	Paula Soria	2 días	5	9/3/2022	10/3/2022
8	**Requisitos**		**6 días**		**11/3/2022**	**18/3/2022**
9	Definición de objetivos	Paula Soria	4 días	6	11/3/2022	16/3/2022
10	Definición de requisitos	Paula Soria	2 días	9	17/3/2022	18/3/2022
11	Definición de actores	Paula Soria	2 días	9	17/3/2022	18/3/2022
12	**Análisis**		**2 días**		**9/3/2022**	**10/3/2022**
13	Análisis de lenguajes	Paula Soria	2 días	5	9/3/2022	10/3/2022
14	Análisis de tecnologías	Paula Soria	2 días	5	9/3/2022	10/3/2022
15	**Diseño**		**5 días**		**9/3/2022**	**15/3/2022**
16	Diseño de prototipo con Adobe XD	Paula Soria	5 días	5	9/3/2022	15/3/2022
17	Fin de Iteración 1		1 día	4 8 +2	21/3/2022	21/3/2022
18	**Iteración 2**		**7 días**	17	**22/3/2022**	**30/3/2022**
19	**Requisitos**		**7 días**		**22/3/2022**	**30/3/2022**
20	Identificación de actores	Paula Soria	3 días		22/3/2022	24/3/2022
21	Identificación de escenarios	Paula Soria	3 días		22/3/2022	24/3/2022
22	Especificación de casos de uso	Paula Soria	4 días	20 21	25/3/2022	30/3/2022
23	**Análisis**		**4 días**		**22/3/2022**	**25/3/2022**
24	Identificación de paquetes	Paula Soria	3 días	16	22/3/2022	24/3/2022
25	Realización de diagramas de análisis	Paula Soria	4 días	16	22/3/2022	25/3/2022
26	**Diseño**		**2 días**		**28/3/2022**	**29/3/2022**
27	Realización de diagramas de diseño	Paula Soria	2 días	25	28/3/2022	29/3/2022
28	Fin de Iteración 2		1 día	19 23 26	31/3/2022	31/3/2022
29	Fin de Inicio		1 día	28	1/4/2022	1/4/2022

Figura 26: Calendario planificación de tareas

En la Figura 27 se muestra la estructura del diagrama de Gantt para tener una idea gráfica del transcurso de las tareas y sus dependencias a lo largo del tiempo:

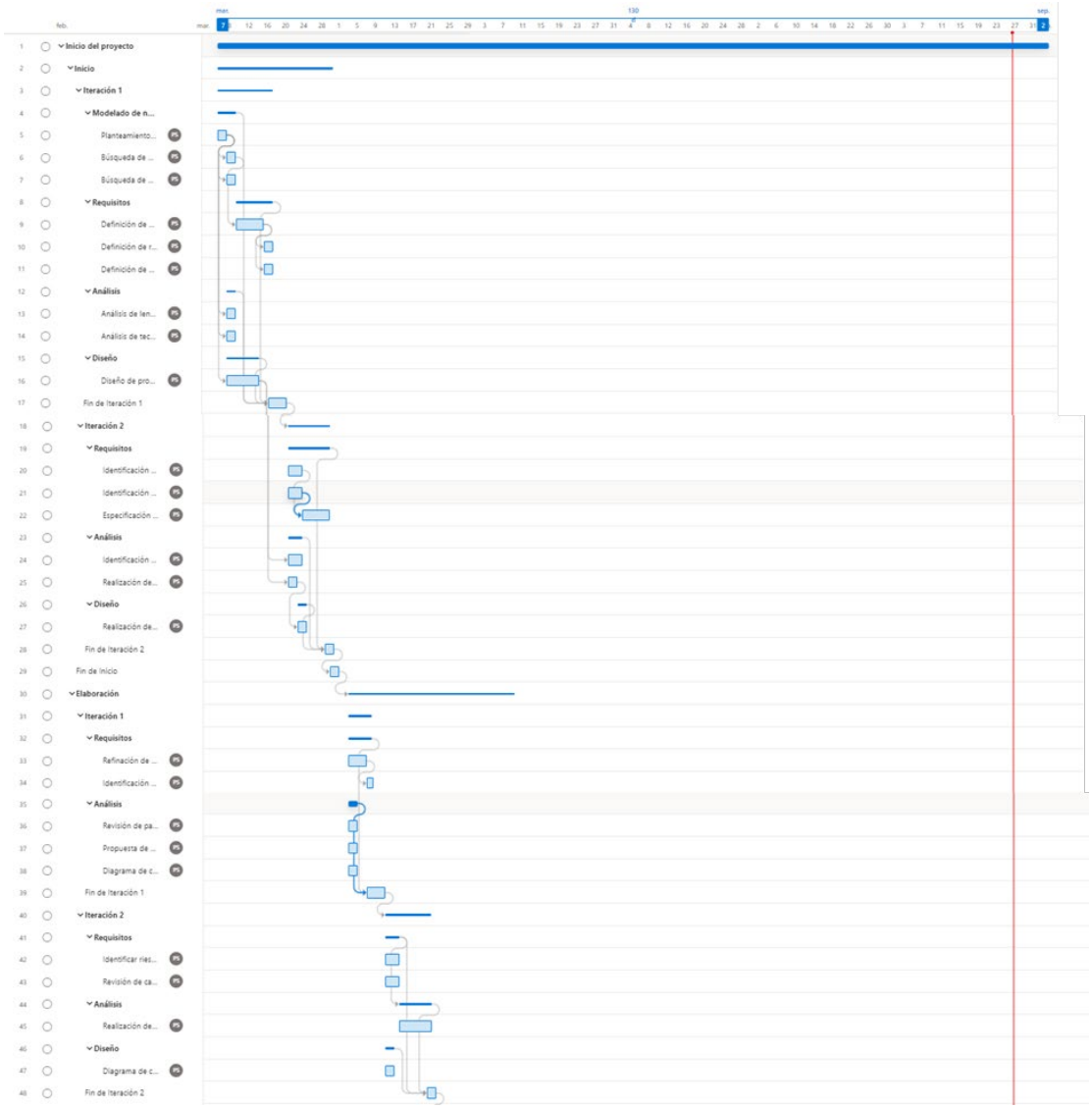


Figura 27: Diagrama de Gantt

Para ver tanto la estimación del esfuerzo en detalle, como la planificación de tareas y el diagrama de Gantt completo, además de las fases y dependencias entre tareas, consultar el “Anexo I: Plan de proyecto”.

6.4. Especificación de requisitos

Para realizar la especificación de requisitos se ha recogido información acerca de los participantes del proyecto, los principales objetivos que debe cumplir el sistema y los requisitos de información, funcionales y no funcionales [38]. De este modo, se definen todos los aspectos que debe cubrir nuestro proyecto, pues esta etapa del Proceso Unificado es la base del resto de las etapas.

Toda la información detallada está recogida en el "Anexo II: Especificación de requisitos" [39] de la memoria. En este apartado se expondrá un ejemplo de la estructura que sigue cada sección que compone el anexo.

Participantes

Los participantes que han formado parte del desarrollo del proyecto son:

- Paula Soria Ullán
- Gabriel Villarrubia González
- Álvaro Lozano Murciego
- André Filipe Sales Mendes.

En la Tabla 5 se puede ver un ejemplo de la estructura que siguen las tablas de participantes:

Participante	Paula Soria Ullán
Organización	Universidad de Salamanca
Rol	Desarrolladora
Es desarrollador	Sí
Es cliente	No
Es usuario	No
Comentarios	Ninguno

Tabla 5: Participante Paula Soria Ullán

Objetivos del sistema

Los objetivos definidos durante la fase de especificación de requisitos y que el sistema debe satisfacer son:

- **Gestión de usuarios:** El sistema deberá permitir dar de alta y baja a los usuarios, y realizar modificaciones sobre su perfil.
- **Gestión de llamadas:** El sistema deberá permitir realizar y recibir llamadas entre familiares, y visualizar el historial de llamadas.
- **Gestión de alertas:** El sistema deberá permitir configurar y recibir alertas asociadas a un familiar.
- **Gestión de solicitudes:** El sistema deberá permitir enviar y recibir solicitudes para añadir familiares.

En la Tabla 6 se puede ver un ejemplo de la estructura que siguen las tablas de objetivos del sistema:

OBJ-0001	Gestión de usuarios
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Descripción	El sistema deberá permitir dar de alta y baja a los usuarios, y realizar modificaciones sobre su perfil.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 6: OBJ-0001 Gestión de usuarios

Requisitos de información

Los requisitos de información, que son los datos que se deben gestionar y almacenar en el sistema de forma persistente son:

- Información sobre usuarios
- Información sobre llamadas
- Información sobre alertas
- Información sobre solicitudes

En la Tabla 7 se puede ver un ejemplo de la estructura que siguen las tablas de requisitos de información:

IRQ-0001	Información sobre usuarios
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Dependencias	Ninguna
Descripción	El sistema deberá almacenar la información correspondiente a los usuarios. En concreto:
Datos específicos	● Identificador ● Email ● Contraseña ● Proveedor ● Rol ● Nombre ● Apellido ● Foto
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 7: IRQ-0001 Información sobre usuarios

Requisitos funcionales

Los requisitos funcionales son los servicios que debe proporcionar el sistema y la manera en que este se debe comportar.

Estos requisitos se han agrupado en paquetes en función de los requisitos de información que manejan, para realizar una división del sistema y tener una mejor organización. Esta división, que también se puede ver en la ilustración de la Figura 28, es la siguiente:

- Gestión de autenticación
- Gestión de usuarios
- Gestión de llamadas
- Gestión de alertas
- Gestión de familiares
- Gestión de solicitudes

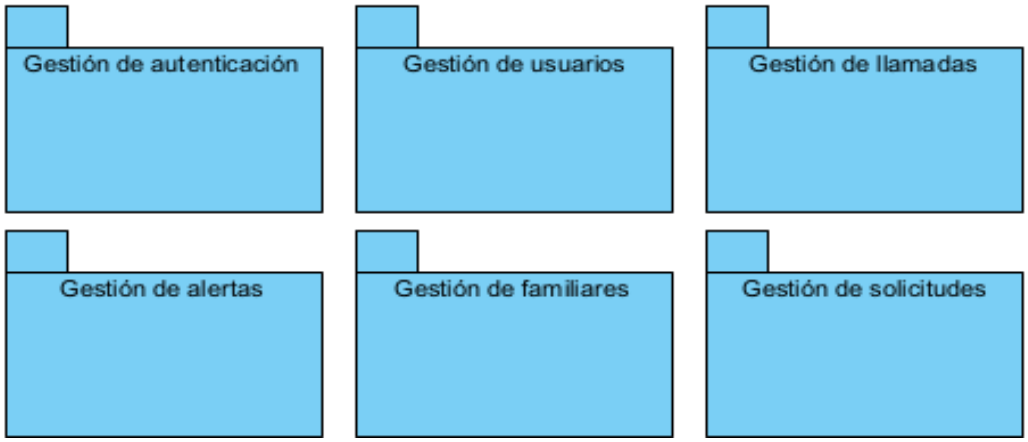


Figura 28: Diagrama de paquetes

Los actores son los que desencadenen los servicios que realiza el sistema, es decir, los casos de uso, y son los siguientes:

- **Usuario sin autenticar**: Representa al usuario antes de acceder al sistema, es decir, antes de autenticarse.
- **Usuario autenticado**: Representa al usuario después de acceder al sistema, es decir, después de autenticarse.
- **Usuario administrador**: Representa al usuario después de acceder al sistema, es decir, después de autenticarse, y además añadiendo la funcionalidad de un administrador.

Figura 29: Actores

En la Tabla 8 se puede ver un ejemplo de la estructura que siguen las tablas de actores:

ACT-0001	Usuario sin autenticar
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Descripción	Representa al usuario antes de acceder al sistema, es decir, antes de autenticarse.
Comentarios	Ninguno

Tabla 8: ACT-0001 Usuario sin autenticar

Para describir los requisitos funcionales se han realizado tablas de casos de uso que explican el servicio que deben dar al sistema. En la Tabla 9 se puede ver un ejemplo de la estructura que siguen las tablas de casos de uso:

UC-0005	Cerrar sesión	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [OBJ-0001] Gestión de usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite cerrar sesión.	
Precondición	El usuario tiene sesión iniciada, y por tanto está en estado de autenticado.	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita al sistema cerrar sesión en la aplicación.
	2	El sistema efectúa el cierre de sesión.
Postcondición	El usuario pasa al estado sin autenticar.	
Excepciones	Paso	Acción
	-	-
Frecuencia	6 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 9: UC-0005 Cerrar sesión

Requisitos no funcionales

Los requisitos no funcionales son aquellos que imponen restricciones en el diseño y en la implementación, así como estándares de calidad. No están relacionados directamente con la funcionalidad del sistema, describen características de funcionamiento.

Los requisitos no funcionales establecidos son los siguientes:

- **Usabilidad:** El sistema deberá proporcionar una interacción sencilla e intuitiva para todos los usuarios, independientemente de su edad.
- **Escalabilidad:** El sistema deberá ser escalable, es decir, que se adapte correctamente al crecimiento de la aplicación.
- **Disponibilidad:** El sistema deberá responder a las peticiones de los usuarios dentro de unos límites de tiempo adecuados.
- **Concurrencia:** El sistema debe de poder ser utilizado por varios usuarios a la vez.
- **Mantenimiento:** Se deben garantizar revisiones periódicas de la aplicación para garantizar su correcto funcionamiento.

En la Tabla 10 se puede ver un ejemplo de la estructura que siguen las tablas de los requisitos no funcionales:

NFR-0001	Usabilidad
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Dependencias	Ninguno
Descripción	El sistema deberá proporcionar una interacción sencilla e intuitiva para todos los usuarios, independientemente de su edad.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 10: NFR-0001 Usabilidad

Para ver todas las tablas de participantes, objetivos y requisitos al detalle, consultar el “Anexo II: Especificación de requisitos software”.

6.5. Análisis de requisitos

A partir de los requisitos obtenidos en la fase anterior del Proceso Unificado, se procede a realizar el análisis de los requisitos capturados donde se especifica el modelo del dominio, se realizan los casos de uso mediante diagramas de secuencia, se lleva a cabo la clase de análisis y se elabora la vista de arquitectura del modelo de análisis.

Toda la información detallada está recogida en el "Anexo III: Análisis de requisitos software" [39] de la memoria. En este apartado se expondrá un ejemplo de la estructura que sigue cada sección que compone el anexo.

Modelo del dominio

El modelo de dominio es un diagrama que permite determinar los objetos de negocio que el sistema debe gestionar y almacenar.

En la Figura 30 es posible ver el modelo del dominio desarrollado:

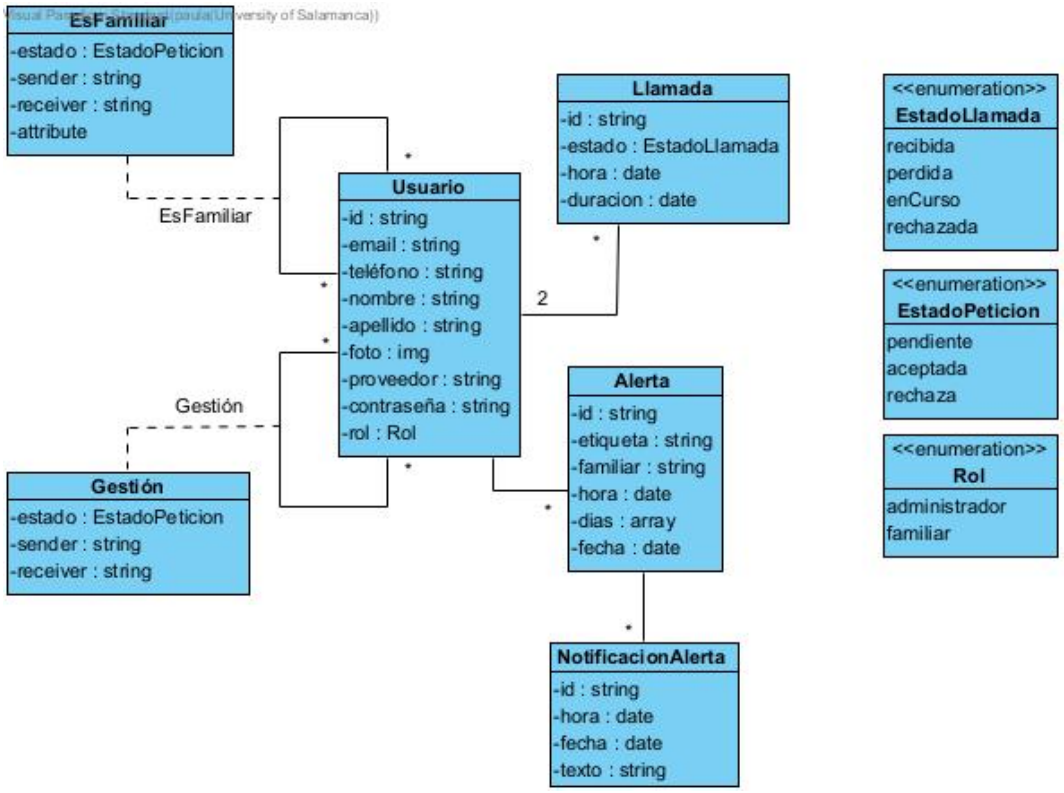


Figura 30: Modelo del dominio

Realización de casos de uso

Para realizar los casos de uso identificados en la etapa anterior, se refinan y se realizan diagramas de secuencia que muestran el intercambio de mensajes, para cada uno de ellos.

En la Figura 31 está representada la estructura que siguen los diagramas de secuencia que ilustran los casos de uso identificados:

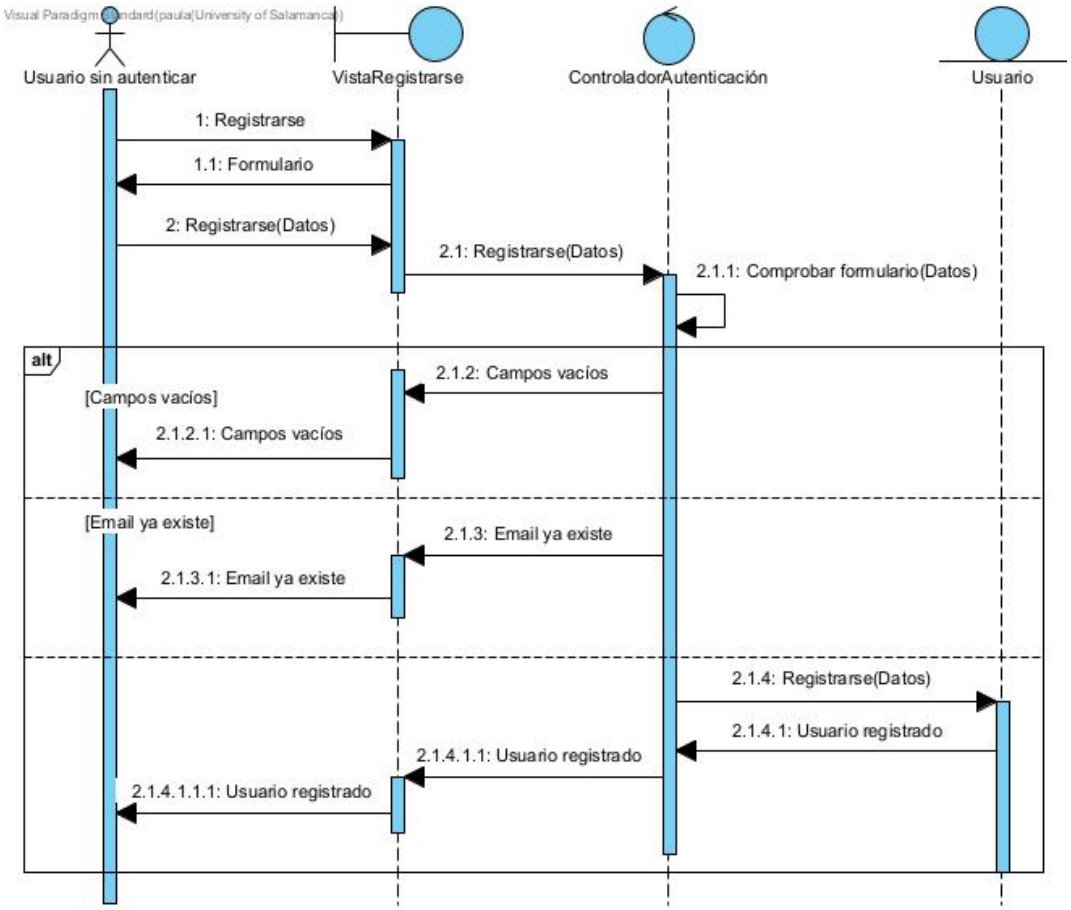


Figura 31: Diagrama de secuencia UC-0001 Registrarse

Clase de Análisis

En los diagramas de la clase de análisis se muestra la comunicación y distribución de las clases recogidas en el apartado anterior, separadas en los paquetes correspondientes.

En la Figura 32 se muestra la estructura que siguen los diagramas de comunicación que representan las clases de análisis:

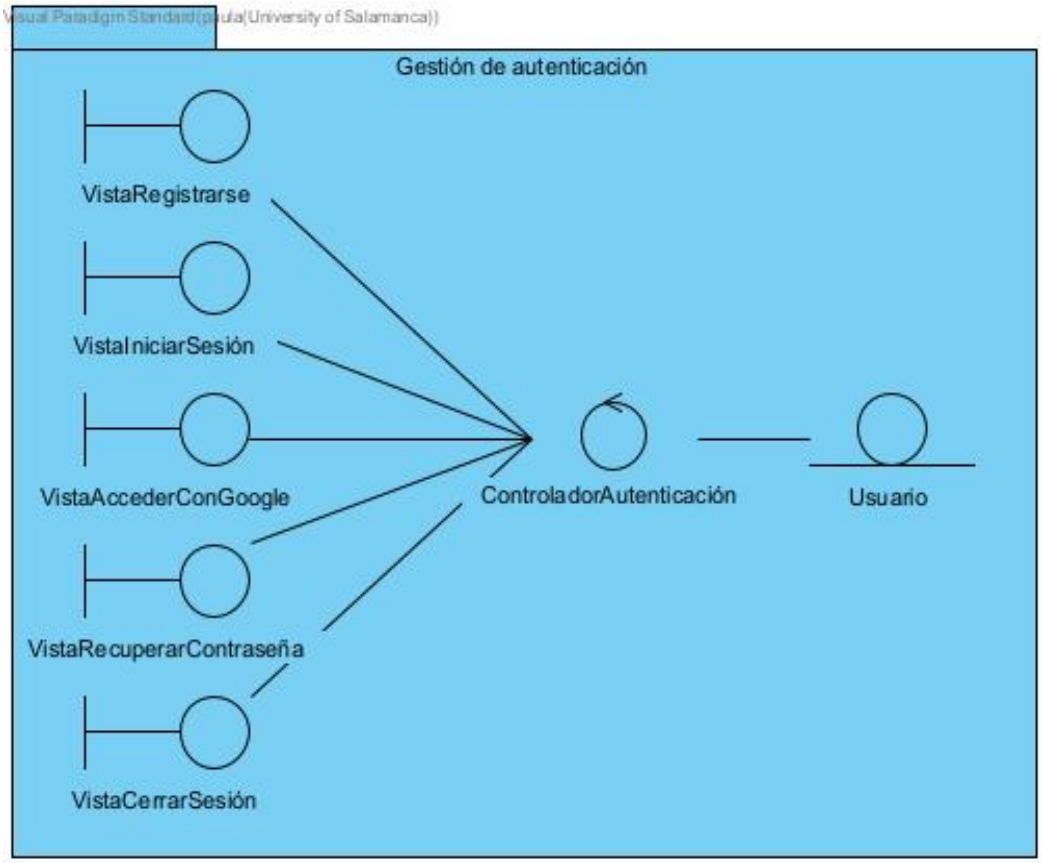


Figura 32: Diagrama de comunicación Gestión de autenticación

Vista de arquitectura del modelo de análisis

En la vista de arquitectura se recoge la vista completa de la arquitectura inicial que se ha obtenido en la fase de análisis, como se puede ver en la Figura 33:

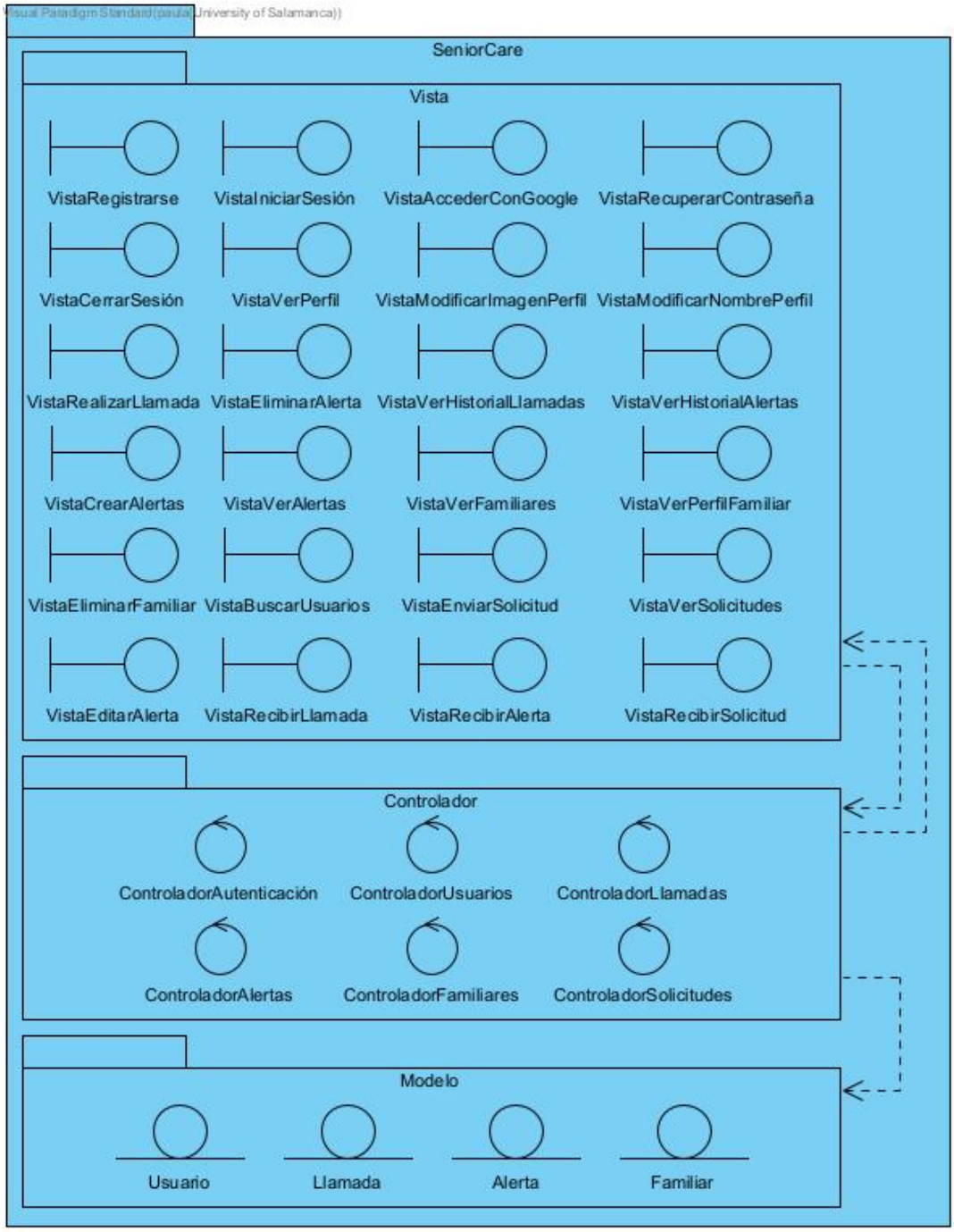


Figura 33: Vista de arquitectura

Para ver todos los diagramas de secuencia y diagramas de comunicación, consultar el “Anexo III: Análisis de requisitos software”.

6.6. Diseño del sistema

A partir del análisis de requisitos realizado en la fase anterior del Proceso Unificado, se realiza el diseño del sistema. Aquí se lleva a cabo el modelo de diseño, el diseño de la base de datos y el modelo de despliegue.

Toda la información detallada está recogida en el “Anexo IV: Diseño del sistema software” [39] de la memoria. En este apartado se expondrá un ejemplo de la estructura que sigue cada sección que compone el anexo.

Diseño de la interfaz

La interfaz gráfica de usuario, que serían los ficheros layout con extensión XML de la aplicación, se ha desarrollado empleando las guías de estilo marcadas por Material Design, para obtener una interacción sencilla e intuitiva.

Al iniciar el proyecto, primero se seleccionó la paleta de colores de la Figura 34 acorde con los valores que se querían transmitir. Se seleccionó una gama de colores azules grisáceos para transmitir seguridad, tranquilidad, protección, salud, confianza, calma y estabilidad.

Figura 34: Paleta de colores

Para crear una paleta de colores bien diseñada, hay que escoger un color principal, que sería el primero de los azules. A continuación, se escogen algunos colores neutros a partir de este color principal para dar dimensión en la aplicación. También hay que seleccionar un color de acento, que en este caso es el azul turquesa de más abajo, para mostrar elementos seleccionados. Por último, se escogieron los colores semánticos.

Después se escogió un nombre para la aplicación y se diseñó un logo que intentase representar adecuadamente los valores de la aplicación. Para ello se escogieron los colores de la paleta anterior y se seleccionó un icono adecuado.

Este logo se puede ver en la Figura 35:

Figura 35: Logo SeniorCare

Para desarrollar una primera idea de la aplicación se empleó la herramienta Adobe XD con la que se diseñó un boceto de las vistas de las pantallas, además de la interacción con los botones.

En las Figuras 36, 37, 38, 39 y 40 se pueden ver los bocetos creados:

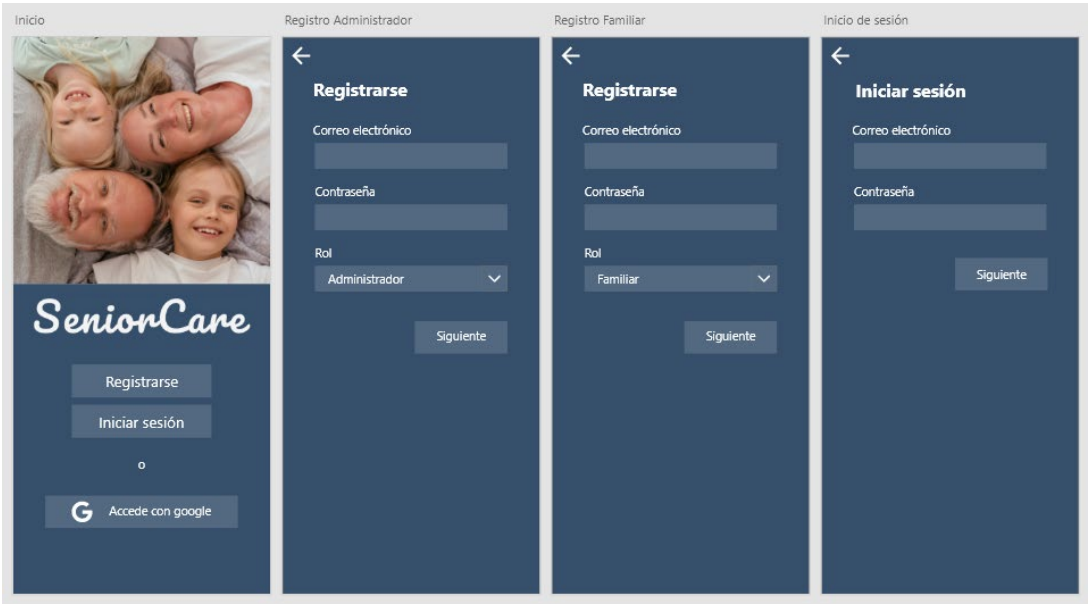


Figura 36: Boceto de la aplicación 1

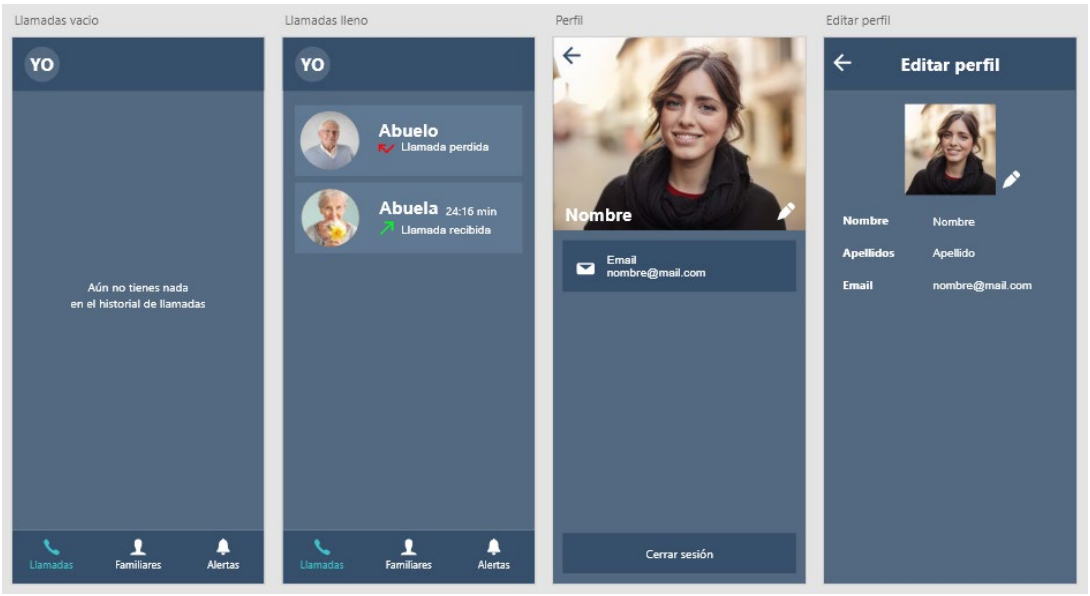


Figura 37: Boceto de la aplicación 2

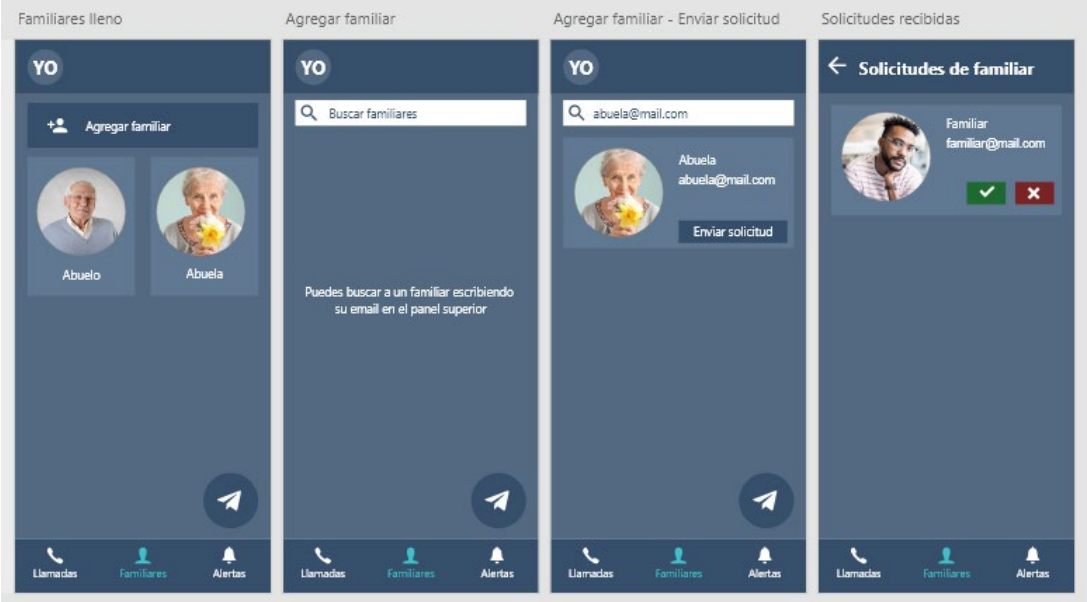


Figura 38: Boceto de la aplicación 3

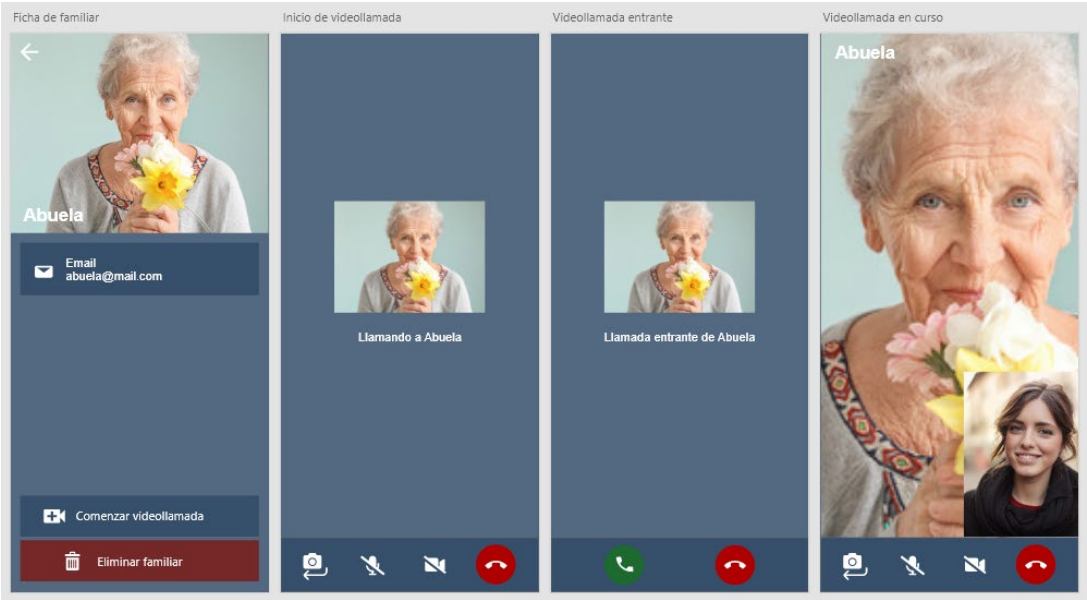


Figura 39: Boceto de la aplicación 4

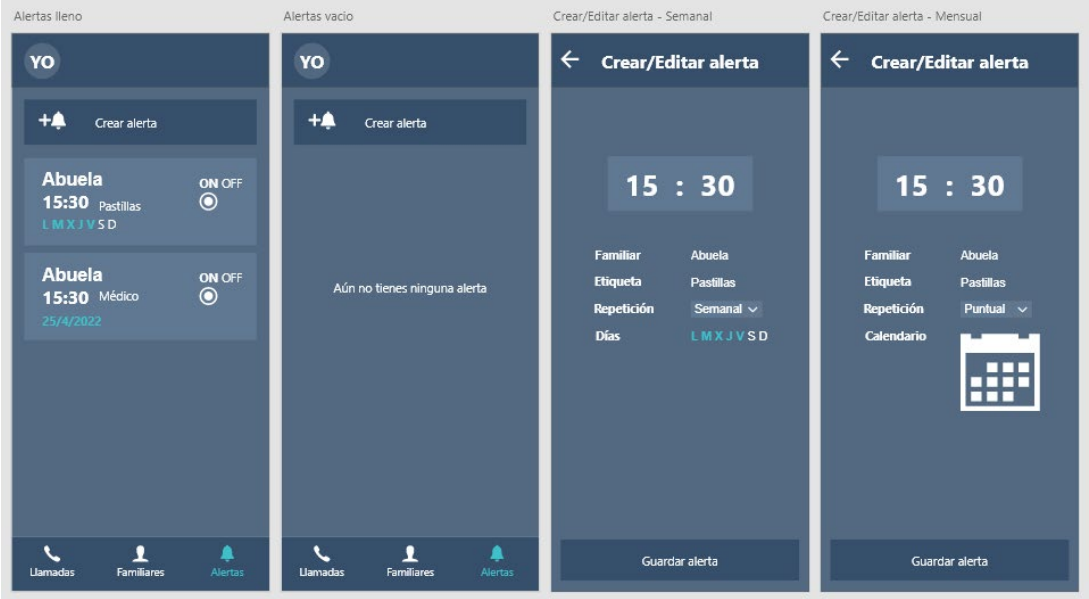


Figura 40: Boceto de la aplicación 5

Posteriormente, se empezó con la codificación de la aplicación en el entorno de Android Studio, y se empezó siguiendo un orden de paquetes: Gestión de autenticación, gestión de usuarios, gestión de familiares, gestión de solicitudes, gestión de alertas y gestión de llamadas.

Modelo de diseño

En este apartado se explican los patrones arquitectónicos utilizados para aplicar unas buenas prácticas de diseño de software. También se desarrollan las clases de diseño, la vista arquitectónica y la realización de casos de uso.

Se han empleado las siguientes tecnologías:

- **Kotlin**: Se ha utilizado como lenguaje de programación en el frontend empleando el entorno de desarrollo de Android Studio.
- **XML**: Se ha utilizado para desarrollar las interfaces de usuario de la aplicación.
- **Firebase**: Se han empleado los distintos servicios que ofrece Firebase para desarrollar el servicio de autenticación, la base de datos, el almacenamiento de imágenes, el servicio de notificaciones y la API.
- **JavaScript**: Se ha utilizado como lenguaje de programación en el backend mediante el uso del entorno de ejecución Node.js.

Patrones arquitectónicos

En este proyecto se ha utilizado el patrón arquitectónico MVP o Model View Presenter, que es un patrón que permite separar la capa de vista de la capa de lógica, de forma que el funcionamiento de la interfaz quede separado de la representación.

En el caso de Android, consigue desacoplar las actividades de la interfaz gráfica de usuario, que es uno de los problemas de estas aplicaciones.

En la Figura 41 se puede ver una ilustración de la organización que sigue el patrón:

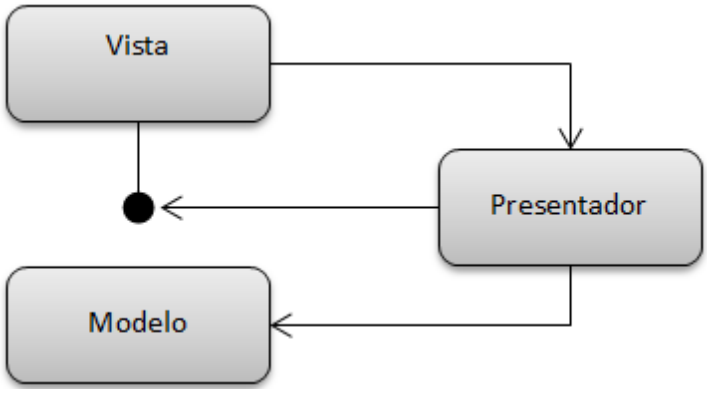


Figura 41: Patrón MVP

Se divide el código en tres capas diferentes:

- **Modelo**: Capa encargada de la lógica de negocio o dominio. Define los datos que se mostrarán o sobre los que actuará la interfaz de usuario.
- **Vista**: Capa encargada de diseñar la UI, realizar peticiones y mostrar resultados. En esta capa no debe existir lógica de negocio. Es una interfaz pasiva que exhibe datos y órdenes de usuario de las rutas al presentador para actuar sobre los datos.
- **Presentador**: Capa encargada de interactuar con la Vista y Modelo. Recupera datos de los repositorios y los formatea para mostrarlos en la vista.

Subsistemas de diseño

Para realizar una mejor organización de la aplicación y tener unos elementos más manejables, se han creado unos subpaquetes de diseño. Se han dividido en dos paquetes principales llamados “Aplicación” y “Servidor” que representan el *frontend* y *backend* respectivamente, y dentro de ellos tendremos otra clasificación más detallada de subpaquetes junto con sus dependencias.

En la Figura 42 podemos ver la organización de los subsistemas de diseño:

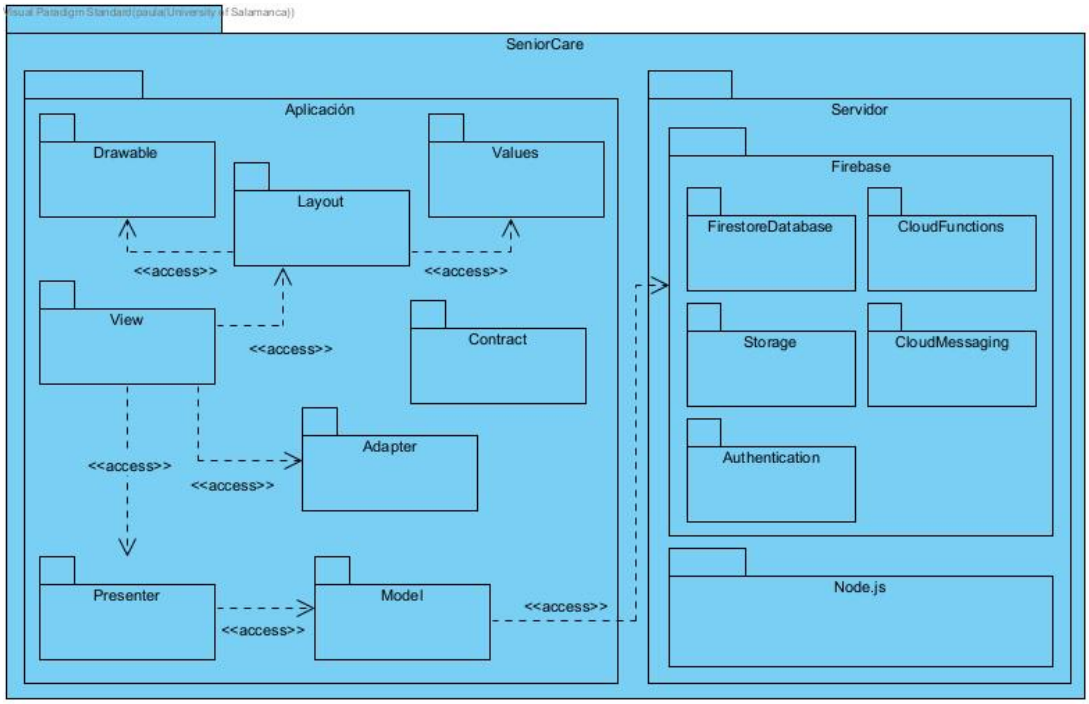


Figura 42: Subsistemas de diseño

Clases de diseño

En las clases de diseño se detalla el contenido que tendrá cada uno de los subpaquetes que componen el patrón arquitectónico MVP. Se encuentran los paquetes descritos en el apartado anterior con sus respectivas clases, y a su vez se describen los métodos que contiene cada una.

En la Figura 43 se puede ver un ejemplo de la estructura que siguen los paquetes con las clases de diseño:

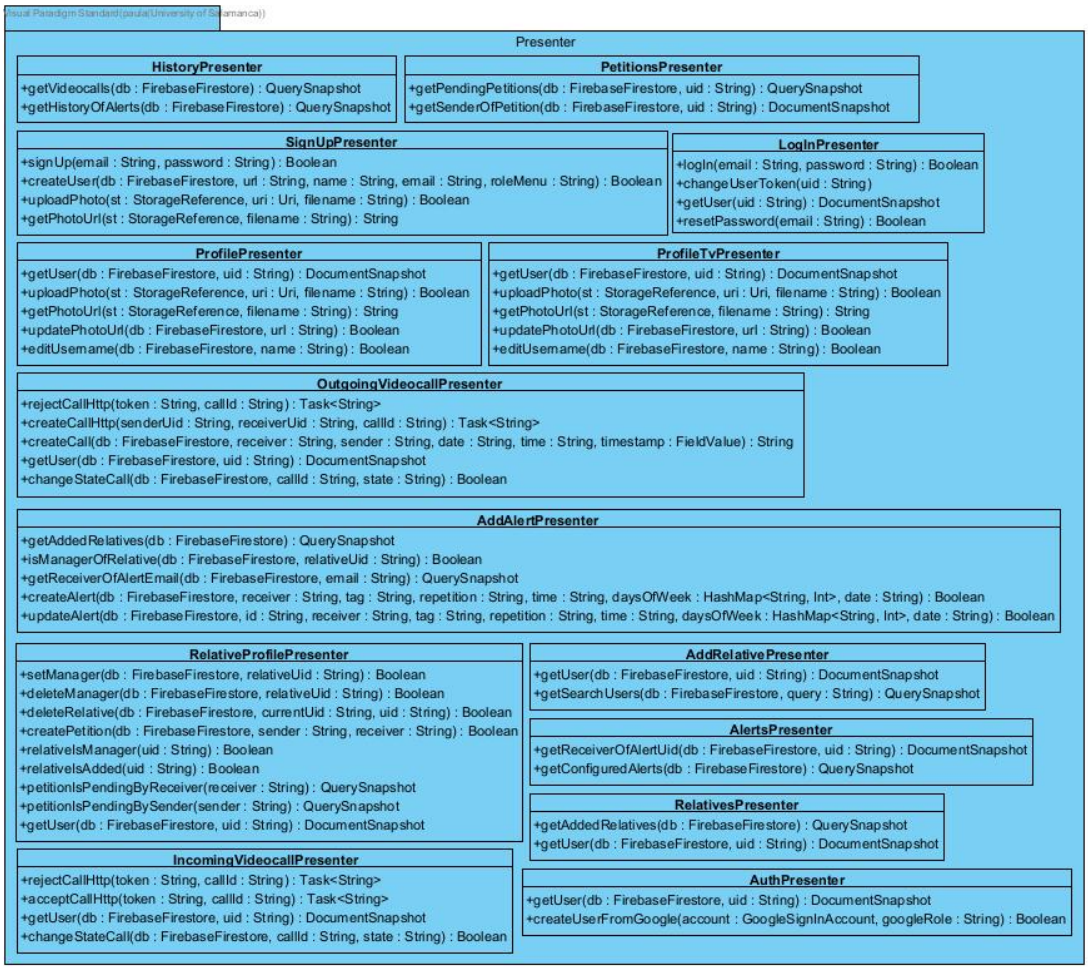


Figura 43: Subsistema Presenter

Vista arquitectónica

Se han organizado las clases de diseño anteriores en función del patrón MVP para tener una vista de las capas y las clases que las forman. En la Figura 44 se puede ver dicha organización de clases:

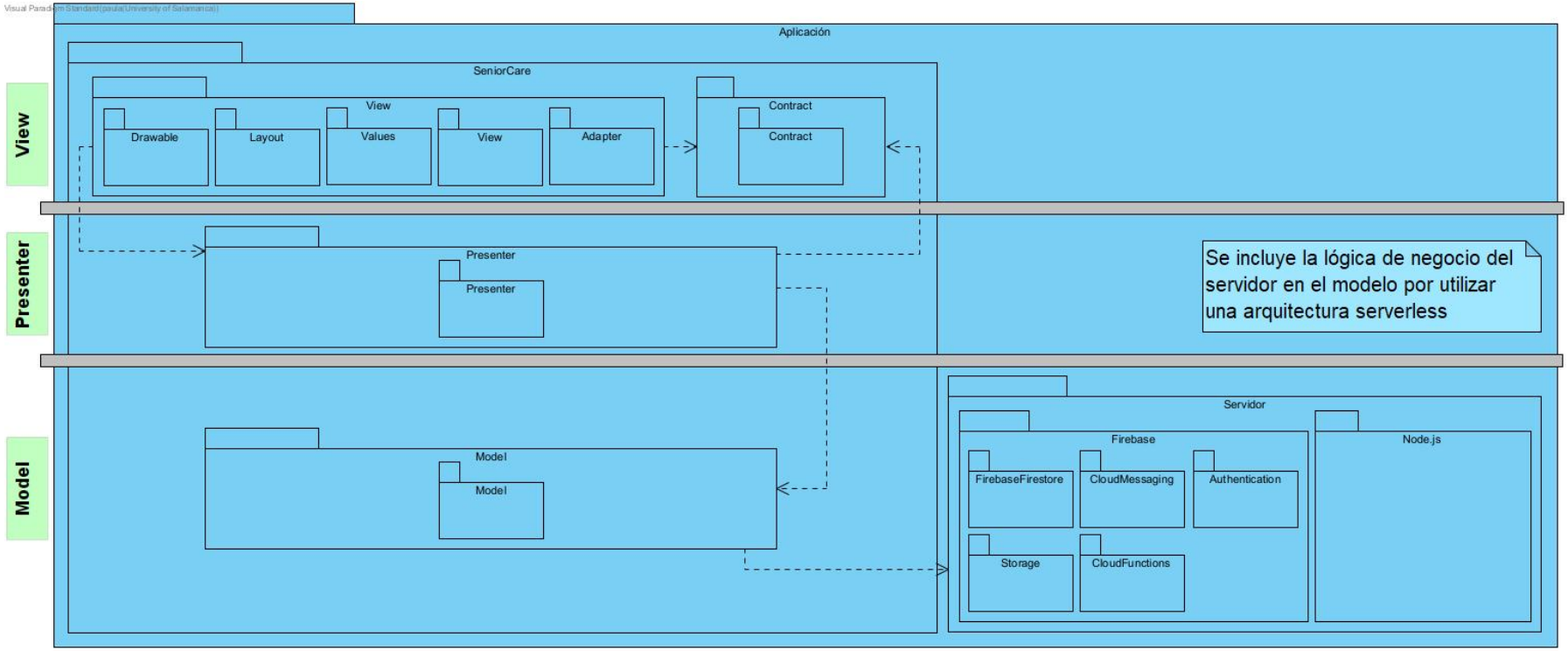


Figura 44: Vista arquitectónica

Realización de casos de uso

A continuación, se han refinado los diagramas de secuencia realizados en el análisis de requisitos hasta obtener una aproximación final de los métodos que se han implementado en la aplicación y empleando las clases de diseño descritas en apartados anteriores.

En la Figura 45 se puede ver un ejemplo de la estructura que siguen los diagramas de secuencia refinados:

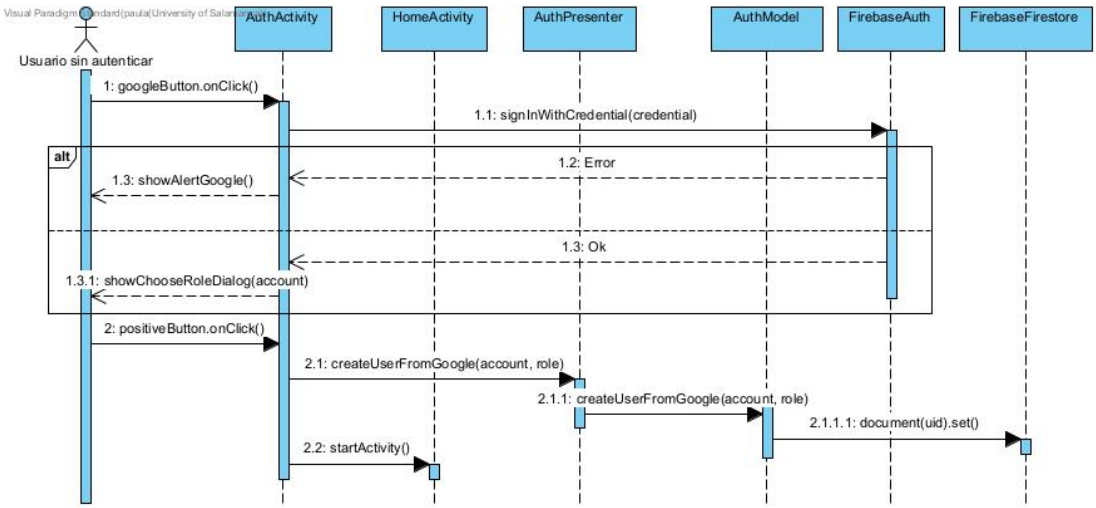


Figura 45: Diagrama de secuencia UC-0002 Iniciar sesión

Diseño de la Base de Datos

Para almacenar y gestionar información, se ha utilizado la base de datos NoSQL de Firebase Firestore, que organiza los datos en colecciones y documentos, es decir, no sigue el esquema tradicional de una base de datos SQL tradicional. Se ha seguido el modelo del dominio realizado antes para satisfacer los requisitos del dominio de la solución.

En la Figura 46 se puede ver el diseño de la base de datos:

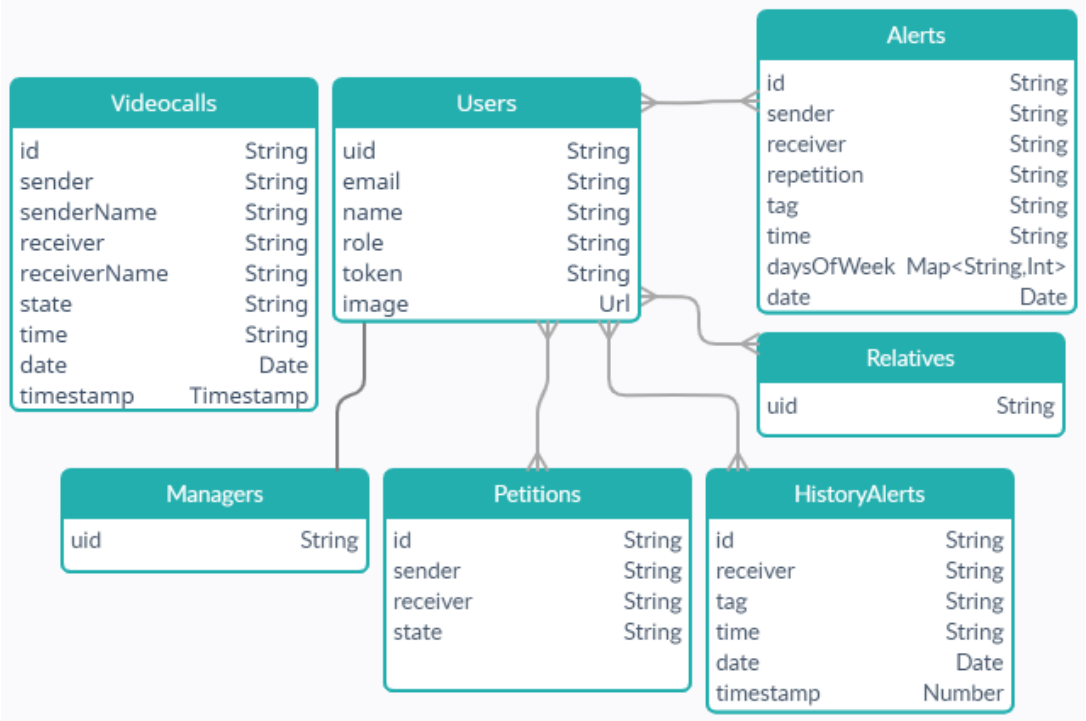


Figura 46: Diseño de la base de datos

Modelo de despliegue

A continuación, en la Figura 47, se muestra el modelo de despliegue desarrollado para el sistema, donde podemos ver la representación física de los componentes software en los nodos físicos, además de los canales a través de los que se comunican.

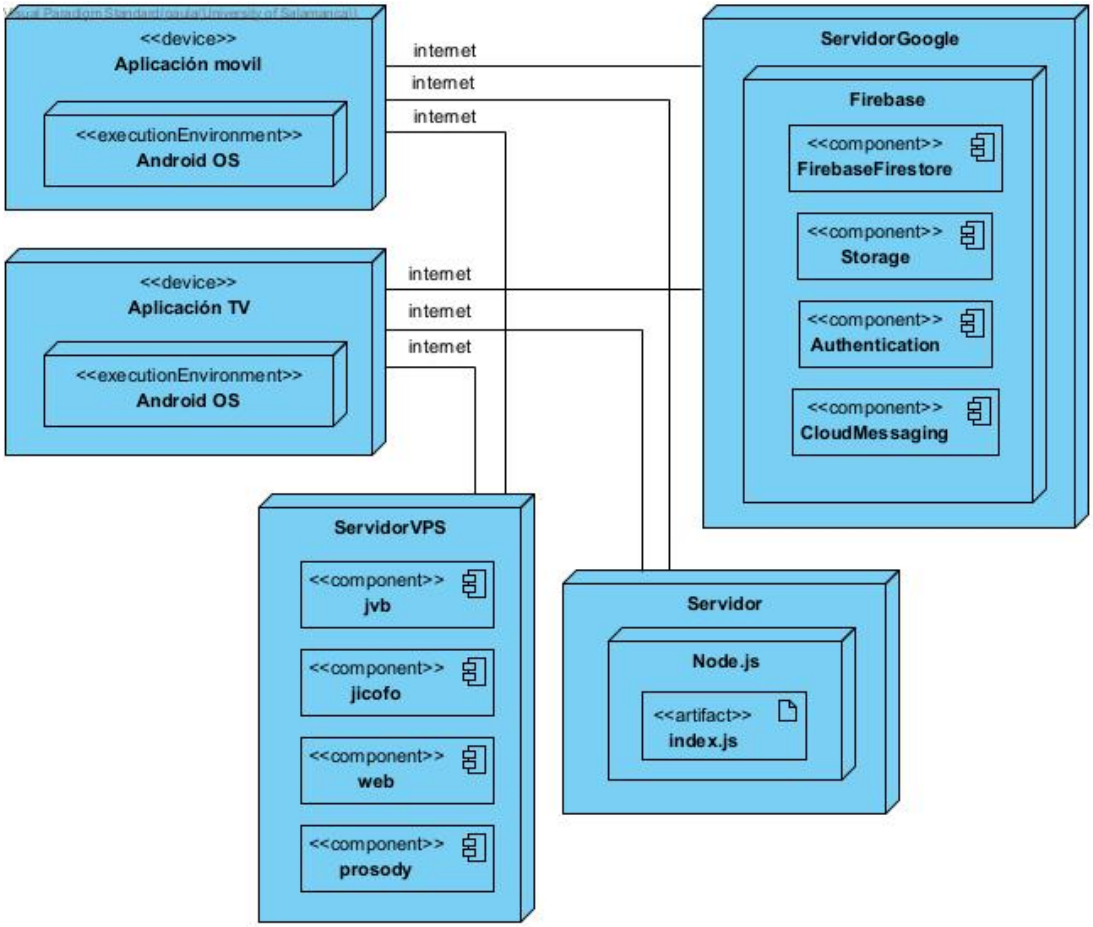


Figura 47: Modelo de despliegue

Para ver los subsistemas de diseño, las clases de diseño y los diagramas de secuencia de los casos de uso en profundidad, o ejemplos de la base de datos, consultar el “Anexo IV: Diseño del sistema software”.

6.7. Implementación

En este apartado se desarrollan los detalles acerca del desarrollo de todo el software necesario para la realización del proyecto, tanto la parte de frontend como la parte de backend. Para realizar la documentación del sistema se ha empleado Dokka [40] y JSDoc [41].

Para controlar las versiones de código y backups, de forma que se puedan administrar cambios en el proyecto a la vez que evoluciona, se ha empleado GitHub [35]. Se puede consultar dicho repositorio en el siguiente enlace: https://github.com/paulasoria/TFG y ver el fichero README.md del proyecto en la Figura 48:

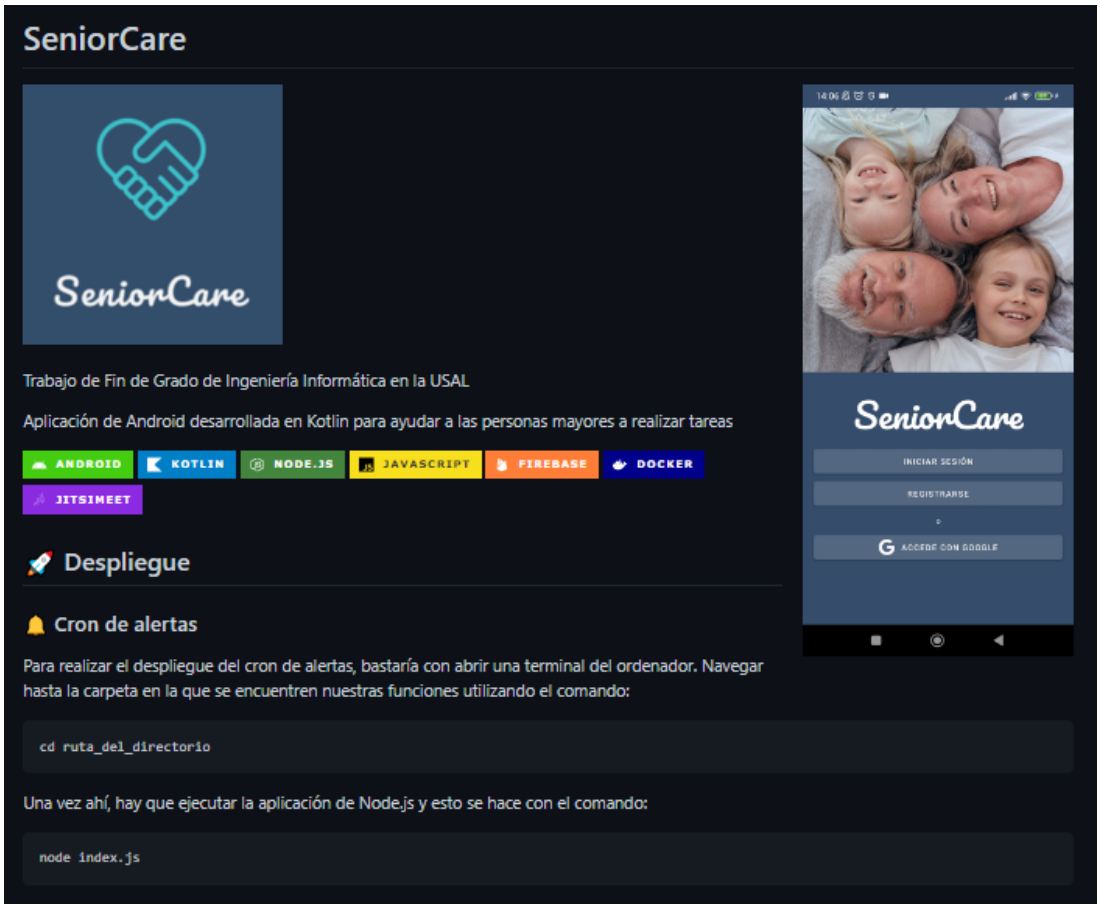


Figura 48: Readme de GitHub

Toda la información detallada está recogida en el “Anexo V: Documentación técnica” [39] de la memoria.

Aplicación Android

El desarrollo de la aplicación ha llevado un tiempo de aprendizaje, porque, aunque se conocían aspectos fundamentales de un proceso de desarrollo, nunca se había implementado un sistema de grandes dimensiones al completo, ni se conocía el lenguaje de programación empleado u otros aspectos avanzados. Además, el uso de herramientas de código abierto ha facilitado la programación al no tener que implementar algunos aspectos de la funcionalidad desde cero.

Servidor

Cron de alertas

Para realizar el cron de alertas en el servidor de Node.js se utilizó el lenguaje JavaScript. Se emplearon también los módulos de “firebase-admin” para acceder a los datos de Firebase, y “cron” para establecer un temporizador que comprueba cada minuto, las alertas establecidas en la base de datos. En el caso de que encuentre alguna, se encarga de enviar una notificación mediante Cloud Messaging y añadir dicha alerta al historial de alertas del usuario.

Jitsi Meet

Para emplear este software de videollamadas, se ha empleado Docker, donde se establecen los parámetros necesarios para la conexión a Internet y se vincula el DNS del dominio creado, y por último, se descargan e inician los contenedores.

Google Cloud Functions

Las funciones de Google Cloud se han desarrollado de manera paralela e incremental, donde algunas funcionan a modo de disparador o trigger cuando se crea un documento en la base de datos, y otras se activan mediante HTTP cuando se produce un evento como, por ejemplo, una videollamada entrante.

En la Figura 49 se muestran algunas de las funciones desplegadas en Google Cloud Functions:

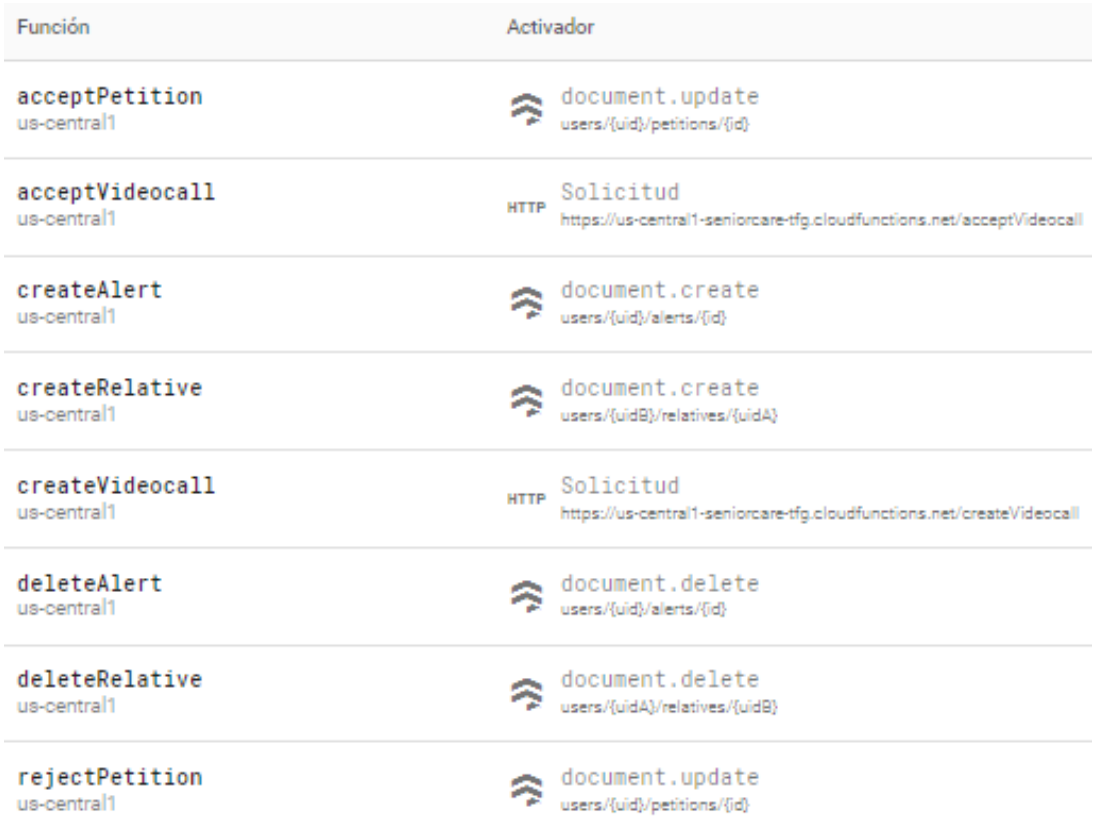


Figura 49: Google Cloud Functions

Para ver detalles del despliegue de las diferentes partes, la estructura del sistema o especificación de la documentación, consultar el “Anexo V: Documentación técnica”.

6.8. Pruebas

Para comprobar y garantizar el correcto funcionamiento del sistema desarrollado se han realizado dos tipos de pruebas, que son:

- **Pruebas unitarias de componentes:** Son aquellas que comprueban cómo funciona una única entidad de la aplicación de forma independiente, es decir, sin tener en cuenta el resto de la interfaz. Se realizaron estas pruebas desde el momento que comenzó el desarrollo del sistema.
- **Pruebas finales de integración:** Son aquellas que comprueban cómo dos entidades se comunican e interactúan entre ellas. Se realizaron estas pruebas cuando se había finalizado la mayor parte del desarrollo del sistema.
- **Pruebas de interfaz de usuario:** Son aquellas en los que se simula cómo interactúa un usuario con la aplicación.

6.9. Funcionalidad del sistema

En esta sección se hace un resumen del funcionamiento e interacción de la aplicación de forma que sea comprensible por el usuario. Toda la información detallada está recogida en el “Anexo VI: Manual de usuario”[39] de la memoria.

Se puede consultar un vídeo del funcionamiento de la aplicación en el siguiente enlace de Google Drive: https://drive.google.com/file/d/1T6qbutjrM_4z3Z9Zcs0uW_X2WMO0Lc6p/view?usp=sharing

Aplicación Android

En la Figura 50 tenemos la pantalla inicial desde la que podemos acceder a la aplicación de tres formas diferentes.

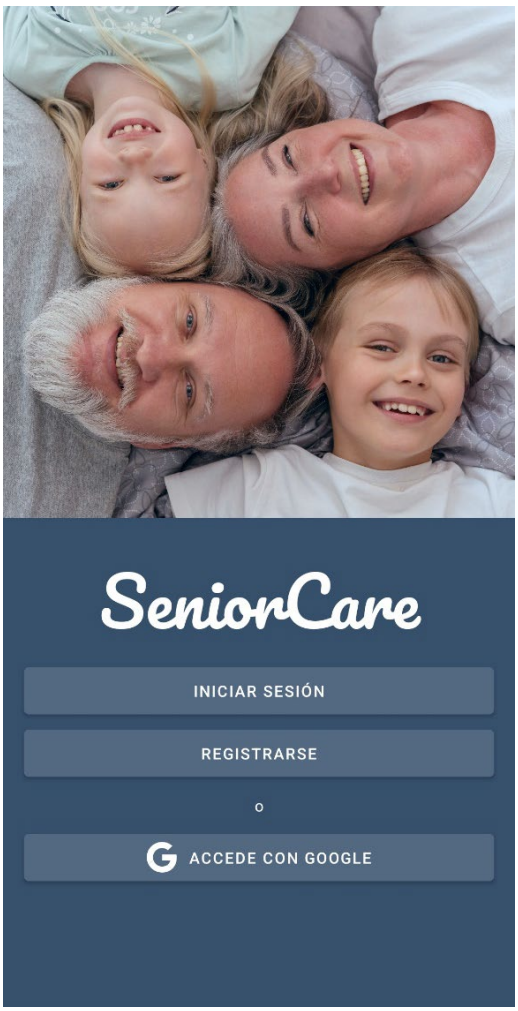


Figura 50: Pantalla inicial

Al hacer clic sobre el botón de iniciar sesión nos llevaría a la ventana que vemos en la figura de abajo, donde tendremos que introducir nuestros datos, como vemos en la Figura 51.

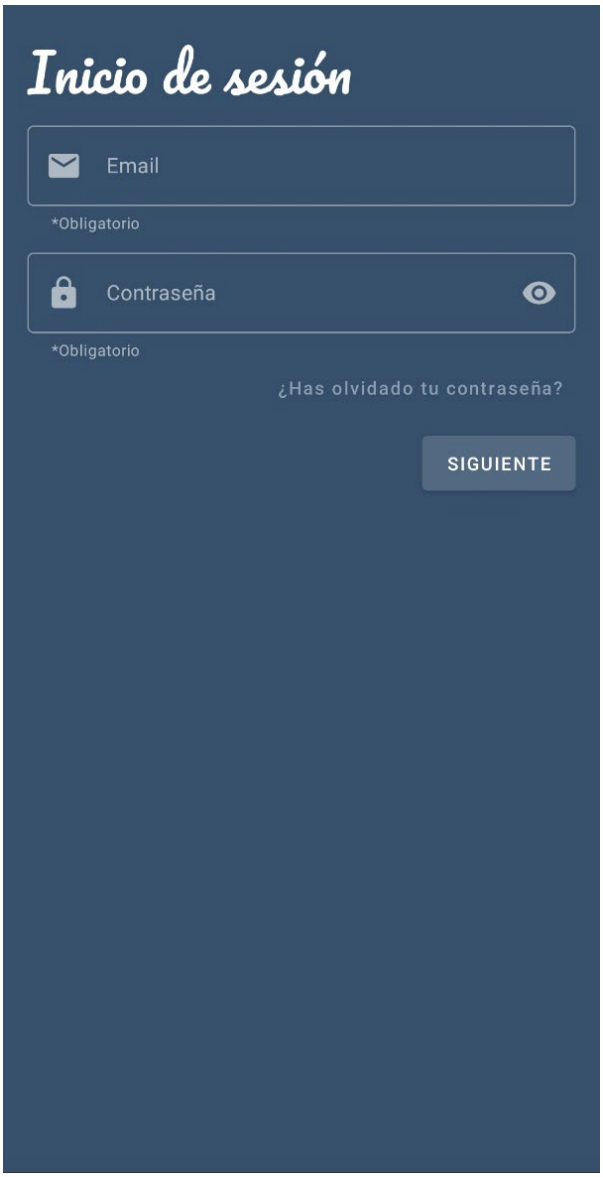


Figura 51: Inicio de sesión

Tenemos un botón para solicitar la recuperación de la contraseña, como se muestra en la Figura 52.

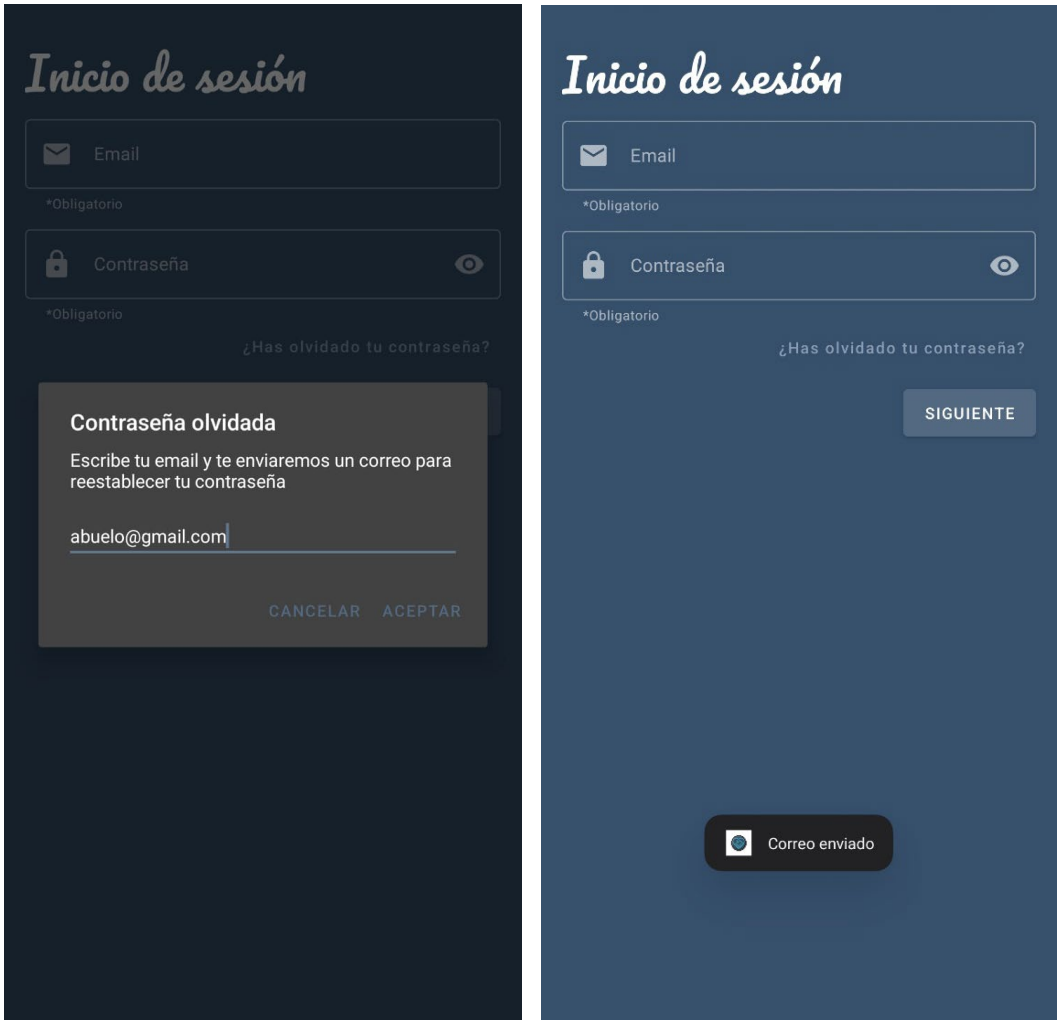


Figura 52: Contraseña olvidada

Si hacemos clic sobre el botón de registro nos llevaría a la ventana que vemos en la Figura 53, donde tendremos que introducir nuestros datos.

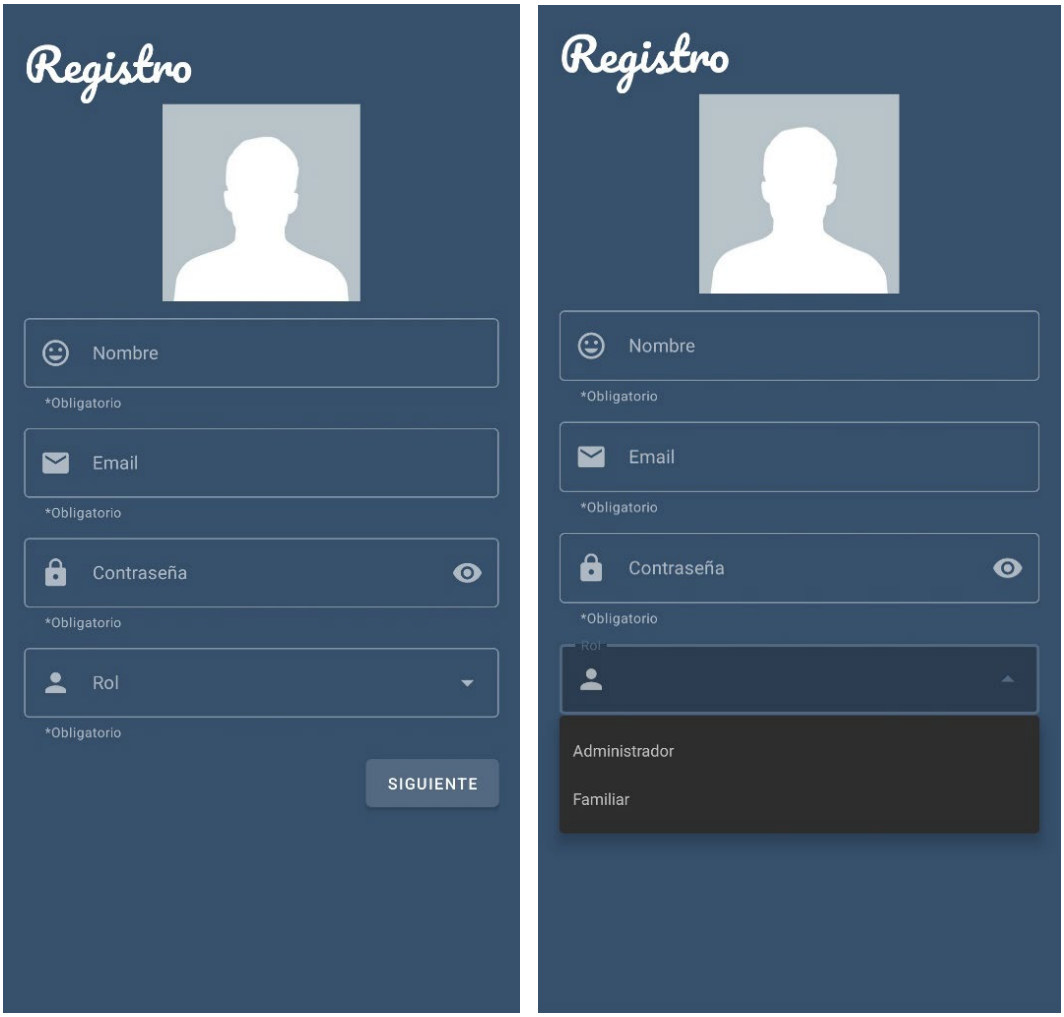


Figura 53: Registro

Si hacemos clic en el botón de acceder con Google nos llevaría a la ventana que vemos en la Figura 54, donde seleccionaríamos nuestro correo.

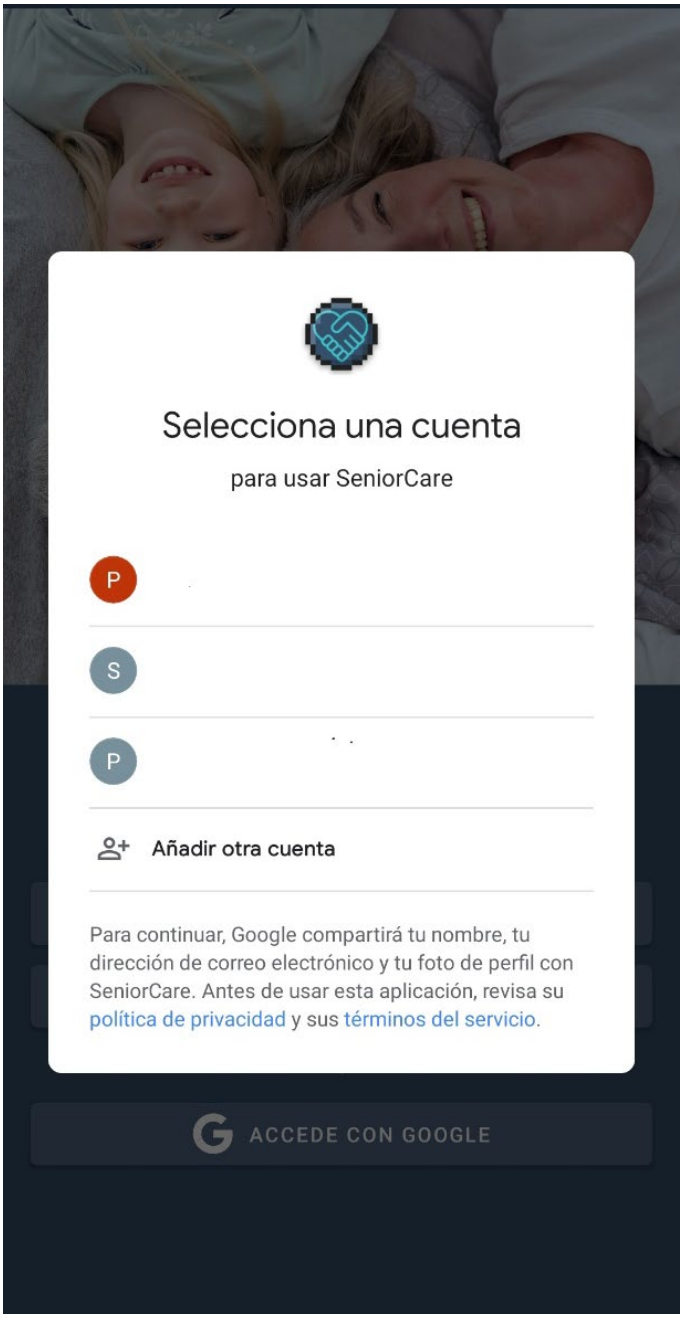


Figura 54: Acceso con Google

Nada más acceder a la aplicación podemos ver el historial de videollamadas o el historial de alertas, como se muestra en la Figura 55.

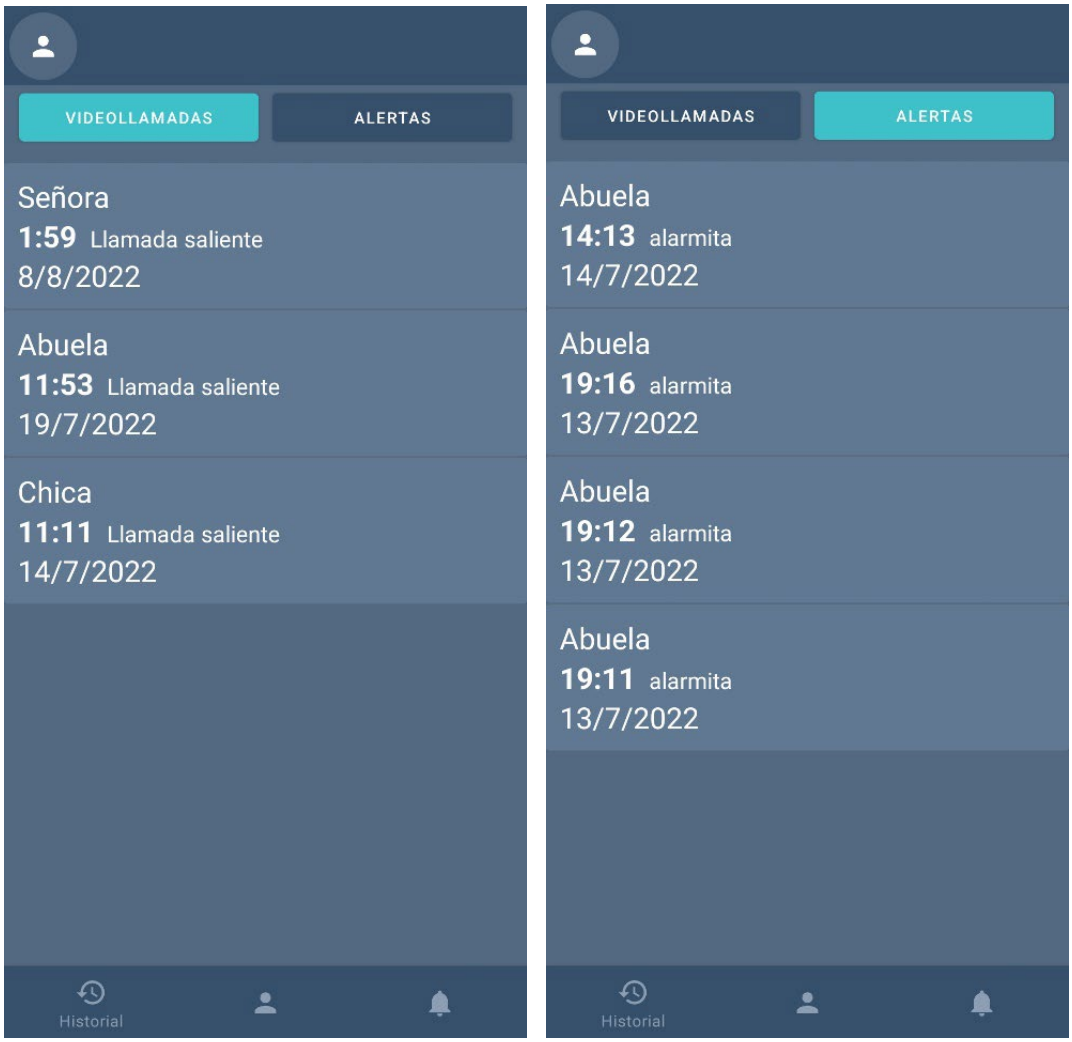


Figura 55: Historial de videollamadas y alertas

Si navegamos a la pantalla de familiares veremos un listado junto con algunos detalles de estos usuarios, como se ve en la Figura 56.

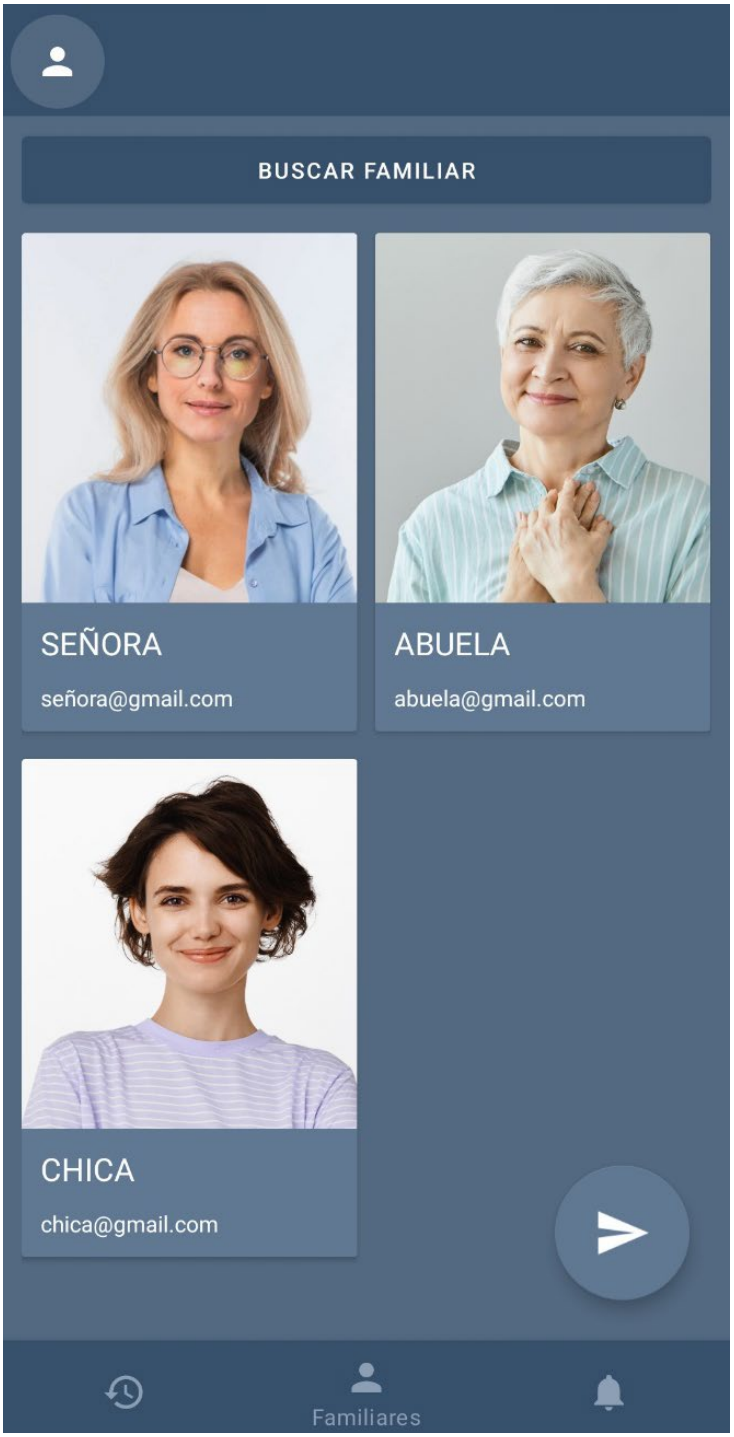


Figura 56: Familiares añadidos

En el botón de búsqueda de familiares tendremos la opción de buscar usuarios dentro de la base de datos del sistema, como se ve en la Figura 57.

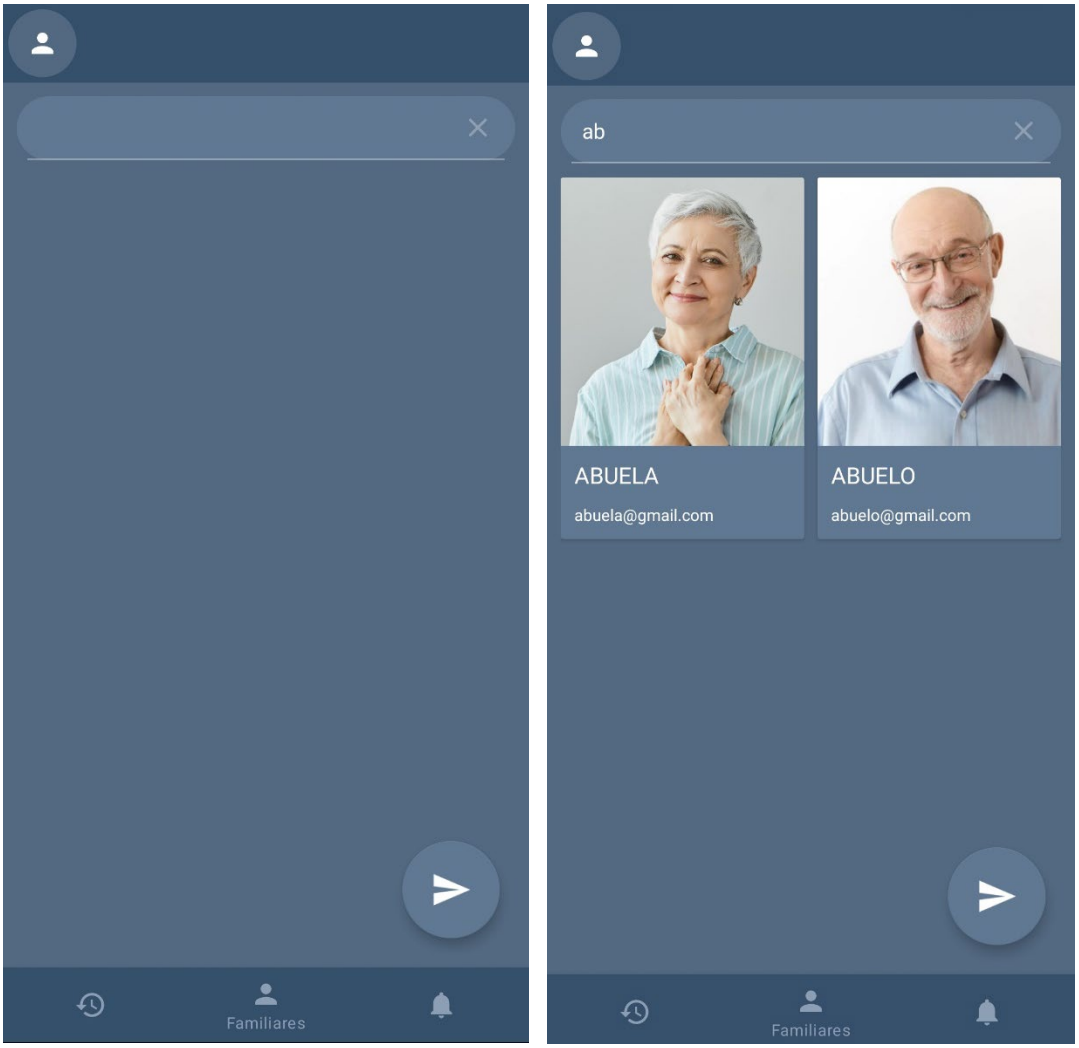


Figura 57: Buscar familiares

Podemos consultar el perfil del usuario que queramos, teniendo más información y funcionalidad que desde el listado. En la Figura 58 se muestra el perfil de un usuario:

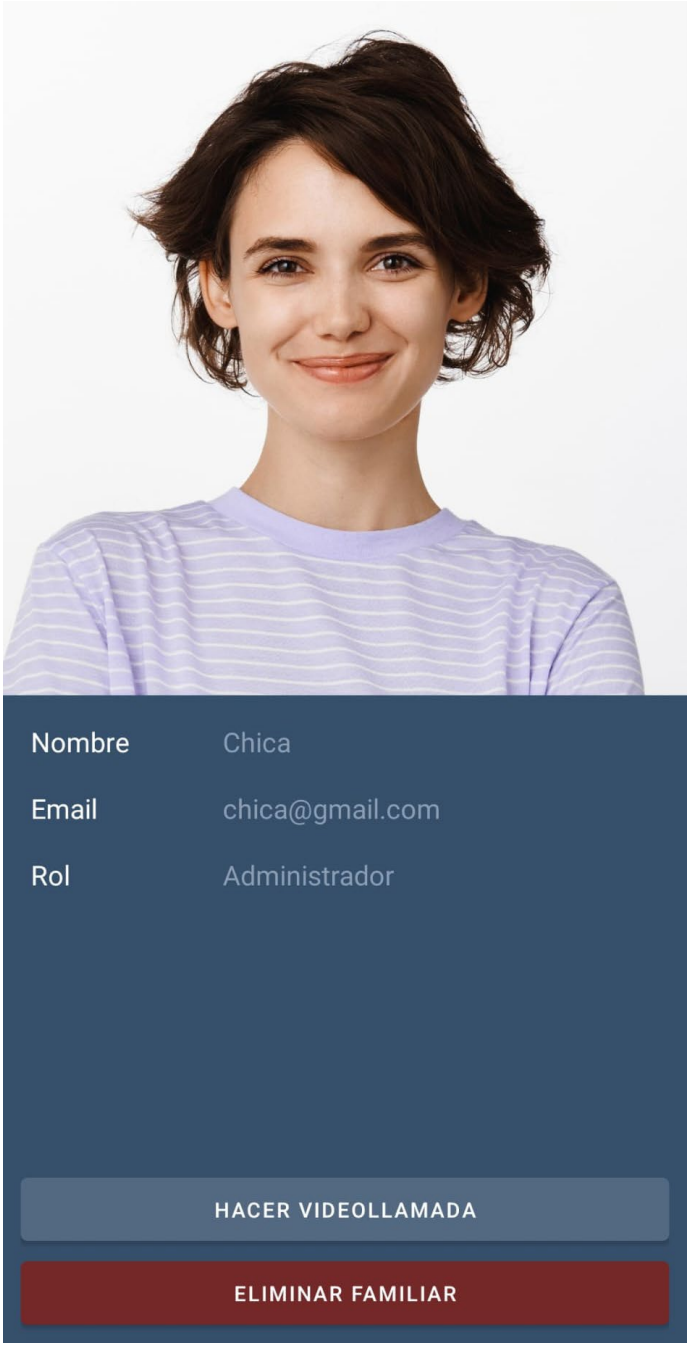


Figura 58: Perfil del familiar

Si hacemos clic sobre el botón de realizar una videollamada, se enviará una solicitud de llamada al familiar, como se ve en la Figura 59.

Figura 59: Videollamada saliente

Y el familiar receptor verá la pantalla de la Figura 60:

Figura 60: Videollamada entrante

Podemos enviar solicitudes de familiar a aquellos que no tengamos agregados, como se ve en la Figura 61.

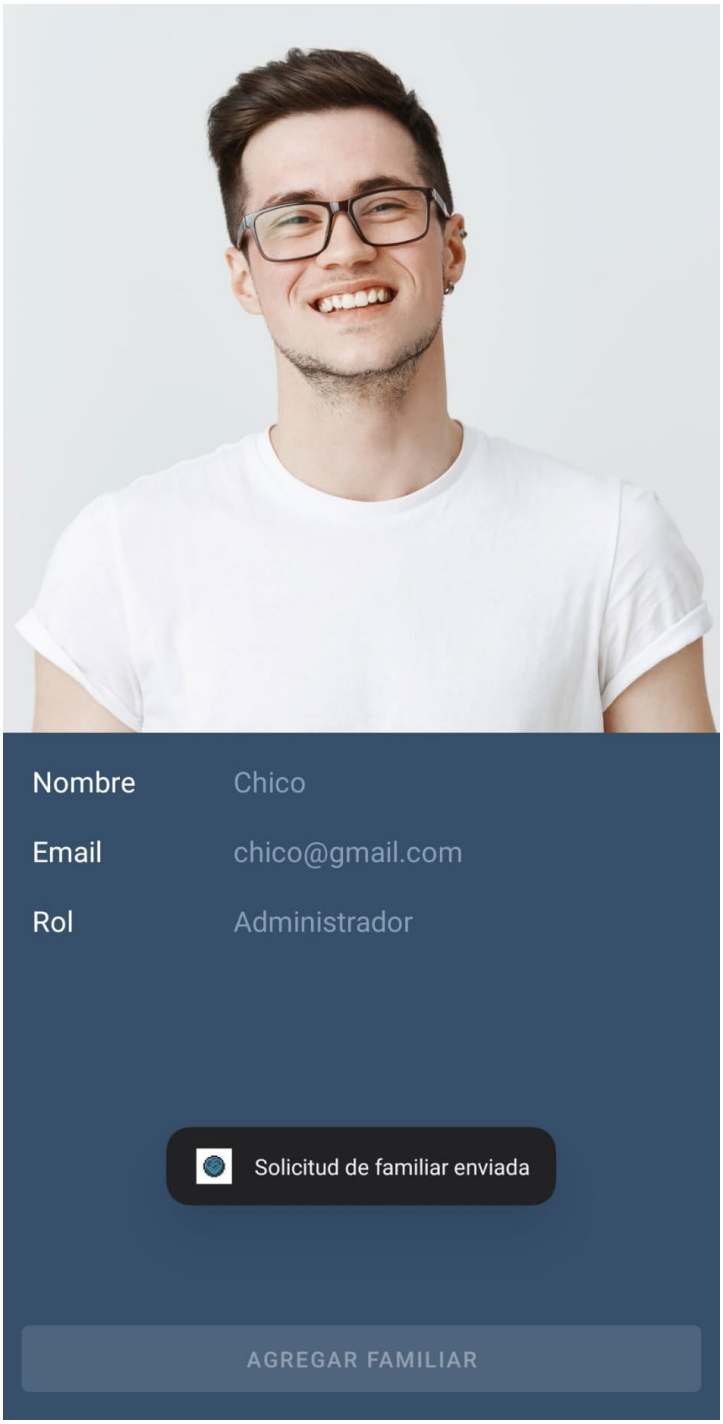


Figura 61: Petición de familiar enviada

Al hacer clic sobre el botón flotante nos lleva a la ventana de solicitudes, que es la que se muestra en la Figura 62.

Figura 62: No hay familiares agregados

En la Figura 63 se muestra la pantalla donde veremos las solicitudes pendientes, y podremos aceptarlas o rechazarlas.

Figura 63: Petición pendiente

Podemos desplazarnos hasta la ventana de alertas, que se muestra en la Figura 64, donde veremos un listado de las alertas que tenemos establecidas.

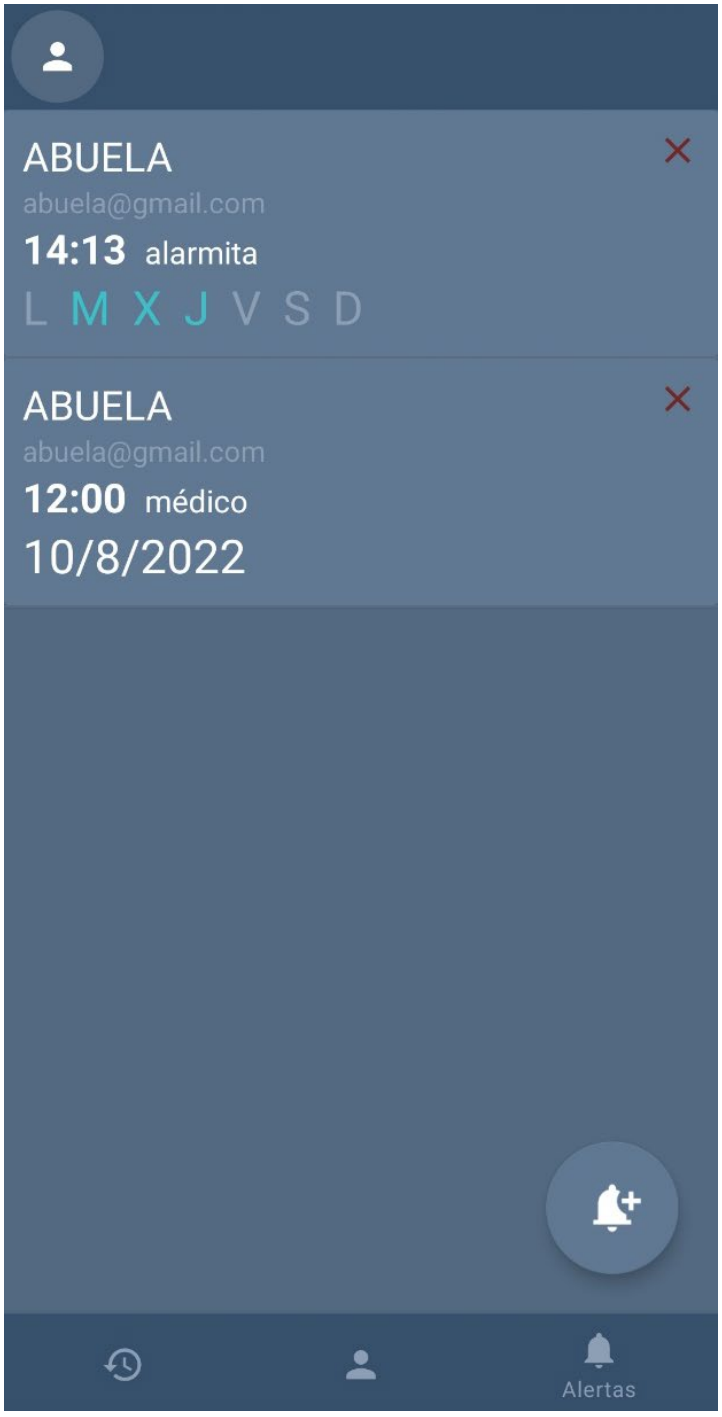


Figura 64: Alertas creadas

En la Figura 65 se muestra cómo podemos crear una nueva alerta introduciendo los datos solicitados.

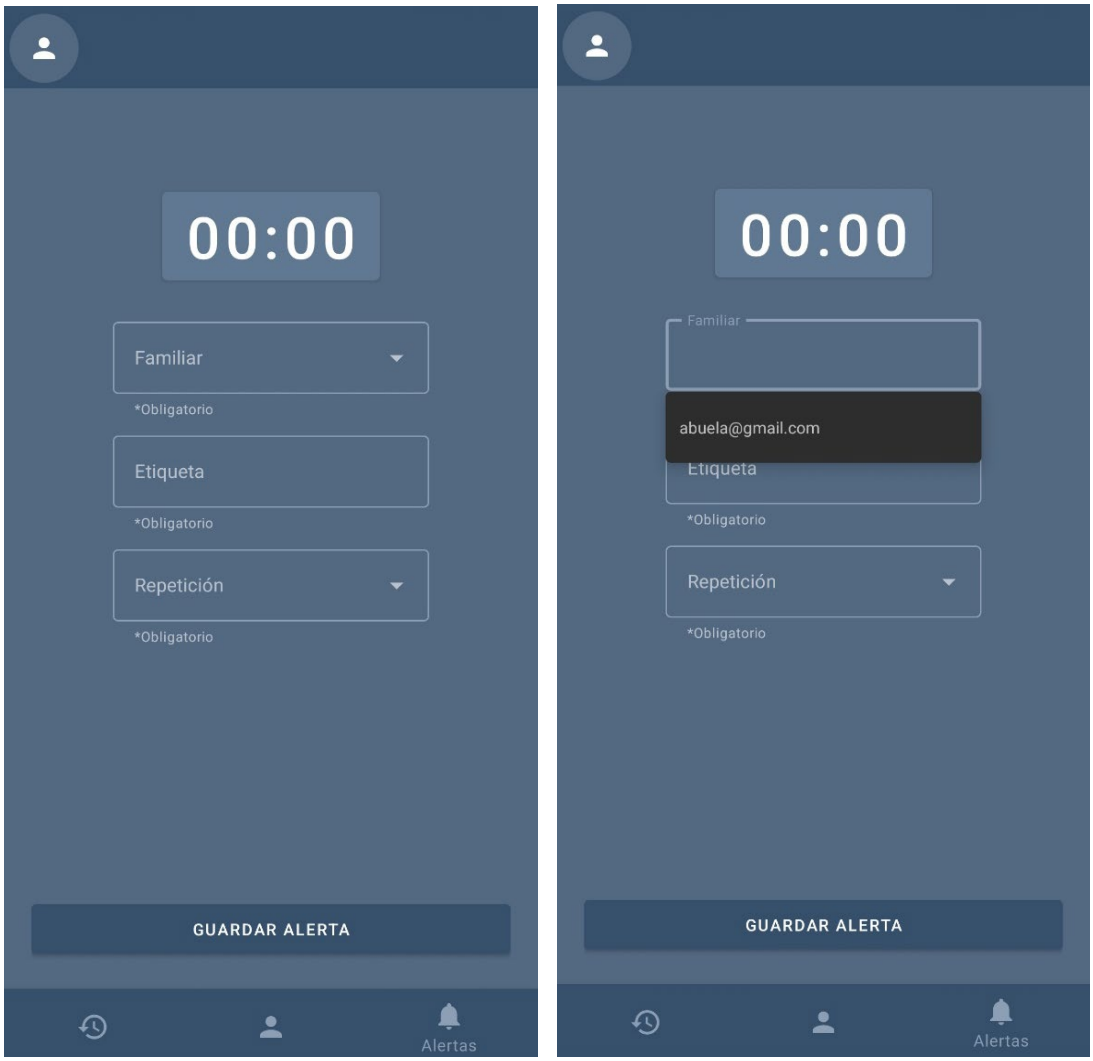


Figura 65: Crear nueva alerta

En la Figura 66 se muestra cómo nos permite introducir la hora de dicha alerta en dos formatos distintos.

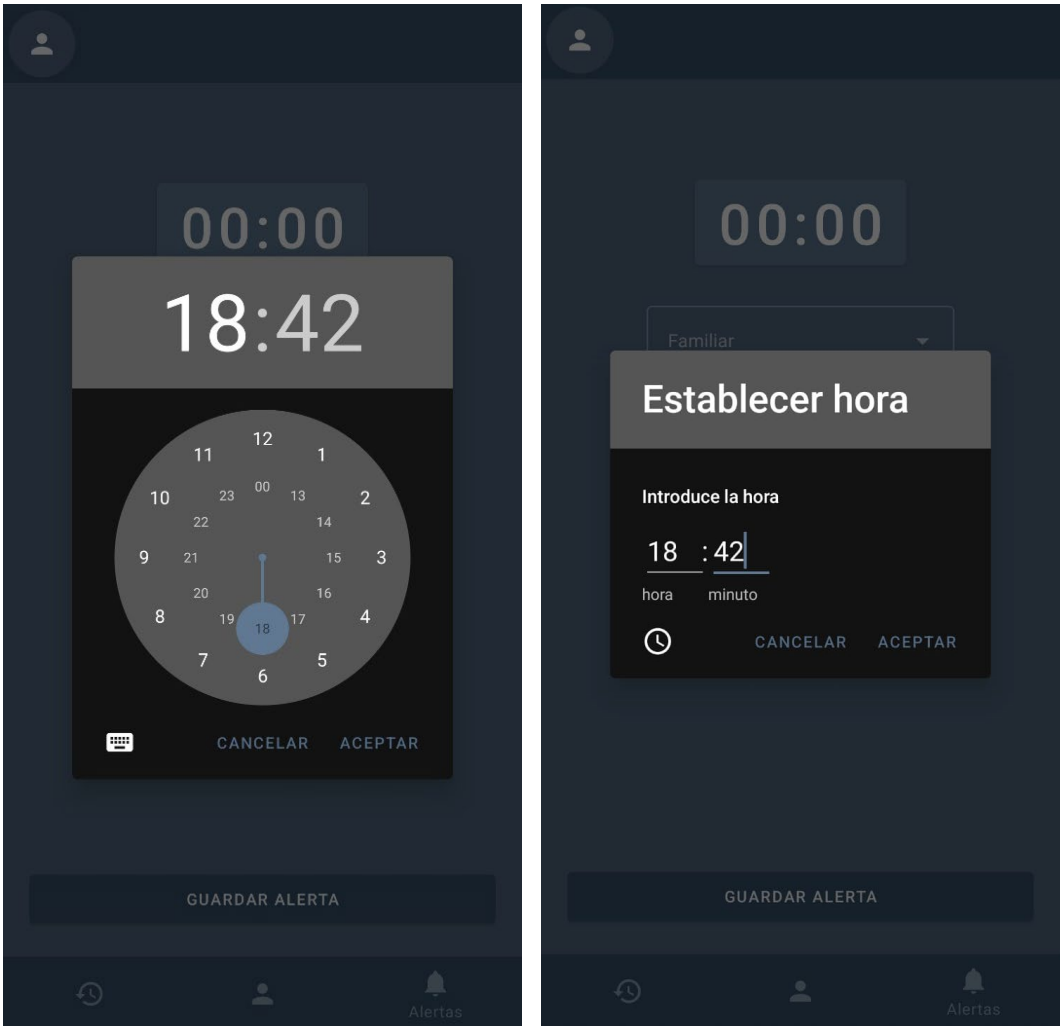


Figura 66: Seleccionar hora de la alerta

La alerta creada puede ser de tipo “Eventual” como se muestra en la Figura 67:

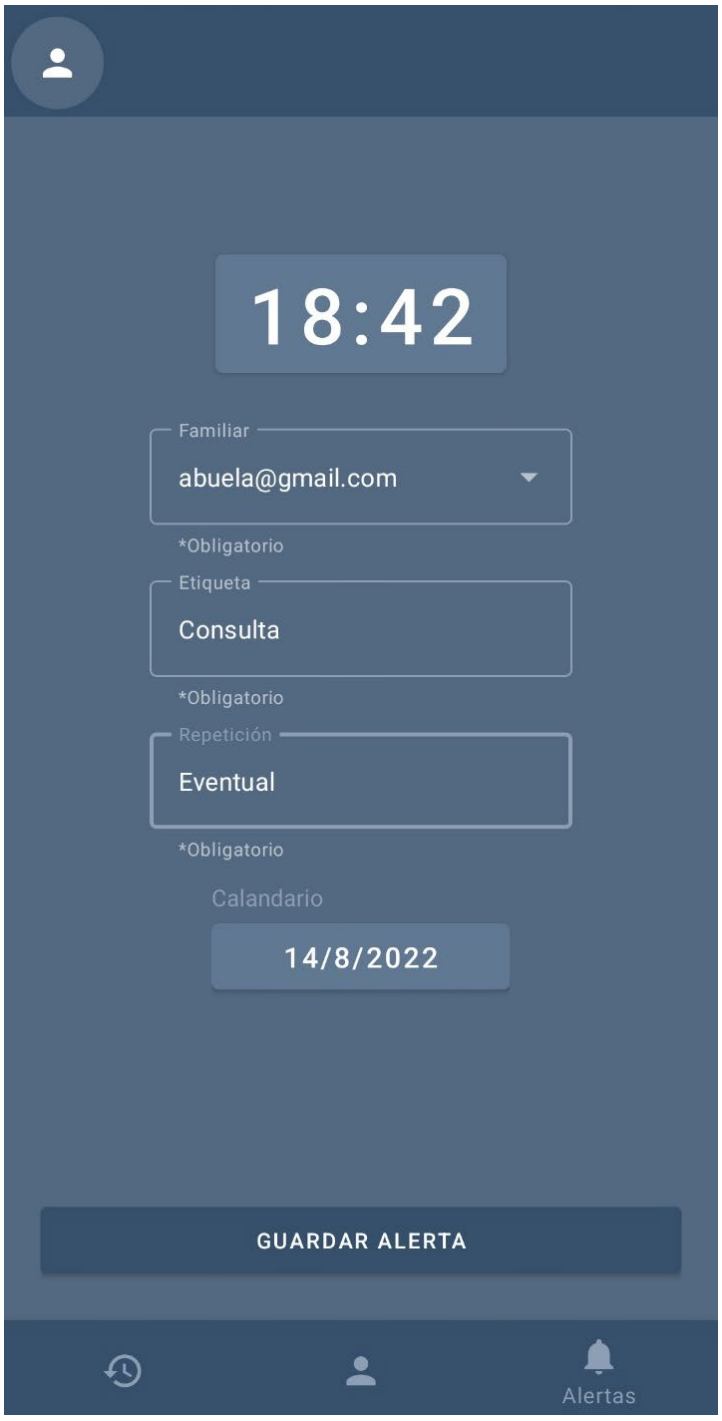


Figura 67: Alerta eventual

En la Figura 68 se puede establecer la fecha en qué sucede dicho evento.

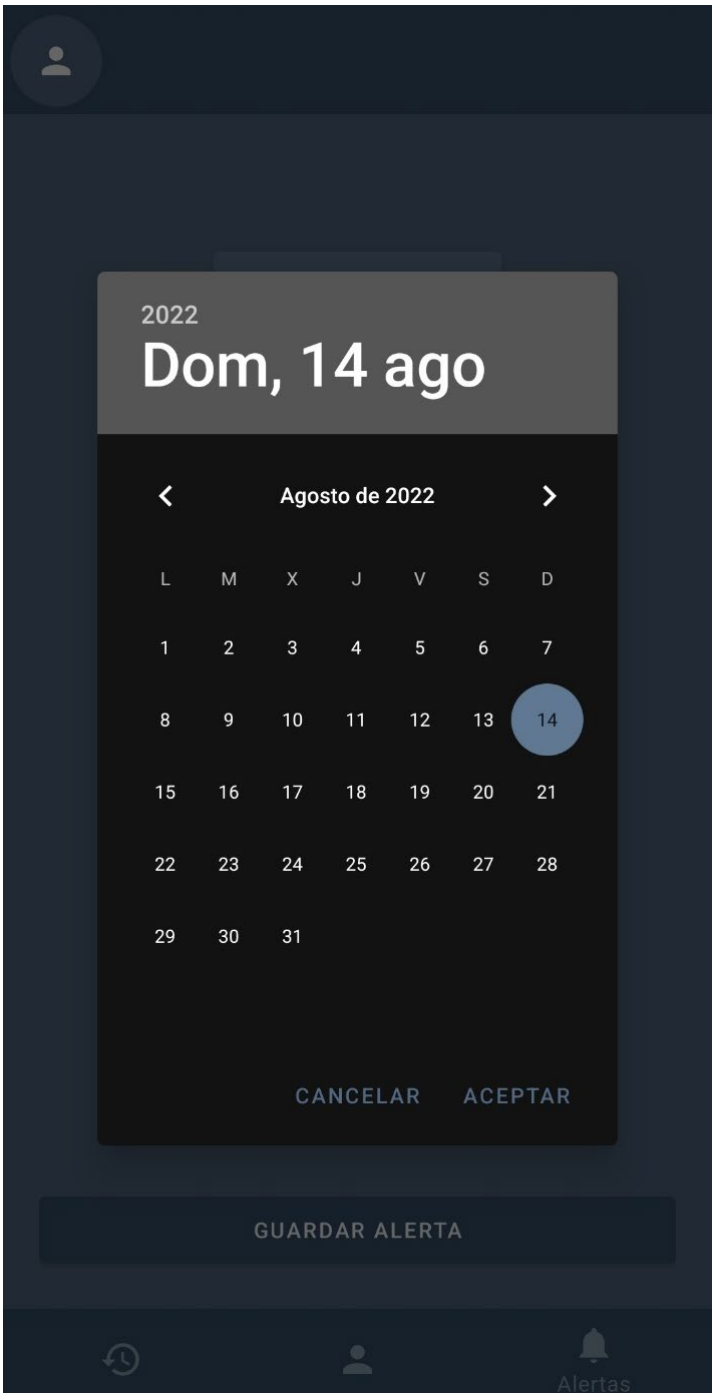


Figura 68: Seleccionar fecha

O puede ser una alerta del tipo “Semanal” y seleccionar los días de la semana, como se ve en la Figura 69.

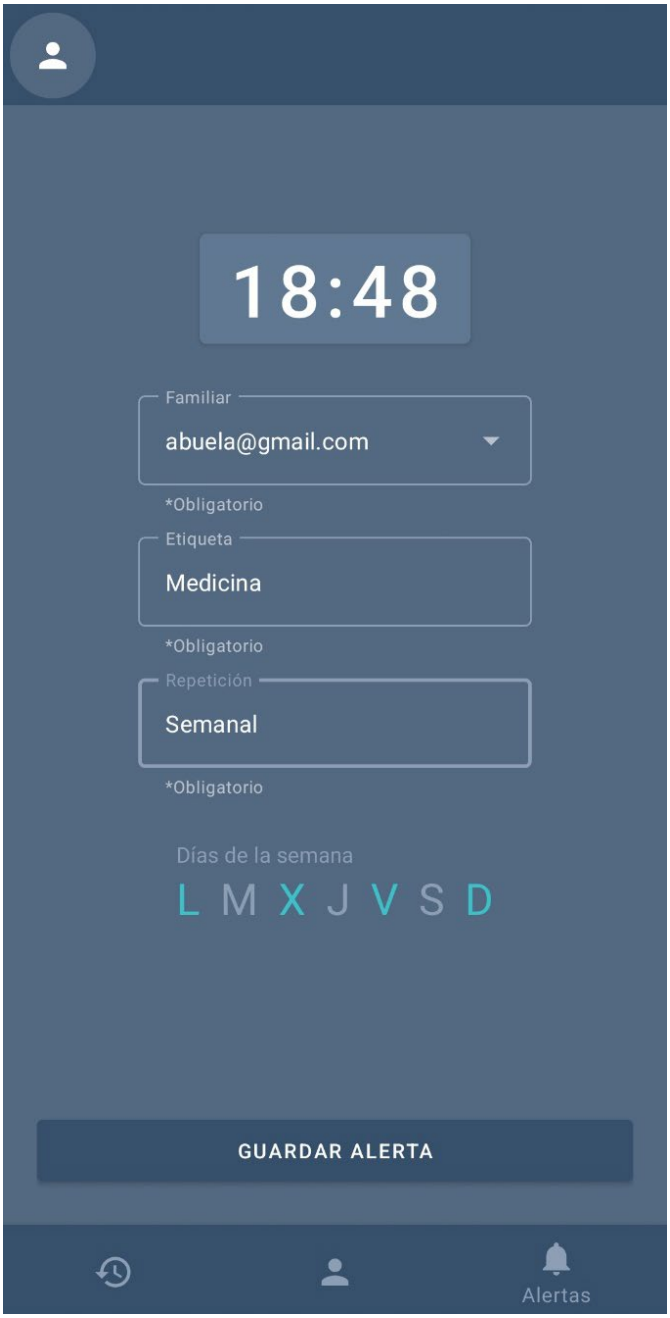


Figura 69: Alerta semanal

También tenemos la opción de eliminar alertas establecidas, como se puede ver en la Figura 70.

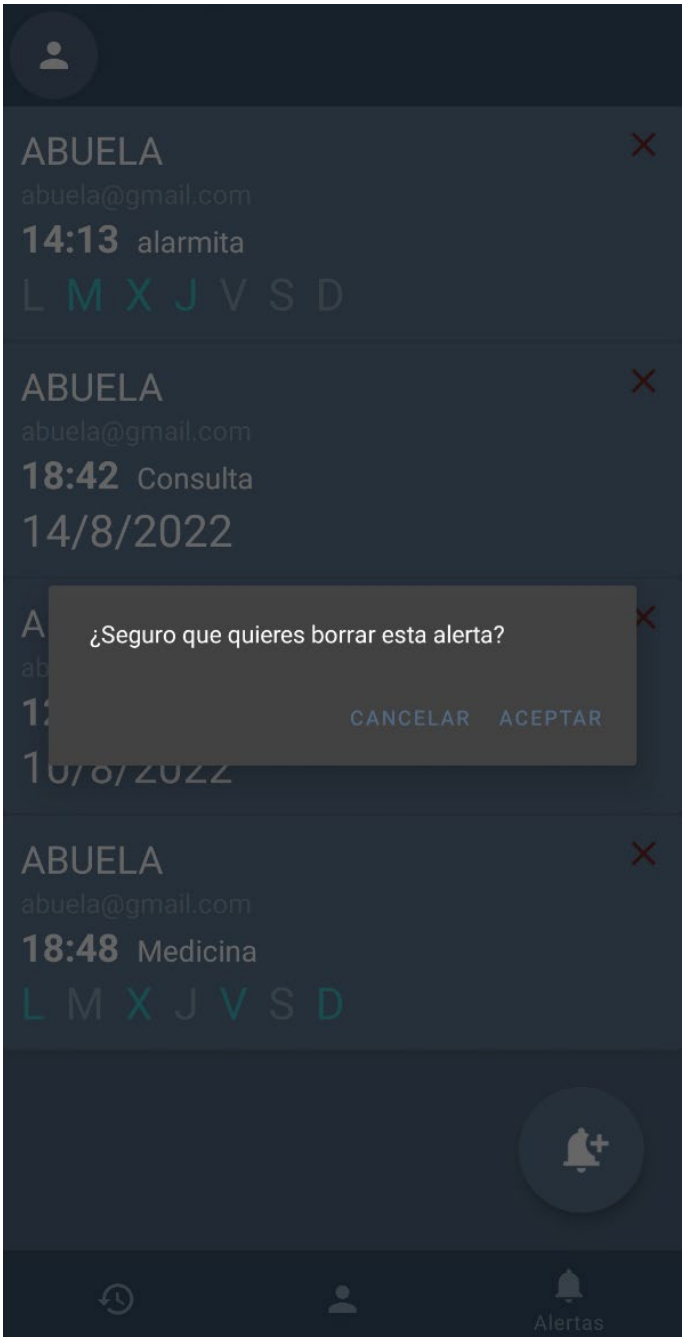


Figura 70: Eliminar alerta

Haciendo clic sobre el icono superior izquierdo se ve la pantalla de la Figura 71 y se accede al perfil del usuario.

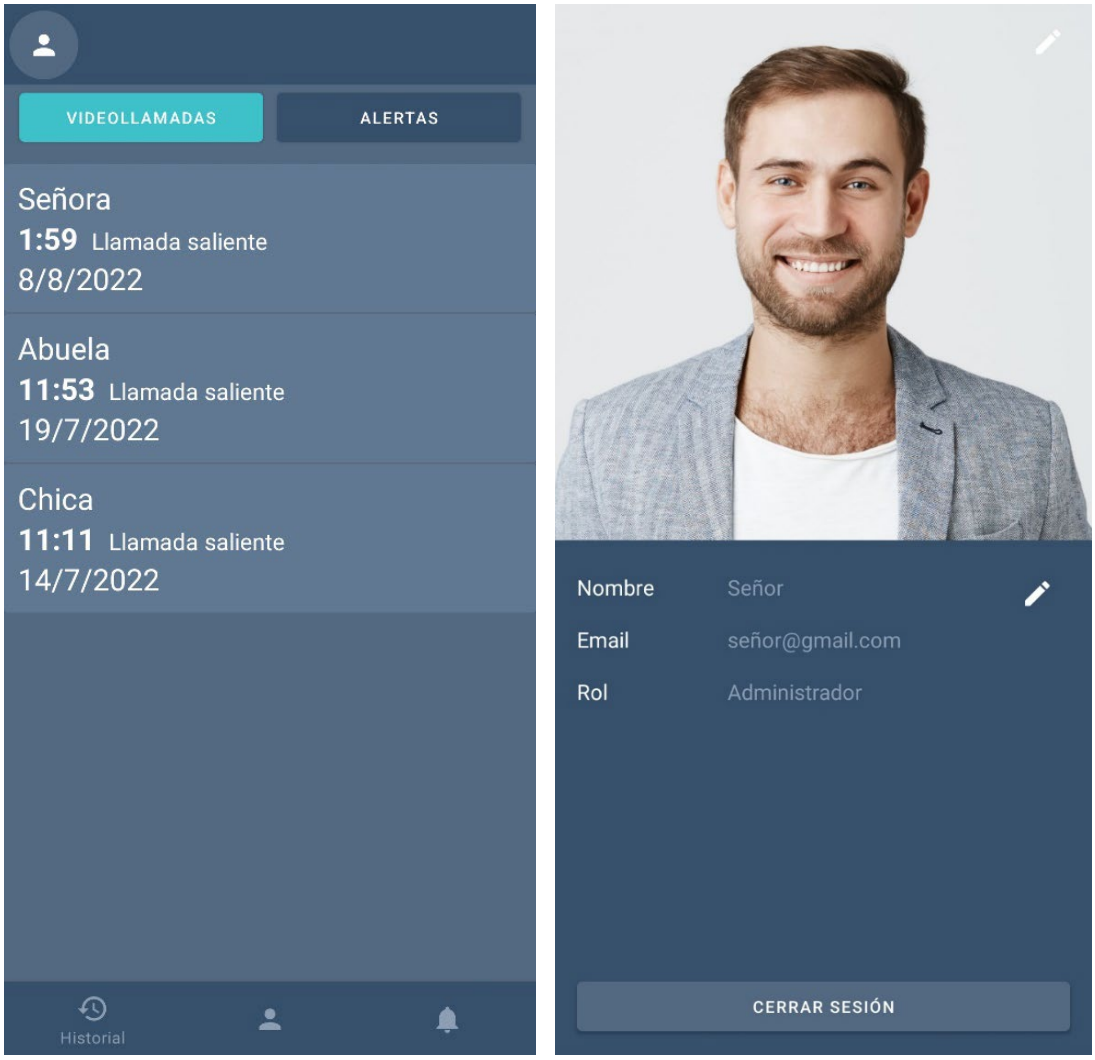


Figura 71: Perfil de usuario

Se puede modificar nuestra foto de perfil escogiendo una de la galería, como se muestra en la Figura 72.

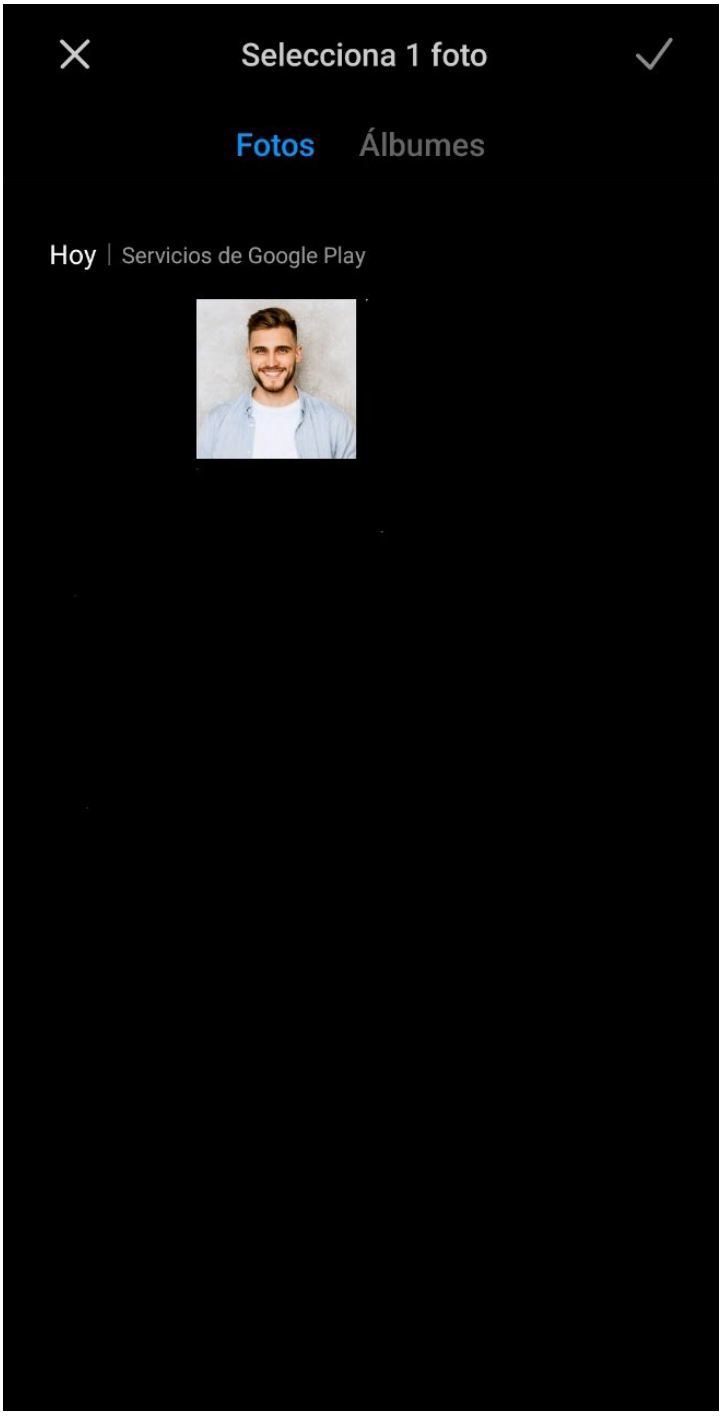


Figura 72: Modificar imagen de usuario

También se puede modificar el nombre de usuario, como se ve en la Figura 73.

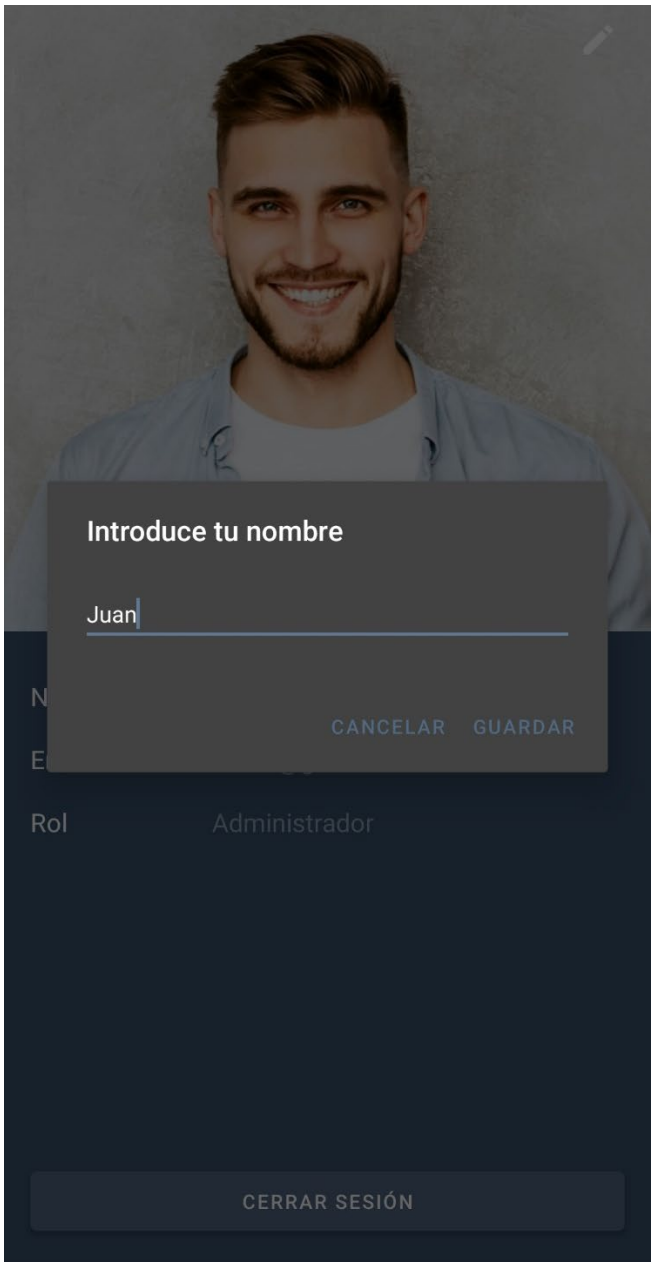


Figura 73: Modificar nombre de usuario

Al darle al botón de cerrar sesión de la Figura 74, se sale a la pantalla inicial y se borran nuestros datos guardados.

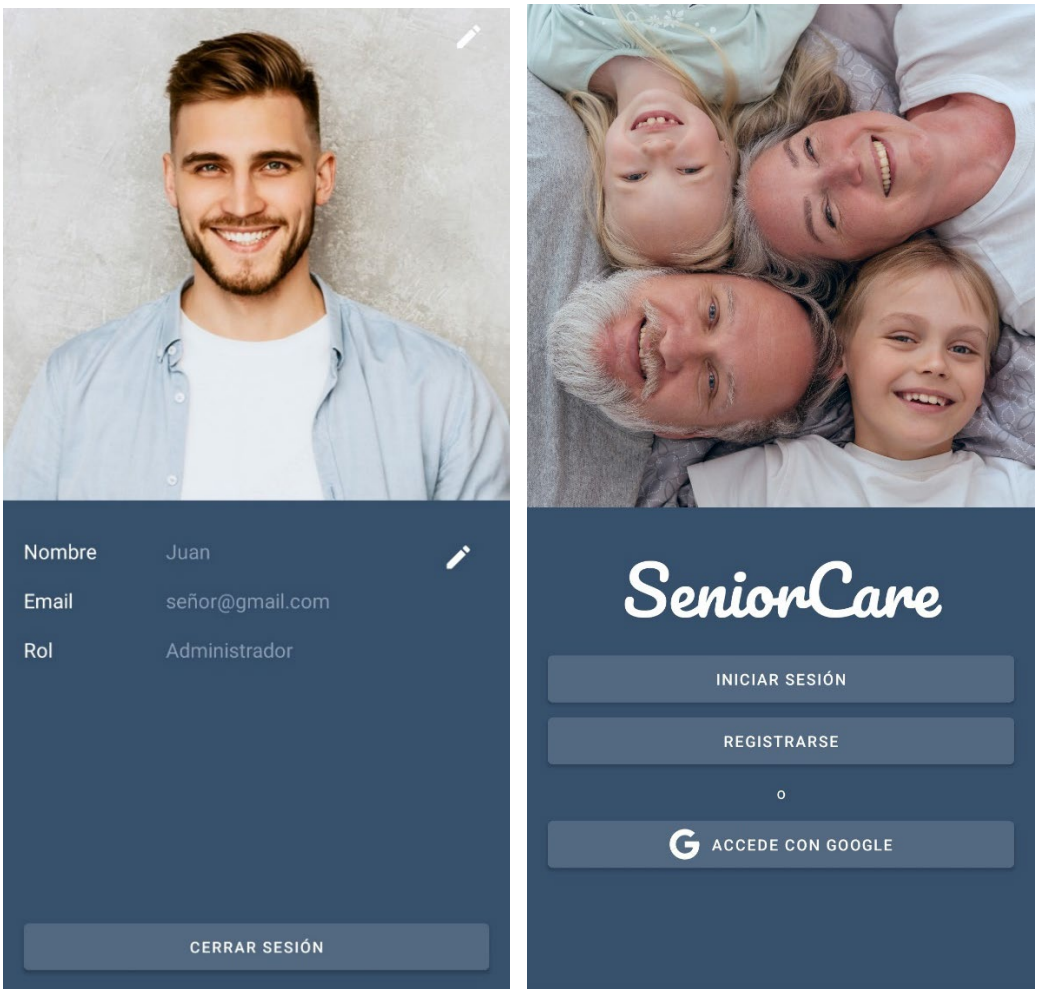


Figura 74: Cerrar sesión

Para ver una explicación más detallada del funcionamiento de la aplicación y ver los detalles que diferencian la aplicación móvil de la aplicación de TV, consultar el “Anexo VI: Manual de usuario”.

7. Conclusiones y líneas de trabajo futuras

Al terminar el desarrollo del sistema, es posible sacar una serie de conclusiones del trabajo realizado y proponernos unas líneas de trabajo futuras para mejorarlo.

7.1. Conclusiones

Podemos concluir que hemos cumplido los objetivos y requisitos propuestos:

- La aplicación es capaz de permitir el registro y acceso de usuarios, además de poder ver y modificar diferentes aspectos de su perfil.
- Los usuarios de la aplicación pueden buscar a través del email, enviarles solicitudes de familiar y recibirlas, además de tener agregados a familiares.
- Se pueden realizar con éxito videollamadas entre usuarios y que quede registrado de forma correcta sobre el historial.
- También se puede realizar con éxito la creación y modificación de alertas, se reciben por los otros usuarios y también quedan registradas de forma correcta sobre el historial.
- El sistema almacena y recupera de forma eficiente los datos acerca de usuarios, alertas o videollamadas de la base de datos, además de mostrar correctamente esta información.
- La aplicación es compatible con dispositivos móviles y televisiones que tengan Android TV.
- La navegación por la aplicación es fluida y sencilla como se esperaba conseguir para que pudiese ser utilizada por cualquier usuario.
- También se ha aprendido a desarrollar un entorno de *backend* utilizando Node.js y se ha aprendido el lenguaje de JavaScript.
- Se ha aprendido a aplicar herramientas útiles como Firebase con nuevas formas de estructuración de bases de datos NoSQL.
- Se ha puesto en práctica el uso de metodologías y patrones que caracterizan la Ingeniería de Software.
- Se ha cumplido el objetivo personal de aprender a desarrollar aplicaciones Android, utilizando un entorno nuevo como es Android Studio, y aprendiendo nuevos lenguajes de programación como Kotlin.
- Por último, he conseguido solucionar de forma autónoma algunos problemas que han surgido durante el desarrollo y he mejorado mi capacidad de buscar los recursos necesarios.

7.2. Líneas de trabajo futuras

Se van a proponer nuevas características que se podrían añadir a la aplicación, así como líneas de trabajo futuras:

- Añadir la funcionalidad de videollamadas grupales entre varios usuarios de la aplicación en lugar de sólo entre dos familiares. De forma que la comunicación entre una familia pueda ser en una única reunión conjunta.
- Añadir un servicio de mensajería entre los usuarios donde haya otros métodos de comunicación que no sean sólo mediante videollamadas, donde poder enviar mensajes de texto, fotos, videos, etc.
- Realizar un mantenimiento correcto del sistema y una ampliación de la funcionalidad a largo plazo. Por ejemplo, adaptando la interfaz gráfica del sistema a tablets o añadiendo un calendario donde ver alertas eventuales programadas.
- Elaborar una aplicación que sirva para otros sistemas operativos que no sean Android para abarcar mayor público, como por ejemplo Apple.

8. Referencias

[1] María Sosa Troya, «Más de 850.000 personas mayores de 80 años viven solas en España», *El País*, abr. 2019, Accedido: sep. 01, 2022. [En línea]. Available: https://elpais.com/sociedad/2019/04/02/actualidad/1554207493_844264.html

[2] «La brecha digital preocupa al 76% de las personas mayores de 80 años», *La Razón*, jul. 2022, Accedido: sep. 01, 2022. [En línea]. Available: https://www.la-razon.com/la-revista/2022/07/06/la-brecha-digital-preocupa-al-76-de-las-personas-mayores-de-80-anos/

[3] «DepenCare». Accedido: ago. 08, 2022. [En línea]. Available: https://play.google.com/store/apps/details?id=com.dependcare&gl=ES

[4] «Cuidum». Accedido: ago. 08, 2022. [En línea]. Available: https://play.google.com/store/apps/details?id=com.cuidum.cuidum_android&gl=ES

[5] «MedControl». Accedido: ago. 08, 2022. [En línea]. Available: https://play.google.com/store/search?q=medcontrol&c=apps&gl=ES

[6] «MediSafe». Accedido: ago. 08, 2022. [En línea]. Available: https://play.google.com/store/search?q=medisafe&c=apps&gl=ES

[7] «Aplicaciones nativas». https://abamobile.com/web/que-son-aplicaciones-nativas-y-ventajas/ (accedido ago. 25, 2022).

[8] «Android TV». https://developer.android.com/tv (accedido ago. 25, 2022).

[9] «Jitsi Meet Docker». https://jitsi.github.io/handbook/docs/devops-guide/devops-guide-docker/ (accedido ago. 25, 2022).

[10] «WebRTC». https://blogs.trilogy-lte.com/post/77427158750/how-webrtc-is-revolutionizing-telephony (accedido ago. 25, 2022).

[11] «Material Design». https://material.io/design (accedido ago. 08, 2022).

[12] «Patrón MVP». https://devexperto.com/mvp-android/ (accedido ago. 08, 2022).

[13] «Arquitectura Cliente-Servidor». https://blog.infranetworking.com/modelo-cliente-servidor/ (accedido ago. 08, 2022).

[14] «Arquitectura Serverless». https://blog.mdcloud.es/arquitectura-serverless/ (accedido ago. 08, 2022).

[15] «Android Studio». Accedido: ago. 08, 2022. [En línea]. Available: https://developer.android.com/studio?hl=es&gclid=Cj0KCQjw9ZGYBhCEARIsAEUXITU25YxeJRBeBaYMGEqnMZ2PcDs5WTOw8MG2ugxOSKA3LGWeOFS9yIUaAoPcEALw_wcB&gclsrc=aw.ds

[16] «XML». https://developer.mozilla.org/es/docs/Web/XML/XML_introduction (accedido ago. 08, 2022).

[17] «Kotlin». https://developer.android.com/kotlin?hl=es-419 (accedido ago. 08, 2022).
[18] «Gradle». https://gradle.org/ (accedido ago. 25, 2022).
[19] «Jitsi Meet SDK». https://jitsi.github.io/handbook/docs/dev-guide/dev-guide-android-sdk/ (accedido ago. 08, 2022).
[20] «Json Web Token». https://jwt.io/ (accedido ago. 08, 2022).
[21] «Glide». https://github.com/bumptech/glide (accedido ago. 08, 2022).
[22] «Firebase». https://firebase.google.com/ (accedido ago. 08, 2022).
[23] «Firebase Authentication». https://firebase.google.com/products/auth (accedido ago. 08, 2022).
[24] «Firebase Firestore». https://firebase.google.com/products/firestore (accedido ago. 08, 2022).
[25] «Firebase Cloud Storage». https://firebase.google.com/products/storage (accedido ago. 08, 2022).
[26] «Firebase Cloud Functions». https://firebase.google.com/products/functions (accedido ago. 08, 2022).
[27] «Firebase Cloud Messaging». https://firebase.google.com/products/cloud-messaging (accedido ago. 08, 2022).
[28] «Visual Studio Code». Accedido: ago. 08, 2022. [En línea]. Available: https://code.visualstudio.com/
[29] «Node.js». Accedido: ago. 08, 2022. [En línea]. Available: https://nodejs.org/es/
[30] «JavaScript». https://developer.mozilla.org/en-US/docs/Web/JavaScript (accedido ago. 08, 2022).
[31] «Herramientas CASE». http://agrega.juntadeandalucia.es/repositorio/20022017/58/es-an_2017022012_9123032/31_tipos_de_herramientas_case.html (accedido ago. 08, 2022).
[32] «UML». https://es.slideshare.net/zury27/uml-gota-a-gota-martin-fowler-con-kendall-scott (accedido ago. 08, 2022).
[33] «Visual Paradigm». Accedido: ago. 08, 2022. [En línea]. Available: https://www.visual-paradigm.com/support/documents/vpuserguide.jsp
[34] «Microsoft Office Project». Accedido: ago. 03, 2022. [En línea]. Available: https://project.microsoft.com/
[35] «GitHub». https://github.com/ (accedido ago. 08, 2022).
[36] «Proceso Unificado». https://www.scribd.com/document/399151346/El-Proceso-Unificado-de-Desarrollo-de-Software-James-Rumbaugh-Ivar-Jacobson-Grady-Booch (accedido ago. 08, 2022).

[37] María Navarro Cáceres y María Navelonga Moreno García, «Gestión de Proyectos - Planificación temporal».

[38] Amador Durán Toro y Beatriz Bernárdez Jiménez, «Metodología para la Elicitación de Requisitos de Sistemas Software».

[39] Roger S. Pressman, *Ingeniería del Software: Un Enfoque Práctico*.

[40] «Dokka». https://github.com/Kotlin/dokka (accedido ago. 05, 2022).

[41] «JSDoc». https://jsdoc.app/index.html (accedido ago. 05, 2022).

Anexo I: Plan de proyecto

Aplicación para facilitar las tareas de personas mayores

Trabajo de Fin de Grado de Ingeniería Informática

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Septiembre de 2022

Autora:

Paula Soria Ullán

Tutores:

Gabriel Villarrubia González

Álvaro Lozano Murciego

André Filipe Sales Mendes

Índice

Índice de figuras

Índice de tablas

Índice de ecuaciones

1. Introducción

En este anexo se va a documentar el plan de proyecto software donde se explicarán los detalles relacionados con la gestión de proyectos.

Se trata de estimar los costes que puede suponer para el cliente y la duración del desarrollo del proyecto. También se llevará a cabo una planificación temporal, asignando tareas a los recursos humanos disponibles.

Para realizar el proyecto se ha seguido el Proceso Unificado, que es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura, y por ser iterativo e incremental.

Este se divide en cuatro fases diferentes, y estas a su vez se dividen en seis disciplinas diferentes que dependiendo de la etapa en la que nos encontremos tendrán más o menos importancia. En la Figura 1 se puede ver una representación del marco de desarrollo:

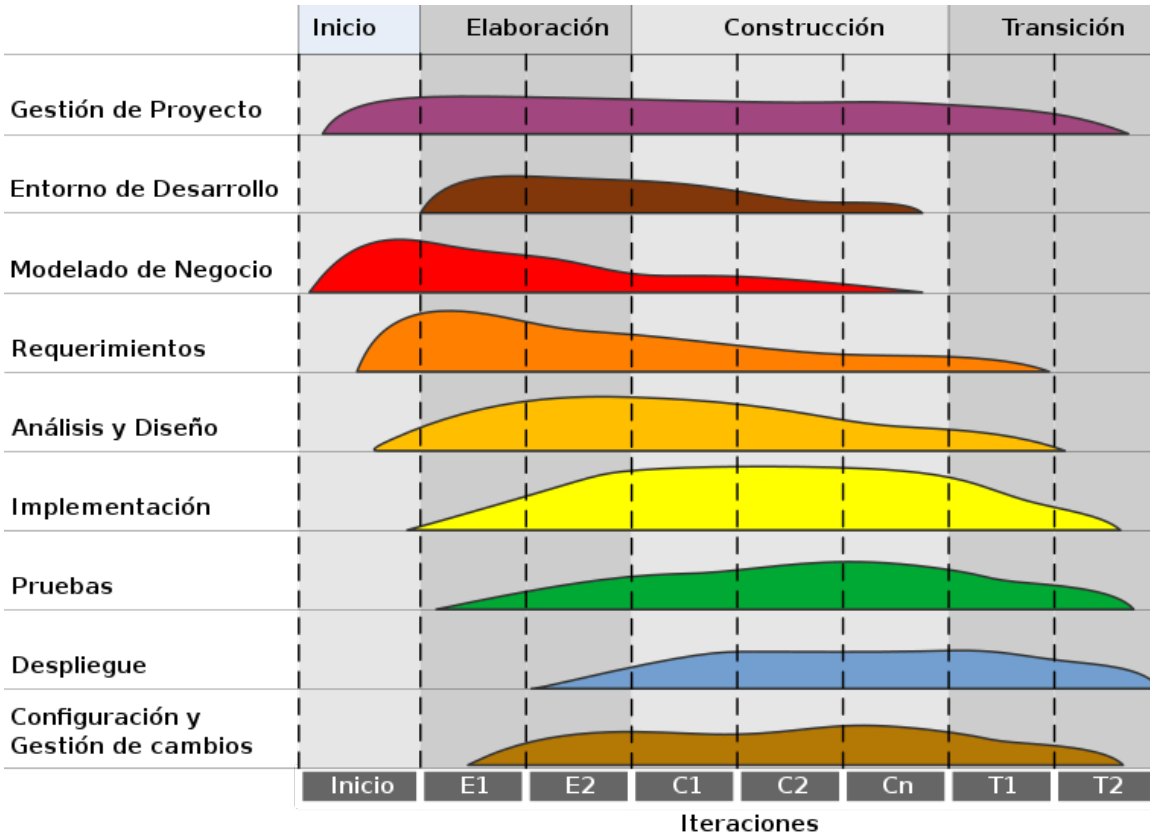


Figura 1: Proceso Unificado [1]

2. Estimación del esfuerzo

En este apartado se realizará una estimación del esfuerzo y del tiempo del proyecto. Se realizará una primera aproximación a los casos de uso y posteriormente se identificarán los actores que intervienen en el sistema.

Esta estimación se va a realizar con la métrica de UCP (*Use Case Points*) o Puntos de caso de uso, que se emplea para evaluar la funcionalidad en forma de casos de uso. Para calcular esta métrica se emplean actores, escenarios y factores técnicos y de entorno. Las variables que intervienen son las siguientes:

- **UUCP (*Unadjusted Use Case Points*)**: Puntos de casos de uso desajustados. Es la suma de las siguientes dos variables:
  - **UUCW (*Unadjusted Use Case Weight*):** Peso de los casos de usos desajustado.
  - **UAW (*Unadjusted Actor Weight*):** Peso de los actores desajustado.
- **TCF (*Technical Complexity Factor*):** Factores de complejidad técnica.
- **ECF (*Environment Complexity Factor*):** Factores de complejidad del entorno.

A partir de los parámetros explicados antes, se puede calcular la métrica de UCP mediante la siguiente ecuación:

$$\mathbf{UCP = UUCP * TCF * ECF}$$

Ecuación 1: Ecuación del UCP

Después de obtener el resultado anterior, se puede calcular el esfuerzo empleando un factor de conversión que determina el número de horas de personas por UCP, y se obtiene mediante la siguiente ecuación:

$$\mathbf{Esfuerzo = UCP * Factor\ de\ conversión}$$

Ecuación 2: Ecuación del esfuerzo

Para realizar todos estos cálculos se ha empleado la herramienta EZEstimate, con la que se obtendrá el tiempo estimado por persona para desarrollar el sistema al completo.

2.1. Complejidad de los actores

Se han definido 3 actores que interactuarán con el sistema, y estos son: Usuario sin autenticar, Usuario autenticado y Usuario administrador. Para cada uno de ellos se ha establecido una complejidad en función de la siguiente tabla:

Complejidad	Peso	Descripción
Simple	1	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API).
Media	2	Otro sistema que interactúa mediante un protocolo (por ejemplo, TCP/IP) o una persona que interactúa a través de una interfaz en modo texto.
Compleja	3	Una persona que interactúa con el sistema mediante una interfaz gráfica de usuario (GUI).

Tabla 1: Complejidad de los actores

Todos los actores definidos interactúan con el sistema a través de la interfaz gráfica de usuario, por lo tanto, los tres tendrán una complejidad de tipo compleja.

2.2. Complejidad de los casos de uso

Se han definido un total de 24 casos de uso, y para cada uno de ellos se ha establecido una complejidad en función del número de transacciones. Podemos verlo en la siguiente tabla:

Complejidad	Peso	Descripción
Simple	5	Menos de 4 transacciones Menos de 5 clases
Media	10	Entre 4 y 7 transacciones Entre 5 y 10 clases
Compleja	15	Más de 7 transacciones Más de 10 clases

Tabla 2: Complejidad de los casos de uso

La mayoría de los casos de uso definidos en el sistema tienen complejidad de tipo simple porque realizan menos de 4 transacciones. También existe algún caso de uso con complejidad de tipo media que realiza entre 4 y 7 transacciones.

2.3. Factores de complejidad técnica

Los factores de complejidad técnica son una serie de factores predefinidos a los que se les da un valor entre 0 y 5, de modo que se pueda representar el esfuerzo que supondrá su implementación para los desarrolladores.

Dichos factores se han establecido en función de la siguiente tabla:

Factor técnico	Descripción	Peso
T1	Sistema distribuido	2
T2	Rendimiento o tiempo de respuesta	1
T3	Eficiencia del usuario final	1
T4	Procesamiento interno complejo	1
T5	Reusabilidad	1
T6	Facilidad de instalación	0.5
T7	Facilidad de uso	0.5
T8	Portabilidad	2
T9	Facilidad de cambio	1
T10	Concurrencia	1
T11	Características especiales de seguridad	1
T12	Provee acceso directo a terceras partes	1
T13	Se requiere entrenamiento especial de usuario	1

Tabla 2: Factores de complejidad técnica

A continuación, se van a asignar los valores de complejidad a cada uno de los factores:

- **Sistemas distribuidos:** 3
  A pesar de que una parte del sistema es *serverless*, la parte de las alertas y videollamadas emplean un servidor. También la aplicación se podrá ejecutar en diferentes dispositivos como teléfonos móviles o televisiones.
- **Rendimiento:** 4
  Se necesitan tiempos de respuesta relativamente rápidos, sobre todo cuando se trata de que suene una alerta o de establecer una videollamada.

- **Eficiencia del usuario final:** 3
  Hay que garantizar que el usuario final puede usar de forma eficiente la aplicación, sobre todo al tratarse de personas mayores.
- **Procesamiento interno complejo:** 2
  El sistema manejará cierto volumen de datos como en la búsqueda de usuarios, pero el procesamiento interno no será demasiado complejo ni se realizarán cálculos costosos.
- **Reusabilidad:** 3
  Poder reutilizar funcionalidad de la aplicación siempre será una característica deseable para satisfacer las buenas prácticas de programación, sobre todo al tratarse de una parte de aplicación adaptada a móvil y otra a la televisión, por lo que tendrán funcionalidad común.
- **Facilidad de instalación:** 1
  La instalación se haría de forma sencilla al tratarse de una aplicación mediante la descarga desde Google Play y el servidor se despliega fácilmente.
- **Facilidad de uso:** 5
  Al tratarse de una aplicación que utilizarán personas mayores, la facilidad de comprensión debe ser la mayor posible, además de ser intuitiva.
- **Portabilidad:** 4
  La aplicación se debe poder instalar en diversos dispositivos de Android, tanto teléfonos móviles como televisiones.
- **Facilidad de cambio:** 2
  No es un factor fundamental, pero debe ser sencillo poder mantener y añadir nueva funcionalidad a la aplicación.
- **Concurrencia:** 3
  Puede haber muchos usuarios conectados a la vez a la aplicación y solicitando por ejemplo realizar una videollamada con el mismo usuario.
- **Características especiales de seguridad:** 1
  El sistema no tiene datos especialmente sensibles, la seguridad se centrará en la autenticación de usuarios y la protección de la base de datos.
- **Acceso directo a terceras partes:** 3
  El sistema emplea servicios de terceros como distintos servicios de Firebase o Jitsi Meet.
- **Se requiere entrenamiento especial de usuario:** 0
  Se necesita que los usuarios manejen la aplicación de forma intuitiva al haber una parte del público de la tercera edad.

2.4. Factores de complejidad del entorno

Los factores de complejidad del entorno son una serie de factores predefinidos a los que se les da un valor entre 0 y 5, que representan la experiencia que tienen los desarrolladores respecto al sistema.

Dichos factores se han establecido en función de la siguiente tabla:

Factor ambiental	Descripción	Peso
E1	Familiaridad con el modelo de proyecto utilizado y familiaridad con UML	1.5
E2	Personal tiempo parcial	-1
E3	Capacidad de los analistas	0.5
E4	Experiencia en la aplicación	0.5
E5	Experiencia en orientación a objetos	1
E6	Motivación	1
E7	Dificultad del lenguaje de programación	-1
E8	Estabilidad de los requisitos	2

Tabla 3: Factores de complejidad del entorno

A continuación, se van a asignar los valores de complejidad a cada uno de los factores:

- **Familiaridad con UML:** 3
  Existe cierta familiaridad con UML al haberlo estudiado durante el grado, pero no se ha aplicado nunca a un proyecto real.
- **Personal a tiempo parcial:** 4
  El sistema se desarrolla por una única persona, pero habrá que compaginar el desarrollo del sistema con el último año del grado y las prácticas de empresa a tiempo completo.
- **Capacidad de los analistas:** 2
  Al igual que antes, existe cierta familiaridad con el análisis, pero no se tienen grandes conocimientos al no tener la experiencia necesaria.
- **Experiencia en la aplicación:** 1
  No se tiene experiencia desarrollando aplicaciones móviles ni utilizando los lenguajes empleados en ellas: Kotlin y JavaScript. Tampoco se tiene

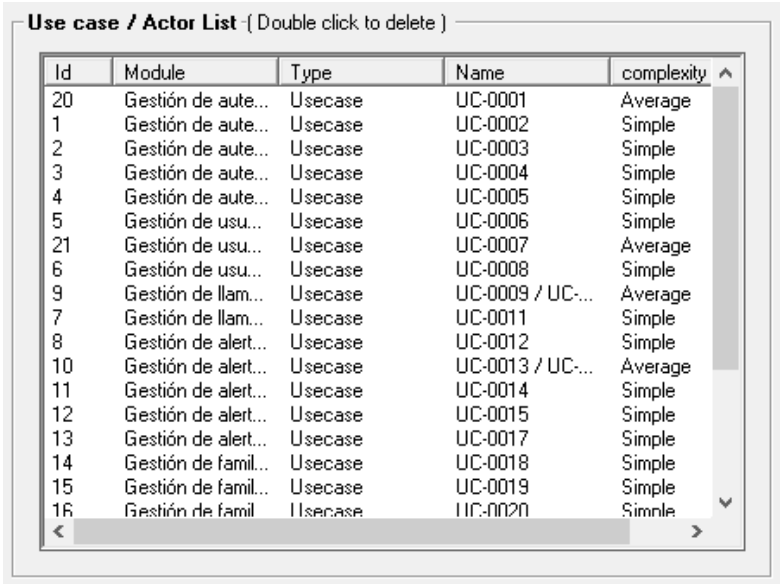
experiencia utilizando ciertas herramientas como Firebase, ni el desarrollo de bases de datos no relacionales.

- **Experiencia en orientación a objetos:** 3
  Se ha realizado algún proyecto académico con lenguajes orientados a objetos, además de trabajar sobre un proyecto de grandes dimensiones que empleaba Java durante las prácticas de empresa.
- **Motivación:** 5
  Hay una gran motivación por aprender y adquirir nuevas capacidades, y por ser el Trabajo Final de Grado.
- **Dificultad del lenguaje de programación:** 4
  No se tiene experiencia utilizando los lenguajes empleados durante el desarrollo, que son Kotlin y JavaScript, por lo que la dificultad será alta, aunque existe mucha documentación acerca de los mismos.
- **Estabilidad de los requisitos:** 4
  Los requisitos se han establecido con anterioridad, por lo tanto, no se esperan grandes cambios a lo largo del desarrollo.

2.5. Resultados

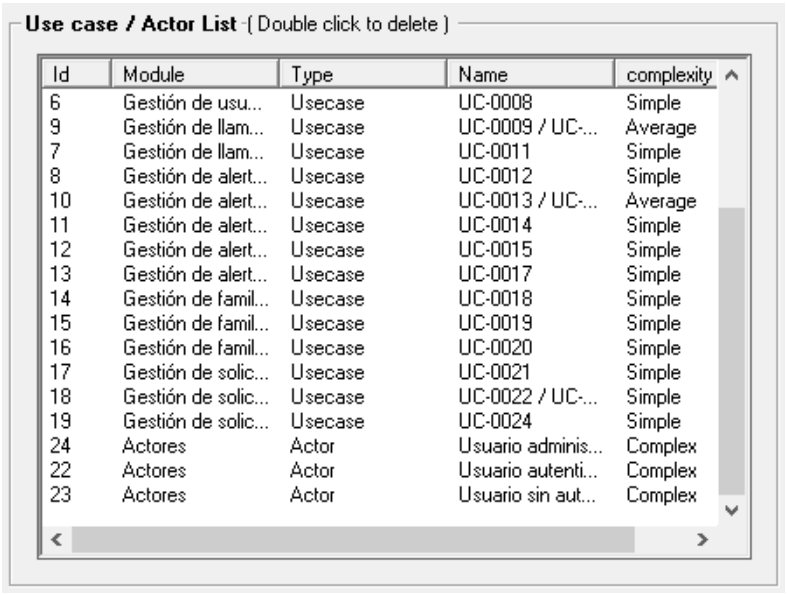
Después de haber identificado la complejidad de los actores y los casos de uso, y de los factores de complejidad técnica y de entorno, se pueden utilizar las fórmulas explicadas anteriormente para realizar los cálculos necesarios que estimen el coste y el esfuerzo del proyecto. Para realizar los cálculos se ha utilizado EZEstimate, como se mencionó al principio.

Primero se han introducido todos los casos de uso por paquetes y los actores definidos, junto con su nivel de complejidad, como se muestra en las Figuras 2, 3 y 4:



Id	Module	Type	Name	complexity
20	Gestión de aute...	Usecase	UC-0001	Average
1	Gestión de aute...	Usecase	UC-0002	Simple
2	Gestión de aute...	Usecase	UC-0003	Simple
3	Gestión de aute...	Usecase	UC-0004	Simple
4	Gestión de aute...	Usecase	UC-0005	Simple
5	Gestión de usu...	Usecase	UC-0006	Simple
21	Gestión de usu...	Usecase	UC-0007	Average
6	Gestión de usu...	Usecase	UC-0008	Simple
9	Gestión de llam...	Usecase	UC-0009 / UC-...	Average
7	Gestión de llam...	Usecase	UC-0011	Simple
8	Gestión de alert...	Usecase	UC-0012	Simple
10	Gestión de alert...	Usecase	UC-0013 / UC-...	Average
11	Gestión de alert...	Usecase	UC-0014	Simple
12	Gestión de alert...	Usecase	UC-0015	Simple
13	Gestión de alert...	Usecase	UC-0017	Simple
14	Gestión de famil...	Usecase	UC-0018	Simple
15	Gestión de famil...	Usecase	UC-0019	Simple
16	Gestión de famil...	Usecase	UC-0020	Simple

Figura 2: Lista de casos de uso y actores en EZEstimate 1



Id	Module	Type	Name	complexity
6	Gestión de usu...	Usecase	UC-0008	Simple
9	Gestión de llam...	Usecase	UC-0009 / UC-...	Average
7	Gestión de llam...	Usecase	UC-0011	Simple
8	Gestión de alert...	Usecase	UC-0012	Simple
10	Gestión de alert...	Usecase	UC-0013 / UC-...	Average
11	Gestión de alert...	Usecase	UC-0014	Simple
12	Gestión de alert...	Usecase	UC-0015	Simple
13	Gestión de alert...	Usecase	UC-0017	Simple
14	Gestión de famil...	Usecase	UC-0018	Simple
15	Gestión de famil...	Usecase	UC-0019	Simple
16	Gestión de famil...	Usecase	UC-0020	Simple
17	Gestión de solic...	Usecase	UC-0021	Simple
18	Gestión de solic...	Usecase	UC-0022 / UC-...	Simple
19	Gestión de solic...	Usecase	UC-0024	Simple
24	Actores	Actor	Usuario adminis...	Complex
22	Actores	Actor	Usuario autenti...	Complex
23	Actores	Actor	Usuario sin aut...	Complex

Figura 3: Lista de casos de uso y actores en EZEstimate 2

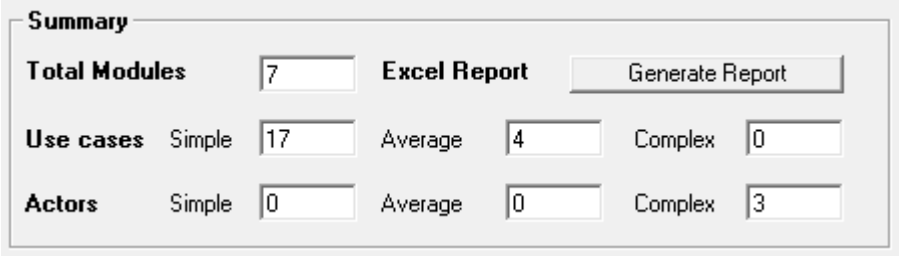


Figura 4: Resumen de casos de uso y actores en EZEstimate

Posteriormente se añadieron los factores de complejidad técnica y los factores de complejidad del entorno, como se puede ver en las Figuras 5 y 6:

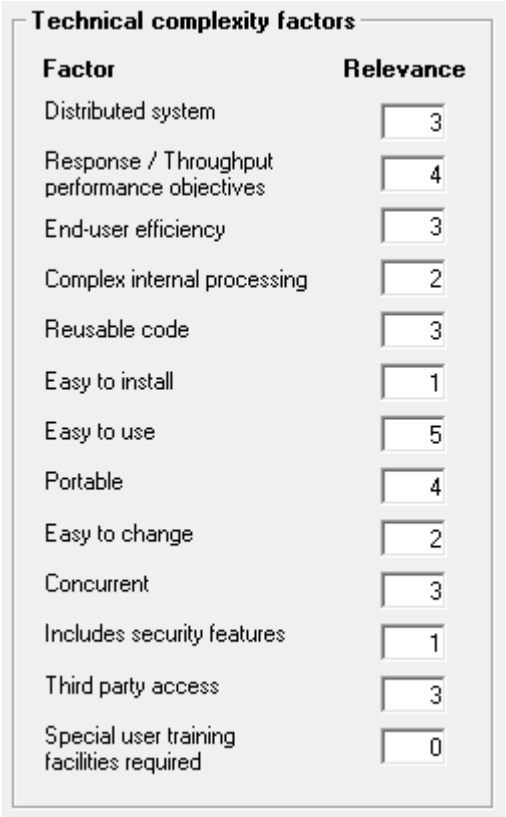


Figura 5: Factores de complejidad técnica en EZEstimate

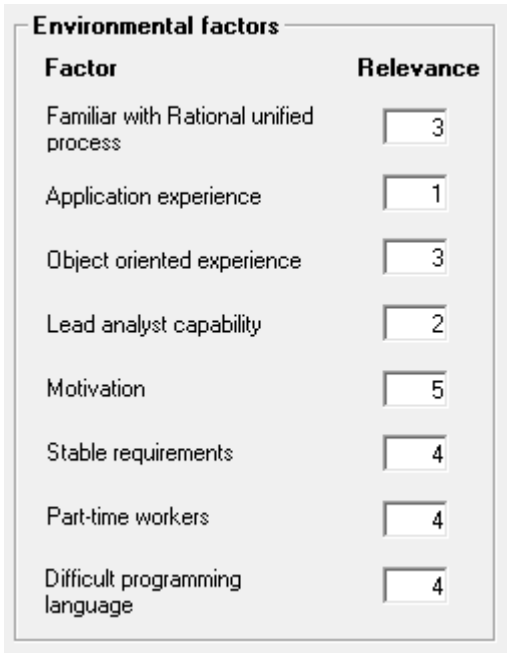


Figura 6: Factores de complejidad del entorno en EZEstimate

Por último, se pueden consultar la estimación obtenida aplicando las ecuaciones vistas anteriormente. Se ha modificado el valor de horas por caso de uso de 20 a 5. La estimación resultante se muestra en la Figura 7:

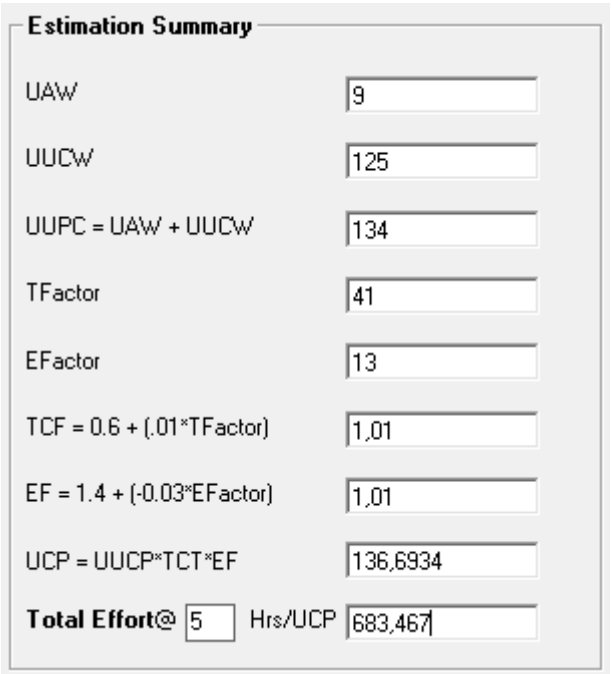


Figura 7: Resultados en EZEstimate

Si dividimos el resultado obtenido entre 6 horas que se trabajará al día aproximadamente, se obtiene una estimación temporal de 114 días.

3. Planificación temporal

En este apartado se planificarán las tareas que hay que realizar durante el proyecto para determinar los tiempos que requieren cada una y así poder estimar el tiempo total.

Para la realización de este anexo se ha empleado la herramienta Microsoft Office Project [2] y se han consultado diversas fuentes como el libro de Ingeniería del Software: Un Enfoque Práctico [3] o los apuntes de Planificación temporal de la asignatura Gestión de Proyectos [4].

Primero se estableció la fecha de inicio del proyecto junto con el horario asignado para la realización del proyecto o calendario, que en este caso es el establecido por defecto. En la siguiente figura se puede observar también la duración del proyecto después de establecer las tareas y la fecha de finalización:

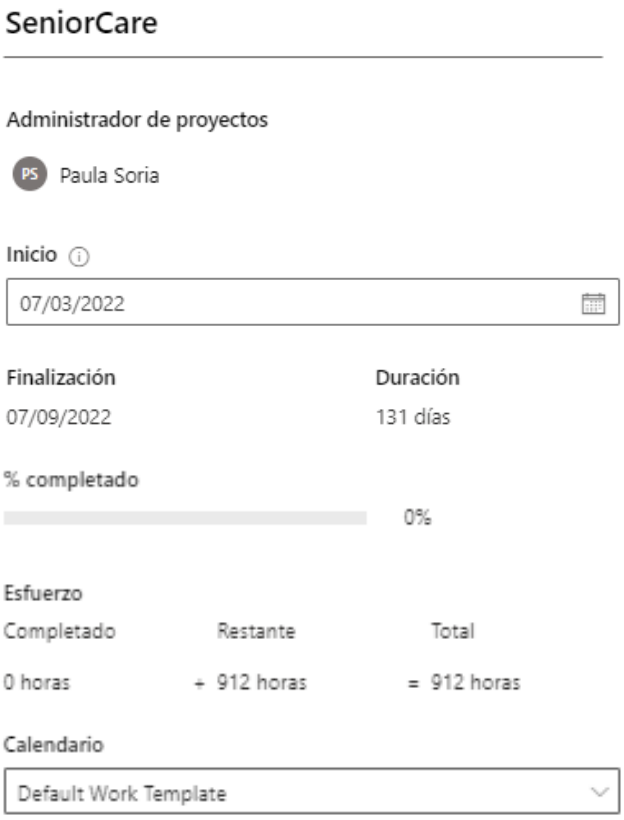


Figura 8: Inicio de proyecto

A continuación, se realizará una planificación de las tareas siguiendo el esquema del Proceso Unificado, siendo las fases por dividir las siguientes:

- **Inicio:** Se define el alcance del proyecto y se desarrollan los casos de negocio. Tiene 2 iteraciones.
- **Elaboración:** Se planifica el proyecto, se especifican en detalle la mayoría de los casos de uso y se diseña la arquitectura del sistema. Tiene 3 iteraciones.
- **Construcción:** Se construye el producto. Tiene 5 iteraciones.
- **Transición:** El producto se convierte en versión beta. Se corrigen problemas y se incorporan mejoras sugeridas en la revisión. Tiene una única iteración.

Las fases anteriores se dividirán en iteraciones, y dentro de ellas se encuentran organizadas en disciplinas, que son las siguientes:

- **Modelado de negocio:** Se realiza un análisis de aspectos relativos al proyecto como investigación y búsqueda de información.
- **Requisitos:** Se realiza un análisis textual y se definen objetivos, además de realizar la captura de requisitos y una especificación de casos de uso.
- **Análisis:** Se realiza un análisis en profundidad de los requisitos recogidos en la etapa anterior, creando un modelo del dominio, definiendo paquetes de análisis y realizando los casos de uso de la fase anterior.
- **Diseño:** Se definen los patrones arquitectónicos a utilizar, se elaboran clases de diseño, se realiza una representación de los casos de uso elaborados y se crea un modelo de despliegue para describir el diseño final.
- **Implementación:** Se programa el sistema diseñado.
- **Pruebas:** Se realizan pruebas unitarias de componentes y pruebas finales de integración para comprobar el funcionamiento del sistema y corregir posibles errores.

Cada una de estas etapas se dividirá a su vez en tareas específicas para establecer mejor los límites de tiempo de cada fase. En las siguientes figuras podemos ver esta planificación de tareas dividida en iteraciones:

	Nombre	Asignado a	Duración	Depende de	Inicio	Finalización
1	**Inicio del proyecto**		**130 días**		**7/3/2022**	**2/9/2022**
2	**Inicio**		**19 días**		**7/3/2022**	**31/3/2022**
3	**Iteración 1**		**10 días**		**7/3/2022**	**18/3/2022**
4	**Modelado de negocio**		**4 días**		**7/3/2022**	**10/3/2022**
5	Planteamiento del problema	PS Paula Soria	2 días		7/3/2022	8/3/2022
6	Búsqueda de necesidades	PS Paula Soria	2 días	5	9/3/2022	10/3/2022
7	Búsqueda de actores	PS Paula Soria	2 días	5	9/3/2022	10/3/2022
8	**Requisitos**		**6 días**		**11/3/2022**	**18/3/2022**
9	Definición de objetivos	PS Paula Soria	4 días	6	11/3/2022	16/3/2022
10	Definición de requisitos	PS Paula Soria	2 días	9	17/3/2022	18/3/2022
11	Definición de actores	PS Paula Soria	2 días	9	17/3/2022	18/3/2022
12	**Análisis**		**2 días**		**9/3/2022**	**10/3/2022**
13	Análisis de lenguajes	PS Paula Soria	2 días	5	9/3/2022	10/3/2022
14	Análisis de tecnologías	PS Paula Soria	2 días	5	9/3/2022	10/3/2022
15	**Diseño**		**5 días**		**9/3/2022**	**15/3/2022**
16	Diseño de prototipo con Adobe XD	PS Paula Soria	5 días	5	9/3/2022	15/3/2022
17	Fin de Iteración 1		1 día	4 8 +2	21/3/2022	21/3/2022
18	**Iteración 2**		**7 días**	17	**22/3/2022**	**30/3/2022**
19	**Requisitos**		**7 días**		**22/3/2022**	**30/3/2022**
20	Identificación de actores	PS Paula Soria	3 días		22/3/2022	24/3/2022
21	Identificación de escenarios	PS Paula Soria	3 días		22/3/2022	24/3/2022
22	Especificación de casos de uso	PS Paula Soria	4 días	20 21	25/3/2022	30/3/2022
23	**Análisis**		**4 días**		**22/3/2022**	**25/3/2022**
24	Identificación de paquetes	PS Paula Soria	3 días	16	22/3/2022	24/3/2022
25	Realización de diagramas de análisis	PS Paula Soria	4 días	16	22/3/2022	25/3/2022
26	**Diseño**		**2 días**		**28/3/2022**	**29/3/2022**
27	Realización de diagramas de diseño	PS Paula Soria	2 días	25	28/3/2022	29/3/2022
28	Fin de Iteración 2		1 día	19 23 26	31/3/2022	31/3/2022
29	Fin de Inicio		1 día	28	1/4/2022	1/4/2022

Figura 9: Planificación de tareas Inicio

		Nombre	Asignado a	Duración	Depende de	Inicio	Finalización
30		**Elaboración**		**26 días**	29	**4/4/2022**	**9/5/2022**
31		**Iteración 1**		**5 días**		**4/4/2022**	**8/4/2022**
32		**Requisitos**		**5 días**		**4/4/2022**	**8/4/2022**
33		Refinación de casos de uso	PS Paula Soria	4 días		4/4/2022	7/4/2022
34		Identificación de relaciones entre casos de uso	PS Paula Soria	1 día	33	8/4/2022	8/4/2022
35		**Análisis**		**2 días**		**4/4/2022**	**5/4/2022**
36		Revisión de paquetes y sus relaciones	PS Paula Soria	2 días		4/4/2022	5/4/2022
37		Propuesta de arquitectura	PS Paula Soria	2 días		4/4/2022	5/4/2022
38		Diagrama de clases de análisis	PS Paula Soria	2 días		4/4/2022	5/4/2022
39		Fin de Iteración 1		1 día	32 35	11/4/2022	11/4/2022
40		**Iteración 2**		**8 días**	39	**12/4/2022**	**21/4/2022**
41		**Requisitos**		**3 días**		**12/4/2022**	**14/4/2022**
42		Identificar riesgos	PS Paula Soria	3 días		12/4/2022	14/4/2022
43		Revisión de casos de uso	PS Paula Soria	3 días		12/4/2022	14/4/2022
44		**Análisis**		**5 días**	41	**15/4/2022**	**21/4/2022**
45		Realización de casos de uso de análisis	PS Paula Soria	5 días		15/4/2022	21/4/2022
46		**Diseño**		**2 días**		**12/4/2022**	**13/4/2022**
47		Diagrama de clases de diseño	PS Paula Soria	2 días		12/4/2022	13/4/2022
48		Fin de Iteración 2		1 día	41 44 46	22/4/2022	22/4/2022
49		**Iteración 3**		**10 días**	48	**25/4/2022**	**6/5/2022**
50		**Requisitos**		**1 día**		**25/4/2022**	**25/4/2022**
51		Revisión de casos de uso	PS Paula Soria	1 día		25/4/2022	25/4/2022
52		**Diseño**		**5 días**	50	**26/4/2022**	**2/5/2022**
53		Realización de casos de uso de diseño	PS Paula Soria	5 días		26/4/2022	2/5/2022
54		Diagrama de modelo de despliegue	PS Paula Soria	2 días		26/4/2022	27/4/2022
55		**Implementación**		**4 días**	52	**3/5/2022**	**6/5/2022**
56		Diagrama de modelo de implementación	PS Paula Soria	4 días		3/5/2022	6/5/2022
57		Fin de Iteración 3		1 día	50 52 55	9/5/2022	9/5/2022
58		Fin de Elaboración		1 día	57	10/5/2022	10/5/2022

Figura 10: Planificación de tareas Elaboración

		Nombre	Asignado a	Duración	Depende de	Inicio	Finalización
59		**Construcción**		**66 días**	58	**11/5/2022**	**10/8/2022**
60		**Iteración 1**		**12 días**		**11/5/2022**	**26/5/2022**
61		**Requisitos**		**3 días**		**11/5/2022**	**13/5/2022**
62		Revisión de casos de uso	PS Paula Soria	3 días		11/5/2022	13/5/2022
63		**Diseño**		**6 días**		**11/5/2022**	**18/5/2022**
64		Definición de patrones	PS Paula Soria	3 días		11/5/2022	13/5/2022
65		Definición de estructuras de datos	PS Paula Soria	2 días	64	16/5/2022	17/5/2022
66		Diseño de la interfaz	PS Paula Soria	6 días		11/5/2022	18/5/2022
67		**Implementación**		**4 días**		**19/5/2022**	**24/5/2022**
68		Implementación del subsistema de gestión de autenticación	PS Paula Soria	4 días	63 61	19/5/2022	24/5/2022
69		**Pruebas**		**2 días**	67	**25/5/2022**	**26/5/2022**
70		Revisión estática del código	PS Paula Soria	2 días		25/5/2022	26/5/2022
71		Pruebas unitarias del subsistema	PS Paula Soria	2 días		25/5/2022	26/5/2022
72		Fin de Iteración 1		1 día	61 63 +2	27/5/2022	27/5/2022
73		**Iteración 2**		**12 días**	72	**30/5/2022**	**14/6/2022**
74		**Requisitos**		**1 día**		**30/5/2022**	**30/5/2022**
75		Revisión de casos de uso	PS Paula Soria	1 día		30/5/2022	30/5/2022
76		**Diseño**		**1 día**	74	**31/5/2022**	**31/5/2022**
77		Revisión de elementos del diseño	PS Paula Soria	1 día		31/5/2022	31/5/2022
78		**Implementación**		**5 días**	76	**1/6/2022**	**7/6/2022**
79		Implementación del subsistema de gestión de usuarios	PS Paula Soria	5 días		1/6/2022	7/6/2022
80		**Pruebas**		**5 días**	78	**8/6/2022**	**14/6/2022**
81		Revisión estática del código	PS Paula Soria	2 días		8/6/2022	9/6/2022
82		Pruebas unitarias del subsistema	PS Paula Soria	2 días		8/6/2022	9/6/2022
83		Pruebas de integración	PS Paula Soria	3 días	82	10/6/2022	14/6/2022
84		Fin de iteración 2		1 día	74 76 +2	15/6/2022	15/6/2022
85		**Iteración 3**		**9 días**	84	**16/6/2022**	**28/6/2022**
86		**Requisitos**		**1 día**		**16/6/2022**	**16/6/2022**
87		Revisión de casos de uso	PS Paula Soria	1 día		16/6/2022	16/6/2022
88		**Implementación**		**3 días**	86	**17/6/2022**	**21/6/2022**
89		Implementación del subsistema de gestión de solicitudes	PS Paula Soria	3 días		17/6/2022	21/6/2022
90		**Pruebas**		**5 días**	88	**22/6/2022**	**28/6/2022**
91		Revisión estática del código	PS Paula Soria	2 días		22/6/2022	23/6/2022
92		Pruebas unitarias del subsistema	PS Paula Soria	2 días		22/6/2022	23/6/2022
93		Pruebas de integración	PS Paula Soria	3 días	92	24/6/2022	28/6/2022
94		Fin de iteración 3		1 día	86 88 90	29/6/2022	29/6/2022

Figura 11: Planificación de tareas Construcción 1

		Nombre	Asignado a	Duración	Depende de	Inicio	Finalización
95		**Iteración 4**		**13 días**	94	**30/6/2022**	**18/7/2022**
96		**Requisitos**		**1 día**		**30/6/2022**	**30/6/2022**
97		Revisión de casos de uso	PS Paula Soria	1 día		30/6/2022	30/6/2022
98		**Implementación**		**7 días**	96	**1/7/2022**	**11/7/2022**
99		Implementación del subsistema de gestión de alertas	PS Paula Soria	7 días		1/7/2022	11/7/2022
100		**Pruebas**		**5 días**	98	**12/7/2022**	**18/7/2022**
101		Revisión estática del código	PS Paula Soria	2 días		12/7/2022	13/7/2022
102		Pruebas unitarias del subsistema	PS Paula Soria	2 días		12/7/2022	13/7/2022
103		Pruebas de integración	PS Paula Soria	3 días	102	14/7/2022	18/7/2022
104		Fin de iteración 4		1 día	96 98 100	19/7/2022	19/7/2022
105		**Iteración 5**		**15 días**	104	**20/7/2022**	**9/8/2022**
106		**Requisitos**		**1 día**		**20/7/2022**	**20/7/2022**
107		Revisión de casos de uso	PS Paula Soria	1 día		20/7/2022	20/7/2022
108		**Implementación**		**9 días**	106	**21/7/2022**	**2/8/2022**
109		Implementación del subsistema de gestión de llamadas	PS Paula Soria	9 días		21/7/2022	2/8/2022
110		**Pruebas**		**5 días**	108	**3/8/2022**	**9/8/2022**
111		Revisión estática del código	PS Paula Soria	2 días		3/8/2022	4/8/2022
112		Pruebas unitarias del subsistema	PS Paula Soria	2 días		3/8/2022	4/8/2022
113		Pruebas de integración	PS Paula Soria	3 días	112	5/8/2022	9/8/2022
114		Fin de iteración 5		1 día	106 +2	10/8/2022	10/8/2022
115		Fin de Construcción		1 día	114	11/8/2022	11/8/2022

Figura 12: Planificación de tareas Construcción 2

		Nombre	Asignado a	Duración	Depende de	Inicio	Finalización
116		**Transición**		**15 días**	115	**12/8/2022**	**1/9/2022**
117		**Iteración 1**		**14 días**		**12/8/2022**	**31/8/2022**
118		**Implementación**		**7 días**		**12/8/2022**	**22/8/2022**
119		Implementación del sistema completo	PS Paula Soria	7 días		12/8/2022	22/8/2022
120		**Pruebas**		**7 días**	119	**23/8/2022**	**31/8/2022**
121		Pruebas de integración total	PS Paula Soria	4 días		23/8/2022	26/8/2022
122		Pruebas de interfaz	PS Paula Soria	3 días	121	29/8/2022	31/8/2022
123		Pruebas de navegación	PS Paula Soria	3 días		23/8/2022	25/8/2022
124		Validación del proyecto software	PS Paula Soria	4 días	123	26/8/2022	31/8/2022
125		Despliegue público del sistema	PS Paula Soria	4 días		23/8/2022	26/8/2022
126		Fin de Iteración 1		1 día	120	1/9/2022	1/9/2022
127		Fin de Transición		1 día	126	2/9/2022	2/9/2022
128		Fin de Proyecto		1 día	127	5/9/2022	5/9/2022

Figura 13: Planificación de tareas Transición

Una vez se han establecido todas las tareas es posible crear un diagrama de Gantt para ver las relaciones entre las actividades y su evolución a lo largo del tiempo:

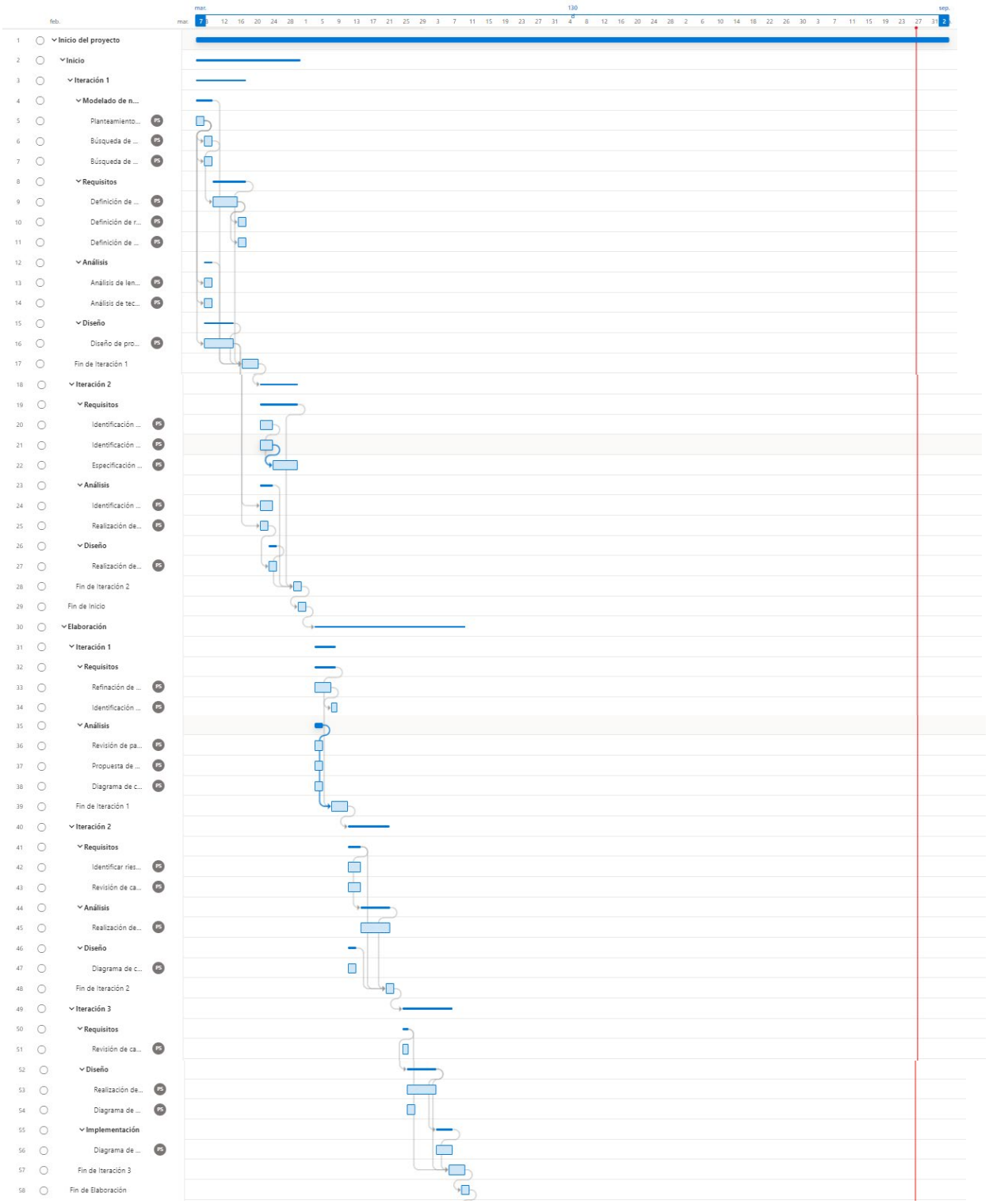


Figura 14: Diagrama de Gantt 1

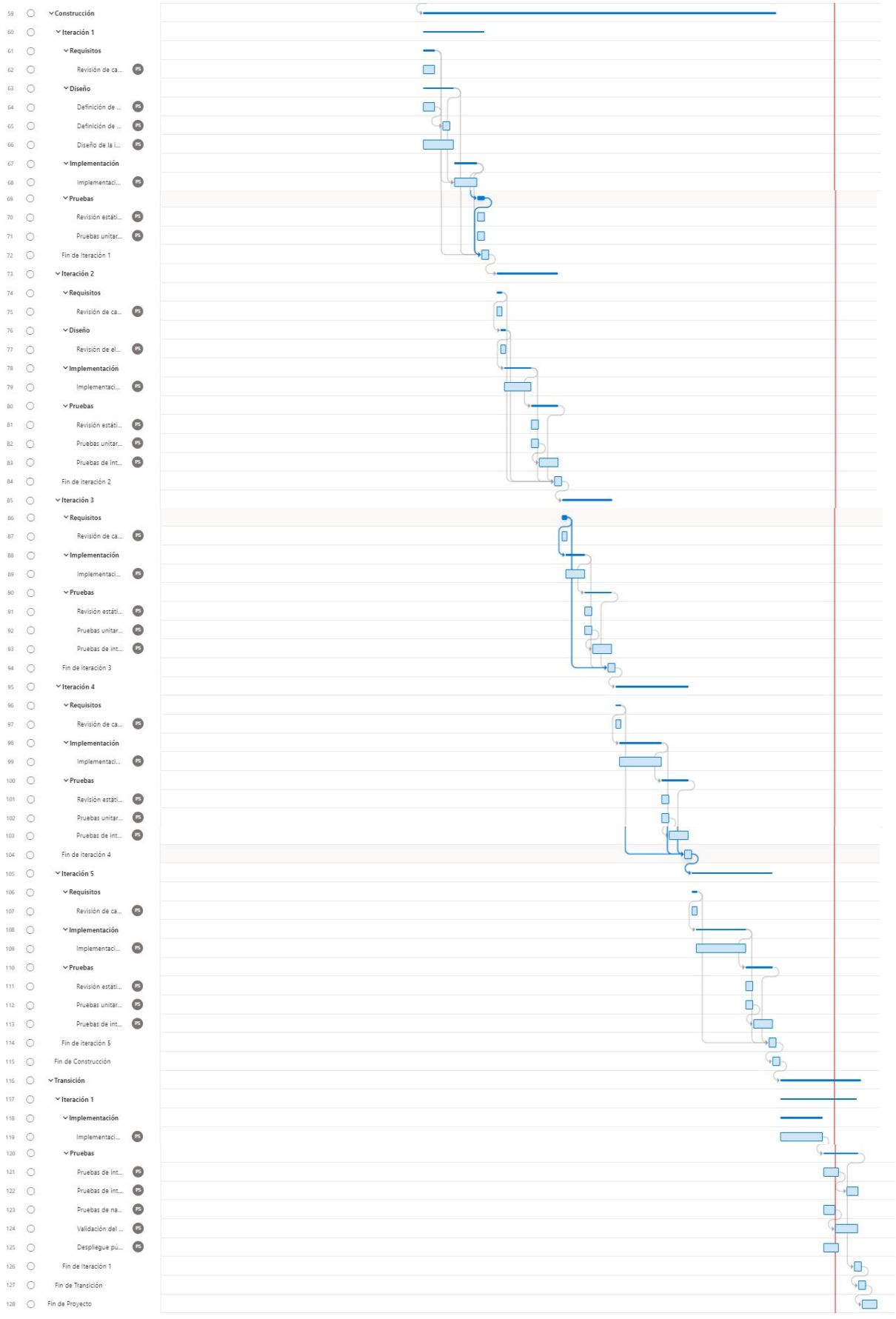


Figura 15: Diagrama de Gantt 2

4. Referencias

[1] Lantolin, «Fases y Flujos de trabajo en el Proceso Unificado».

[2] «Microsoft Office Project». Accedido: ago. 03, 2022. [En línea]. Available: https://project.microsoft.com/

[3] Roger S. Pressman, *Ingeniería del Software: Un Enfoque Práctico*.

[4] María Navarro Cáceres y María Navelonga Moreno García, «Gestión de Proyectos - Planificación temporal».

Anexo II: Especificación de requisitos software

Aplicación para facilitar las tareas de personas mayores

Trabajo de Fin de Grado de Ingeniería Informática

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Septiembre de 2022

Autora:

Paula Soria Ullán

Tutores:

Gabriel Villarrubia González

Álvaro Lozano Murciego

André Filipe Sales Mendes

Índice

Índice de figuras

Índice de tablas

1. Introducción

Este documento tiene como objetivo recoger la especificación de requisitos de software del sistema a desarrollar.

El sistema consiste en el diseño de una aplicación móvil para facilitar las tareas a las personas mayores, cuyo objetivo será poder realizar videollamadas entre familiares y poder facilitar la gestión de tareas para las personas mayores.

Se han consultado diversos temas, como la identificación de participantes, los escenarios de uso o conceptos de UML en el libro de Ingeniería del Software: Un Enfoque Práctico de Roger Pressman [1].

Para la realización de este anexo se va a seguir la estructura de Durán y Bernárdez [2]:

- Participantes
- Objetivos del sistema
- Catálogo de requisitos del sistema
    - Requisitos de información
    - Requisitos funcionales
        - Diagrama de paquetes
        - Definición de actores
        - Casos de uso del sistema
    - Requisitos no funcionales
- Matriz de rastreabilidad

2. Participantes

El proyecto cuenta con cuatro participantes, de los cuales tres son los tutores y uno es el alumno. Todos pertenecen a la misma organización, la Universidad de Salamanca.

Organización	Universidad de Salamanca
Dirección	Plaza Caídos, s/n, 37008 Salamanca
Teléfono	-
Fax	-
Comentarios	Ninguno

Tabla 1: Organización

Participante	Paula Soria Ullán
Organización	Universidad de Salamanca
Rol	Desarrolladora
Es desarrollador	Sí
Es cliente	No
Es usuario	No
Comentarios	Ninguno

Tabla 2: Participante Paula Soria Ullán

Participante	Gabriel Villarrubia González
Organización	Universidad de Salamanca
Rol	Tutor
Es desarrollador	No
Es cliente	No
Es usuario	No
Comentarios	Ninguno

Tabla 3: Participante Gabriel Villarrubia González

Participante	André Filipe Sales Mendes
Organización	Universidad de Salamanca
Rol	Tutor
Es desarrollador	No
Es cliente	No
Es usuario	No
Comentarios	Ninguno

Tabla 4: Participante André Filipe Sales Mendes

Participante	Álvaro Lozano Murciego
Organización	Universidad de Salamanca
Rol	Tutor
Es desarrollador	No
Es cliente	No
Es usuario	No
Comentarios	Ninguno

Tabla 5: Participante Álvaro Lozano Murciego

3. Objetivos del sistema

En esta sección se definen los objetivos que debe cumplir el sistema para satisfacer los requisitos planteados:

OBJ-0001	Gestión de usuarios
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Descripción	El sistema deberá permitir dar de alta y baja a los usuarios.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 6: OBJ-0001 Gestión de usuarios

OBJ-0002	Gestión de llamadas
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Descripción	El sistema deberá permitir realizar y recibir llamadas entre familiares, y visualizar el historial de llamadas.
Importancia	Importante
Urgencia	Urgente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 7: OBJ-0002 Gestión de llamadas

OBJ-0003	Gestión de alertas
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Descripción	El sistema deberá permitir configurar y recibir alertas asociadas a un familiar.
Importancia	Importante
Urgencia	Urgente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 8: OBJ-0003 Gestión de alertas

OBJ-0004	Gestión de solicitudes
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Descripción	El sistema deberá permitir enviar y recibir solicitudes para añadir familiares.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 9: OBJ-0004 Gestión de solicitudes

4. Requisitos del sistema

4.1. Requisitos de información

En este apartado se expondrán los requisitos de información, que son los datos que se deben almacenar en el sistema.

IRQ-0001	Información sobre usuarios
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Dependencias	Ninguna
Descripción	El sistema deberá almacenar la información correspondiente a los usuarios. En concreto:
Datos específicos	● Identificador ● Email ● Contraseña ● Proveedor ● Rol ● Nombre ● Apellido ● Foto
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 10: IRQ-0001 Información sobre usuarios

IRQ-0002	Información sobre llamadas
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Dependencias	Ninguna
Descripción	El sistema deberá almacenar la información correspondiente a las llamadas. En concreto:
Datos específicos	● Identificador ● Fecha ● Hora ● Duración ● Estado ● Identificador del usuario que inició la llamada ● Identificador del usuario que recibió la llamada
Importancia	Importante
Urgencia	Urgente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 11: IRQ-0002 Información sobre llamadas

IRQ-0003	Información sobre alertas
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Dependencias	Ninguna
Descripción	El sistema deberá almacenar la información correspondiente a las alertas. En concreto:
Datos específicos	● Identificador ● Etiqueta ● Hora ● Días de la semana ● Identificador del usuario que asignó la alerta ● Identificador del usuario al que se le asignó la alerta
Importancia	Importante
Urgencia	Urgente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 12: IRQ-0003 Información sobre alertas

IRQ-0004	Información sobre solicitudes
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Dependencias	Ninguna
Descripción	El sistema deberá almacenar la información correspondiente a las solicitudes de familiar. En concreto:
Datos específicos	● Identificador ● Estado ● Identificador del usuario que envió la solicitud ● Identificador del usuario que recibió la solicitud
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 13: IRQ-0004 Información sobre solicitudes

4.2. Requisitos funcionales

En este apartado se definirán los servicios que debe proporcionar el sistema y de qué manera se debe comportar en función de cada situación.

4.2.1. Diagrama de paquetes

El diagrama de paquetes de la Figura 1 muestra la división del sistema durante la fase de dominio del problema. Los paquetes en los que se divide el sistema son: Gestión de autenticación, Gestión de usuarios, Gestión de llamadas, Gestión de alertas, Gestión de familiares y Gestión de solicitudes.

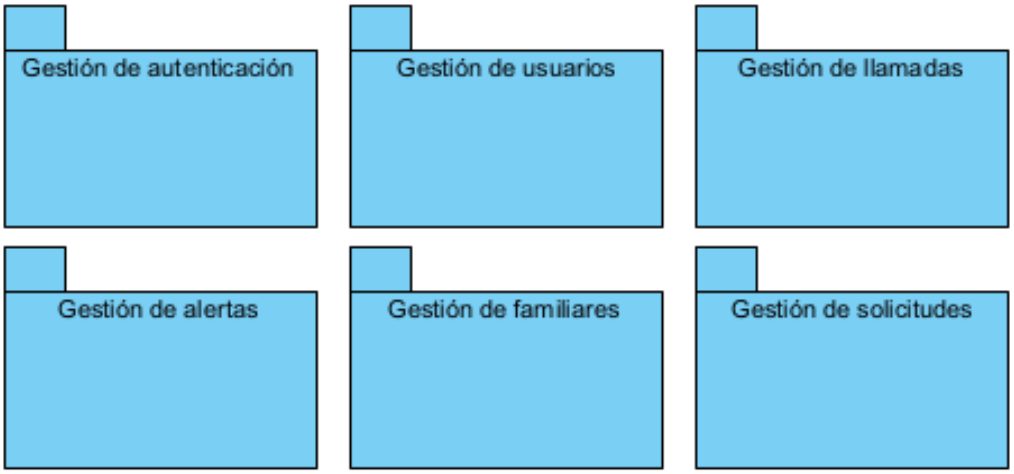


Figura 1: Diagrama de paquetes

4.2.2. Definición de actores

Para poder identificar correctamente las entidades externas que interactúan con el sistema se han consultado los apuntes de UML de Ingeniería de Software I [3].

El sistema está compuesto por tres actores:

- **Usuario sin autenticar**: Representa al usuario antes de acceder al sistema, es decir, antes de autenticarse.
- **Usuario autenticado**: Representa al usuario después de acceder al sistema, es decir, después de autenticarse.
- **Usuario administrador**: Representa al usuario después de acceder al sistema, es decir, después de autenticarse, y además añadiendo la funcionalidad de un administrador.

Figura 2: Actores

A continuación, se muestran las tablas que hacen referencia a los actores del sistema:

ACT-0001	Usuario sin autenticar
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	• Gabriel Villarrubia González • Álvaro Lozano Murciego • André Filipe Sales Mendes
Descripción	Representa al usuario antes de acceder al sistema, es decir, antes de autenticarse.
Comentarios	Ninguno

Tabla 14: ACT-0001 Usuario sin autenticar

ACT-0002	Usuario autenticado
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Descripción	Representa al usuario después de acceder al sistema, es decir, después de autenticarse.
Comentarios	Ninguno

Tabla 15: ACT-0002 Usuario autenticado

ACT-0003	Usuario administrador
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Descripción	Representa al usuario después de acceder al sistema, es decir, después de autenticarse, y además añadiendo la funcionalidad de un administrador.
Comentarios	Ninguno

Tabla 16: ACT-0003 Usuario administrador

4.2.3. Casos de uso

En este apartado se detallarán las diferentes funcionalidades y la secuencia de las interacciones entre el sistema y el usuario. También se ha hecho uso de los apuntes de UML de la asignatura de Ingeniería de Software para elaborar los diagramas de casos de uso [3].

i. Gestión de autenticación

Este paquete hace referencia a los diferentes estados en los que se puede encontrar un usuario en el sistema. Los casos de uso que componen este paquete son los siguientes:

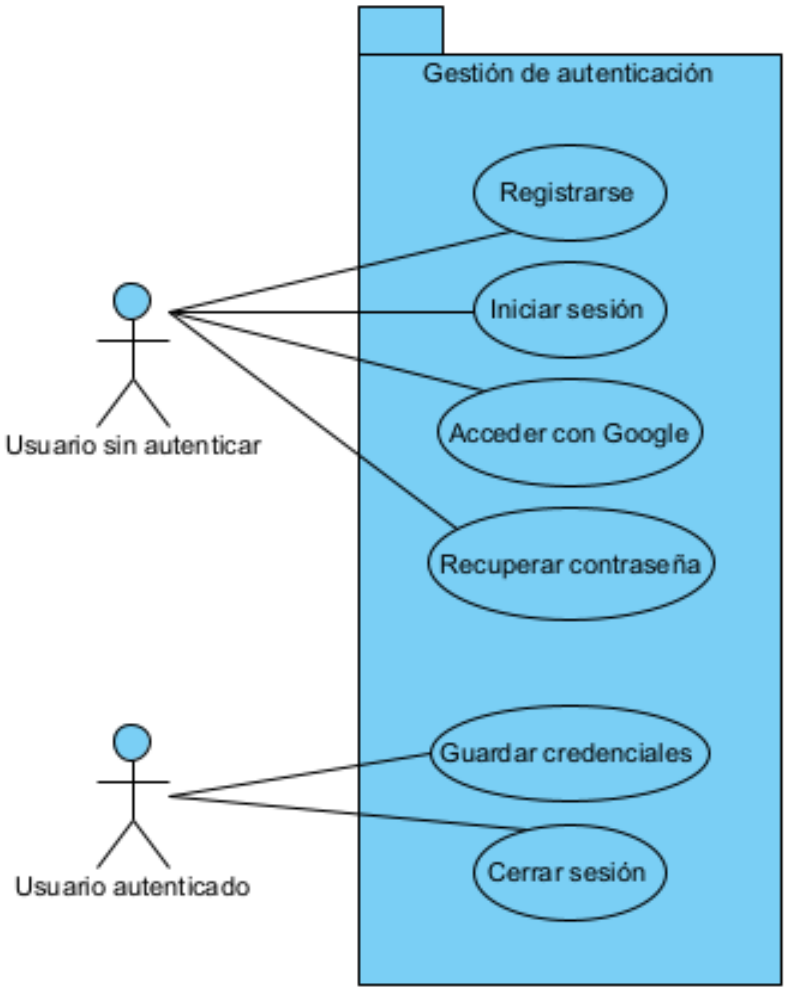


Figura 3: Diagrama de casos de uso: Gestión de autenticación

A continuación, se muestran las tablas que hacen referencia a los casos de uso de este paquete:

UC-0001	Registrarse	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [OBJ-0001] Gestión de usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite hacer un registro de usuario.	
Precondición	El usuario no está registrado.	
Secuencia normal	Paso	Acción
	1	El actor “Usuario sin autenticar (ACT-0001)” solicita al sistema realizar un registro de usuario.
	2	El sistema solicita al usuario los datos correspondientes al email, contraseña y rol.
	3	El actor “Usuario sin autenticar (ACT-0001)” proporciona los datos solicitados y solicita el registro.
	4	El sistema almacena los datos del usuario y le permite acceder a la aplicación.
Postcondición	El usuario pasa al estado de autenticado.	
Excepciones	Paso	Acción
	3	Si alguno de los datos es inválido o ya existe un usuario con dicho email, se muestra un mensaje de error.
Frecuencia	6 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 17: UC-0001 Registrarse

UC-0002	Iniciar sesión	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [OBJ-0001] Gestión de usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite iniciar sesión.	
Precondición	El usuario no ha iniciado sesión, y por tanto no está en estado de autenticado.	
Secuencia normal	Paso	Acción
	1	El actor “Usuario sin autenticar (ACT-0001)” solicita al sistema realizar un inicio de sesión.
	2	El sistema solicita al usuario los datos correspondientes al email y contraseña.
	3	El actor “Usuario sin autenticar (ACT-0001)” proporciona los datos solicitados y solicita el inicio de sesión.
	4	El sistema comprueba los datos del usuario y le permite acceder a la aplicación.
Postcondición	El usuario pasa al estado de autenticado.	
Excepciones	Paso	Acción
	3	Si alguno de los datos es inválido o no existe ningún usuario registrado con ese email, se muestra un mensaje de error.
Frecuencia	10 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 18: UC-0002 Iniciar sesión

UC-0003	Acceder con Google	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [OBJ-0001] Gestión de usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite acceder mediante Google.	
Precondición	El usuario no ha iniciado sesión, y por tanto no está en estado de autenticado.	
Secuencia normal	Paso	Acción
	1	El actor “Usuario sin autenticar (ACT-0001)” solicita al sistema acceder con Google.
	2	El sistema solicita al usuario los datos correspondientes al email de Google.
	3	El actor “Usuario sin autenticar (ACT-0001)” proporciona el email solicitado y solicita el inicio de sesión.
	4	El sistema almacena los datos del usuario y le permite acceder a la aplicación.
Postcondición	El usuario pasa al estado de autenticado.	
Excepciones	Paso	Acción
	3	Si ya existe un usuario con dicho email, se muestra un mensaje de error.
Frecuencia	6 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 19: UC-0003 Acceder con Google

UC-0004	Recuperar contraseña	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [OBJ-0001] Gestión de usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite recuperar la contraseña.	
Precondición	El usuario está registrado pero no ha iniciado sesión.	
Secuencia normal	Paso	Acción
	1	El actor “Usuario sin autenticar (ACT-0001)” solicita al sistema recuperar la contraseña.
	2	El sistema solicita al usuario los datos correspondientes al email.
	3	El actor “Usuario sin autenticar (ACT-0001)” proporciona el email solicitado y solicita recuperar la contraseña.
	4	El sistema envía al usuario un correo con un link para recuperar la contraseña.
Postcondición	-	
Excepciones	Paso	Acción
	3	Si el email es inválido o no existe ningún usuario registrado con ese email, se muestra un error.
Frecuencia	4 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 20: UC-0004 Recuperar contraseña

UC-0005	Cerrar sesión	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [OBJ-0001] Gestión de usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite cerrar sesión.	
Precondición	El usuario tiene sesión iniciada, y por tanto está en estado de autenticado.	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita al sistema cerrar sesión en la aplicación.
	2	El sistema efectúa el cierre de sesión.
Postcondición	El usuario pasa al estado sin autenticar.	
Excepciones	Paso	Acción
	-	-
Frecuencia	6 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 21: UC-0005 Cerrar sesión

ii. Gestión de usuarios

Este paquete hace referencia a la visualización y modificación de datos personales del usuario dentro del sistema. Los casos de uso que componen este paquete son los siguientes:

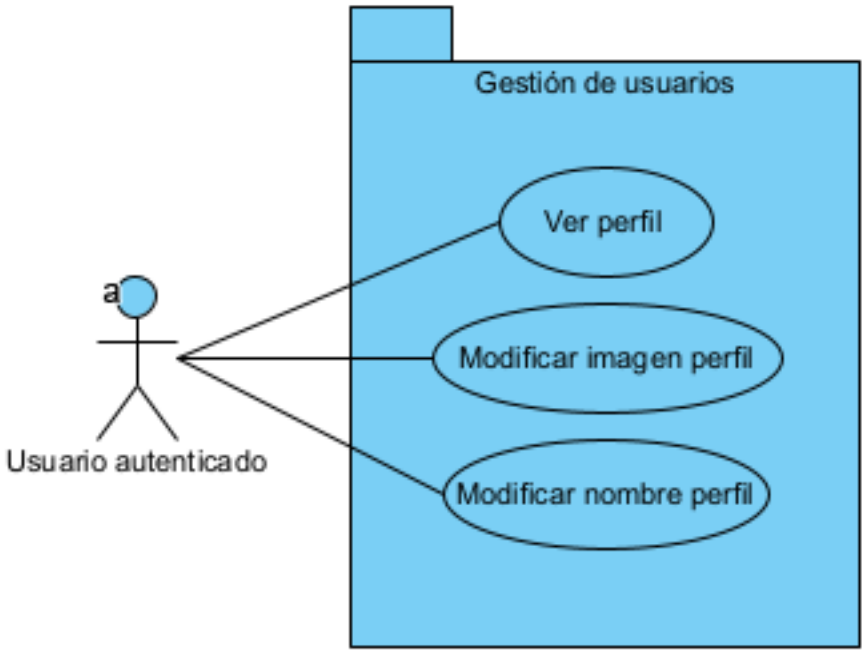


Figura 4: Diagrama de casos de uso: Gestión de usuarios

A continuación, se muestran las tablas que hacen referencia a los casos de uso de este paquete:

UC-0006	Ver perfil	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [OBJ-0001] Gestión de usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite ver el perfil.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita al sistema ver su perfil de usuario.
	2	El sistema le muestra al usuario la información relacionada con el perfil.
Postcondición	-	
Excepciones	Paso	Acción
	-	-
Frecuencia esperada	5 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 22: UC-0006 Ver perfil

UC-0007	Modificar imagen del perfil	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [OBJ-0001] Gestión de usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite modificar la imagen del perfil.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita al sistema modificar la imagen de su perfil.
	2	El sistema solicita al usuario la nueva imagen de su perfil a elegir de la galería.
	3	El actor “Usuario autenticado (ACT-0002)” proporciona la imagen solicitada y confirma los cambios.
	4	El sistema guarda los cambios y muestra la información actualizada del perfil.
Postcondición	-	
Excepciones	Paso	Acción
	3	Si el usuario cancela los cambios, el sistema mantiene la información anterior del usuario.
Frecuencia esperada	3 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 23: UC-0007 Modificar imagen del perfil

UC-0008	Modificar nombre del perfil	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [OBJ-0001] Gestión de usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite modificar el nombre del perfil.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita al sistema modificar el nombre de su perfil.
	2	El sistema solicita al usuario el nuevo nombre de su perfil.
	3	El actor “Usuario autenticado (ACT-0002)” proporciona el nombre solicitado y confirma los cambios.
	4	El sistema guarda los cambios y muestra la información actualizada del perfil.
Postcondición	-	
Excepciones	Paso	Acción
	3	Si el usuario cancela los cambios, el sistema mantiene la información anterior del usuario.
Frecuencia esperada	3 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 24: UC-0008 Modificar nombre del perfil

iii. Gestión de llamadas

Este paquete hace referencia a la realización y recepción de llamadas entre usuarios, así como la visualización del historial de llamadas. Los casos de uso que componen este paquete son los siguientes:

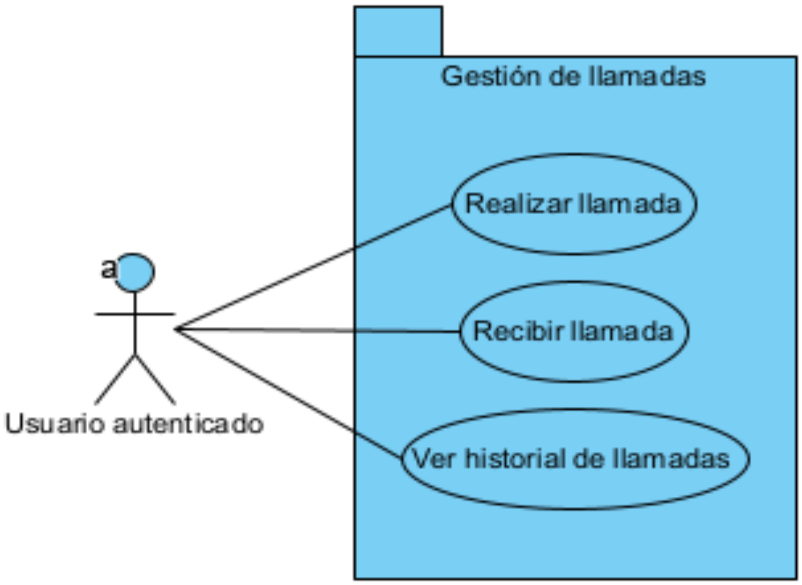


Figura 5: Diagrama de casos de uso: Gestión de llamadas

A continuación, se muestran las tablas que hacen referencia a los casos de uso de este paquete:

UC-0009	Realizar llamada	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0002] Información sobre llamadas ● [OBJ-0002] Gestión de llamadas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite realizar una llamada.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita al sistema realizar una llamada.
	2	El sistema envía una notificación de llamada al familiar seleccionado y almacena información.
	3.1 alt1	El otro familiar recibe la llamada y se almacena como “aceptada” si el familiar la acepta.
	3.2 alt1	El otro familiar recibe la llamada y se almacena como “rechazada” si el familiar la rechaza.
	3.3 alt1	El otro familiar recibe la llamada y se almacena como “perdida” si no responde al cabo de un timeout.
	4	El sistema finaliza la notificación de la llamada.
	5	El sistema almacena la información resultante.
Postcondición	-	
Excepciones	Paso	Acción
	-	-
Frecuencia	30 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	

Tabla 25: UC-0009 Realizar llamada

UC-0010	Recibir llamada	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0002] Información sobre llamadas ● [OBJ-0002] Gestión de llamadas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se reciba una llamada.	
Precondición	Se debe haber realizado dicha llamada.	
Secuencia normal	Paso	Acción
	1	El sistema envía una notificación de llamada al actor “Usuario autenticado (ACT-0002)” y almacena información.
	2	El actor “Usuario autenticado (ACT-0002)” acepta, rechaza o no responde dicha llamada.
	3	El sistema finaliza la llamada.
	4	El sistema almacena la información resultante.
Postcondición	-	
Excepciones	Paso	Acción
	-	-
Frecuencia	30 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 26: UC-0010 Recibir llamada

UC-0011	Ver historial de llamadas	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [IRQ-0002] Información sobre llamadas ● [OBJ-0002] Gestión de llamadas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite ver el historial de llamadas.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita al sistema ver el historial de llamadas.
	2	El sistema le muestra al usuario un listado con el historial de llamadas ordenadas por orden cronológico.
Postcondición	-	
Excepciones	Paso	Acción
	2	Si el sistema no encuentra ninguna llamada en el historial, se muestra un mensaje que lo indique.
Frecuencia esperada	20 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 27: UC-0011 Ver historial de llamadas

iv. Gestión de alertas

Este paquete hace referencia a la creación y recepción de alertas de los usuarios, incluida la notificación de estas. Los casos de uso que componen este paquete son los siguientes:

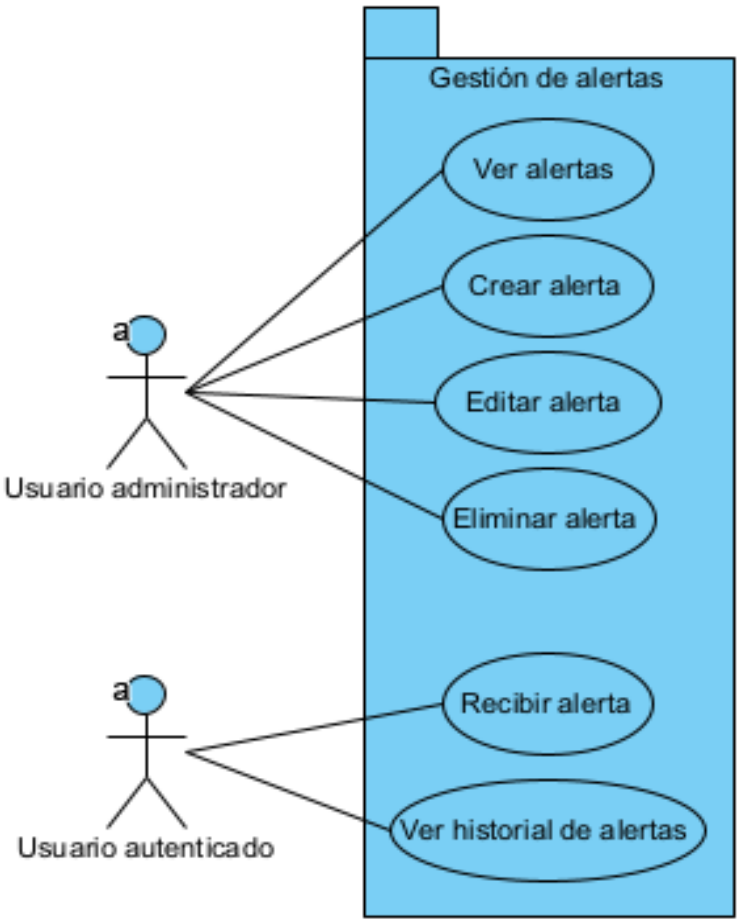


Figura 6: Diagrama de casos de uso: Gestión de alertas

A continuación, se muestran las tablas que hacen referencia a los casos de uso de este paquete:

UC-0012	Ver alertas	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0003] Información sobre alertas ● [OBJ-0003] Gestión de alertas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite ver las alertas.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario administrador (ACT-0003)” solicita al sistema ver las alertas establecidas.
	2	El sistema le muestra al usuario un listado con las alertas ordenadas por orden cronológico.
Postcondición	-	
Excepciones	Paso	Acción
	2	Si el sistema no encuentra ninguna alerta creada, se muestra un mensaje que lo indique.
Frecuencia esperada	20 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 28: UC-0012 Ver alertas

UC-0013	Crear alerta	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0003] Información sobre alertas ● [OBJ-0003] Gestión de alertas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite crear una alerta.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario administrador (ACT-0003)” solicita al sistema crear una alerta.
	2	El sistema le muestra al usuario un formulario para crear las alertas.
	3	El sistema solicita al usuario los datos correspondientes al familiar asignado, la hora, etc.
	4	El actor “Usuario administrador (ACT-0003)” proporciona los datos solicitados y solicita crear la alerta.
	5	El sistema almacena la información resultante.
Postcondición	-	
Excepciones	Paso	Acción
	-	-
Frecuencia	10 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 29: UC-0013 Crear alerta

UC-0014	Editar alerta	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0003] Información sobre alertas ● [OBJ-0003] Gestión de alertas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite editar una alerta.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario administrador (ACT-0003)” solicita al sistema editar una alerta.
	2	El sistema le muestra al usuario un formulario para editar las alertas.
	3	El sistema le muestra al usuario los datos correspondientes a la alerta para que los pueda modificar.
	4	El actor “Usuario administrador (ACT-0003)” proporciona los datos y solicita la modificación.
	5	El sistema almacena la información resultante.
Postcondición	-	
Excepciones	Paso	Acción
	4	Si el usuario cancela los cambios, el sistema mantiene la información anterior de la alerta.
Frecuencia	5 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 30: UC-0014 Editar alerta

UC-0015	Eliminar alerta	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0003] Información sobre alertas ● [OBJ-0003] Gestión de alertas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite eliminar una alerta.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario administrador (ACT-0003)” solicita al sistema eliminar una alerta.
	2	El sistema solicita la confirmación para eliminar dicha alerta.
	3	El actor “Usuario administrador (ACT-0003)” confirma la eliminación de la alerta.
	4	El sistema almacena la información resultante.
Postcondición	-	
Excepciones	Paso	Acción
	3	Si el usuario cancela la operación, el sistema cancela la eliminación de la alerta.
Frecuencia esperada	5 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 31: UC-0015 Eliminar alerta

UC-0016	Recibir alerta	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0003] Información sobre alertas ● [OBJ-0003] Gestión de alertas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se reciba una alerta.	
Precondición	Se debe haber establecido dicha alarma.	
Secuencia normal	Paso	Acción
	1	El sistema envía una notificación de alerta al actor “Usuario autenticado (ACT-0002)” y almacena información.
	2	El actor “Usuario autenticado (ACT-0002)” detiene dicha alarma.
	3	El sistema finaliza la notificación de la alerta.
	4	El sistema almacena la información resultante.
Postcondición	-	
Excepciones	Paso	Acción
	-	-
Frecuencia esperada	10 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 32: UC-0016 Recibir alerta

UC-0017	Ver historial de alertas	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [IRQ-0003] Información sobre alertas ● [OBJ-0003] Gestión de alertas	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite visualizar el historial de alertas.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita al sistema ver el historial de alertas.
	2	El sistema le muestra al usuario un listado con el historial de alertas.
Postcondición	-	
Excepciones	Paso	Acción
	-	-
Frecuencia esperada	20 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 33: UC-0017 Ver historial de alertas

v. Gestión de familiares

Este paquete hace referencia a la visualización de contactos o familiares añadidos y el acceso a sus respectivos perfiles. Los casos de uso que componen este paquete son los siguientes:

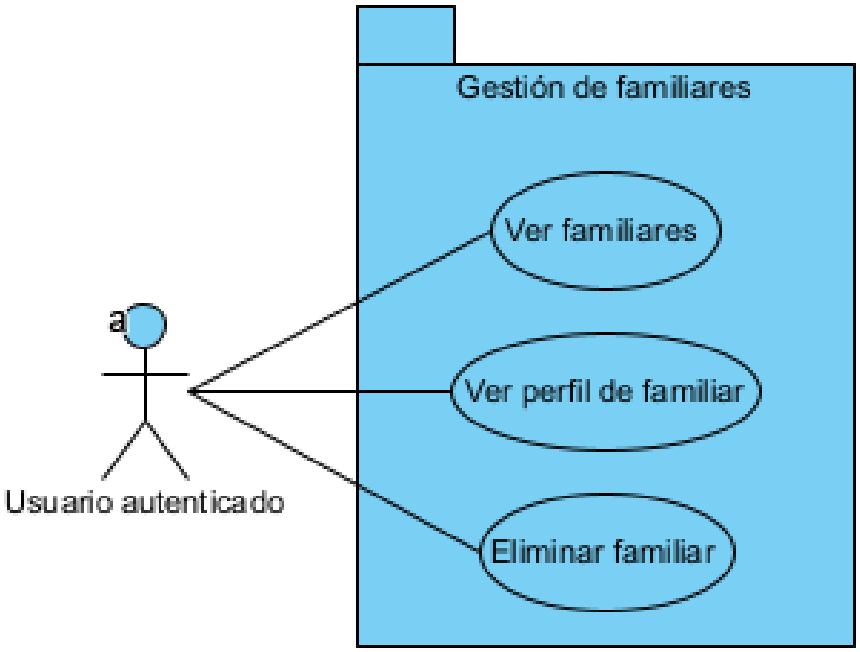


Figura 7: Diagrama de casos de uso: Gestión de familiares

A continuación, se muestran las tablas que hacen referencia a los casos de uso de este paquete:

UC-0018	Ver familiares	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [OBJ-0001] Gestión de usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite ver los familiares.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita al sistema ver los familiares añadidos.
	2	El sistema le muestra al usuario un listado con los familiares ordenados por orden alfabético.
Postcondición	-	
Excepciones	Paso	Acción
	2	Si el sistema no encuentra ningún familiar agregado, se muestra un mensaje que lo indique.
Frecuencia esperada	12 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 34: UC-0018 Ver familiares

UC-0019	Ver perfil de familiar	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [OBJ-0001] Gestión de usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite ver el perfil de un familiar.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita al sistema ver el perfil de un familiar.
	2	El sistema le muestra al usuario la información del perfil del familiar.
Postcondición	-	
Excepciones	Paso	Acción
	-	-
Frecuencia esperada		
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 35: UC-0019 Ver perfil de familiar

UC-0020	Eliminar familiar	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [OBJ-0001] Gestión de usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite eliminar un familiar.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita al sistema eliminar un familiar.
	2	El sistema solicita la confirmación para eliminar dicho familiar.
	3	El actor “Usuario autenticado (ACT-0002)” confirma la eliminación del familiar.
	4	El sistema almacena la información resultante.
Postcondición	-	
Excepciones	Paso	Acción
	-	-
Frecuencia esperada	3 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 36: UC-0020 Eliminar familiar

vi. Gestión de solicitudes

Este paquete hace referencia al envío y recepción de solicitudes de familiares, así como la visualización de solicitudes pendientes o enviadas. Los casos de uso que componen este paquete son los siguientes:

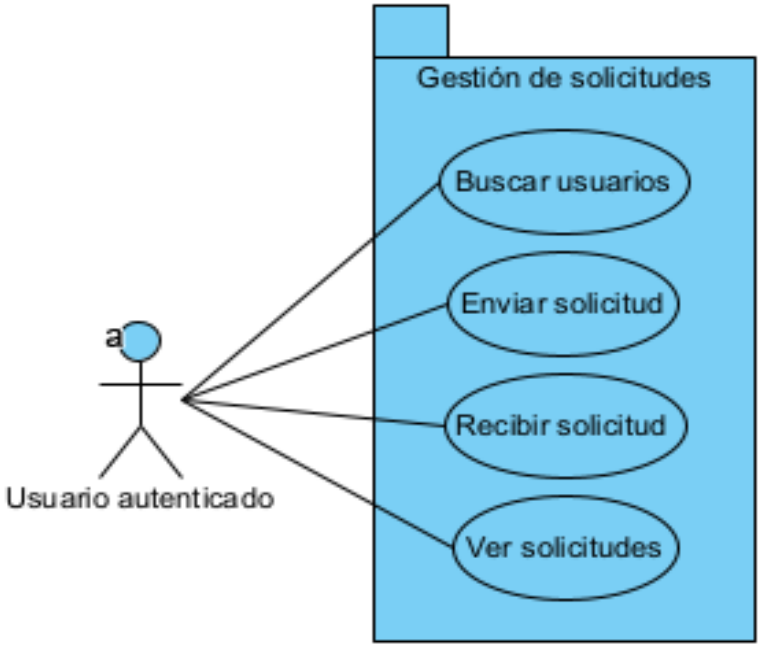


Figura 8: Diagrama de casos de uso: Gestión de solicitudes

A continuación, se muestran las tablas que hacen referencia a los casos de uso de este paquete:

UC-0021	Buscar usuarios	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0001] Información sobre usuarios ● [OBJ-0001] Gestión de usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite buscar un usuario.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita al sistema buscar un usuario.
	2	El sistema solicita al usuario el correo electrónico del familiar a buscar.
	3	El actor “Usuario autenticado (ACT-0002)” proporciona el correo electrónico y acepta la búsqueda.
	4	El sistema muestra los resultados obtenidos de la búsqueda.
Postcondición	-	
Excepciones	Paso	Acción
	4	Si el sistema no encuentra resultados, se muestra un mensaje que lo indique.
Frecuencia esperada	15 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 37: UC-0021 Buscar usuarios

UC-0022	Enviar solicitud	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0004] Información sobre solicitudes ● [OBJ-0004] Gestión de solicitudes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite enviar una solicitud.	
Precondición	El familiar seleccionado no puede estar ya agregado.	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita agregar a un familiar.
	2	El sistema envía una solicitud a dicho familiar y almacena información sobre el estado de la solicitud.
Postcondición	-	
Excepciones	Paso	Acción
	-	-
Frecuencia esperada	5 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 38: UC-0022 Enviar solicitud

UC-0023	Recibir solicitud	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0004] Información sobre solicitudes ● [OBJ-0004] Gestión de solicitudes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se reciba una solicitud.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El sistema envía una solicitud de familiar al actor “Usuario autenticado (ACT-0002)” y almacena información.
	2	El actor “Usuario autenticado (ACT-0002)” acepta o rechaza dicha solicitud.
	3	El sistema almacena la información resultante.
Postcondición	-	
Excepciones	Paso	Acción
	-	-
Frecuencia esperada	5 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 39: UC-0023 Recibir solicitud

UC-0024	Ver solicitudes	
Versión	1.0	
Autores	Paula Soria Ullán	
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes	
Dependencias	● [IRQ-0004] Información sobre solicitudes ● [OBJ-0004] Gestión de solicitudes	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando se solicite ver las solicitudes.	
Precondición	-	
Secuencia normal	Paso	Acción
	1	El actor “Usuario autenticado (ACT-0002)” solicita al sistema ver las solicitudes pendientes.
	2	El sistema le muestra al usuario un listado con el historial de solicitudes pendientes.
Postcondición	-	
Excepciones	Paso	Acción
	2	Si el sistema no encuentra solicitudes pendientes, se muestra un mensaje que lo indique.
Frecuencia esperada	5 veces por mes	
Importancia	Vital	
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 40: UC-0024 Ver solicitudes

4.3. Requisitos no funcionales

En este apartado se definirán los requisitos que imponen restricciones en el diseño y en la implementación, así como estándares de calidad.

NFR-0001	Usabilidad
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Dependencias	Ninguno
Descripción	El sistema deberá proporcionar una interacción sencilla e intuitiva para todos los usuarios, independientemente de su edad.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 41: NFR-0001 Usabilidad

NFR-0002	Escalabilidad
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Dependencias	Ninguno
Descripción	El sistema deberá ser escalable, es decir, que se adapte correctamente al crecimiento de la aplicación.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 42: NFR-0002 Escalabilidad

NFR-0003	Disponibilidad
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Dependencias	Ninguno
Descripción	El sistema deberá responder a las peticiones de los usuarios dentro de unos límites de tiempo adecuados.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 43: NFR-0003 Disponibilidad

NFR-0004	Concurrencia
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Dependencias	Ninguno
Descripción	El sistema debe de poder ser utilizado por varios usuarios a la vez.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 44: NFR-0004 Concurrencia

NFR-0005	Mantenimiento
Versión	1.0
Autores	Paula Soria Ullán
Fuentes	● Gabriel Villarrubia González ● Álvaro Lozano Murciego ● André Filipe Sales Mendes
Dependencias	Ninguno
Descripción	Se deben garantizar revisiones periódicas de la aplicación para garantizar su correcto funcionamiento.
Importancia	Vital
Urgencia	Inmediatamente
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 45: NFR-0005 Mantenimiento

4.4. Matriz de rastreabilidad

La matriz de rastreabilidad nos permitirá conocer la relación entre requisitos, es decir, los casos de uso y los objetivos definidos al principio del proyecto:

TRM-0001	OBJ-0001	OBJ-0002	OBJ-0003	OBJ-0004
UC-0001	X			
UC-0002	X			
UC-0003	X			
UC-0004	X			
UC-0005	X			
UC-0006	X			
UC-0007	X			
UC-0008	X			
UC-0009		X		
UC-0010		X		
UC-0011		X		
UC-0012			X	
UC-0013			X	
UC-0014			X	
UC-0015			X	
UC-0016			X	
UC-0017			X	
UC-0018			X	
UC-0019	X			
UC-0020	X			
UC-0021	X			
UC-0022				X
UC-0023				X
UC-0024				X

Tabla 46: Matriz de rastreabilidad

5. Referencias

[1] Roger S. Pressman, *Ingeniería del Software: Un Enfoque Práctico*.

[2] Amador Durán Toro and Beatriz Bernárdez Jiménez, “Metodología para la Elicitación de Requisitos de Sistemas Software.”

[3] María Moreno García and Francisco José García Peñalvo, “Ingeniería de Software I - UML (Parte 1 de 3).”

Anexo III: Análisis de requisitos software

Aplicación para facilitar las tareas de personas mayores

Trabajo de Fin de Grado de Ingeniería Informática

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Septiembre de 2022

Autora:

Paula Soria Ullán

Tutores:

Gabriel Villarrubia González

Álvaro Lozano Murciego

André Filipe Sales Mendes

Índice

Índice de figuras

1. Introducción

Este anexo tiene como objetivo documentar la etapa de análisis del sistema, la cual consiste en un análisis de los requisitos y la funcionalidad que debe cumplir la aplicación, todos ellos obtenidos en el Anexo II.

Se han consultado diversos temas, como el análisis y modelado del dominio o el desarrollo de casos de uso en el libro de Ingeniería del Software: Un Enfoque Práctico de Roger Pressman [1]. Para obtener información sobre el dominio del problema y el sistema actual, o para controlar la estructura de los productos entregables se ha seguido la metodología de Durán y Bernárdez [2]. Para la realización de los diagramas mostrados en este documento se ha utilizado la herramienta UML Visual Paradigm [3].

El anexo tiene la siguiente estructura:

- **Modelo de dominio**: Se determinan los objetos de negocio que el sistema debe gestionar y almacenar.
- **Realización de casos de uso**: Se analizan todos los casos de uso del Anexo II - Especificación de requisitos con la realización de un diagrama de secuencia, que muestran el intercambio de mensajes, para cada uno de ellos.
- **Clases de análisis**: Se muestran las clases de análisis recogidas en el apartado anterior, separadas en los paquetes correspondientes.
- **Vista arquitectura del modelo de análisis**: Recoge la vista completa de la arquitectura inicial que se ha obtenido en la fase de análisis.

2. Modelo del dominio

El objetivo del modelo de dominio es representar los conceptos clave del mundo real para el dominio del problema, así como los datos relevantes o atributos de cada uno de cada uno de ellos y las relaciones entre los mismos. Se trata de recoger las necesidades de almacenamiento y gestión de la información del sistema. Para la realización del modelo del dominio se han consultado los apuntes de UML de Ingeniería de Software I [4].

El modelo del dominio se representa con el siguiente diagrama de clases:

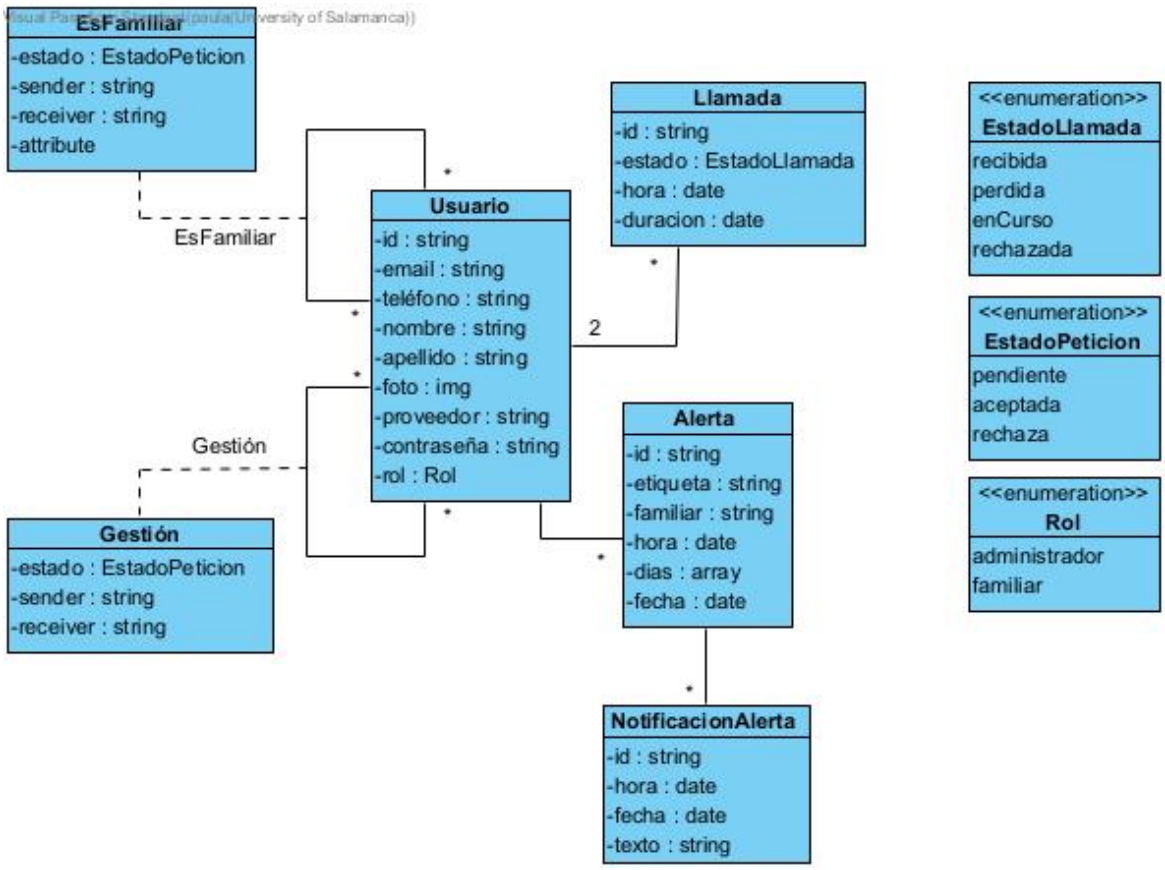


Figura 1: Modelo del dominio

En el diagrama tenemos las siguientes clases:

- **Usuario**: Representa a los usuarios, tanto administradores como familiares.
- **Llamada**: Representa la llamada entre dos usuarios.
- **Alerta**: Representa las alertas, tanto alarmas como recordatorios.
- **NotificaciónAlerta**: Representa la notificación recibida de una alerta.
- **EsFamiliar**: Asociación entre dos usuarios que indica si son familiares.
- **Gestión**: Asociación entre dos usuarios que indica si puede gestionar el administrador al familiar.

3. Realización de casos de uso - Análisis

En este apartado se refinan los casos de uso definidos en las tablas de casos de uso realizadas en el Anexo II y que se encuentran en los paquetes: Gestión de usuarios, Gestión de llamadas, Gestión de alertas y Gestión de solicitudes.

Para ello se realizan diagramas de secuencia donde se mostrarán las interacciones y mensajes entre objetos del sistema. Para la realización de los diagramas de secuencia y de los casos de uso se han consultado los apuntes de UML de Ingeniería de Software [5].

3.1. Diagramas de secuencia del paquete Gestión de autenticación

UC-0001 Registrarse:

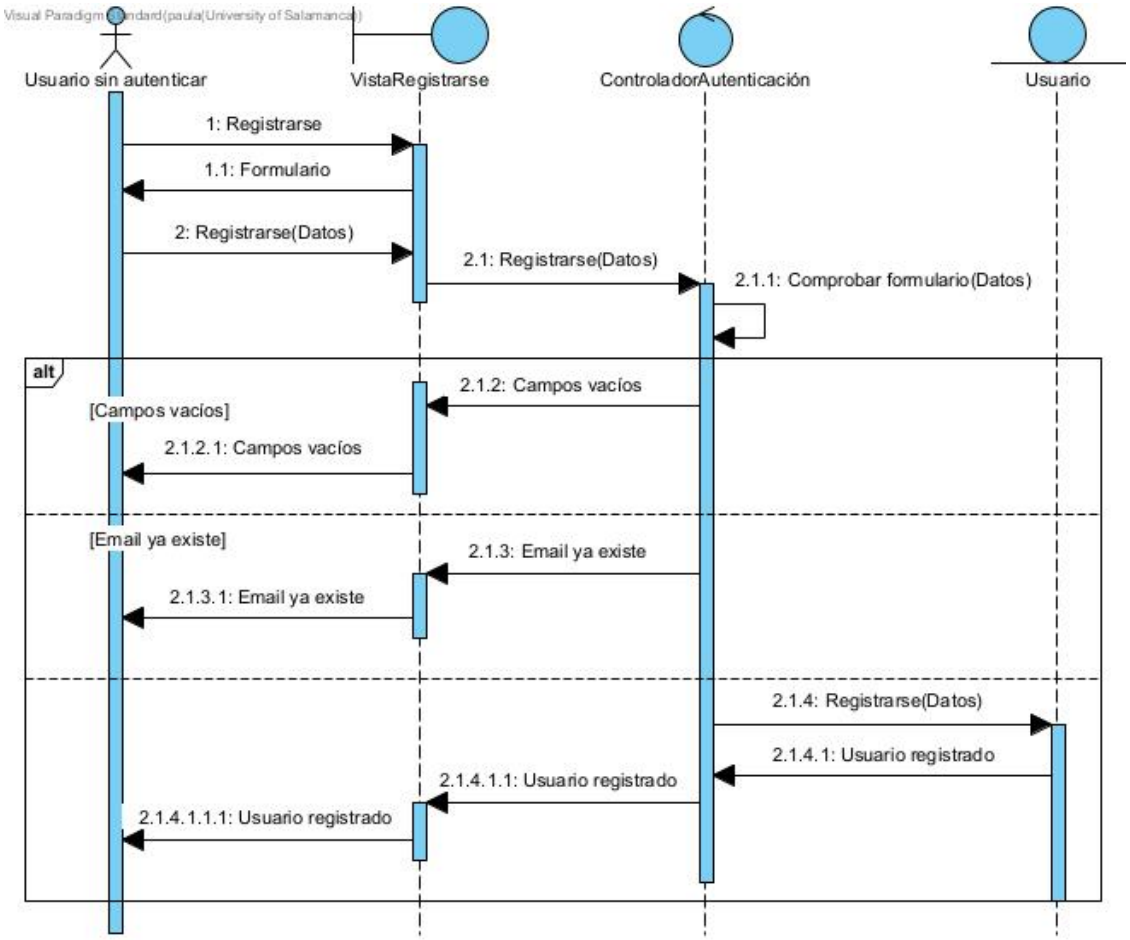


Figura 2: Diagrama de secuencia UC-0001 Registrarse

UC-0002 Iniciar sesión:

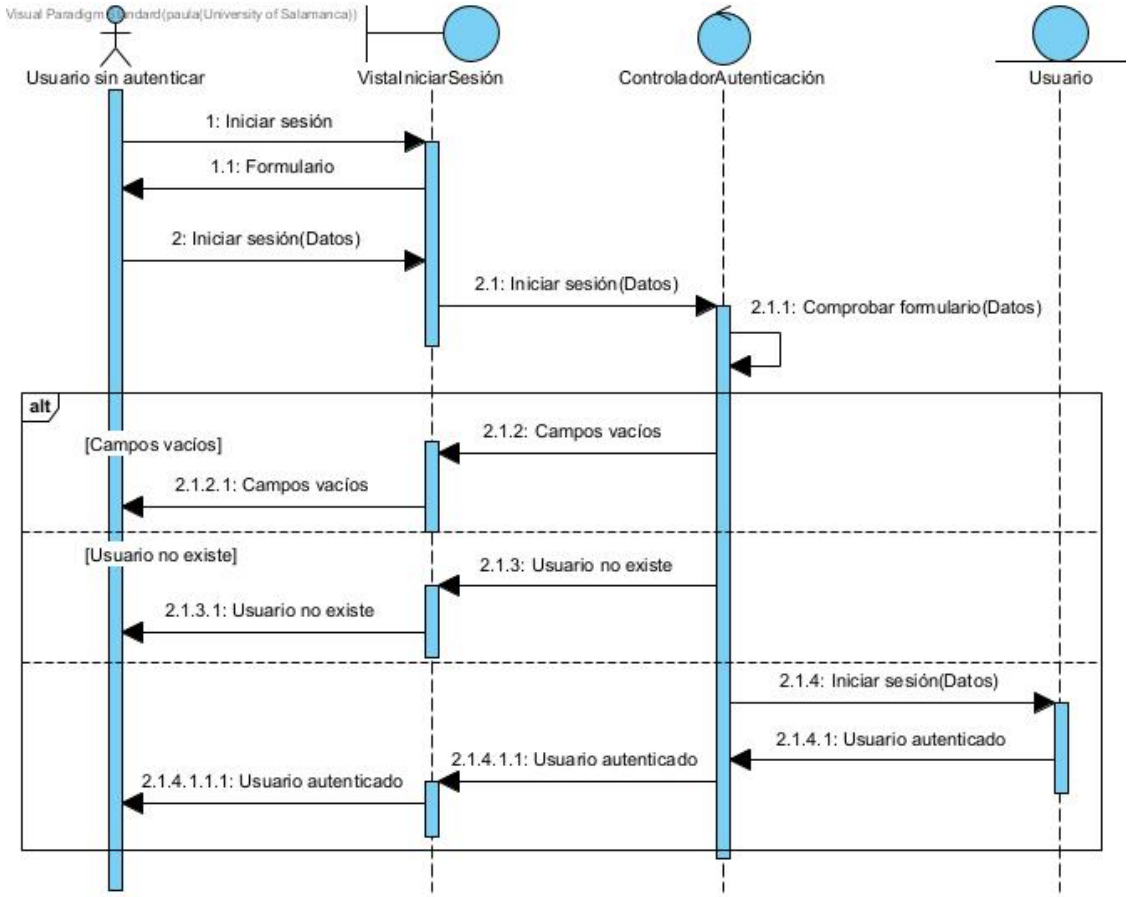


Figura 3: Diagrama de secuencia UC-0002 Iniciar sesión

UC-0003 Acceder con Google:

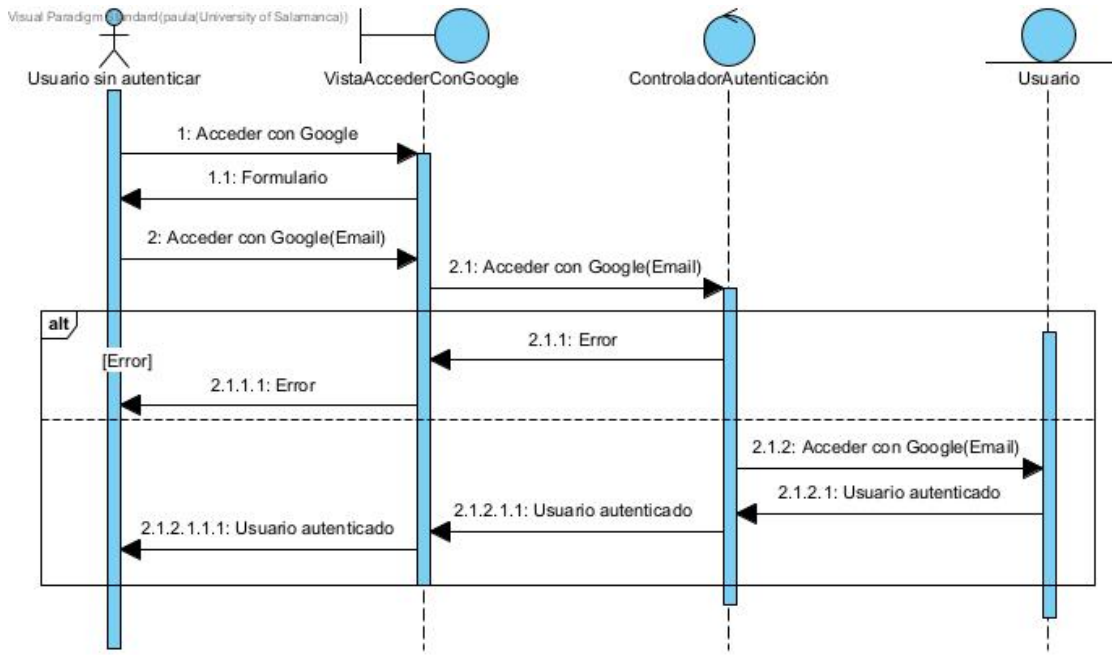


Figura 4: Diagrama de secuencia UC-0003 Acceder con Google

UC-0004 Recuperar contraseña:

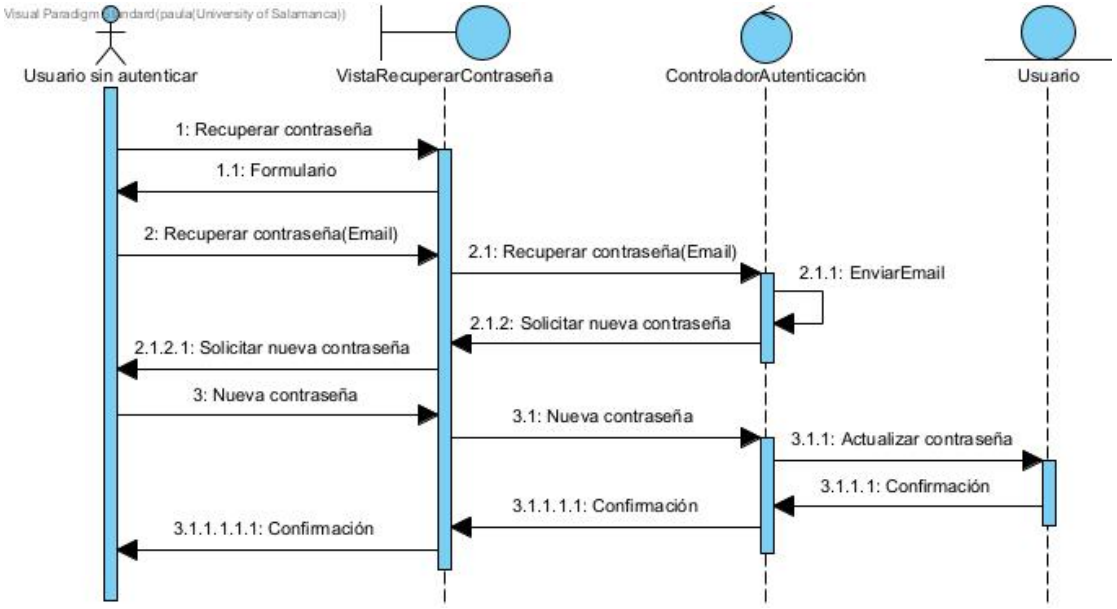


Figura 5: Diagrama de secuencia UC-0004 Recuperar contraseña

UC-0005 Cerrar sesión:

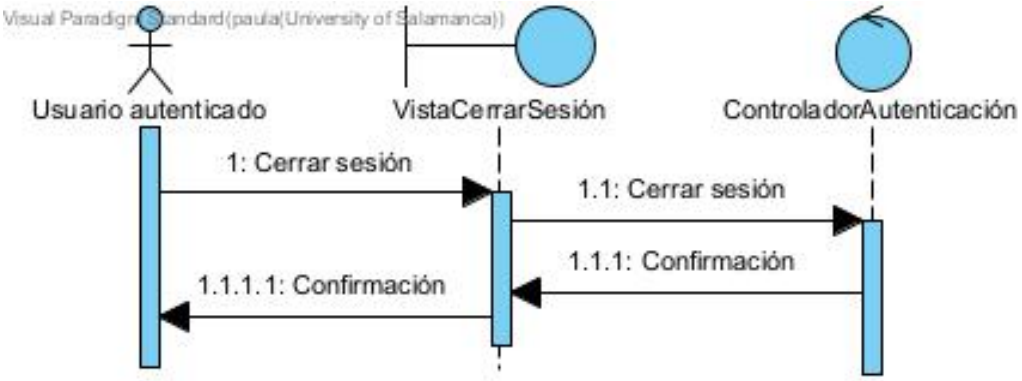


Figura 6: Diagrama de secuencia UC-0005 Cerrar sesión

3.2. Diagramas de secuencia del paquete Gestión de usuarios

UC-0006 Ver perfil:

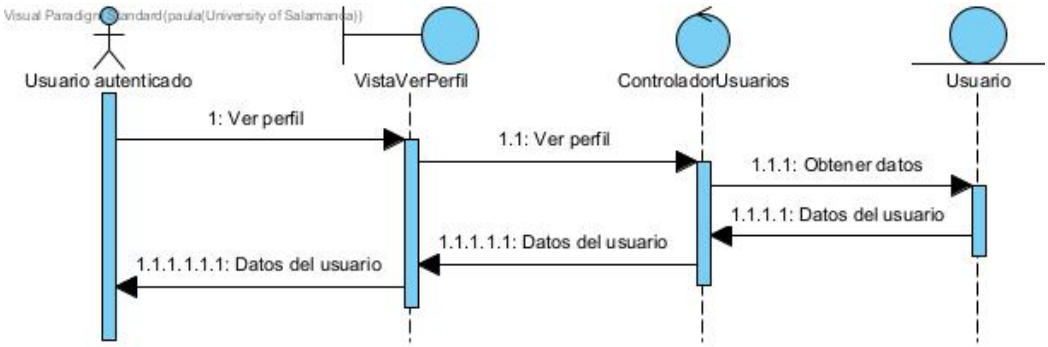


Figura 7: Diagrama de secuencia UC-0006 Ver perfil

UC-0007 Modificar imagen perfil:

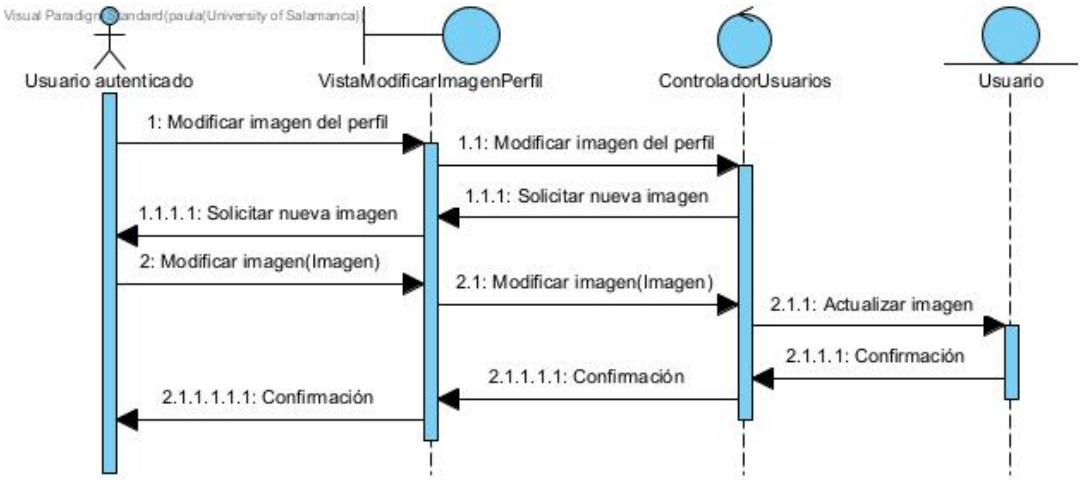


Figura 8: Diagrama de secuencia UC-0007 Modificar imagen perfil

UC-0008 Modificar nombre perfil:

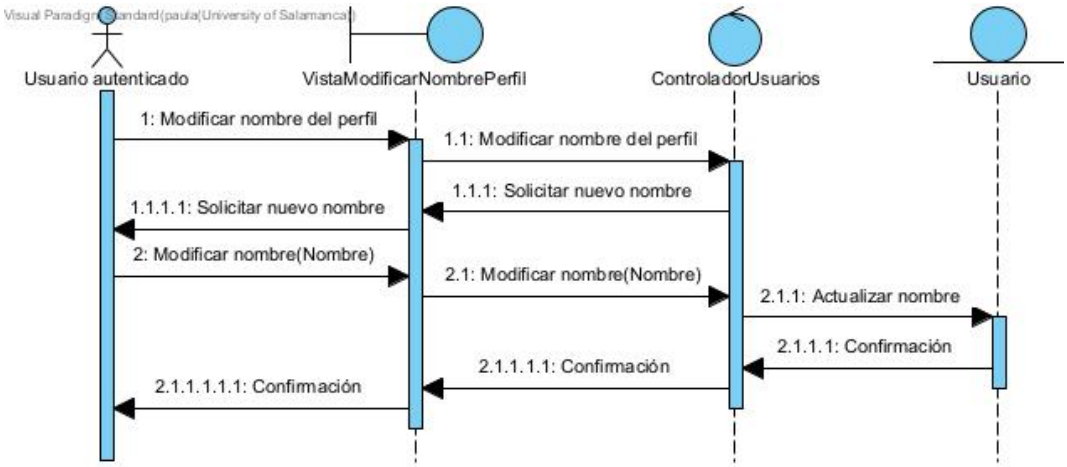


Figura 9: Diagrama de secuencia UC-0008 Modificar nombre perfil

3.3. Diagramas de secuencia del paquete Gestión de llamadas

UC-0009 Realizar llamada / UC-0010 Recibir llamada:

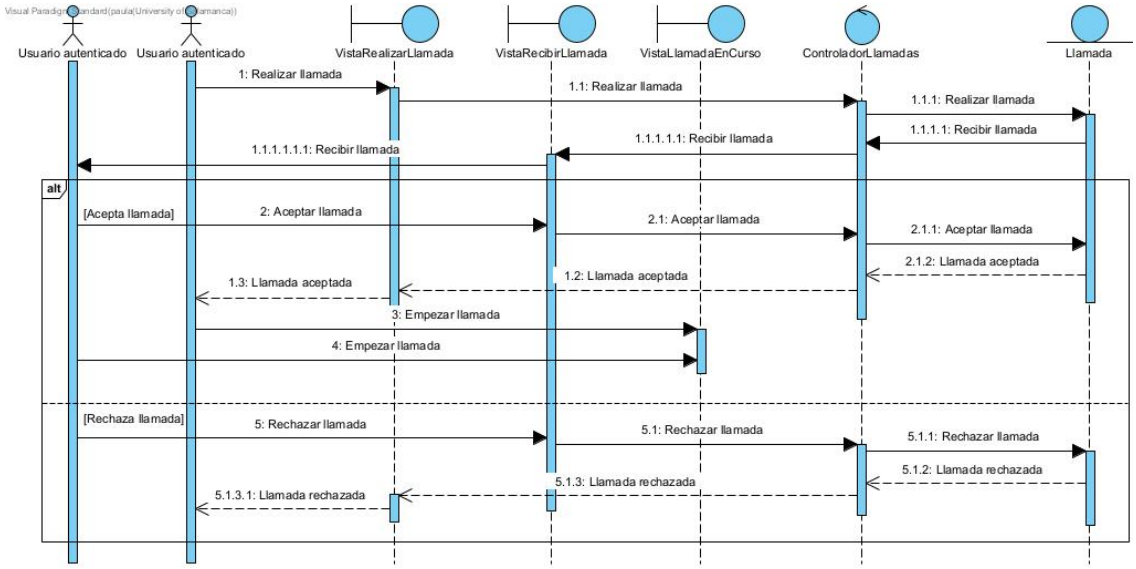


Figura 10: Diagrama de secuencia UC-0009 Realizar llamada / UC-0010 Recibir llamada

UC-0011 Ver historial de llamadas:

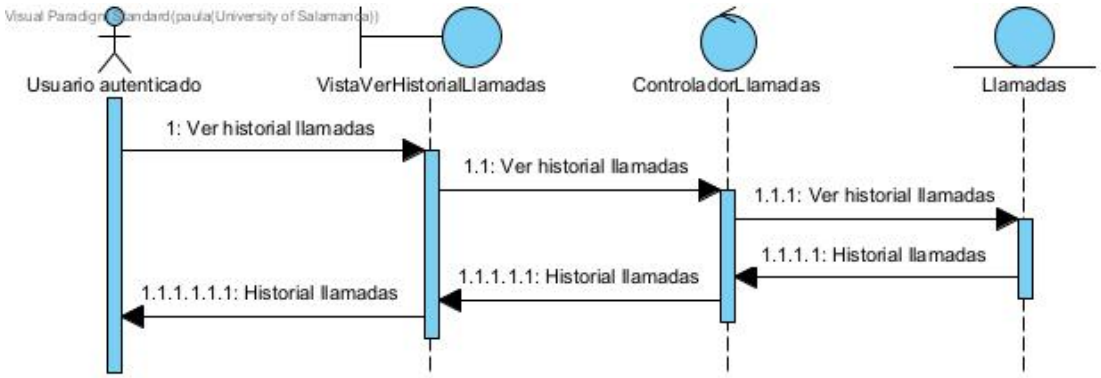


Figura 11: Diagrama de secuencia UC-0011 Ver historial de llamadas

3.4. Diagramas de secuencia del paquete Gestión de alertas

UC-0012 Ver alertas:

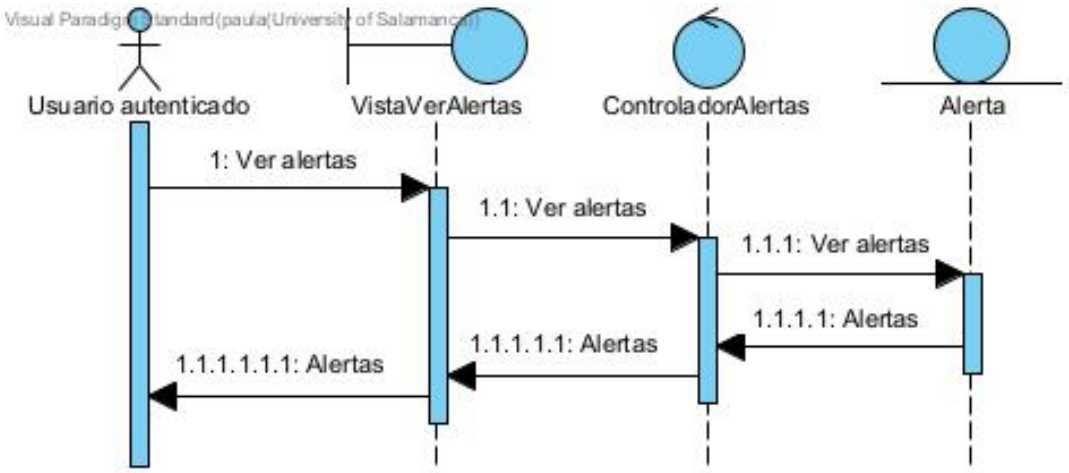


Figura 12: Diagrama de secuencia UC-0012 Ver alertas

UC-0013 Crear alerta / UC-0016 Recibir alerta:

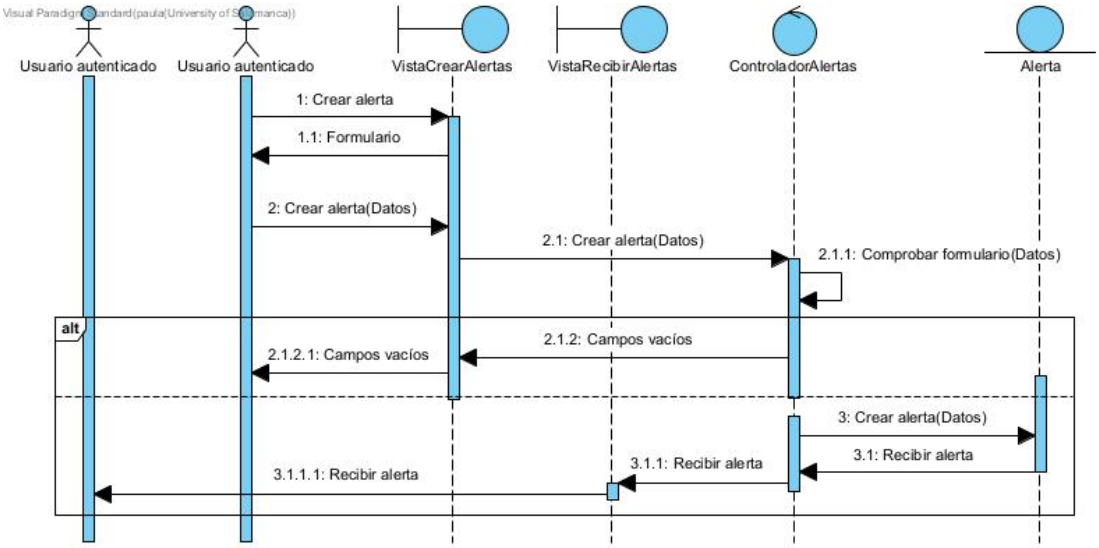


Figura 13: Diagrama de secuencia UC-0013 Crear / UC-0016 Recibir alerta

UC-0014 Editar alerta:

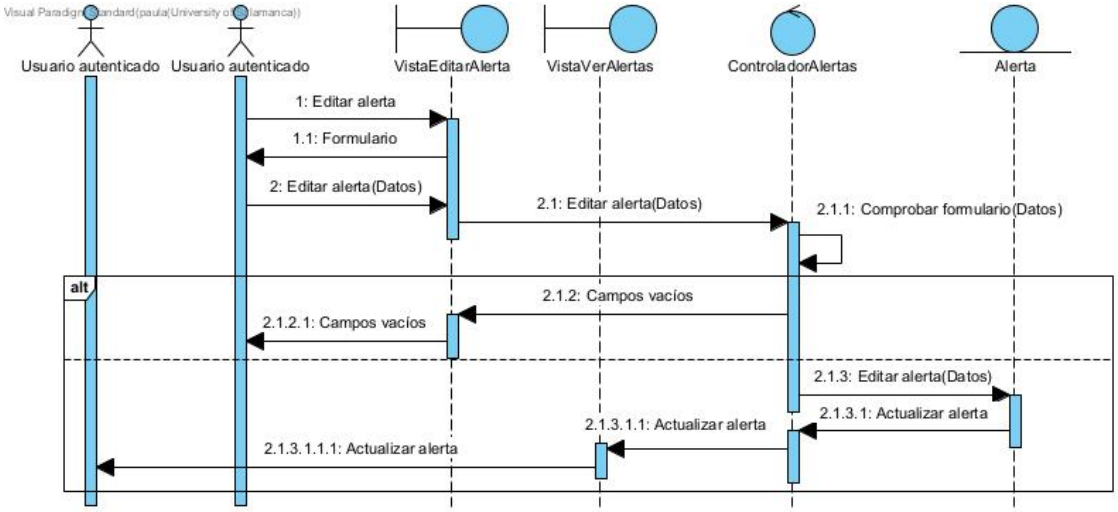


Figura 14: Diagrama de secuencia UC-0014 Editar alerta

UC-0015 Eliminar alerta:

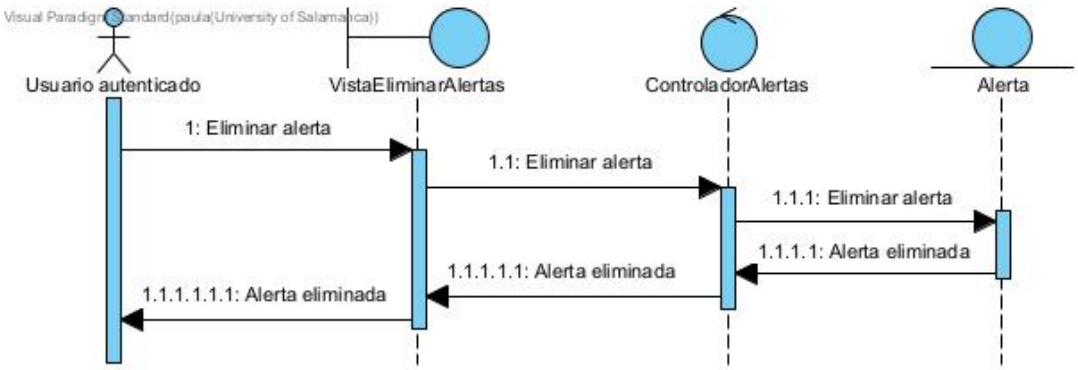


Figura 15: Diagrama de secuencia UC-0015 Eliminar alerta

UC-0017 Ver historial de alertas:

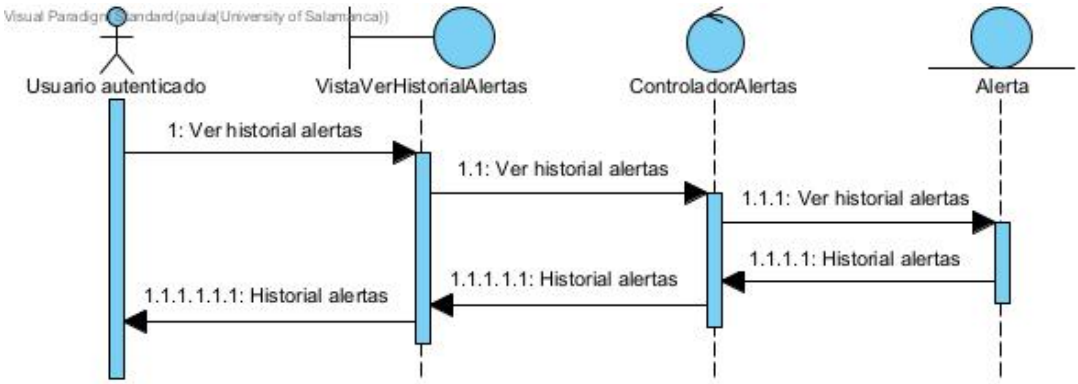


Figura 16: Diagrama de secuencia UC-0017 Ver historial de alertas

3.5. Diagramas de secuencia del paquete Gestión de familiares

UC-0018 Ver familiares:

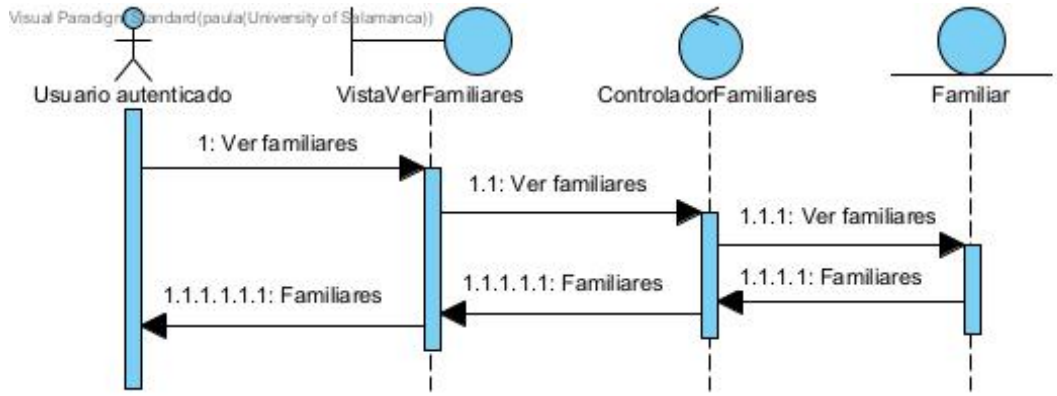


Figura 17: Diagrama de secuencia UC-0018 Ver familiares

UC-0019 Ver perfil de familiar:

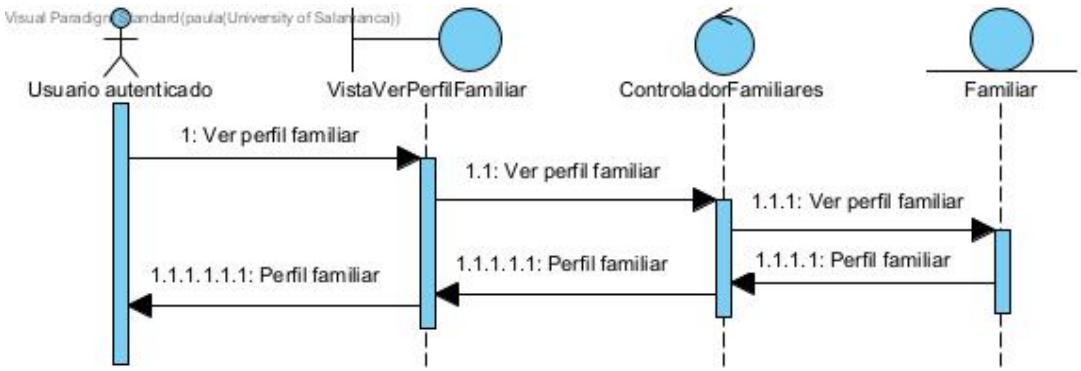


Figura 18: Diagrama de secuencia UC-0019 Ver perfil de familiar

UC-0020 Eliminar familiar:

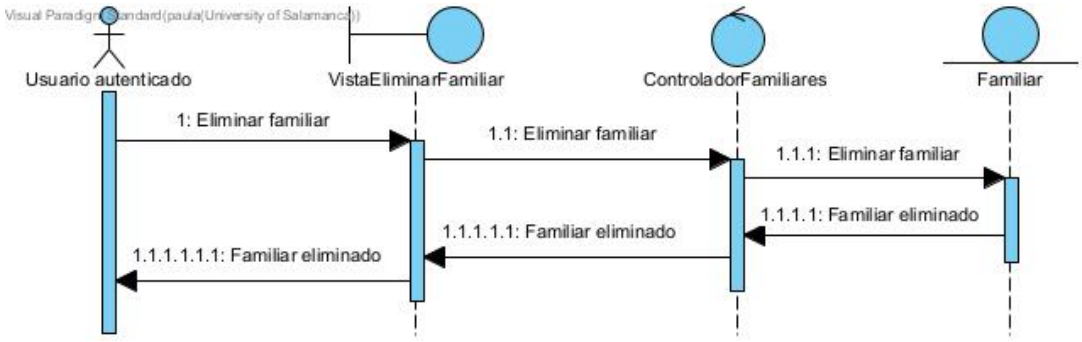


Figura 19: Diagrama de secuencia UC-0020 Eliminar familiar

3.6. Diagramas de secuencia del paquete Gestión de solicitudes

UC-0021 Buscar usuarios:

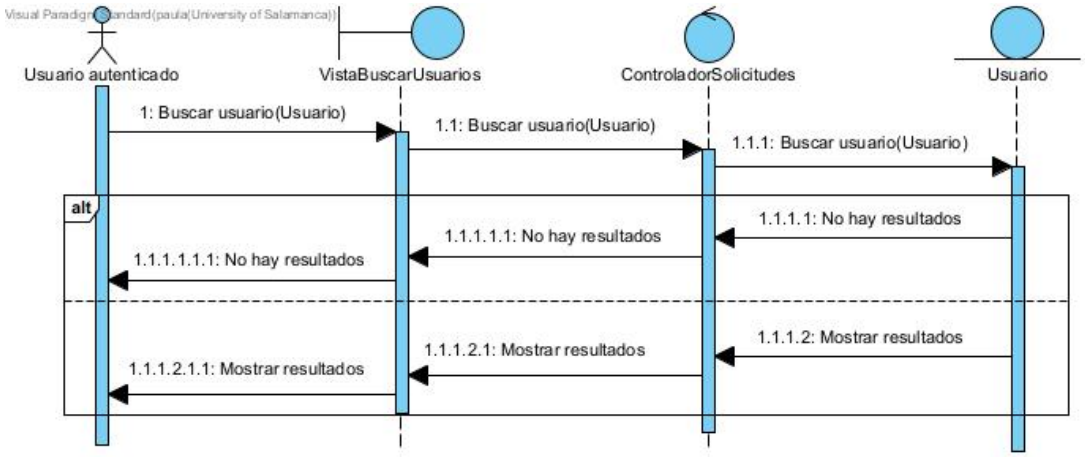


Figura 20: Diagrama de secuencia UC-0021 Buscar usuarios

UC-0022 Enviar solicitud / UC-0023 Recibir solicitud:

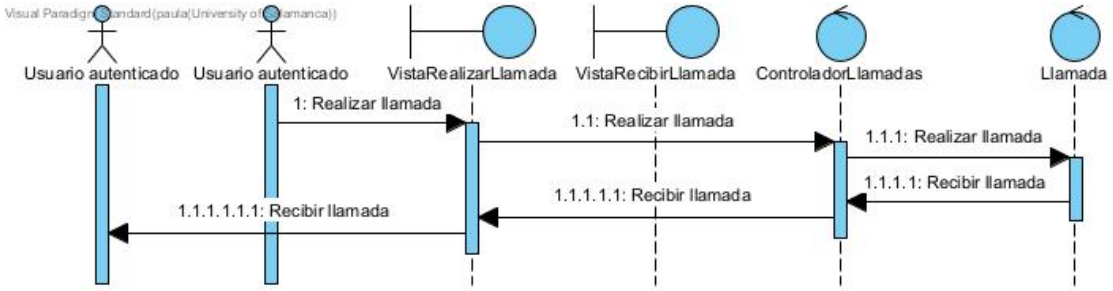


Figura 21: Diagrama de secuencia UC-0022 Enviar solicitud / UC-0023 Recibir solicitud

UC-0024 Ver solicitudes:

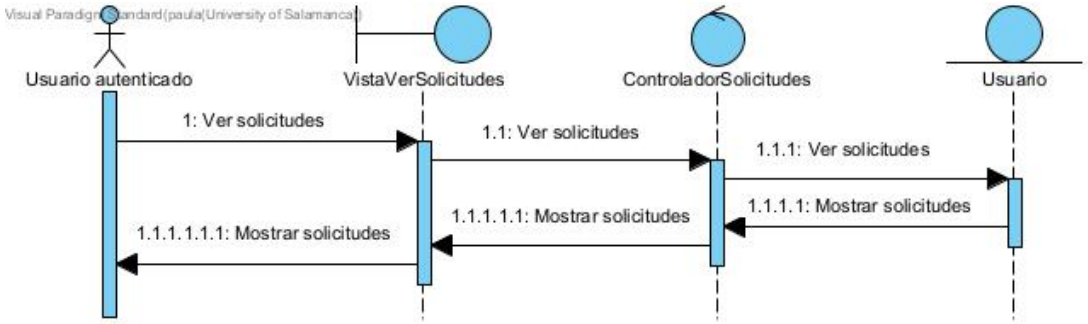


Figura 22: Diagrama de secuencia UC-0024 Ver solicitudes

4. Clases de análisis

En este apartado se representa la comunicación y distribución de las diferentes clases de análisis del apartado anterior. Para ello se utilizarán diagramas de comunicación donde estarán nuevamente agrupadas por paquetes. Si existe una clase que aparece en más de un paquete, representa una relación entre paquetes.

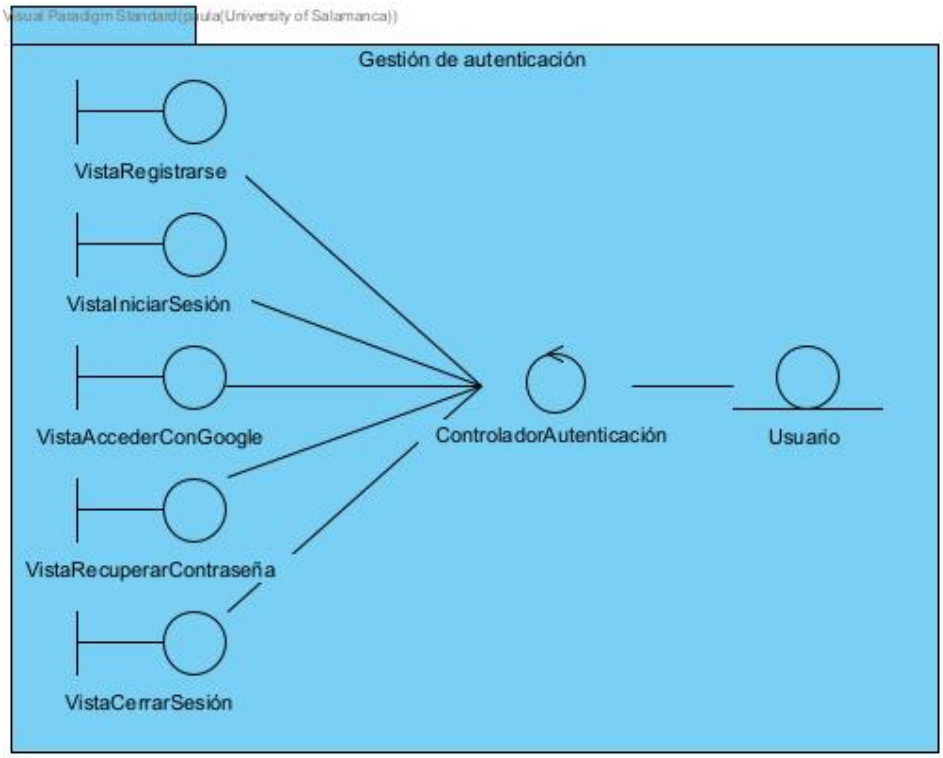


Figura 23: Diagrama de comunicación Gestión de autenticación

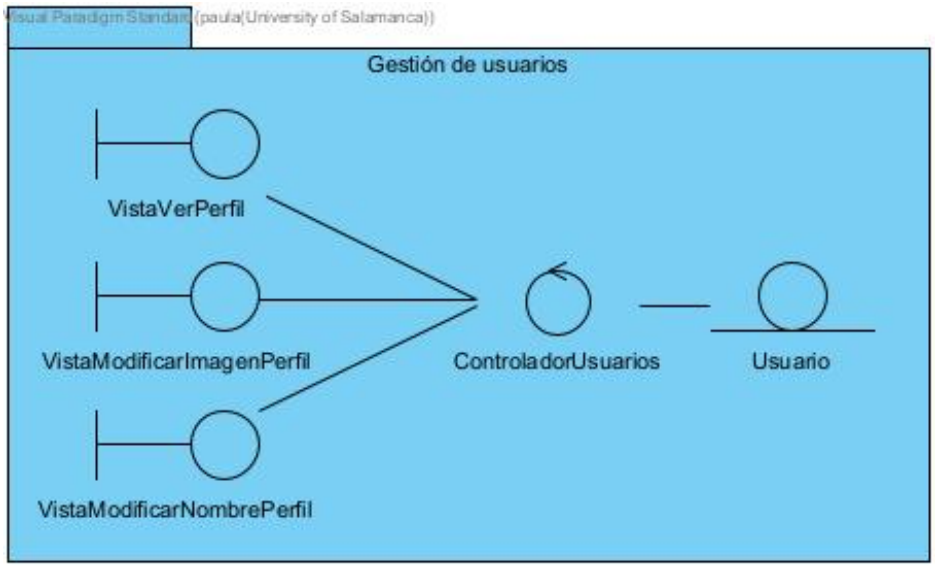


Figura 24: Diagrama de comunicación Gestión de usuarios

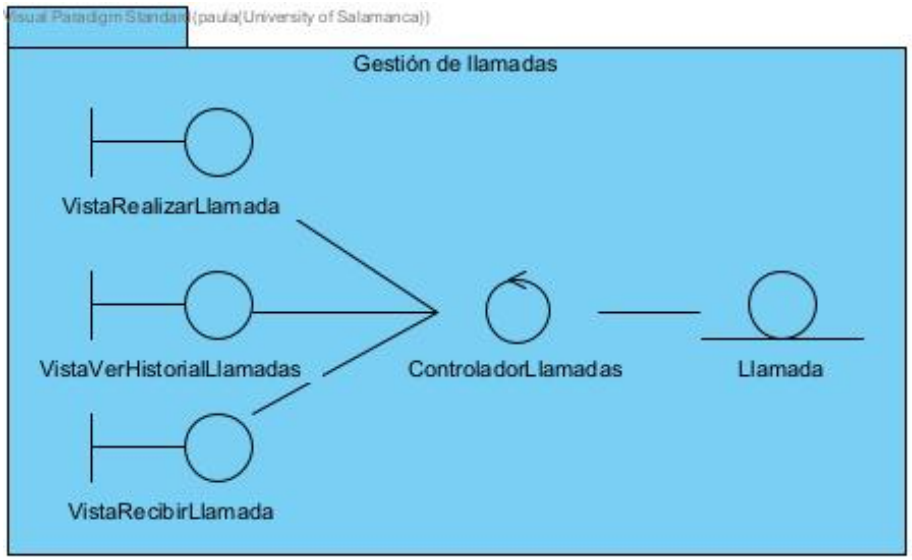


Figura 25: Diagrama de comunicación Gestión de llamadas

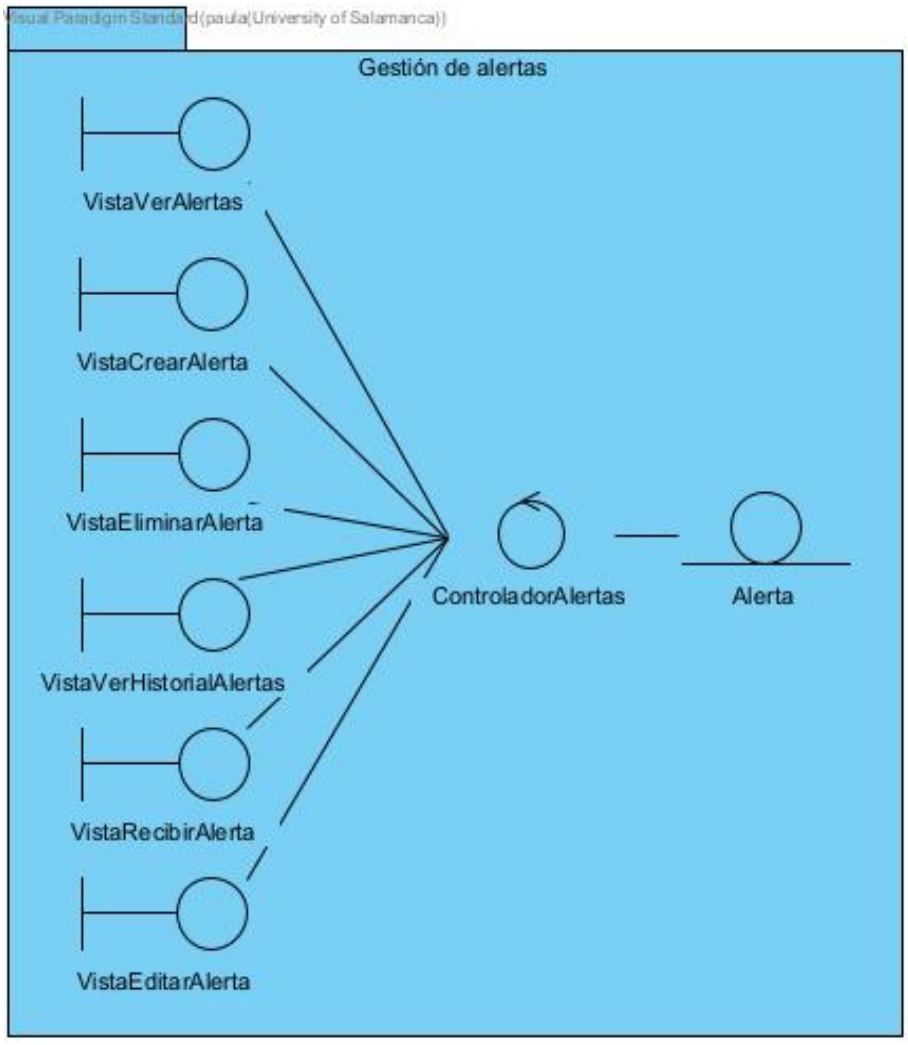


Figura 26: Diagrama de comunicación Gestión de alertas

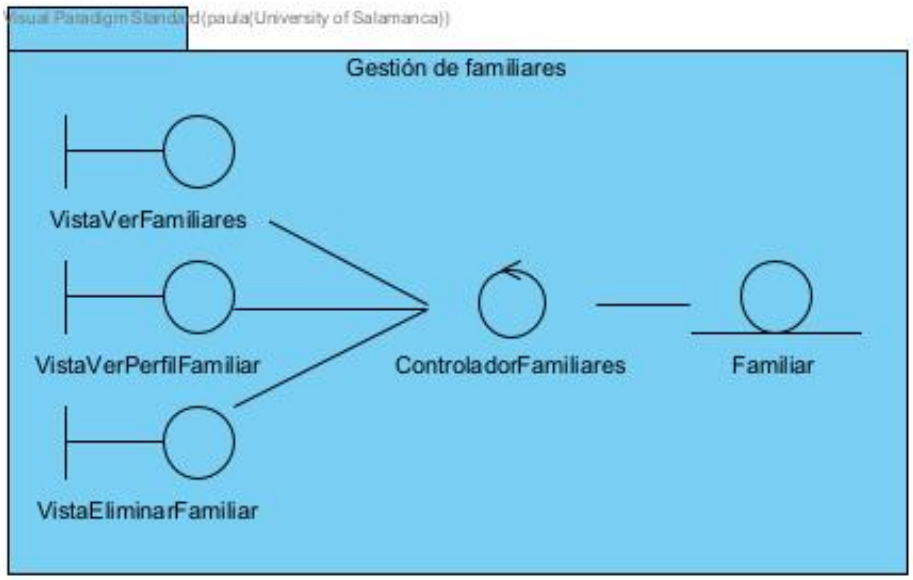


Figura 27: Diagrama de comunicación Gestión de familiares

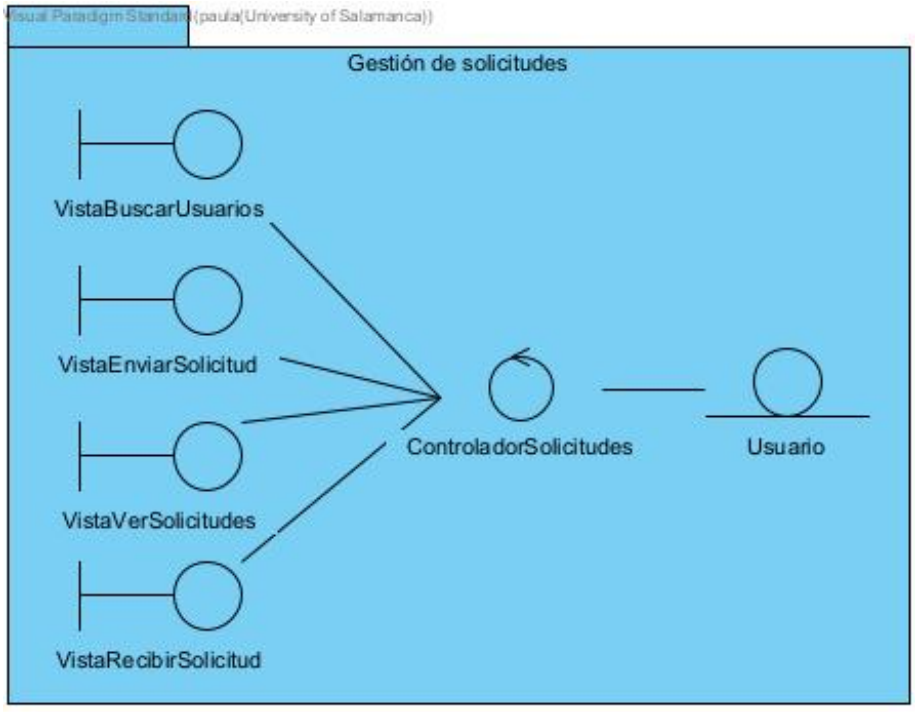


Figura 28: Diagrama de comunicación Gestión de solicitudes

5. Vista arquitectura del modelo de análisis

En este apartado se representa una aproximación de la arquitectura a partir de las clases de análisis del apartado anterior, utilizando el patrón MVC (Modelo Vista Controlador) para agrupar e identificar a qué capa pertenece cada una de las clases.

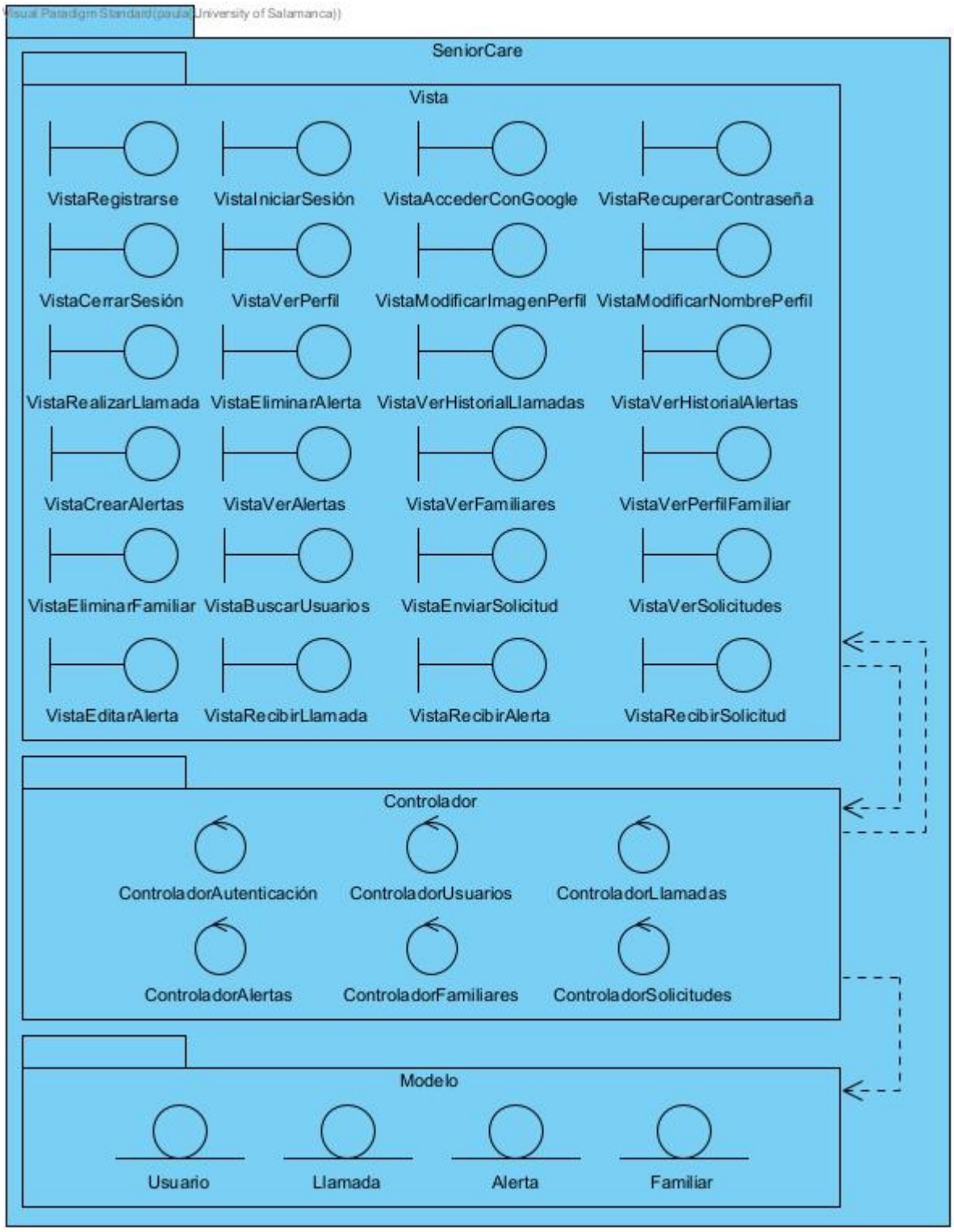


Figura 29: Arquitectura

6. Referencias

[1] Roger S. Pressman, *Ingeniería del Software: Un Enfoque Práctico*.

[2] Amador Durán Toro and Beatriz Bernárdez Jiménez, “Metodología para la Elicitación de Requisitos de Sistemas Software.”

[3] “Visual Paradigm.” https://www.visual-paradigm.com/ (accessed Jun. 21, 2022).

[4] María Moreno García and Francisco José García Peñalvo, “Ingeniería de Software I - UML (Parte 2 de 3).”

[5] María Moreno García and Francisco José García Peñalvo, “Ingeniería de Software I - UML (Parte 3 de 3).”

Anexo IV: Diseño del sistema software

Aplicación para facilitar las tareas de personas mayores

Trabajo de Fin de Grado de Ingeniería Informática

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Septiembre de 2022

Autora:

Paula Soria Ullán

Tutores:

Gabriel Villarrubia González

Álvaro Lozano Murciego

André Filipe Sales Mendes

Índice

Índice de figuras

1. Introducción

Este anexo tiene como objetivo documentar la etapa de diseño del sistema, la cual consiste en realizar una aproximación a la que será la implementación de la aplicación, es decir, trata sobre el dominio del problema. Por lo tanto, en este anexo se detallarán nombres de clases, métodos y atributos que se utilizarán finalmente.

Se han consultado diversos conceptos de Ingeniería del Software, como patrones, arquitectura o elementos del diseño de datos o del despliegue, en el libro: Un Enfoque Práctico de Roger Pressman [1]. Para conocer cómo realizar correctamente los diagramas de secuencia se han utilizado los apuntes de UML de Ingeniería de Software [2]. Para la realización de los diagramas mostrados en este documento se ha utilizado la herramienta UML Visual Paradigm [3].

El anexo tiene la siguiente estructura:

- **Diseño de la interfaz:** Se describe el proceso de diseño de los elementos gráficos de la aplicación, como la paleta de colores, el logo o el prototipado digital.
- **Modelo de diseño**:
  - **Patrones arquitectónicos**: Se detallan los patrones arquitectónicos que se utilizarán en el diseño.
  - **Subsistemas de diseño**: Se muestra la organización de los diferentes sistemas de paquetes que componen el sistema junto con las relaciones entre ellos.
  - **Clases de diseño**: Se describen de forma detallada las clases de diseño junto con los métodos y atributos, y las relaciones entre ellos.
  - **Vista arquitectónica**: Se describen de forma detallada las clases que formarán las distintas capas del patrón MVP (Modelo-Vista-Presentador) de la arquitectura diseñada.
  - **Realización de casos de uso**: Se muestran los diagramas de secuencia de diseño de los casos de uso del sistema, donde se podrá observar el intercambio de mensajes entre los distintos componentes.
- **Diseño de la base de datos**: Visión de las clases en un diagrama de tipo entidad-relación que representará la información de la base de datos.
- **Modelo de despliegue**: Muestra del despliegue de la funcionalidad de nodos y artefactos que componen el sistema.

2. Diseño de la interfaz

Para diseñar la interfaz gráfica de usuario, que serían los ficheros layout con extensión XML de la aplicación, se ha desarrollado empleando las guías de estilo marcadas por Material Design, para obtener una interacción sencilla e intuitiva.

Lo primero que se hizo fue, escoger una paleta de colores acorde con los valores que se querían transmitir. Se seleccionó una gama de colores azules grisáceos para transmitir seguridad, tranquilidad, protección, salud, confianza, calma y estabilidad.

Figura 1: Paleta de colores

Para crear una paleta de colores bien diseñada, hay que escoger un color principal, que sería el primero de los azules. A continuación, se escogen algunos colores neutros a partir de este color principal para dar dimensión en la aplicación. También hay que seleccionar un color de acento, que en este caso es el azul turquesa de más abajo, para mostrar elementos seleccionados. Por último, como colores semánticos que están presentes en todos los diseños, se escogió el rojo para representar errores, peligro o alertas, y el verde para representar éxito, seguridad o algo correcto.

Además de la paleta de colores, también se diseñó un logo que intentase representar adecuadamente los valores de la aplicación. Para ello se escogieron los colores de la paleta anterior y se seleccionó un icono de dos manos agarrándose para transmitir que la aplicación sirve para acompañar y ayudar. También hacen un corazón porque se trata de cuidar a nuestros familiares.

Figura 2: Logo SeniorCare

Por último, se realizó un primer diseño utilizando la herramienta Adobe XD [4] con la que se elaboró un boceto o prototipado digital de las pantallas de la aplicación, de forma que se viese una primera idea antes de programar, y comprobar cómo se interactuaría con los botones y cómo se desplazaría un usuario por la aplicación.

En las siguientes figuras se pueden ver los bocetos creados:

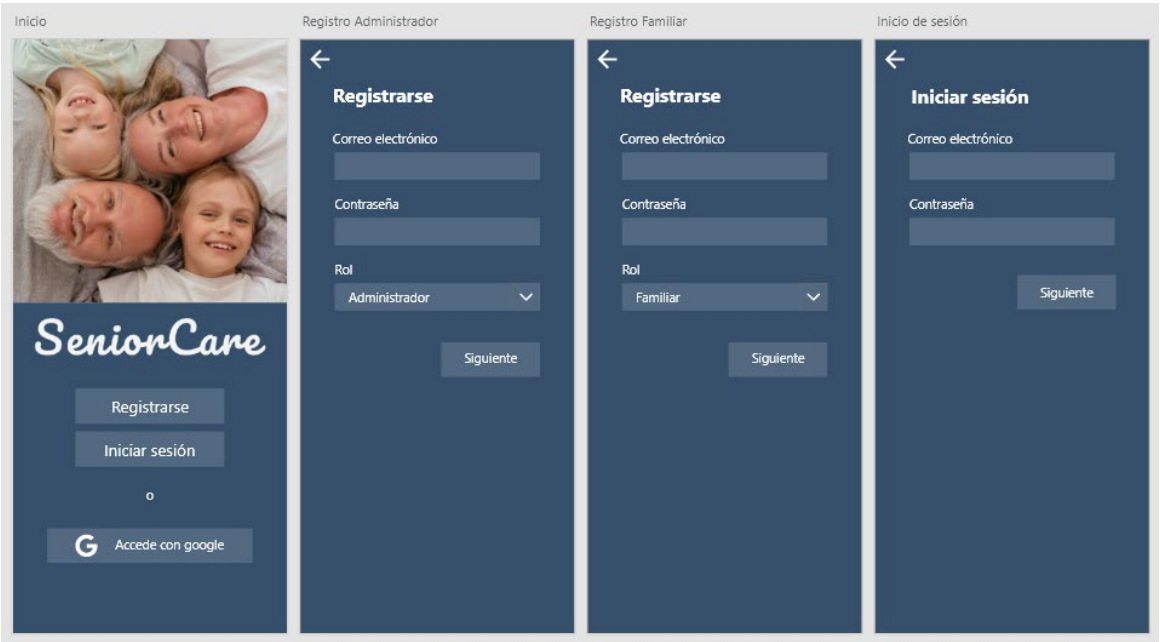


Figura 3: Boceto de la aplicación 1

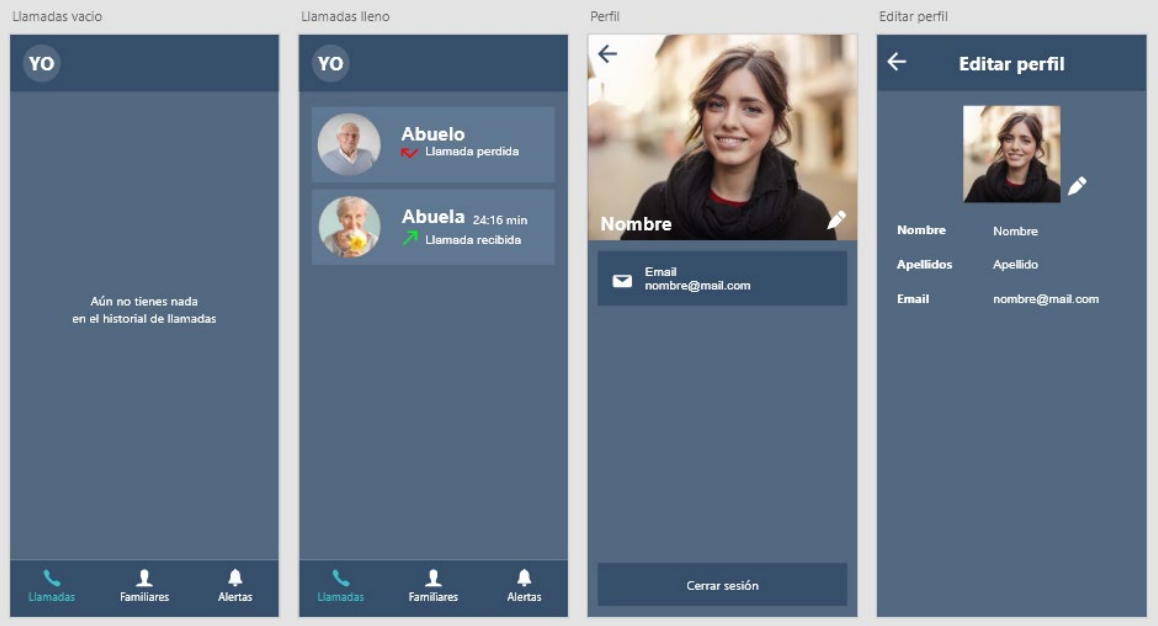


Figura 4: Boceto de la aplicación 2

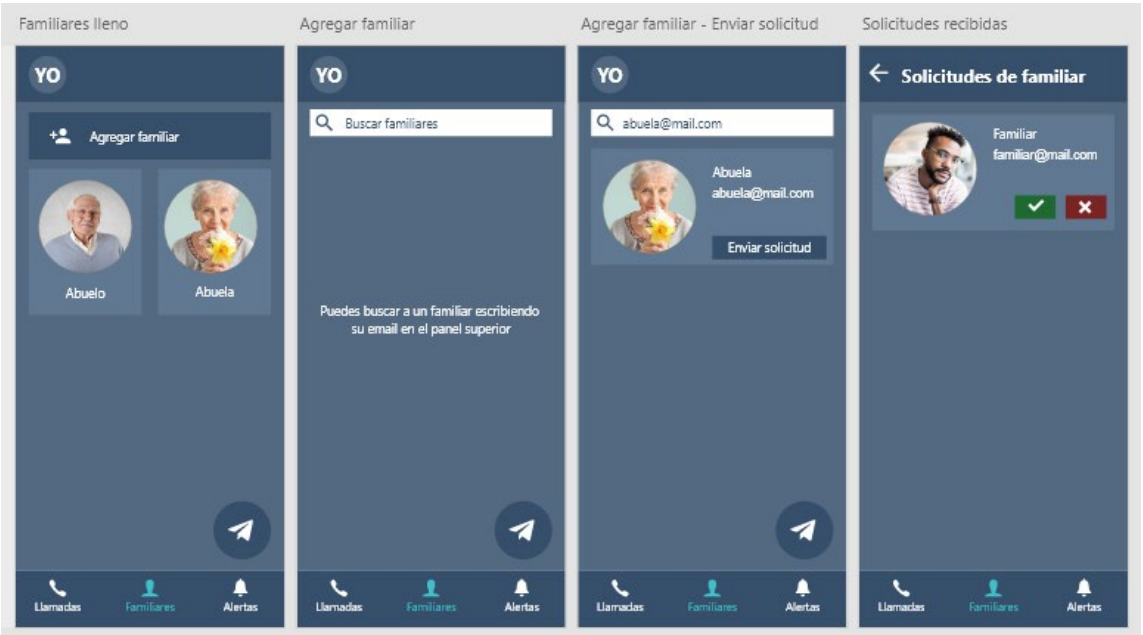


Figura 5: Boceto de la aplicación 3

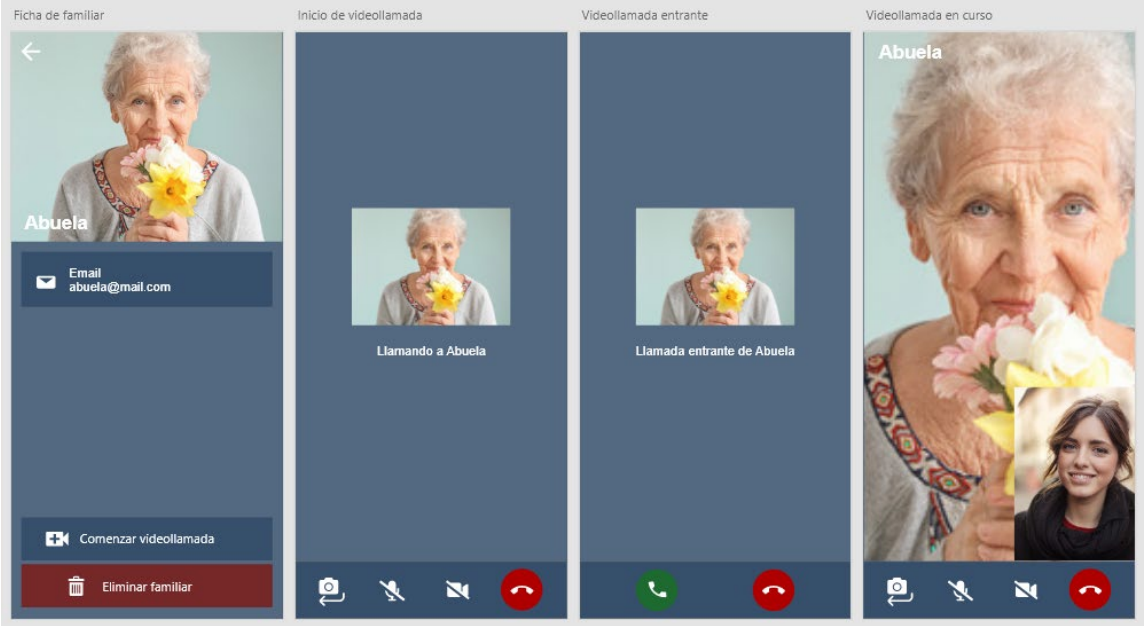


Figura 6: Boceto de la aplicación 4

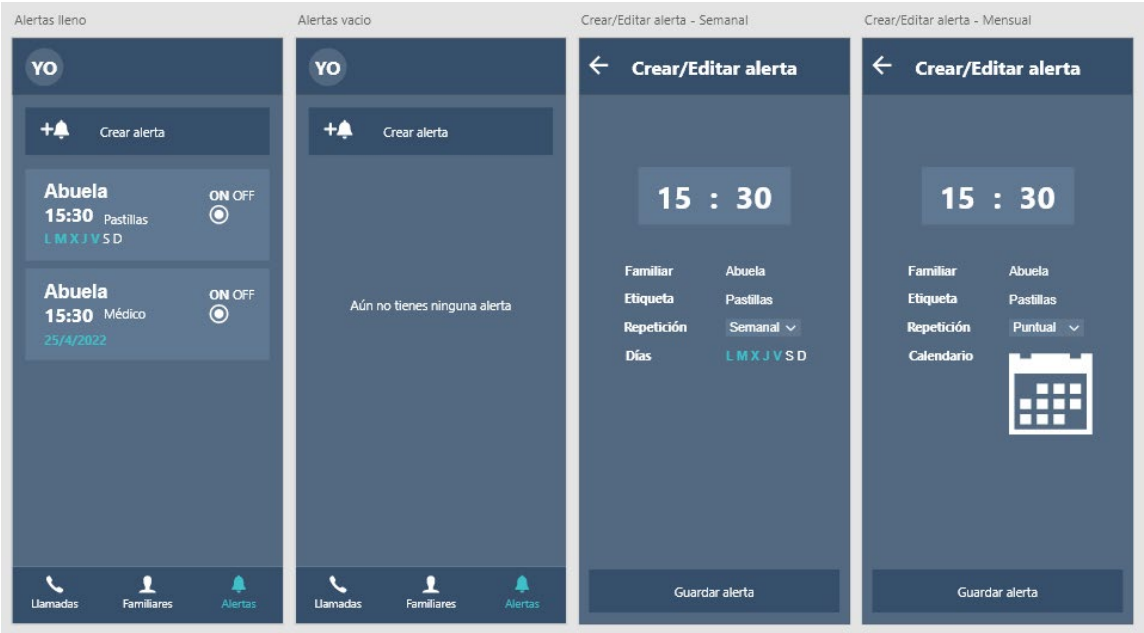


Figura 7: Boceto de la aplicación 5

3. Modelo de diseño

En esta primera fase de diseño se exponen los patrones arquitectónicos utilizados para facilitar el desarrollo del sistema además de proporcionar unas buenas prácticas del diseño de software.

Se han empleado las siguientes tecnologías:

- **Kotlin**: Se ha utilizado como lenguaje de programación en el *frontend* empleando el entorno de desarrollo de Android Studio.
- **XML**: Se ha utilizado para desarrollar las interfaces de usuario de la aplicación.
- **Firebase**: Se han empleado los distintos servicios que ofrece Firebase [5] para desarrollar el servicio de autenticación, la base de datos, el almacenamiento de imágenes, el servicio de notificaciones y la API.
- **JavaScript**: Se ha utilizado como lenguaje de programación en el *backend* mediante el uso del entorno de ejecución Node.js.

4. Patrones arquitectónicos

El patrón más importante utilizado en el desarrollo es el MVP (Modelo-Vista-Presentador), un heredero del patrón MVC (Modelo-Vista-Controlador) que sirve para estructurar correctamente las capas que componen la aplicación y es muy relevante en el desarrollo de aplicaciones Android.

4.1. Modelo - Vista - Controlador

El patrón MVC es un patrón de arquitectura de software que se utiliza para implementar interfaces de usuario, datos y lógica de control. Describe la separación de los componentes software en tres partes:

- **Modelo**: Representa los datos y la lógica de negocio.
- **Vista**: Representa visualmente los datos del modelo, el diseño y la presentación al usuario.
- **Controlador**: Intermediario entre los modelos y las vistas, es decir, controla y gestiona el flujo de datos entre ellos.

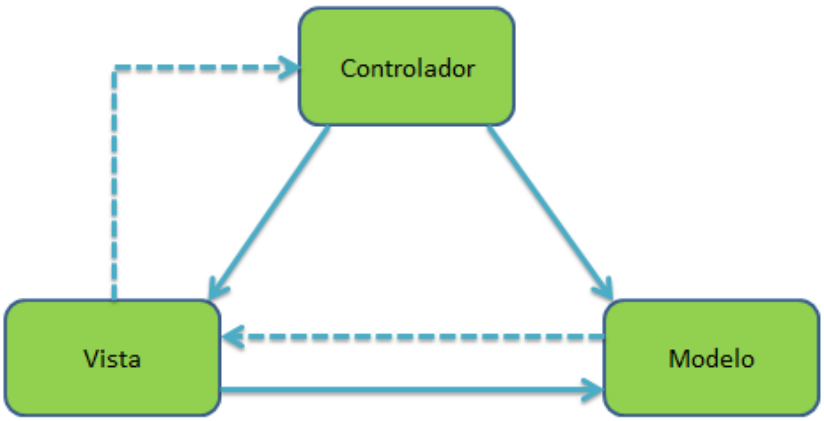


Figura 8: Patrón MVC [6]

Entre las ventajas de este patrón tendremos:

- Facilita agregar nuevos tipos de datos al ser independientes del funcionamiento del resto de capas.
- Facilita la escalabilidad de la aplicación y el mantenimiento en caso de errores.
- Permite una mayor portabilidad de la aplicación.
- Existe una alta cohesión y un bajo acoplamiento entre las capas de software, por lo que favorece la reutilización.

4.2. Modelo - Vista - Presentador

El patrón MVP es un patrón derivado del MVC y uno de los patrones más populares para organizar la capa de presentación en las aplicaciones de Android. Este patrón permite separar la capa de vista de la capa de lógica, de forma que el funcionamiento de la interfaz quede separado de la representación.

Este patrón resuelve el principal problema de las aplicaciones Android, que es el hecho de que las actividades están muy acopladas a la interfaz y a los mecanismos de acceso a datos. De esta forma conseguimos una aplicación más flexible y mantenible mediante la división del código en capas:

- **Modelo**: Capa encargada de la lógica de negocio o dominio.
- **Vista**: Capa encargada de diseñar la UI, realizar peticiones y mostrar resultados. En esta capa no debe existir lógica de negocio.
- **Presentador**: Capa encargada de interactuar con la Vista y Modelo.

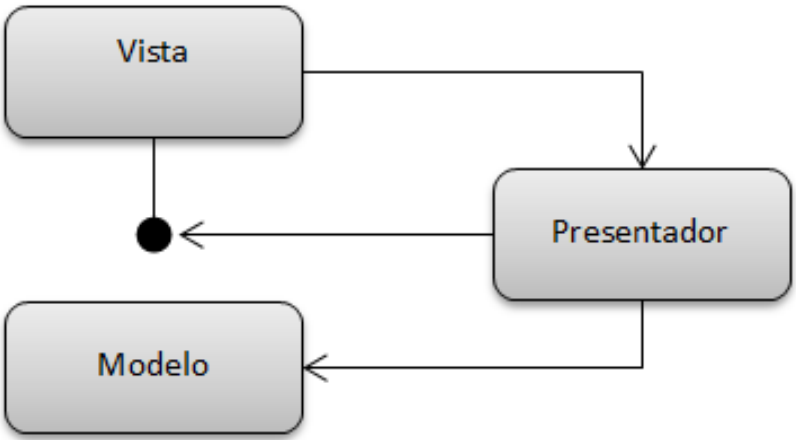


Figura 9: Patrón MVP [7]

5. Subsistemas de diseño

A partir de este punto se ha decidido organizar el sistema en subpaquetes más específicos para tener unos elementos más manejables y tener los métodos, los atributos y las relaciones entre ellos bien definidos. Tendremos los siguientes subpaquetes:

- **Paquete Aplicación**: Se trata de la aplicación Android donde se gestiona el sistema completo, tanto la parte móvil como la parte de televisión.
- **Paquete Servidor**: Tendremos la parte de las Google Cloud Functions, la base de datos de Firebase Firestore y otras funciones de Firebase que no podemos ejecutar desde el lado del cliente. También tenemos la parte de Node.js.

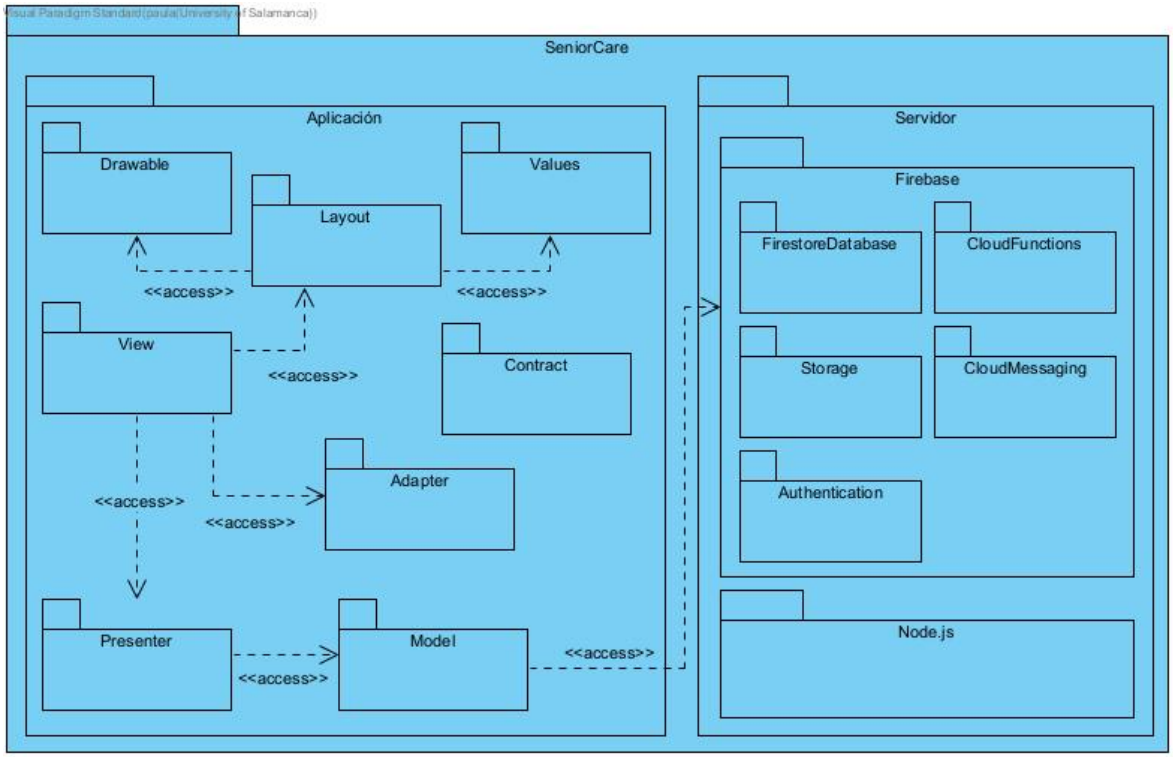


Figura 10: Subsistemas de diseño

5.1. Clases de diseño

5.1.1. Aplicación

Aquí se detallan los subsistemas de diseño del paquete Aplicación, que a su vez se dividen en otros subsistemas más pequeños.

Layout

En este subsistema tendremos los ficheros correspondientes al diseño de la interfaz gráfica de usuario en formato XML.

View

En este subsistema tendremos todas las actividades y los fragmentos, es decir, todo lo que está relacionado con la interfaz gráfica de usuario.

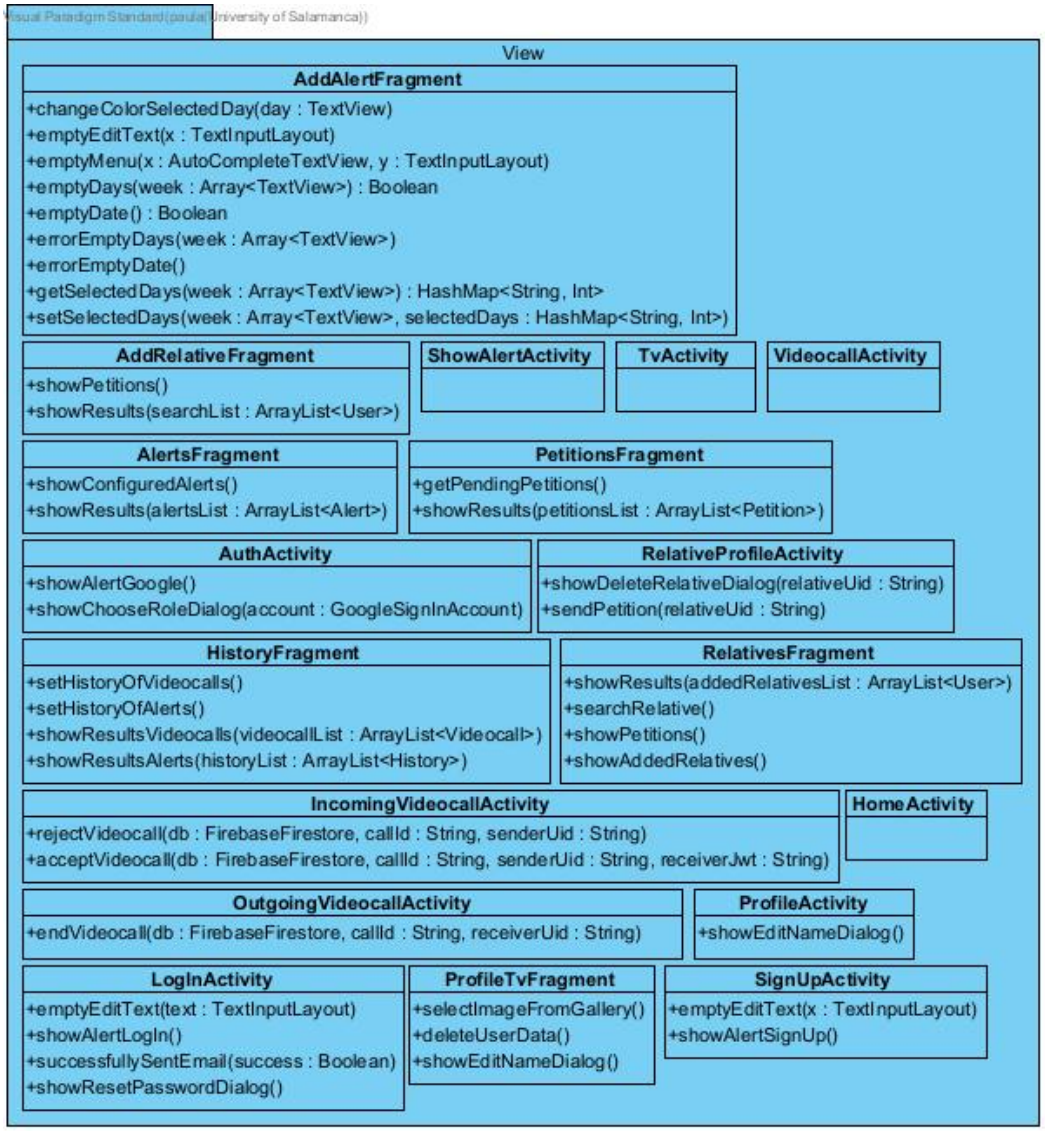


Figura 11: Subsistema View

Drawable

Aquí se almacenan los elementos gráficos de la interfaz como las imágenes y los iconos.

Values

Aquí se almacenan los archivos XML que establecen valores del diseño de la aplicación como los colores y el tema de la aplicación, o cadenas de caracteres que se utilizan para poner textos o mensajes de error.

Presenter

En este subsistema tendremos todas las clases que comunican las vistas con los modelos.

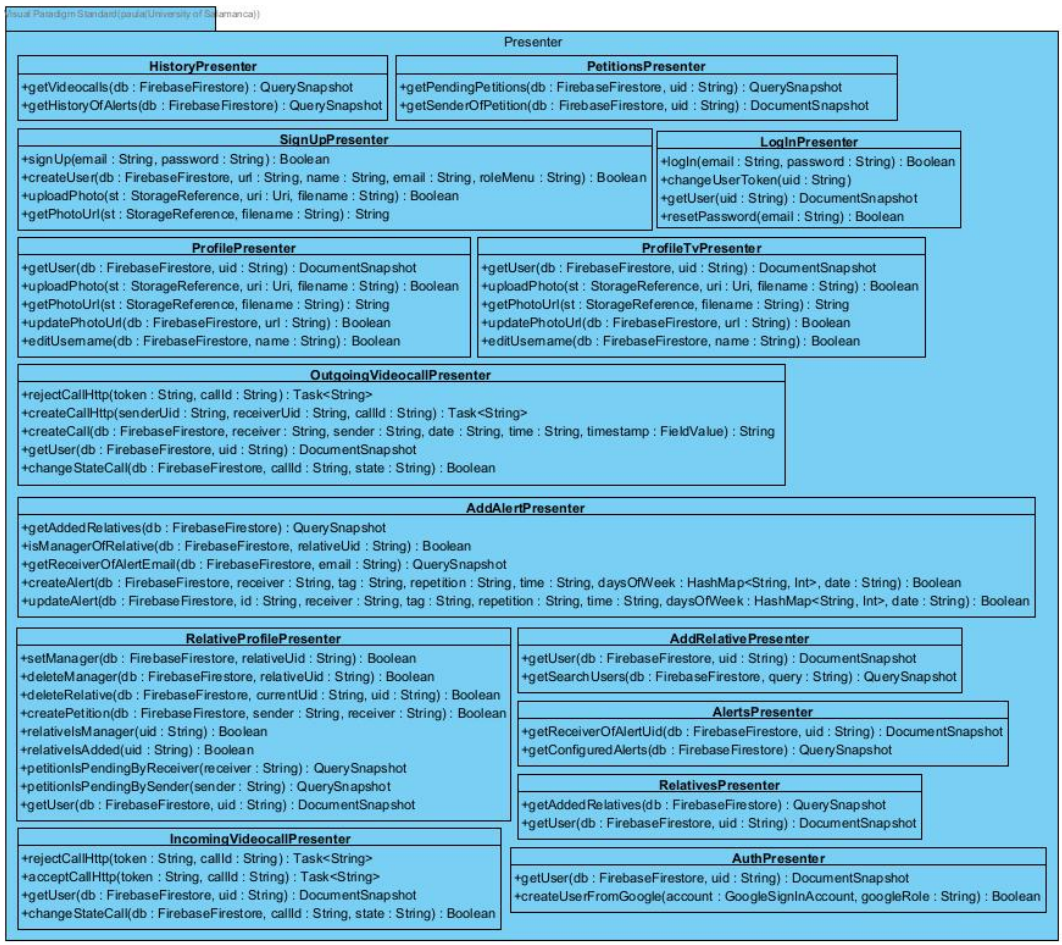


Figura 12: Subsistema Presenter

Modelo

En este subsistema tendremos todas las clases encargadas del acceso a datos, las reglas de negocio y los casos de uso.

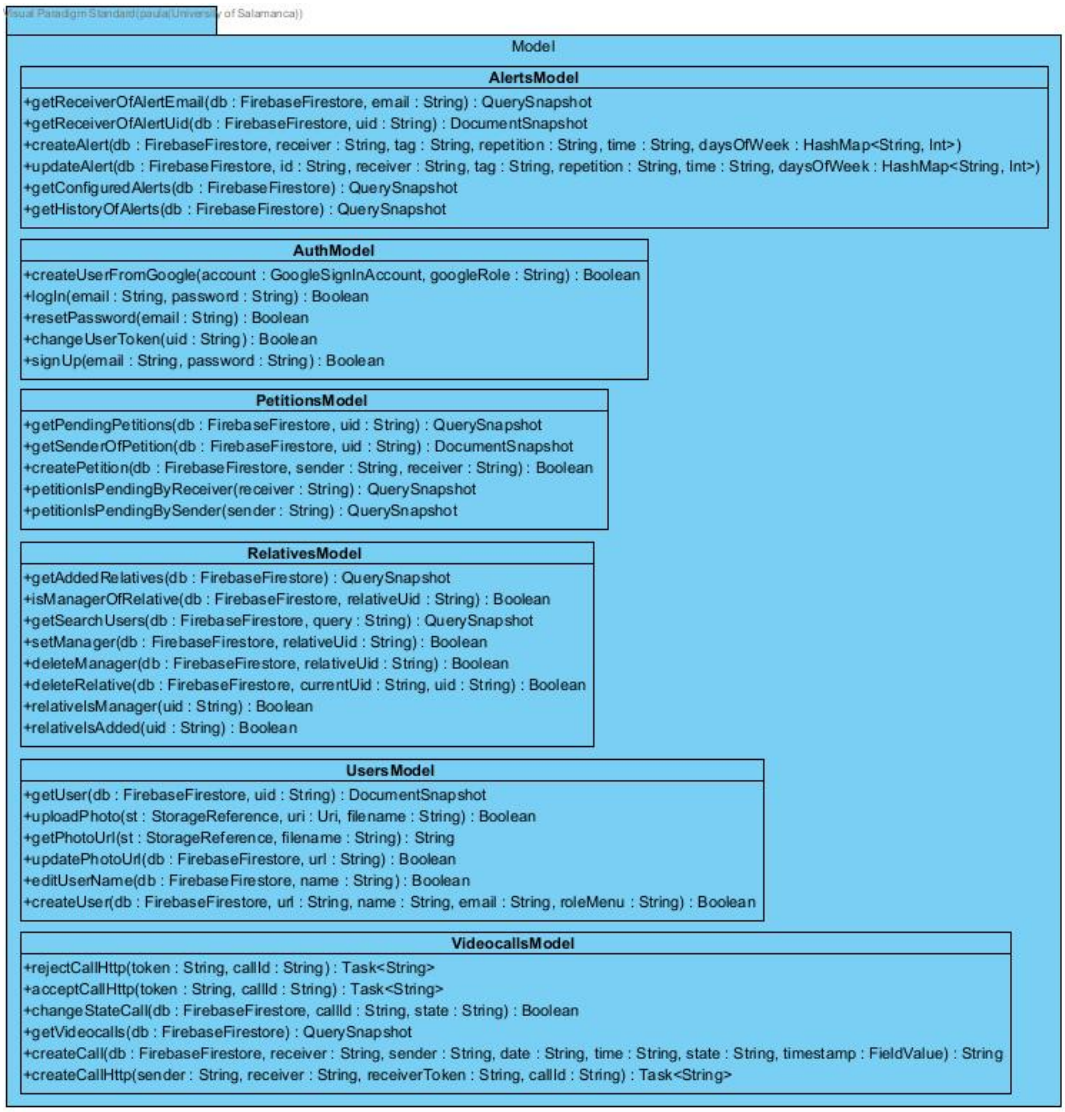


Figura 13: Subsistema Model

Contract

En este subsistema tendremos todas las clases que implementan las interfaces del resto de clases del patrón MVP.

Adapter

En este subsistema tendremos todas las clases que implementan la interfaz Adapter y que serán las encargadas de colocar un conjunto de datos sobre una vista, en este caso se trata de GridView.

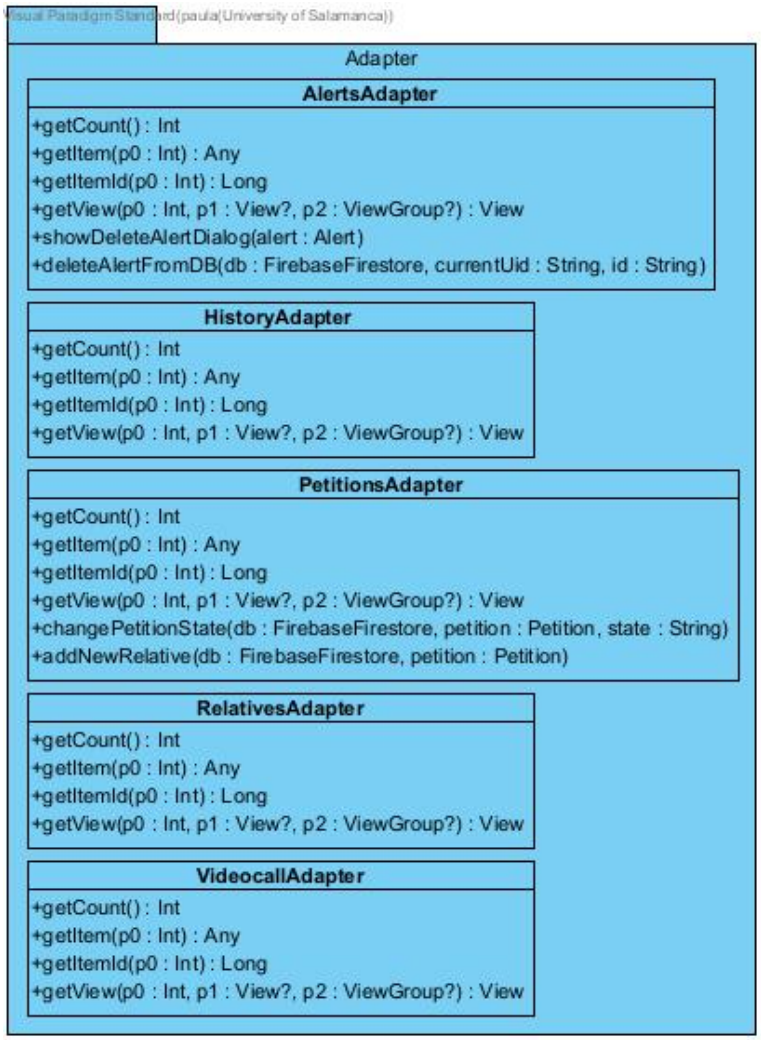


Figura 14: Subsistema Adapter

5.1.2. Servidor

Google Cloud Functions

Aquí se detallan las Google Cloud Functions implementadas que realizan tareas en el lado del servidor y que no se pueden realizar en el lado del cliente.

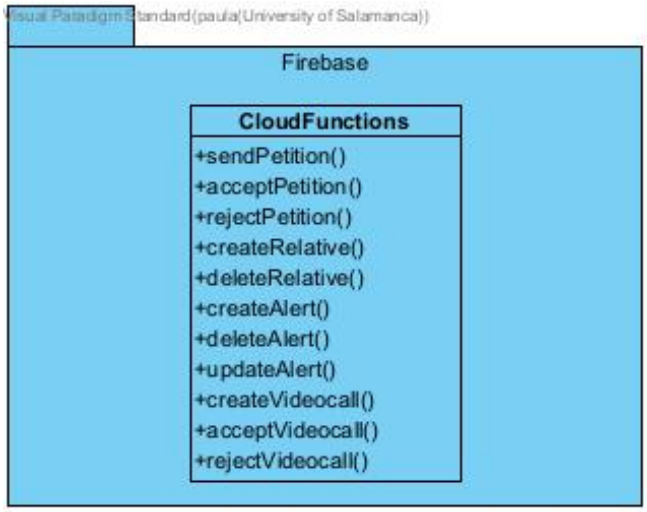


Figura 15: Cloud Functions

Node.js

Aquí se detallan los aspectos relacionados con las alertas, donde tenemos una aplicación en Node.js que se encarga de revisar mediante un cron las alertas establecidas en la base de datos, y cuando debe sonar una alerta, envía una notificación de Firebase Messaging al usuario e incluye una nueva entrada en el historial de alertas.

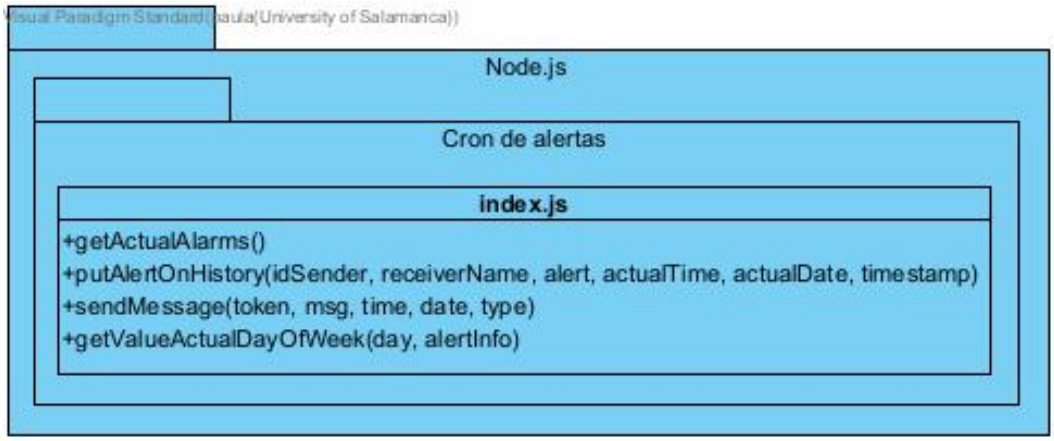


Figura 16: Node.js

Docker Jitsi Meet

Aquí se detallan los aspectos relacionados con las videollamadas.

La parte de videollamadas se encuentra alojada en un servidor VPS desplegado mediante Docker Jitsi Meet [8], que es un repositorio que contiene las herramientas necesarias para ejecutar una pila de Jitsi Meet en Docker usando Docker Compose.

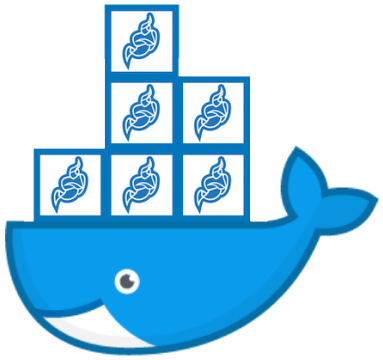


Figura 17: Logo Docker Jitsi Meet

Tenemos 4 contenedores montados en el servidor:

- **jitsi/jvb:** Jitsi Videobridge. El enrutador de video, encargado de recibir y hacer llegar los flujos de audio y vídeo a los participantes de una conferencia.
- **jitsi/jicofo:** Jitsi Conference Focus. El componente de enfoque XMPP, encargado de repartir las conferencias entre los videobridges disponibles en función de la carga, controlar los recursos…
- **jitsi/web:** Jitsi Meet Web UI. Interfaz gráfica de Jitsi Meet, basada en Nginx.
- **jitsi/prosody:** El servidor XMPP o servidor de mensajería instantánea. Se encarga de la autenticación de usuarios, conversaciones con varios participantes, envío de ficheros…

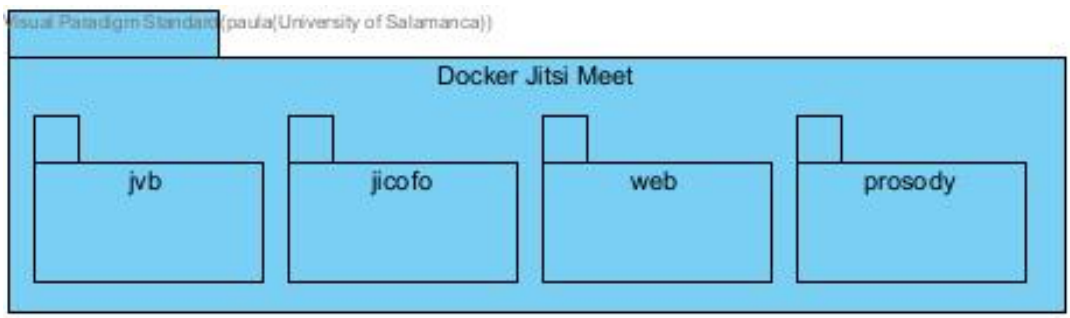


Figura 18: Docker Jitsi Meet

6. Vista arquitectónica

A partir de los subpaquetes descritos en el apartado anterior, podemos realizar la vista arquitectónica del sistema, que se puede ver en la Figura 19:

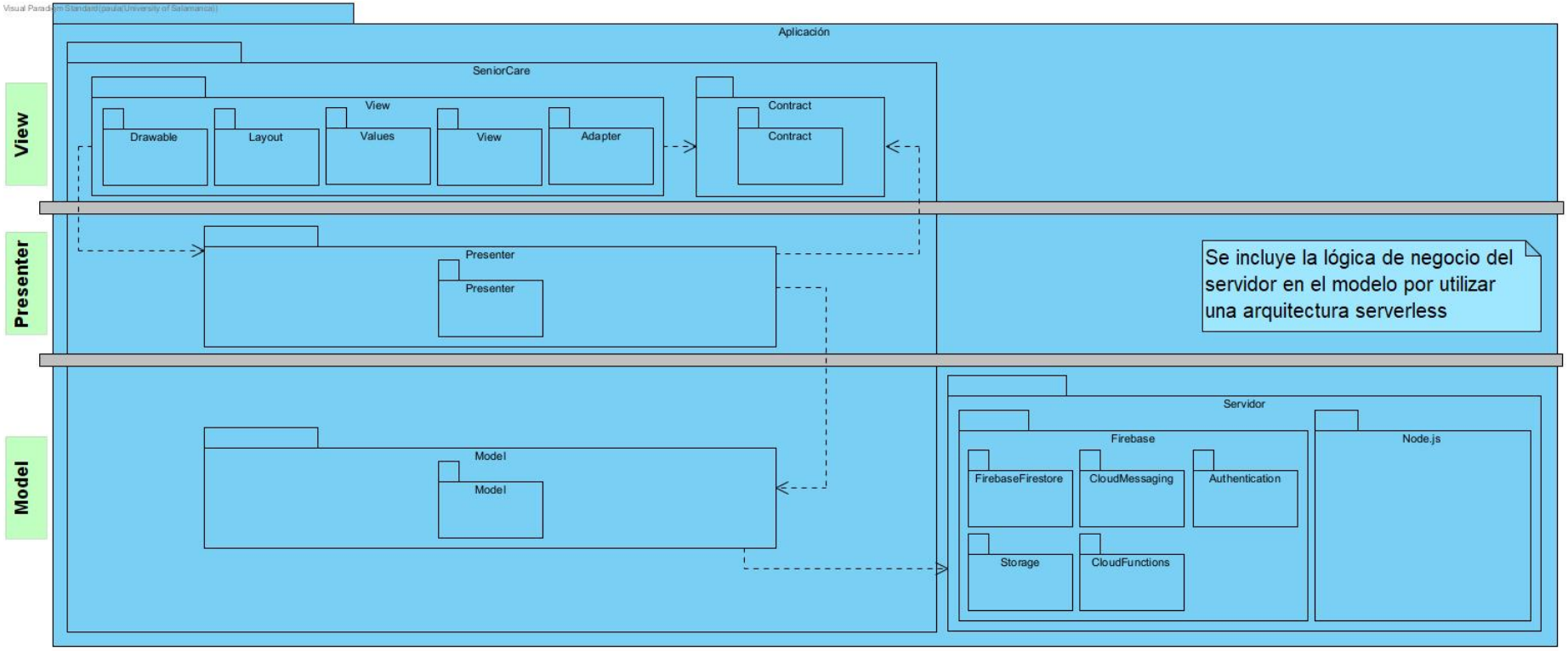


Figura 19: Vista arquitectónica

7. Realización de casos de uso - Diseño

En este apartado se detallarán mediante diagramas de secuencia, los mensajes que se intercambian los diferentes objetos.

7.1. Diagramas de secuencia del paquete Gestión de autenticación

UC-0001 Registrarse:

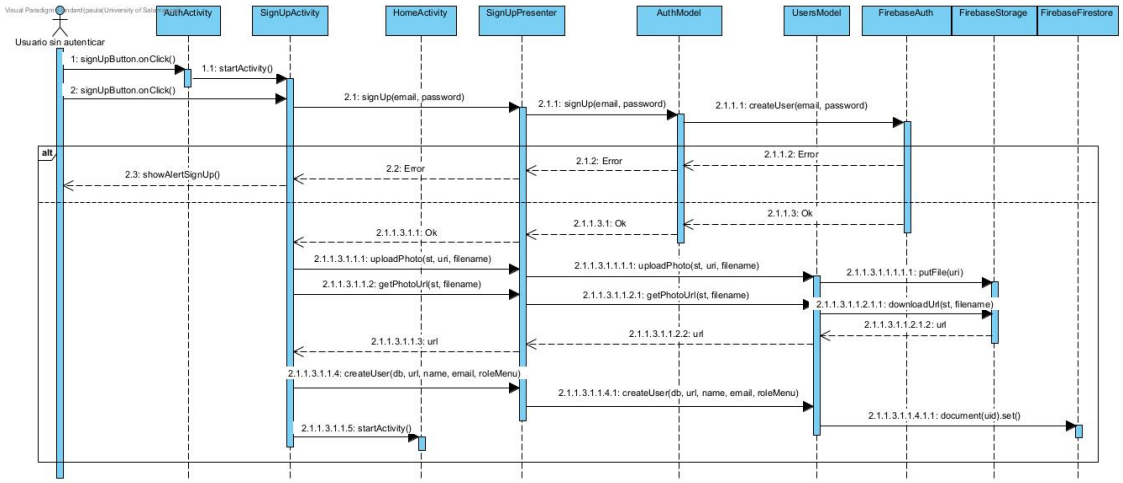


Figura 20: Diagrama de secuencia UC-0001 Registrarse

UC-0002 Iniciar sesión:

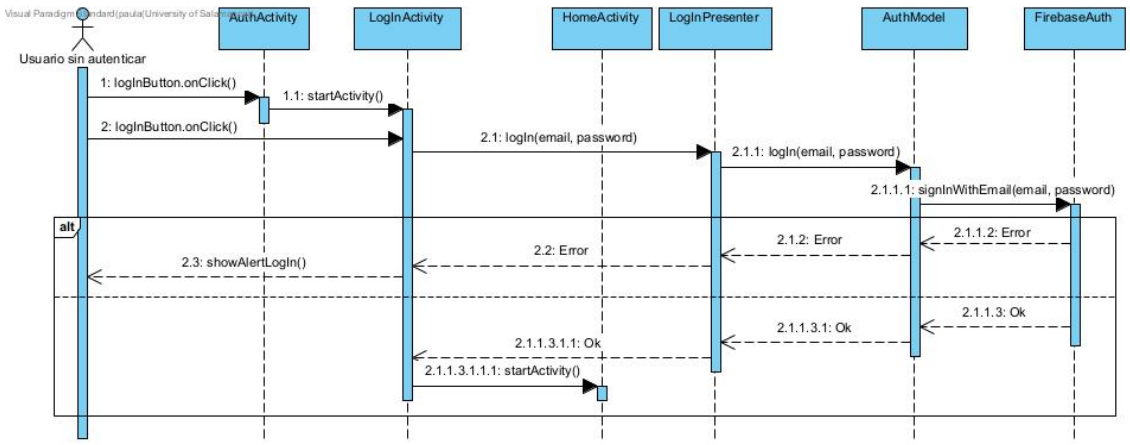


Figura 21: Diagrama de secuencia UC-0002 Iniciar sesión

UC-0003 Acceder con Google:

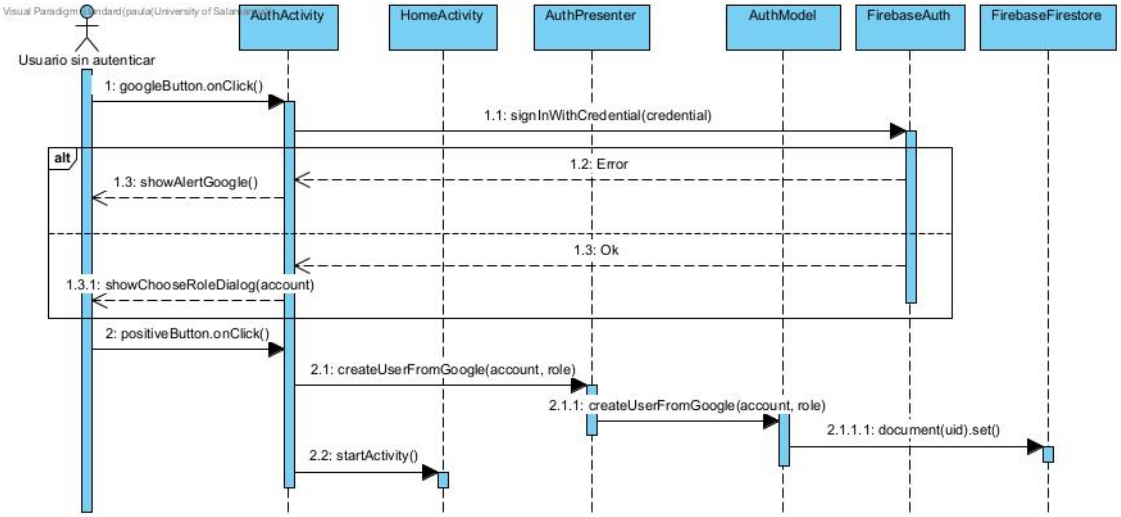


Figura 22: Diagrama de secuencia UC-0003 Acceder con Google

UC-0004 Recuperar contraseña:

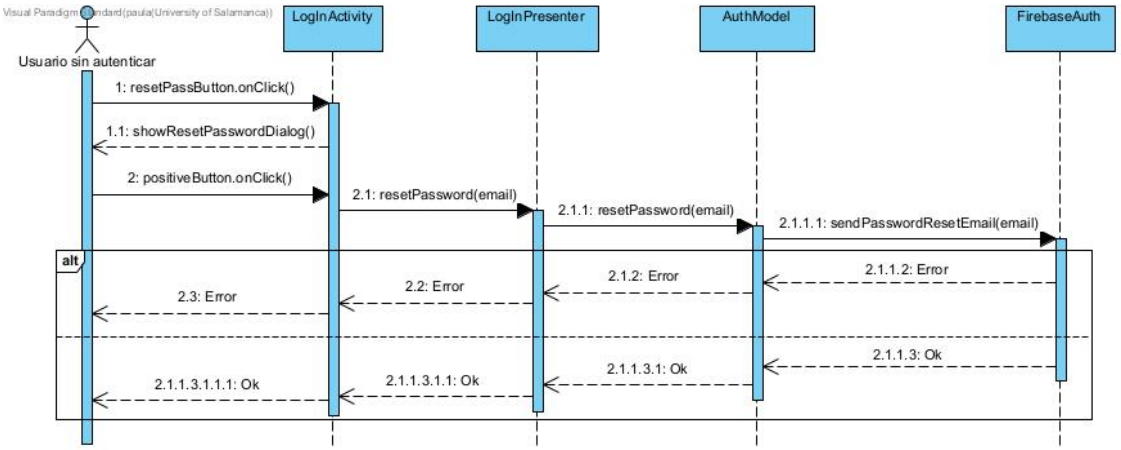


Figura 23: Diagrama de secuencia UC-0004 Recuperar contraseña

UC-0005 Cerrar sesión:

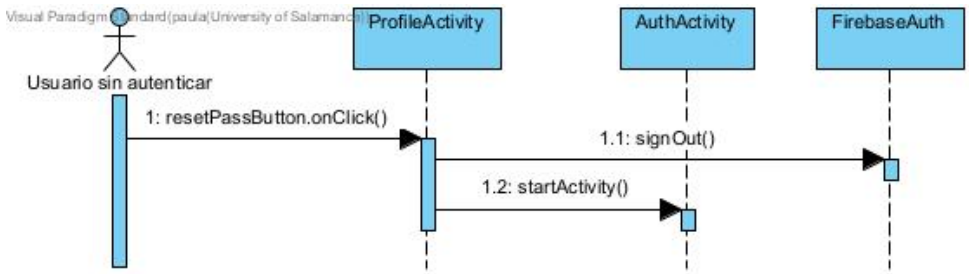


Figura 24: Diagrama de secuencia UC-0005 Cerrar sesión

7.2. Diagramas de secuencia del paquete Gestión de usuarios

UC-0006 Ver perfil:

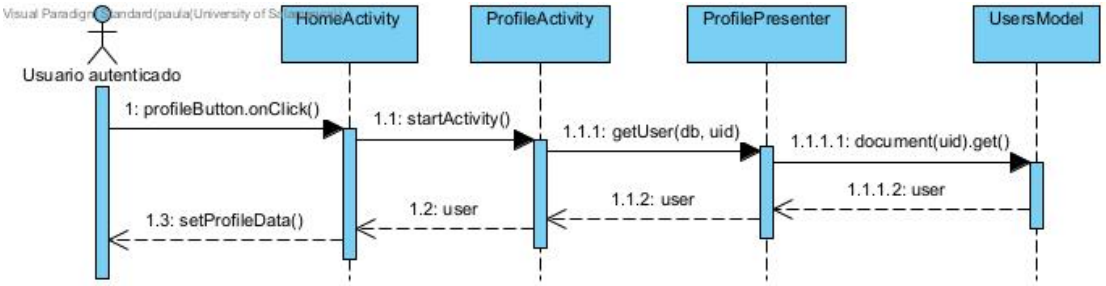


Figura 25: Diagrama de secuencia UC-0006 Ver perfil

UC-0007 Modificar imagen del perfil:

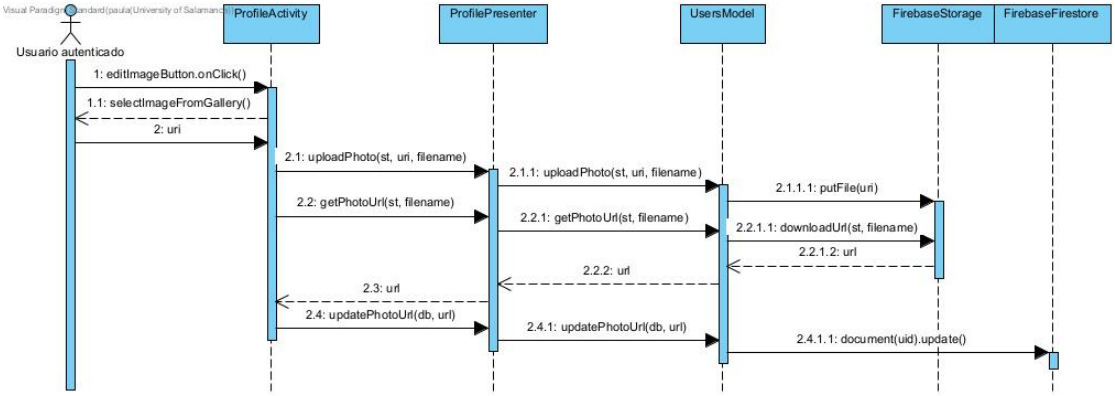


Figura 26: Diagrama de secuencia UC-0007 Modificar imagen del perfil

UC-0008 Modificar nombre del perfil:

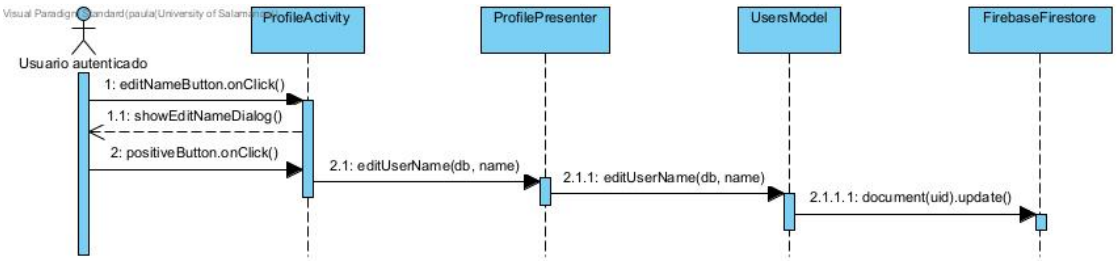


Figura 27: Diagrama de secuencia UC-0008 Modificar nombre del perfil

7.3. Diagramas de secuencia del paquete Gestión de llamadas

UC-0009 Realizar llamada / UC-0010 Recibir llamada:

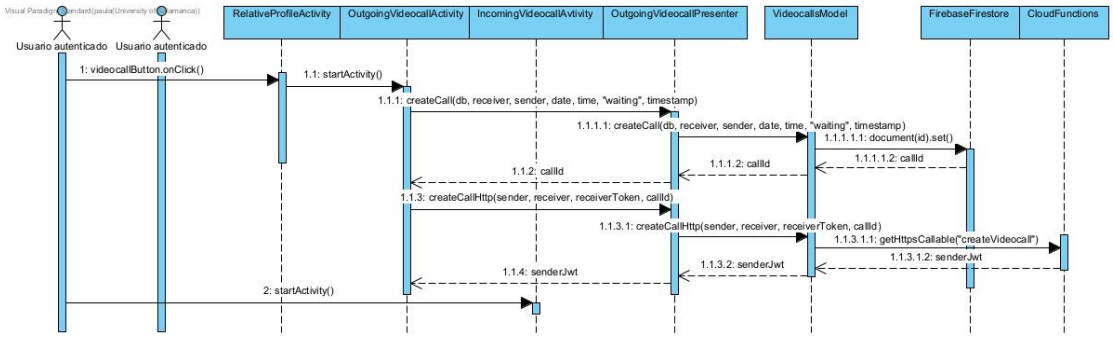


Figura 28: Diagrama de secuencia UC-0009 Realizar llamada / UC-0010 Recibir llamada

UC-0011 Ver historial de llamadas:

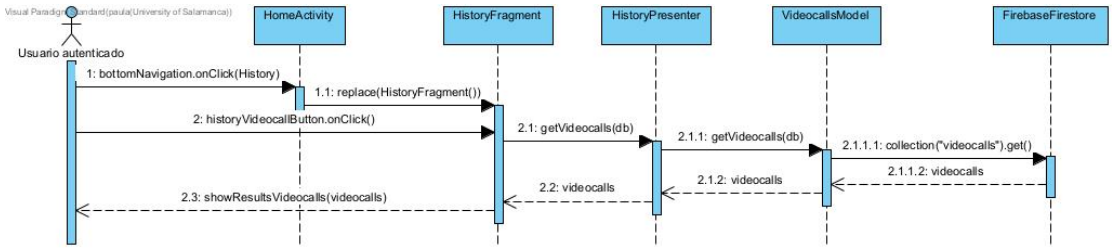


Figura 29: Diagrama de secuencia UC-0011 Ver historial de llamadas

7.4. Diagramas de secuencia del paquete Gestión de alertas

UC-0012 Ver alertas:

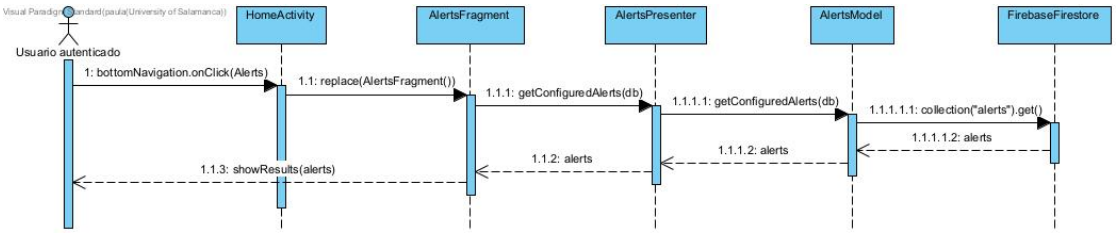


Figura 30: Diagrama de secuencia UC-0012 Ver alertas

UC-0013 Crear alerta / UC-0016 Recibir alerta:

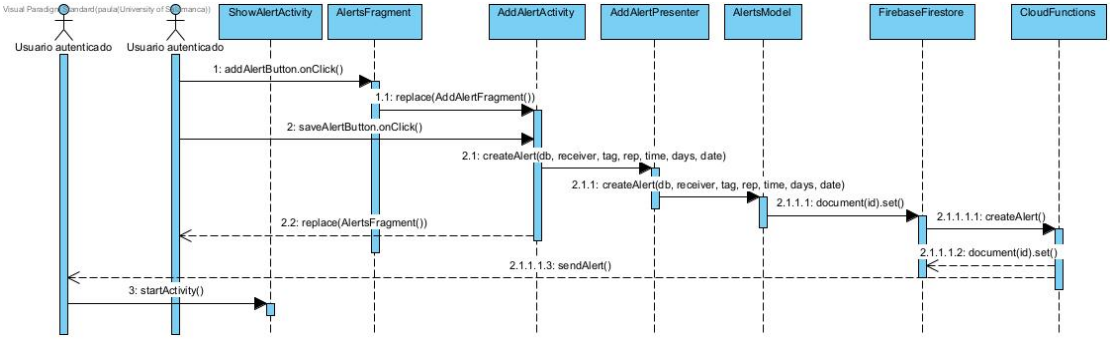


Figura 31: Diagrama de secuencia UC-0013 Crear alerta / UC-0016 Recibir alerta

UC-0014 Editar alerta:

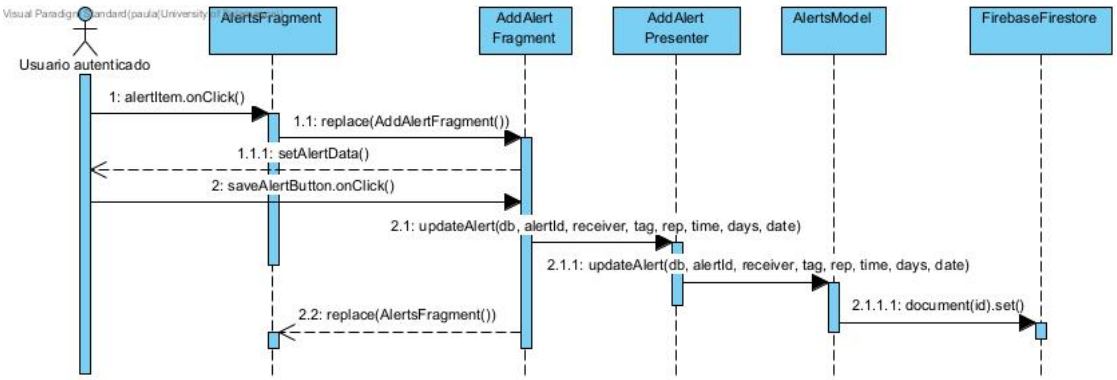


Figura 32: Diagrama de secuencia UC-0014 Editar alerta

UC-0015 Eliminar alerta:

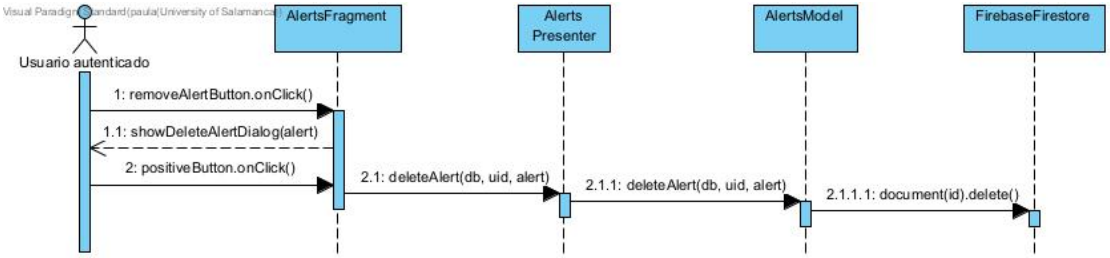


Figura 33: Diagrama de secuencia UC-0015 Eliminar alerta

UC-0017 Ver historial de alertas:

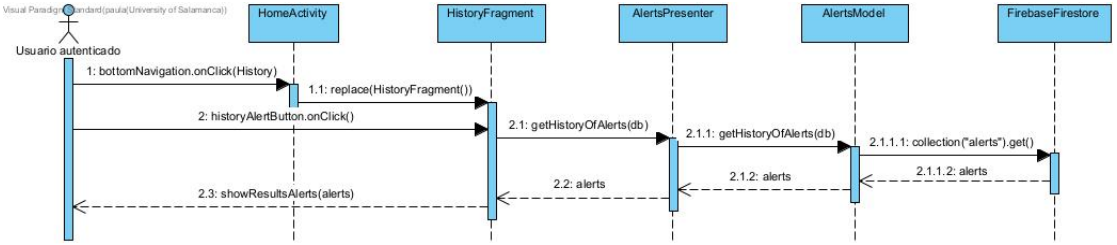


Figura 34: Diagrama de secuencia UC-0017 Ver historial de alertas

7.5. Diagramas de secuencia del paquete Gestión de familiares

UC-0018 Ver familiares:

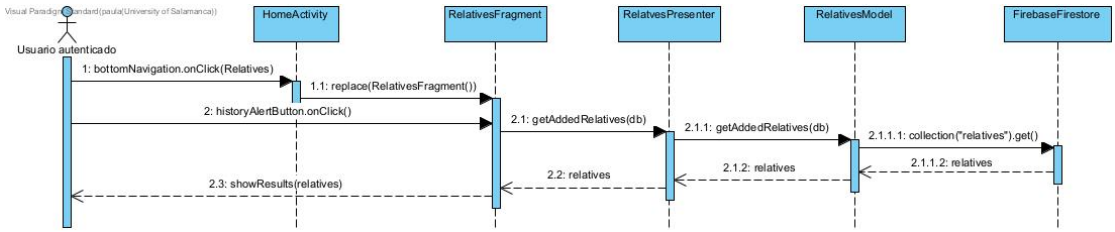


Figura 35: Diagrama de secuencia UC-0018 Ver familiares

UC-0019 Ver perfil de familiar:

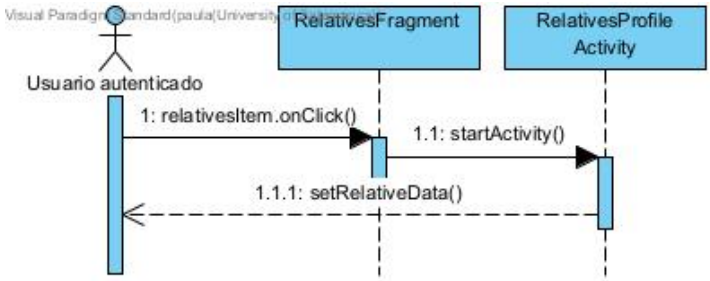


Figura 36: Diagrama de secuencia UC-0019 Ver perfil de familiar

UC-0020 Eliminar familiar:

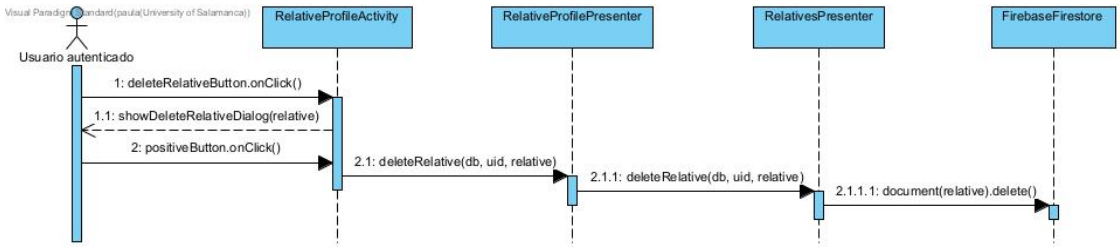


Figura 37: Diagrama de secuencia UC-0020 Eliminar familiar

7.6. Diagramas de secuencia del paquete Gestión de solicitudes

UC-0021 Buscar usuarios:

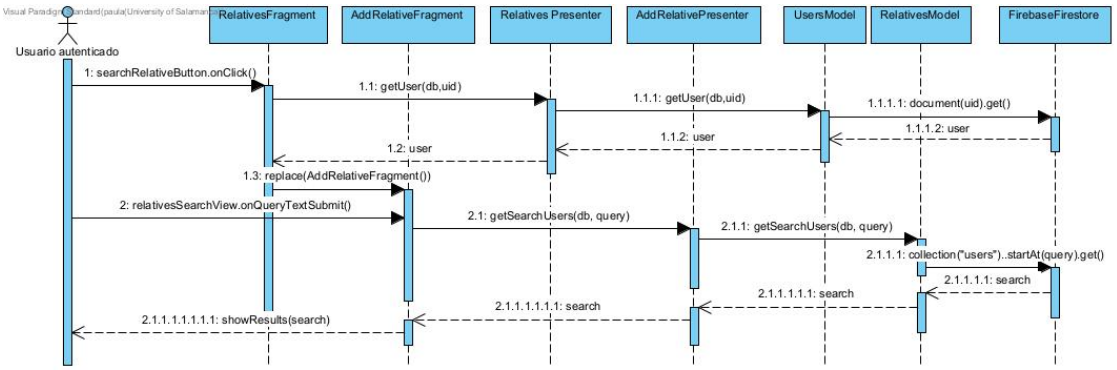


Figura 38: Diagrama de secuencia UC-0021 Buscar usuarios

UC-0022 Enviar solicitud / UC-0023 Recibir solicitud:

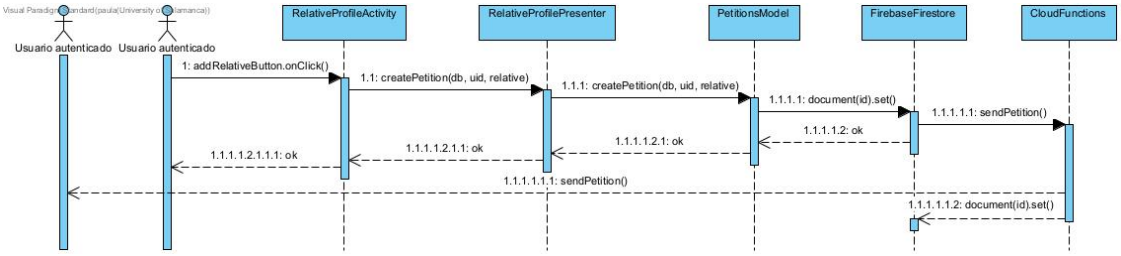


Figura 39: Diagrama de secuencia UC-0022 Enviar solicitud / UC-0023 Recibir solicitud

UC-0024 Ver solicitudes:

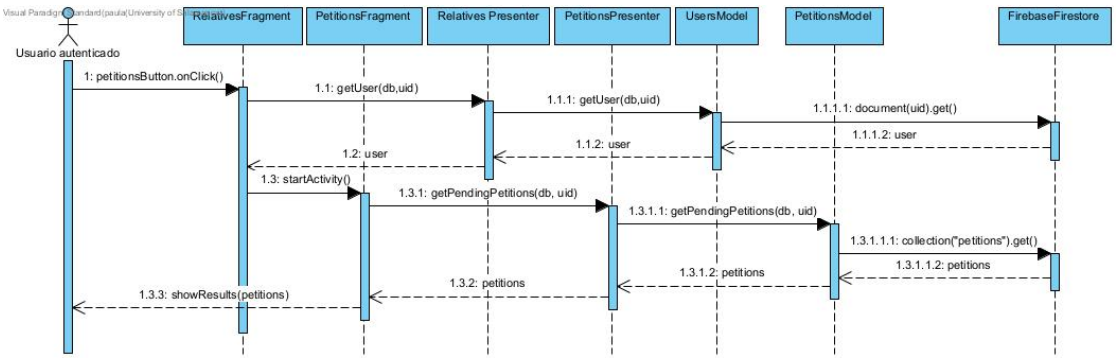


Figura 40: Diagrama de secuencia UC-0024 Ver solicitudes

8. Diseño de la base de datos

Para que el sistema pueda almacenar y gestionar la información, se utilizará una base de datos NoSQL proporcionada por el servicio Firebase Firestore [9], donde se utilizarán colecciones y documentos. Este tipo de bases de datos no sigue el esquema tradicional de las bases de datos SQL. Se seguirá el modelo del dominio realizado en el Anexo III para satisfacer los requisitos del dominio de la solución.

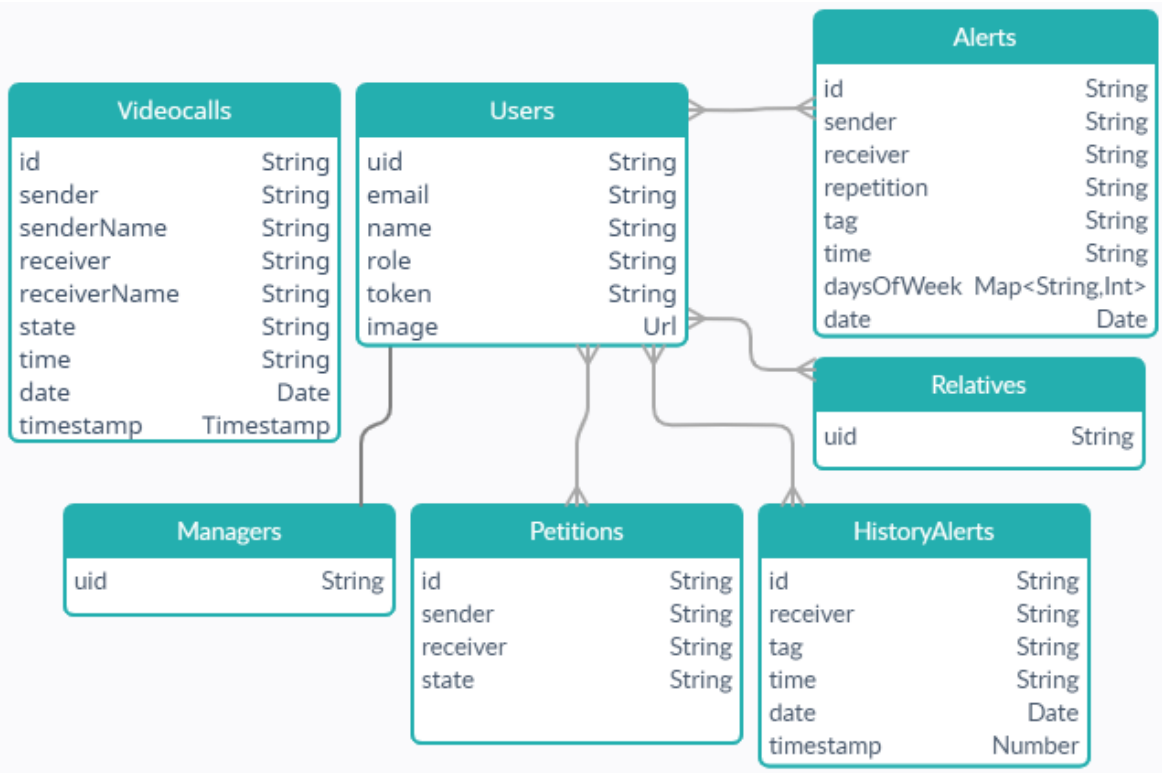


Figura 41: Diseño de la base de datos

A continuación, se va a ver un ejemplo de un documento de cada colección que forma la base de datos. Primero se muestra un documento de la colección de *Users*, en el que se ven los campos descritos anteriormente.

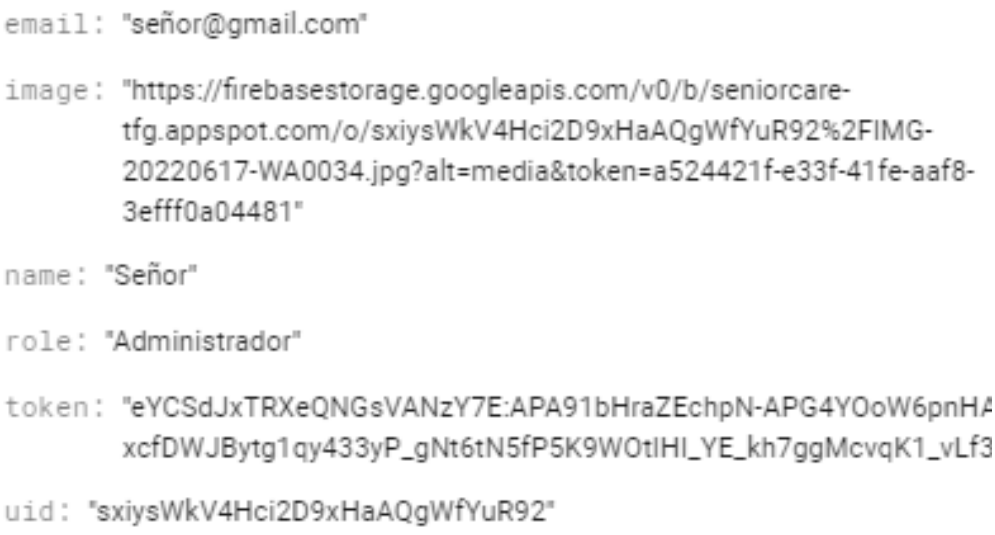


Figura 42: Documento de la colección Users

Dentro de cada documento de esta colección hay una serie de subcolecciones entre las que se encuentran:

- Documento de la subcolección *Alerts*:

Figura 43: Documento de la colección Alerts

- Documento de la subcolección *Relatives*:

Figura 44: Documento de la colección Relatives

- Documento de la subcolección *HistoryAlerts*:

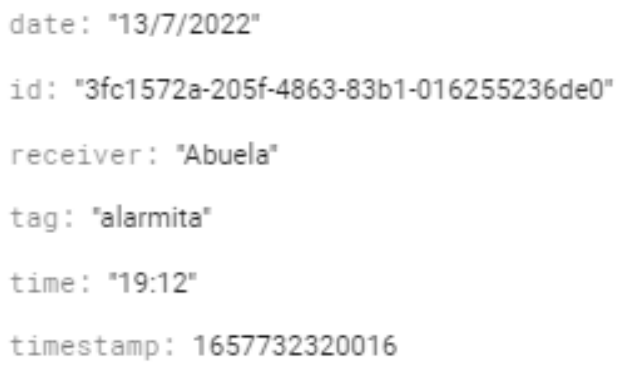


Figura 45: Documento de la colección HistoryAlerts

- Documento de la subcolección *Petitions*:

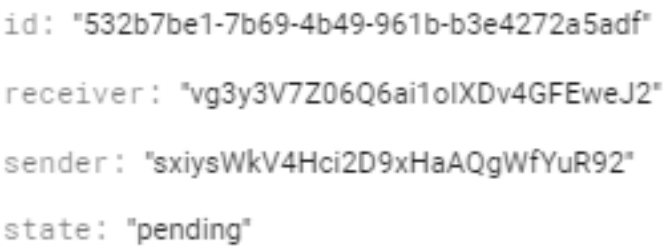


Figura 46: Documento de la colección Petitions

- Documento de la subcolección *Managers*:

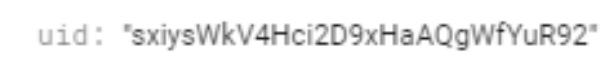


Figura 47: Documento de la colección Managers

Después se muestra un ejemplo de un documento que pertenece a otra colección diferente llamada *Videocalls*, que tiene la siguiente estructura:

Figura 48: Documento de la colección Videocalls

9. Modelo de despliegue

Para representar el despliegue de los componentes del sistema, se ha elaborado un diagrama de despliegue que muestra la arquitectura en ejecución y así poder ver la representación física de los componentes en los nodos físicos.

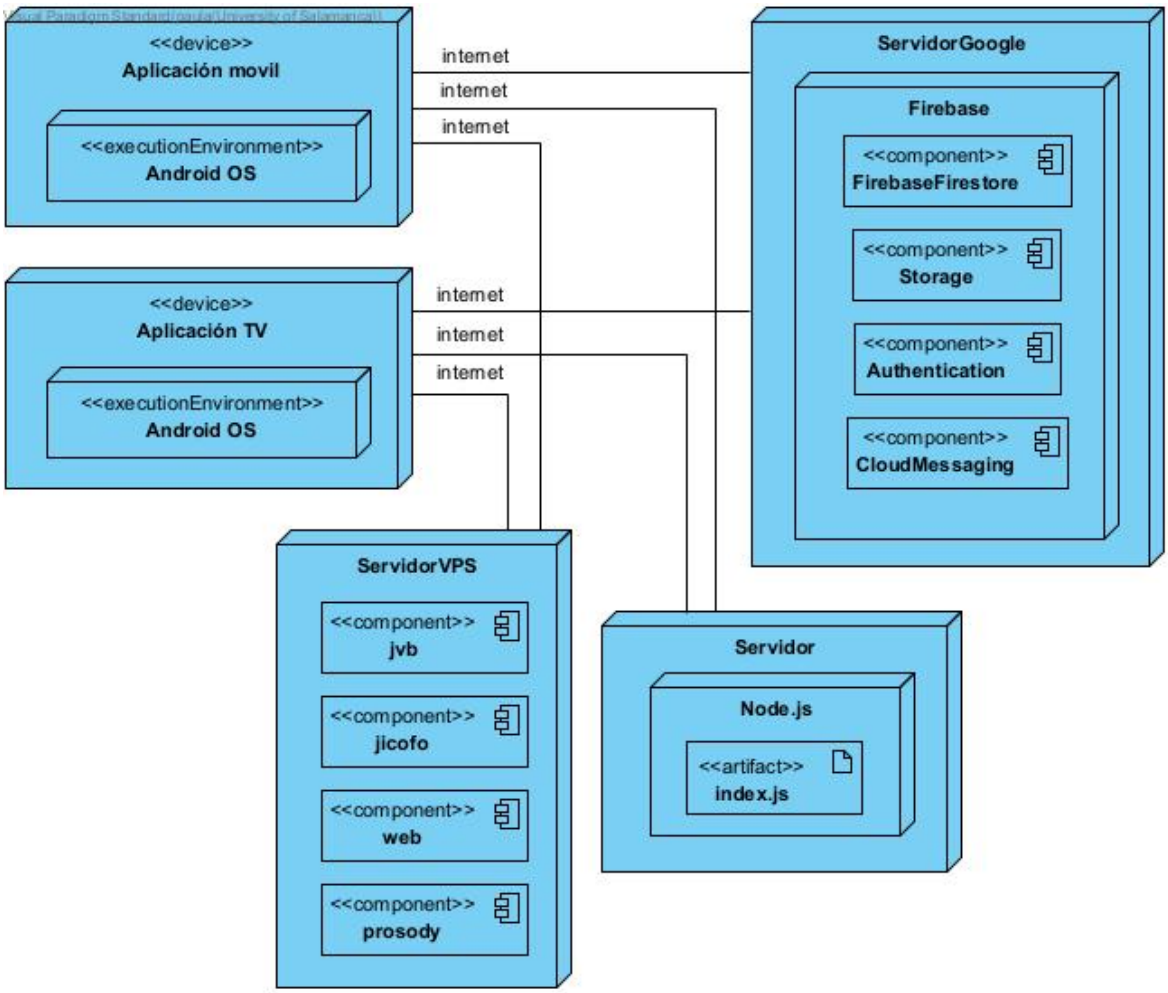


Figura 49: Modelo de despliegue

10. Referencias

[1] Roger S. Pressman, *Ingeniería del Software: Un Enfoque Práctico*.

[2] María Moreno García y Francisco José García Peñalvo, «Ingeniería de Software I - UML (Parte 3 de 3)».

[3] «Visual Paradigm». https://www.visual-paradigm.com/ (accedido jun. 21, 2022).

[4] «Adobe XD». Accedido: sep. 01, 2022. [En línea]. Available: https://www.adobe.com/es/products/xd.html

[5] «Firebase». https://firebase.google.com/ (accedido ago. 08, 2022).

[6] Óscar Blancarte, «Patrón MVC». https://www.oscarblancarteblog.com/2014/07/21/patron-de-diseno-modelo-vista-controlador-mvc/ (accedido ago. 25, 2022).

[7] Joaquín Medina Serrano, «Patrón MVP». http://joaquin.medina.name/web2008/documentos/informatica/documentacion/logica/patrones/patronMVP/2012_06_09_PatronMVP.html (accedido ago. 25, 2022).

[8] «Jitsi Meet Docker». https://jitsi.github.io/handbook/docs/devops-guide/devops-guide-docker/ (accedido ago. 25, 2022).

[9] «Firebase Firestore». https://firebase.google.com/products/firestore (accedido ago. 08, 2022).

Anexo V: Documentación técnica

Aplicación para facilitar las tareas de personas mayores

Trabajo de Fin de Grado de Ingeniería Informática

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Septiembre de 2022

Autora:

Paula Soria Ullán

Tutores:

Gabriel Villarrubia González

Álvaro Lozano Murciego

André Filipe Sales Mendes

Índice

Índice de figuras

1. Introducción

Este anexo tiene como objetivo facilitar la comprensión del código de la aplicación a programadores o a las personas interesadas en conocer el software empleado, además de facilitar la comprensión y modificación por parte de terceros.

El proyecto se divide en 3 partes, una de ellas forma parte del *frontend* y las otras dos forman parte del *backend*, siendo estas:

- Aplicación Android.
- Google Cloud Functions.
- Servidor.

En el anexo se verán las siguientes secciones:

- Estructura del sistema.
- Especificación de la documentación.

2. Estructura del sistema

Para controlar las versiones de código y backups, de forma que se puedan administrar cambios en el proyecto a la vez que evoluciona, se ha empleado GitHub [1], que es un repositorio online gratuito. Está basado en Git, que es un sistema de control de versiones distribuido de código abierto.

En los siguientes apartados se verá la estructura que tiene cada parte del sistema y que se puede ver en el repositorio de GitHub del enlace: https://github.com/paulasoria/TFG

2.1. Frontend

2.1.1. Aplicación Android

La aplicación Android es la parte principal del sistema, formada por la parte de la aplicación móvil y la parte de la aplicación de televisión. En esta se realizan la mayoría de las interacciones entre el usuario y el sistema. Se han empleado XML y Kotlin a través de Android Studio, que es un entorno de desarrollo integrado para crear aplicaciones Android.

El código sigue la siguiente estructura:

- **SeniorCare_App/app/src/main/java/com/paula/seniorcare_app**: Directorio que contiene todas las clases que conforman las tareas de la aplicación Android en sí. Contiene las siguientes carpetas Adapter, Contract, Dataclass, Model, Presenter, Service y View, que contienen las clases correspondientes a su nombre.
- **SeniorCare_App/app/src/main/res:** Directorio que contiene todos los elementos gráficos de la aplicación Android. Algunas de las carpetas que contiene son:
  - **Drawable:** Directorio que contiene ficheros XML que corresponden a objetos gráficos.
  - **Font:** Directorio que contiene las fuentes con extensión ".ttf" utilizadas en los textos de la aplicación.
  - **Layout:** Directorio que contiene todas las clases XML que corresponden a la representación de las vistas en orden y estructura de los elementos.
  - **Menu:** Directorio que contiene ficheros XML que representan menús de opciones.

- **Mipmap:** Directorio que contiene imágenes en distintas calidades representadas en la aplicación.
- **Values:** Directorio que contiene los ficheros que representan los colores, las cadenas de texto y el tema utilizado en la aplicación.

- **SeniorCare_App/app/src/main/AndroidManifest.xml:** Fichero que describe información esencial sobre la aplicación para las herramientas de creación de Android, el sistema operativo Android y Google Play. También contiene referencias a todas las actividades de la aplicación.

2.2. Backend

2.2.1. Cron de alertas

El cron de alertas se encarga de monitorizar las alertas establecidas por los usuarios de forma periódica de forma que avise a los usuarios receptores de una alarma o recordatorio en el momento indicado.

Para realizar el despliegue del cron de alertas, bastaría con abrir una terminal del ordenador. Navegar hasta la carpeta en la que se encuentren nuestras funciones utilizando el comando “cd *ruta_del_directorio*”. Una vez ahí, hay que ejecutar la aplicación de Node.js y esto se hace con el comando “node index.js”.

Sigue la siguiente estructura:

- **SeniorCare_App/server/node_modules:** Fichero que contiene los módulos utilizados por Node.js para ejecutar el cron.
- **SeniorCare_App/server/index.js:** Fichero que contiene el código del cron de alertas.
- **SeniorCare_App/server/package.json:** Fichero de especificación con las dependencias de Node.js.

2.2.2. Google Cloud Functions

Las Google Cloud Functions no están relacionadas directamente con la aplicación de Android, pero se encuentran dentro de la carpeta del proyecto, aunque estén subidas y desplegadas en Firebase.

Para realizar el despliegue de las funciones de Google Cloud, bastaría con abrir una terminal del ordenador. Navegar hasta la carpeta en la que se encuentren nuestras funciones utilizando el comando "cd *ruta_del_directorio*". Una vez ahí, hay que desplegar las funciones y esto se hace con el comando "firebase deploy --only functions" y tras unos segundos o minutos, en función de las dimensiones del fichero, se habrán desplegado en Google Cloud Functions.

Esta parte del sistema sigue la siguiente estructura:

- **SeniorCare_App/functions/node_modules:** Fichero que contiene las bibliotecas utilizadas por las Firebase Cloud Functions.
- **SeniorCare_App/functions/index.js:** Fichero que contiene el código de las Firebase Cloud Functions.
- **SeniorCare_App/functions/package.json:** Fichero de especificación con las dependencias de las Firebase Cloud Functions.

2.2.3. Jitsi Meet

Jitsi Meet es una aplicación que permite realizar de videoconferencias.

Para realizar el despliegue de Jitsi Meet, primero habría que instalar Docker mediante el comando "apt-get install docker.io docker-compose". Tras la instalación, hay que descargar el repositorio de Jitsi Meet Docker [2] de GitHub mediante el comando "git clone https://github.com/jitsi/docker-jitsi-meet". Después, hay que entrar en el directorio, copiar el fichero de configuración de ejemplo y pegarlo en el mismo directorio con la extensión .env utilizando el comando "cp env.example .env". Ahora hay que editar el fichero, cambiando el valor de "HTTP_PORT" a 80 y "HTTPS_PORT" a 443, porque debe poder conectarse a Internet. También hay que modificar el valor de "PUBLIC_URL" a "https://jitsi.paulasoria.tk", que es el DNS de mi dominio. Finalmente, para cargar la configuración, y descargar e iniciar los contenedores, se utilizará el comando "docker-compose up -d".

3. Especificación de la documentación

Para generar la documentación técnica se han utilizado herramientas que generan código HTML donde se explica con mayor detalle cada fichero.

3.1. Frontend

3.1.1. Aplicación Android

La generación de documentación para la parte de la aplicación Android se ha realizado utilizando la herramienta Dokka [3], que es un motor de documentación para Kotlin, que realiza la misma función que javadoc para Java. Al igual que el propio Kotlin, Dokka es totalmente compatible con proyectos Java/Kotlin de lenguaje mixto. Esta documentación se encuentra en el directorio “SeniorCare_App\app\build\dokka\html” en el archivo “index.html”.

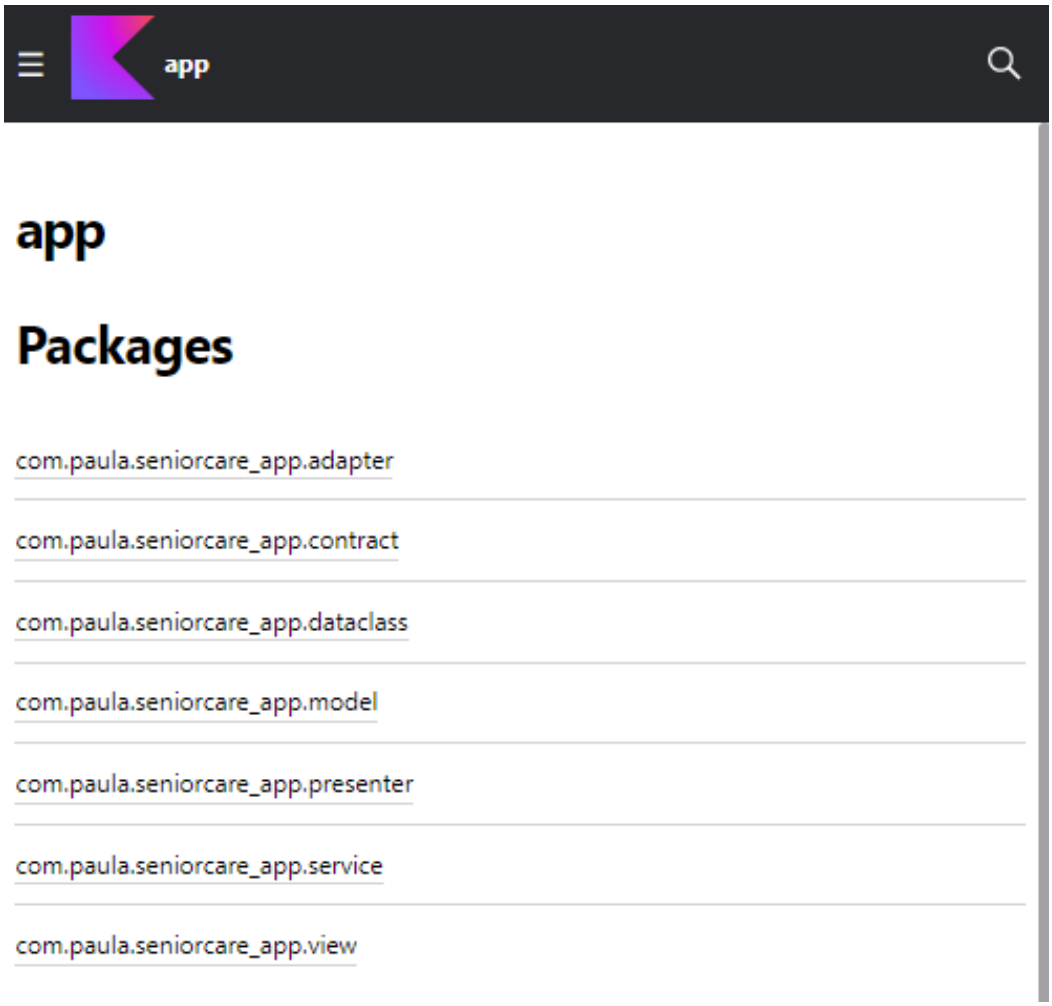


Figura 1: Documentación con Dokka

3.2. Backend

3.2.1. Google Cloud Functions

La generación de documentación para la parte de las Google Cloud Functions se ha realizado utilizando la herramienta JSDoc [4]. Esta documentación se encuentra en el directorio “SeniorCare_App\server\out” en el archivo “Index.html”.

3.2.2. Cron de alertas

Se ha empleado la misma herramienta que en el caso anterior. Esta documentación se encuentra en el directorio “SeniorCare_App\functions\out” en el archivo “Index.html”.

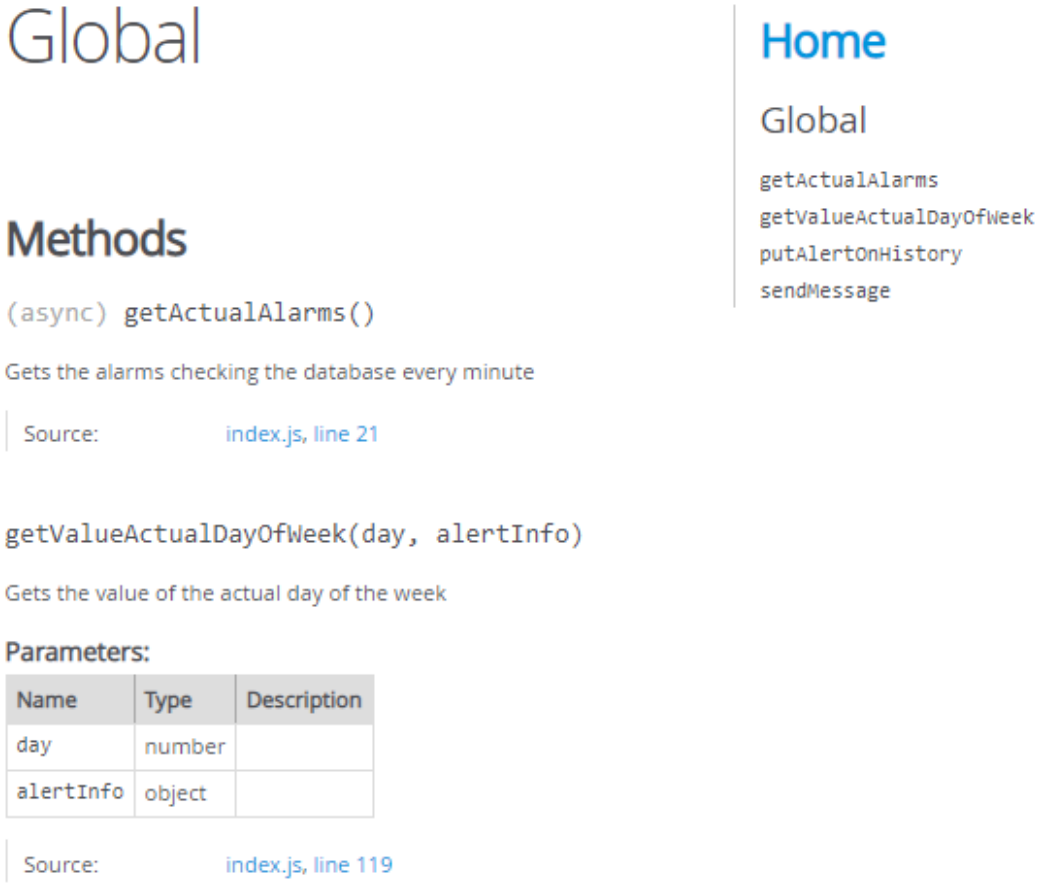


Figura 2: Documentación con JSDoc

4. Referencias

[1] «GitHub». https://github.com/ (accedido ago. 08, 2022).

[2] «Jitsi Meet Docker». https://jitsi.github.io/handbook/docs/devops-guide/devops-guide-docker/ (accedido ago. 25, 2022).

[3] «Dokka». https://github.com/Kotlin/dokka (accedido ago. 05, 2022).

[4] «JSDoc». https://jsdoc.app/index.html (accedido ago. 05, 2022).

Anexo VI: Manual de usuario

Aplicación para facilitar las tareas de personas mayores

Trabajo de Fin de Grado de Ingeniería Informática

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Septiembre de 2022

Autora:

Paula Soria Ullán

Tutores:

Gabriel Villarrubia González

Álvaro Lozano Murciego

André Filipe Sales Mendes

Índice

Índice de figuras

1. Introducción

Este anexo tiene como objetivo explicar la funcionalidad e interacción con el sistema de forma que la aplicación sea comprensible por el usuario.

La aplicación se puede ejecutar en dispositivos móviles como cualquier aplicación de Android, pero también se ha diseñado una parte especialmente para ser empleada en Android TV de forma que los familiares o personas mayores tengan una forma sencilla e intuitiva de acceder a la aplicación. Aunque todos los usuarios pueden acceder a los dos caminos de la aplicación, la parte móvil está diseñada especialmente para los usuarios normales mientras que la parte de televisión está diseñada especialmente para los familiares.

Para realizar la instalación de la aplicación, será necesario que el usuario disponga de un dispositivo móvil con el sistema operativo de Android o una televisión que disponga de Android TV. A partir de aquí bastará con tener el SDK de la aplicación y la instalación se realizará de forma automática.

Se puede consultar un vídeo del funcionamiento de la aplicación en el siguiente enlace de Google Drive:
https://drive.google.com/file/d/1T6qbutjrM_4z3Z9Zcs0uW_X2WMO0Lc6p/view?usp=sharing

El contenido del anexo está dividido en dos apartados:

- Aplicación móvil
- Aplicación TV

2. Manual Aplicación

En este apartado se explicará la forma de acceder y utilizar la aplicación móvil para realizar diferentes tareas.

2.1. Instalación

La aplicación no se encuentra subida a ninguna plataforma, pero se va a explicar cómo se procedería a su descarga en el caso de que estuviera.

Para poder utilizar esta aplicación desde el dispositivo móvil, habría que acceder a la Play Store y descargar la aplicación “SeniorCare”. A continuación, se puede acceder a ella pulsando sobre el icono.

Figura 1: Icono SeniorCare

Para poder utilizar esta aplicación desde la televisión, habría que ir al apartado de búsqueda de aplicaciones, descargar la aplicación “SeniorCare” y desplazarse hasta nuestras apps. Nuevamente, se puede acceder a ella pulsando sobre el icono.

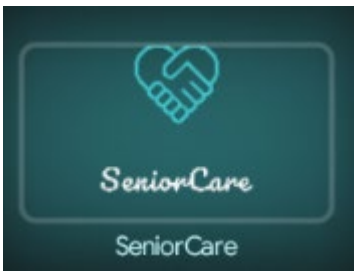


Figura 2: Icono SeniorCare TV

2.2. Aplicación móvil

Al iniciar la aplicación móvil se encuentra esta primera pantalla, donde se puede acceder de diferentes formas a la aplicación, como se muestra en la Figura 3.

La primera forma es iniciando sesión en el caso de que ya se tenga una cuenta, la segunda forma es mediante registro de forma manual y la tercera forma es accediendo mediante una cuenta de Google vinculada al dispositivo.

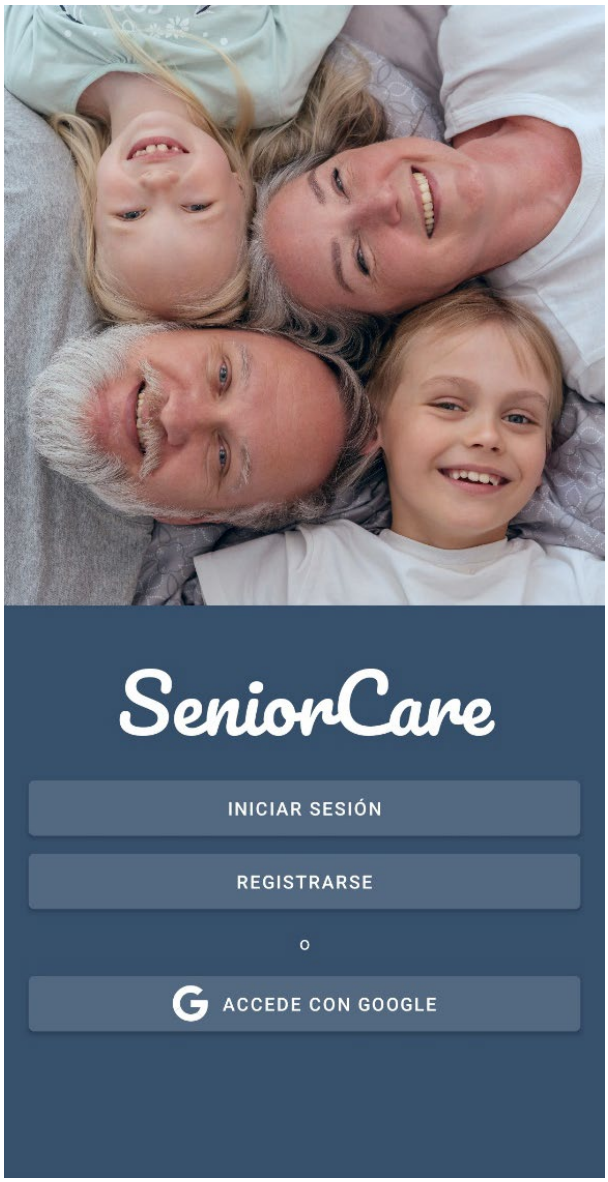


Figura 3: Pantalla inicial

Al hacer clic sobre el botón de iniciar sesión de la ventana anterior, se navega a la siguiente pantalla, como se puede ver en la Figura 4. Aquí es posible introducir el correo electrónico y contraseña para iniciar sesión y también es posible hacer clic sobre un botón de recuperación de contraseña en el caso de que no recordar la contraseña.

Figura 4: Inicio de sesión

Al hacer clic sobre el botón de recuperación de contraseña, se navega hasta la siguiente pantalla, como se puede ver en la Figura 5, donde se abre un panel flotante que solicita el email de recuperación. Posteriormente, se le muestra al usuario un mensaje de confirmación en el caso de que se haya podido enviar correctamente el correo de restablecimiento de contraseña.

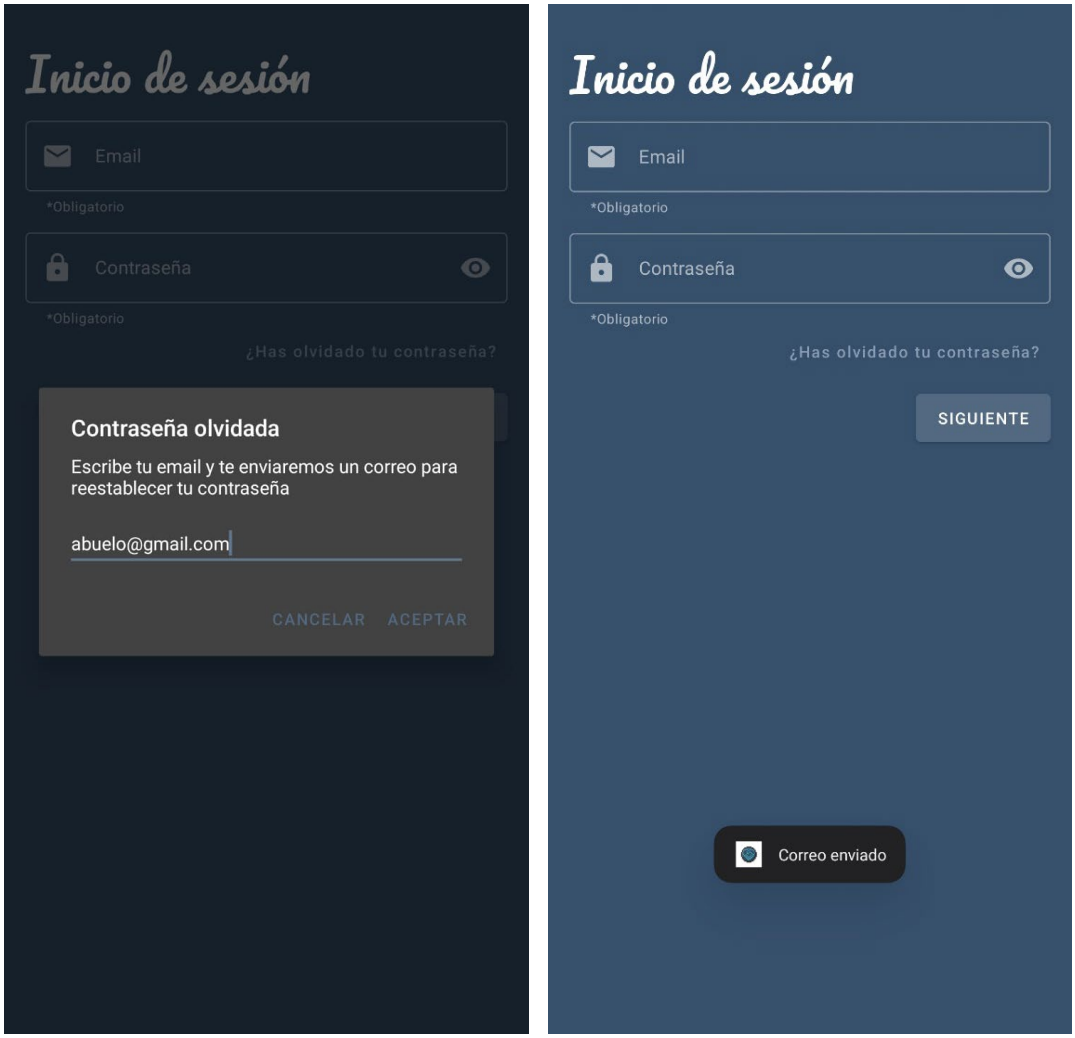


Figura 5: Contraseña olvidada

La segunda forma de acceder es mediante el segundo botón de la ventana inicial mediante registro, como se puede ver en la Figura 6. Aquí se solicitan datos personales, como la imagen de perfil (opcional), el nombre, el email, la contraseña y el rol de usuario, que será “Administrador” en el caso de que se administre a un pariente o “Familiar” si es el administrado y habrá menos interfaces para ver.

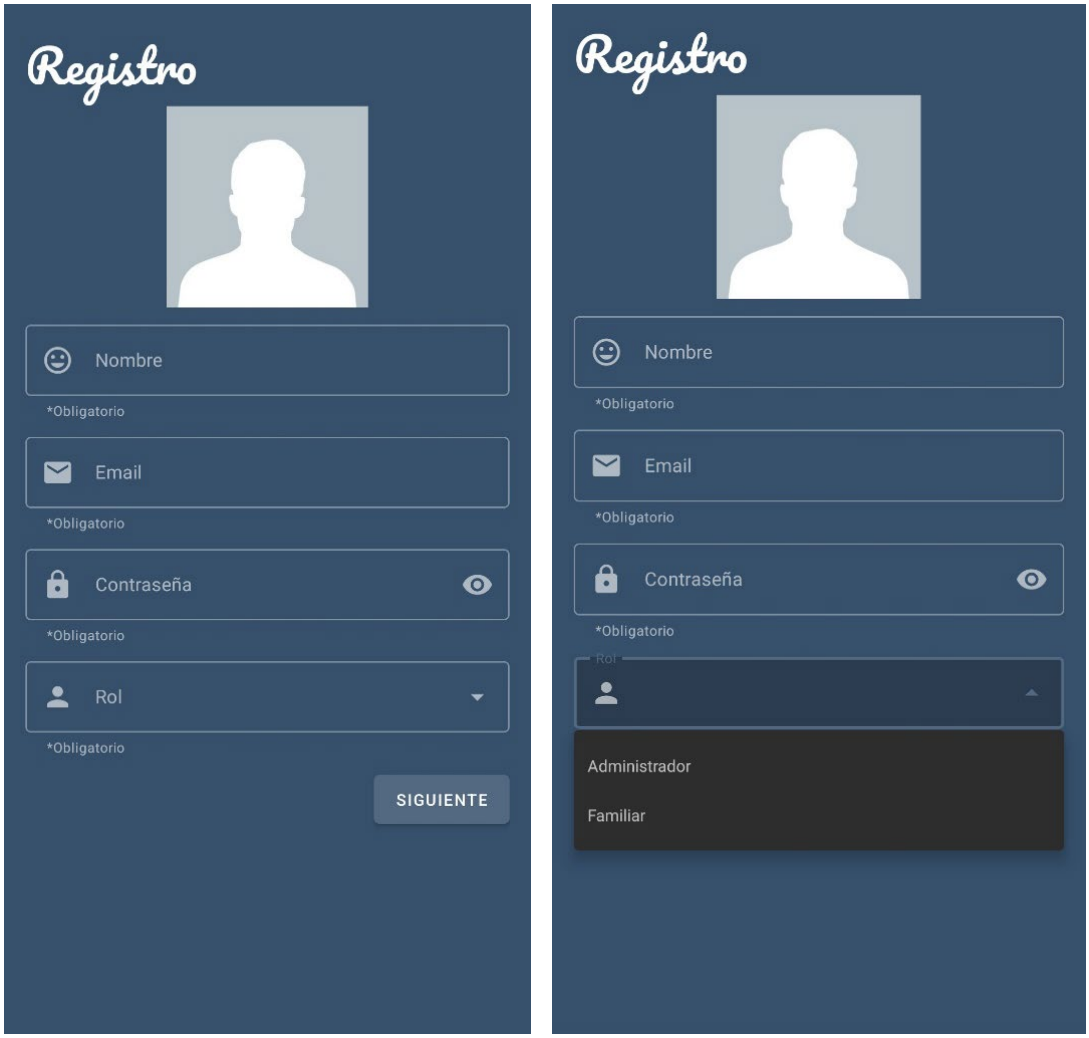


Figura 6: Registro

En el tercer caso, se hace clic sobre el tercer botón de la pantalla inicial y por lo tanto se accede a la aplicación seleccionando una de las cuentas que haya vinculadas al dispositivo móvil, como se puede ver en la Figura 7.

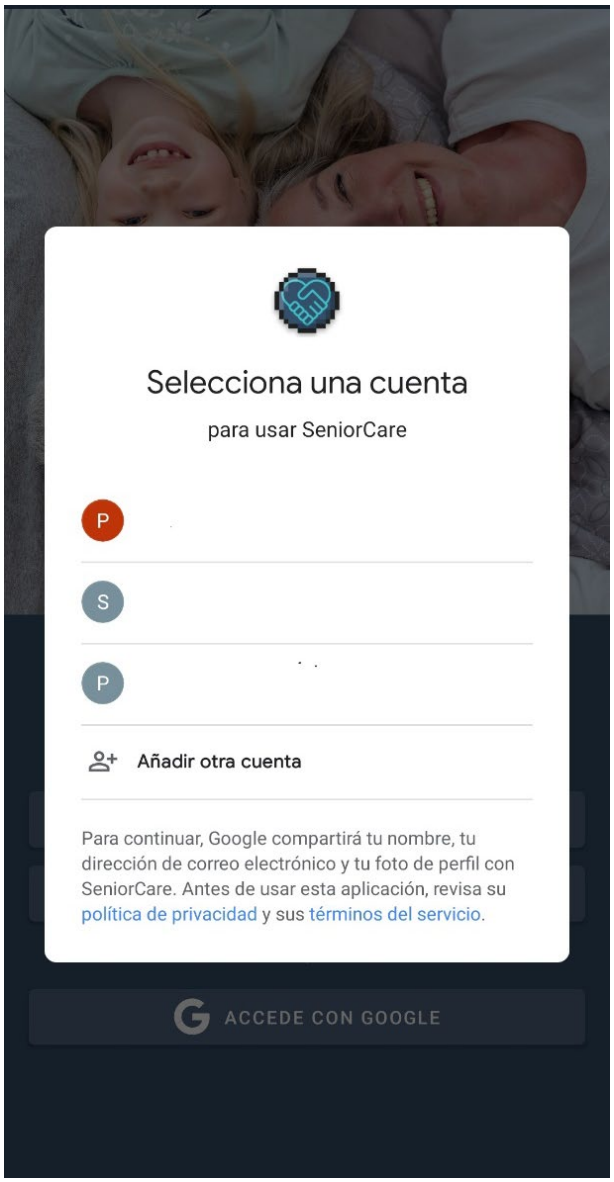


Figura 7: Acceso con Google

Al acceder a la aplicación después de iniciar sesión de alguna de las formas comentadas anteriormente, se llega a la pantalla de home, como se puede ver en la Figura 8, donde se podrá navegar a tres ventanas diferentes haciendo clic sobre alguno de los botones de la parte inferior. Por defecto, se encuentra en la ventana del historial, donde se puede escoger entre ver el historial de llamadas o de alertas, junto a la información de cada una y la hora y el día en el que ocurrieron.

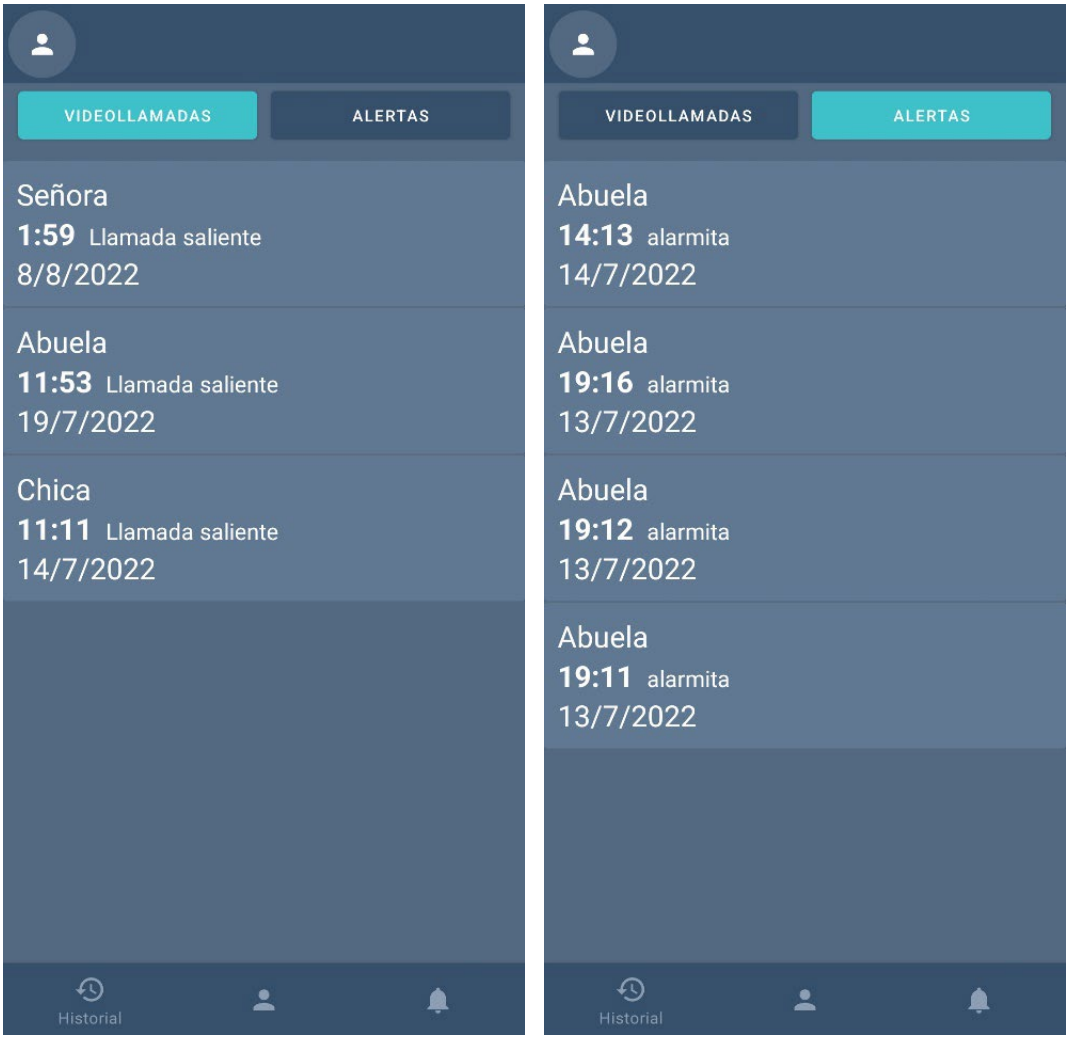


Figura 8: Historial de videollamadas y alertas

Al hacer clic sobre el botón central de la parte inferior, se navega hasta la ventana de familiares agregados, como se puede ver en la Figura 9, donde se muestra una vista rápida de los perfiles de los familiares donde se muestra la imagen de perfil, el nombre y el email. Si se hace clic sobre el botón superior que dice “Buscar familiar”, se abre una ventana de búsqueda de usuarios, y al hacer clic sobre alguna de las cartas, se puede consultar el perfil completo del familiar.

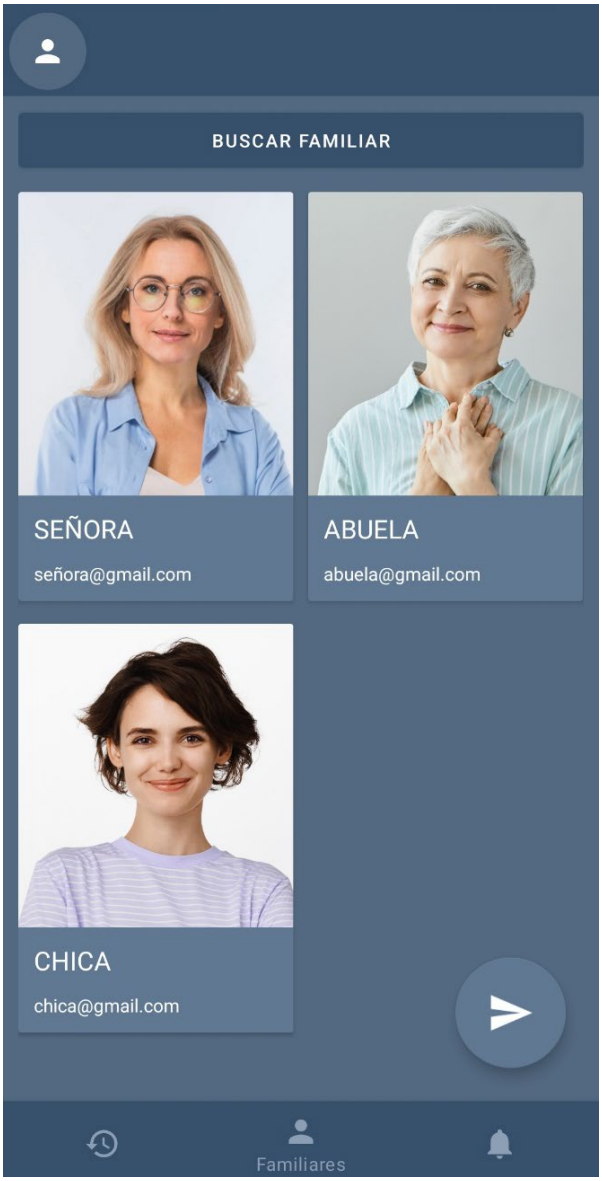


Figura 9: Familiares añadidos

Al hacer clic sobre el botón superior, se abre un desplegable de búsqueda, como se puede ver en la Figura 10, donde se pueden realizar búsquedas en función del email del usuario y se muestran los resultados más afines de todos los usuarios registrados en la aplicación.

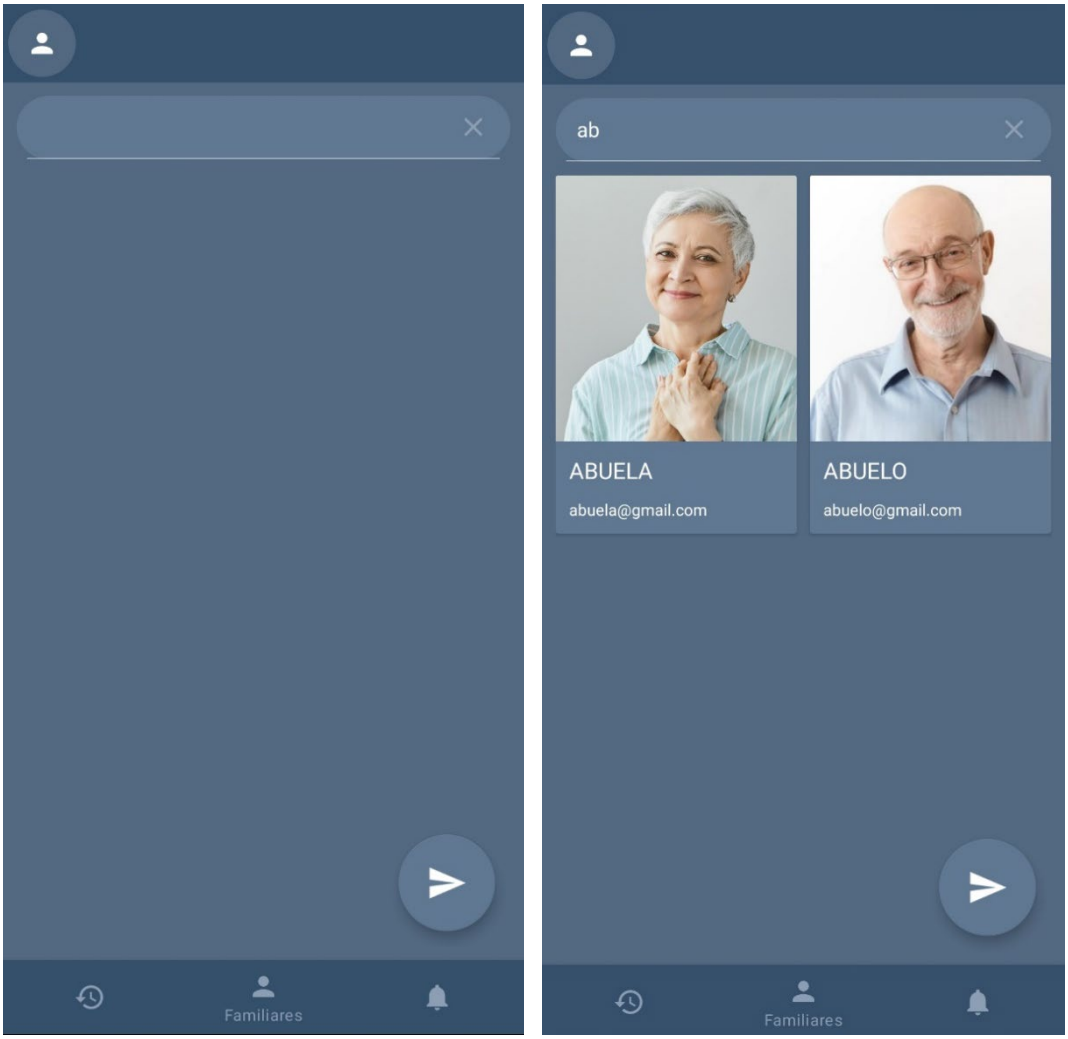


Figura 10: Buscar familiares

Haciendo clic sobre la carta de uno de los familiares, se puede ver una vista detallada de su perfil, como se puede ver en la Figura 11, donde se muestra la foto de perfil, el nombre, el email y el rol del usuario. También hay dos botones que permiten hacer una videollamada o eliminarlo como familiar.

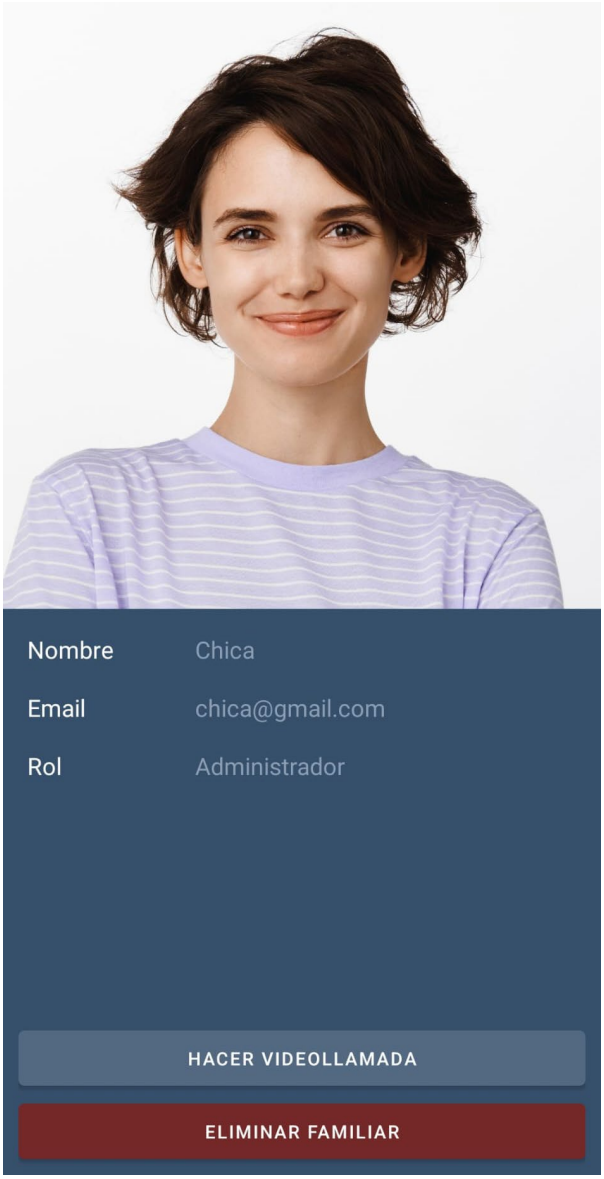


Figura 11: Perfil del familiar

Si se hace clic sobre el botón de realizar una videollamada, se abre una ventana de espera para la videollamada, como se puede ver en la Figura 12.

Figura 12: Videollamada saliente

Mientras el usuario que inicia la videollamada espera a que el familiar responda, este recibe una notificación de la llamada entrante y se le abre la siguiente ventana para aceptar o rechazar nuestra videollamada, como se puede ver en la Figura 13. Si acepta la videollamada, se pasaría a establecerla entre los dos usuarios por medio de jitsi.

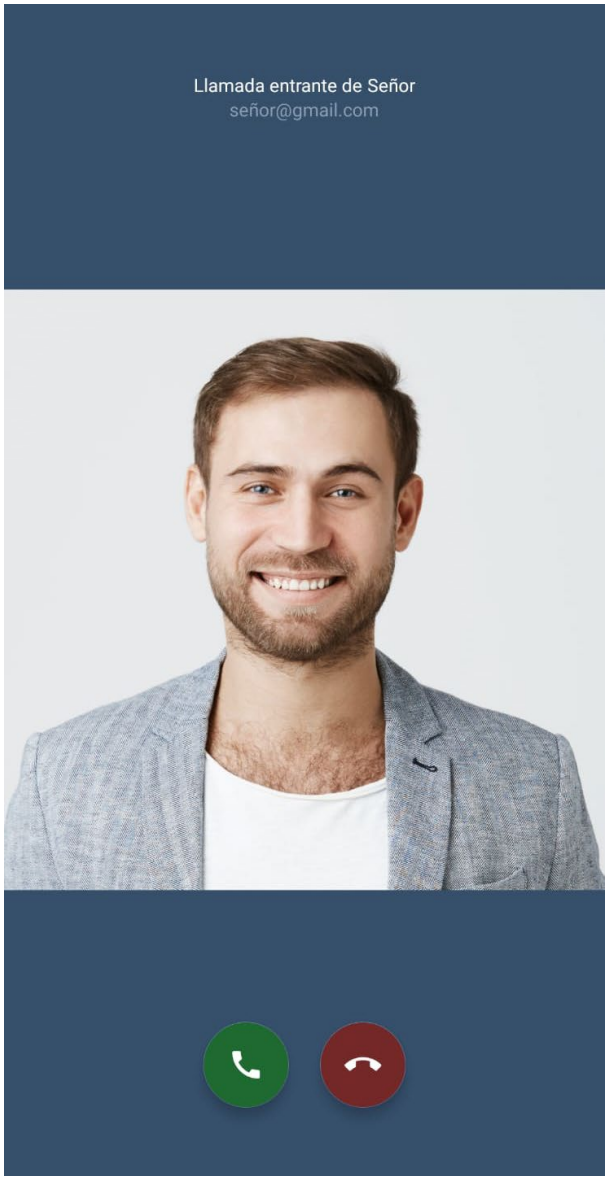


Figura 13: Videollamada entrante

Si ahora se busca a algún usuario que no se haya agregado y se hace clic sobre su carta, se mostrará la información de su perfil, como se puede ver en la Figura 14, pero esta vez con la opción de enviar una petición de familiar. Al hacer clic sobre el botón de enviar la solicitud, se muestra un mensaje informativo al usuario.

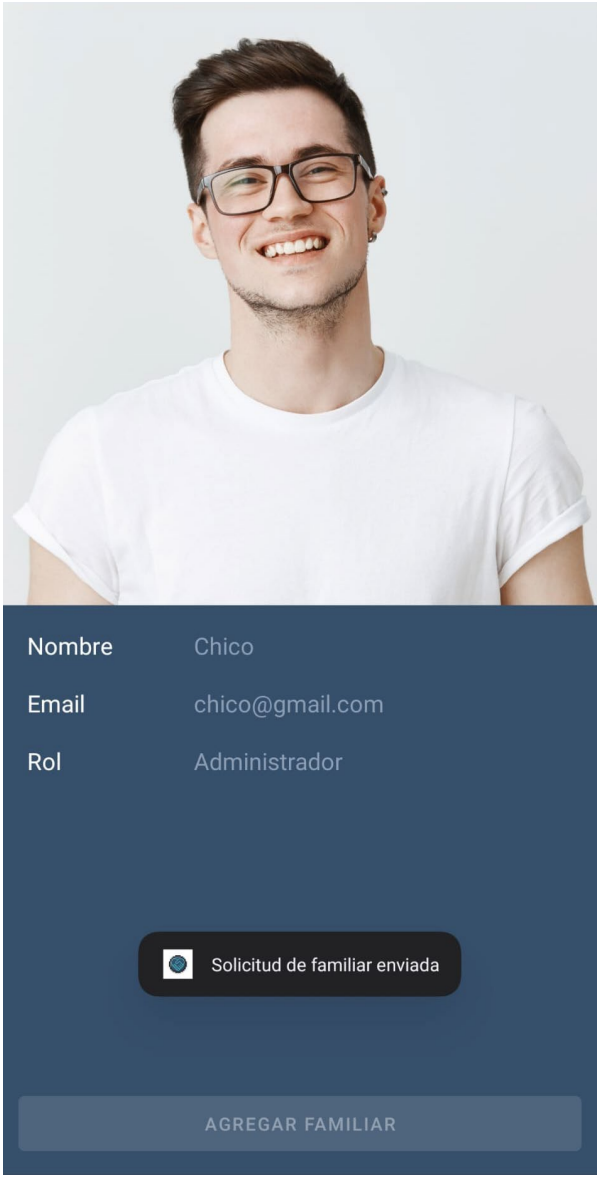


Figura 14: Petición de familiar enviada

Si el otro usuario entra en la aplicación, va al apartado de familiares y hace clic sobre el botón flotante de peticiones de familiares, accederá a ver las peticiones pendientes por aceptar o rechazar, como se puede ver en la Figura 15.

Figura 15: No hay familiares agregados

En esta ventana se mostrarán las peticiones pendientes, como se puede ver en la Figura 16. Si el usuario hace clic sobre aceptar, este pasará a formar parte de los familiares de ambos. En el caso contrario, se rechazará la solicitud y no serán familiares.

Figura 16: Petición pendiente

El último botón de la parte inferior contiene las pantallas de alertas donde se pueden ver las alertas que se han establecido y a qué familiares, junto con información de los detalles de la propia alerta, como se puede ver en la Figura 17. Si se hace clic sobre el botón flotante, se llega a una pantalla de creación de alertas.

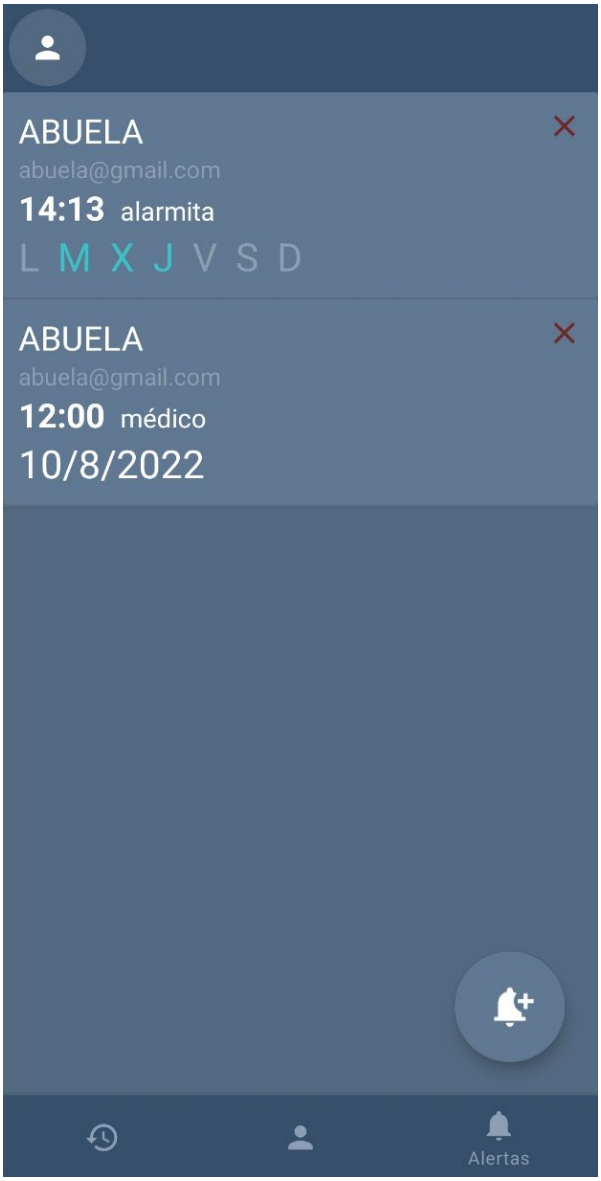


Figura 17: Alertas creadas

En esta pantalla se da la opción de rellenar y seleccionar diferentes campos, como se puede ver en la Figura 18. Los familiares que se pueden seleccionar serán sólo aquellos que hayan marcado como “Gestor” al usuario (se verá más adelante).

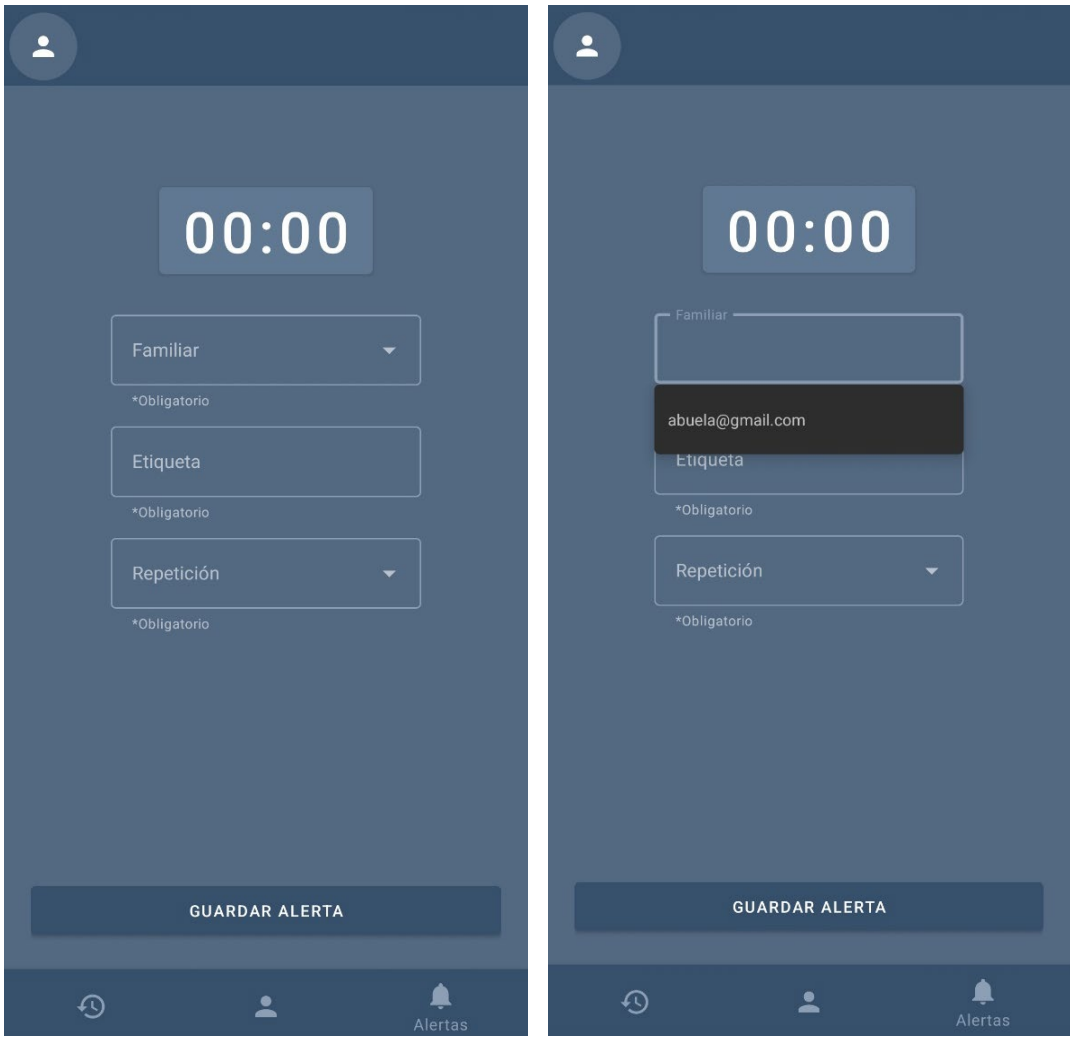


Figura 18: Crear nueva alerta

Se puede seleccionar la hora a la que se quiere establecer la alerta tanto en analógico como en digital, como se puede ver en la Figura 19.

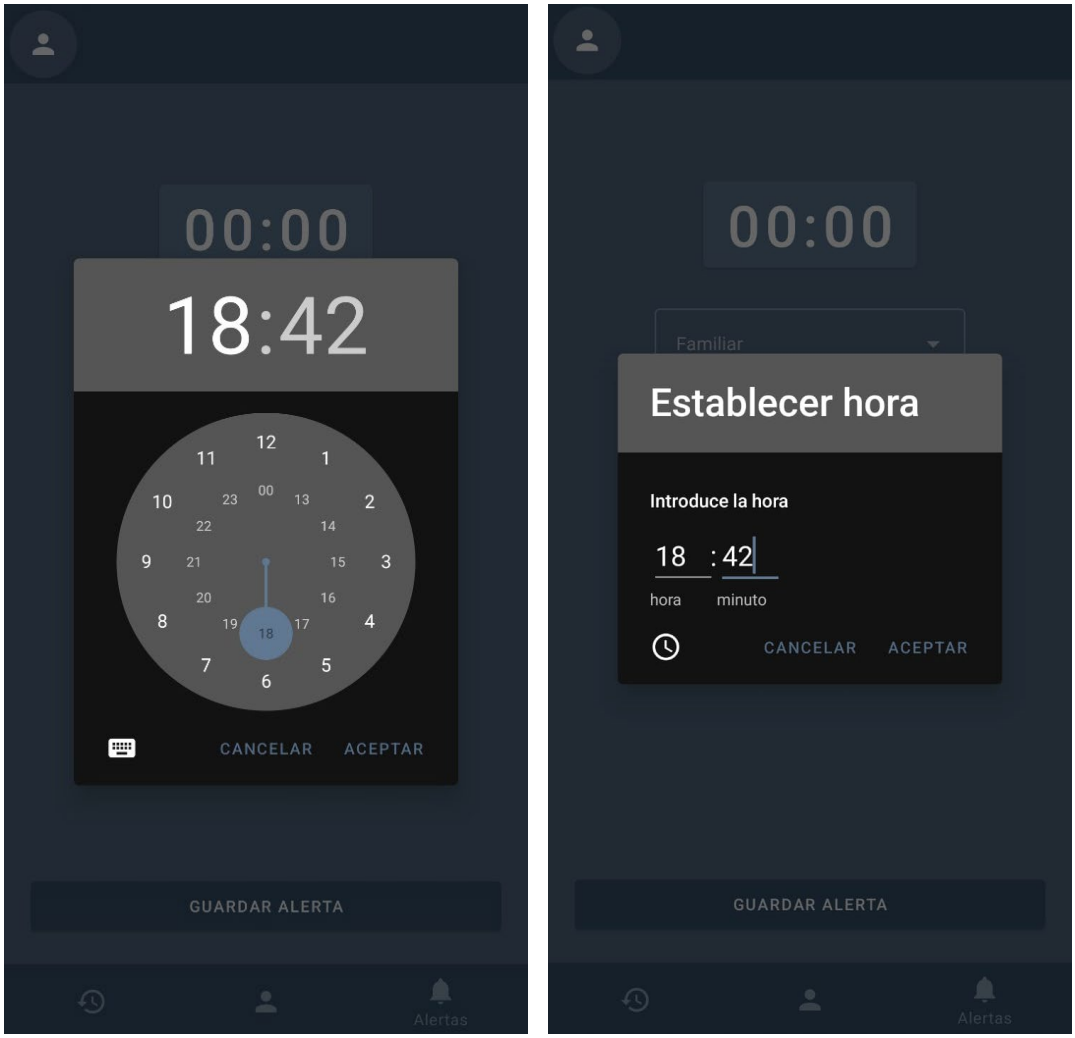


Figura 19: Seleccionar hora de la alerta

El primer tipo de alerta que se puede crear es la alerta eventual, como se puede ver en la Figura 20, que será aquella que sólo se repite una vez en el tiempo como una consulta con el médico.

Figura 20: Alerta eventual

Hay que seleccionar la fecha de esta alerta eventual mediante un calendario, como se puede ver en la Figura 21.

Figura 21: Seleccionar fecha

El otro tipo de alerta será la alerta semanal, que es aquella que tiene que repetirse durante todas las semanas, como se puede ver en la Figura 22. La funcionalidad es la misma que para una alerta eventual, pero en este caso se marcan los días de la semana en los que tiene que sonar.

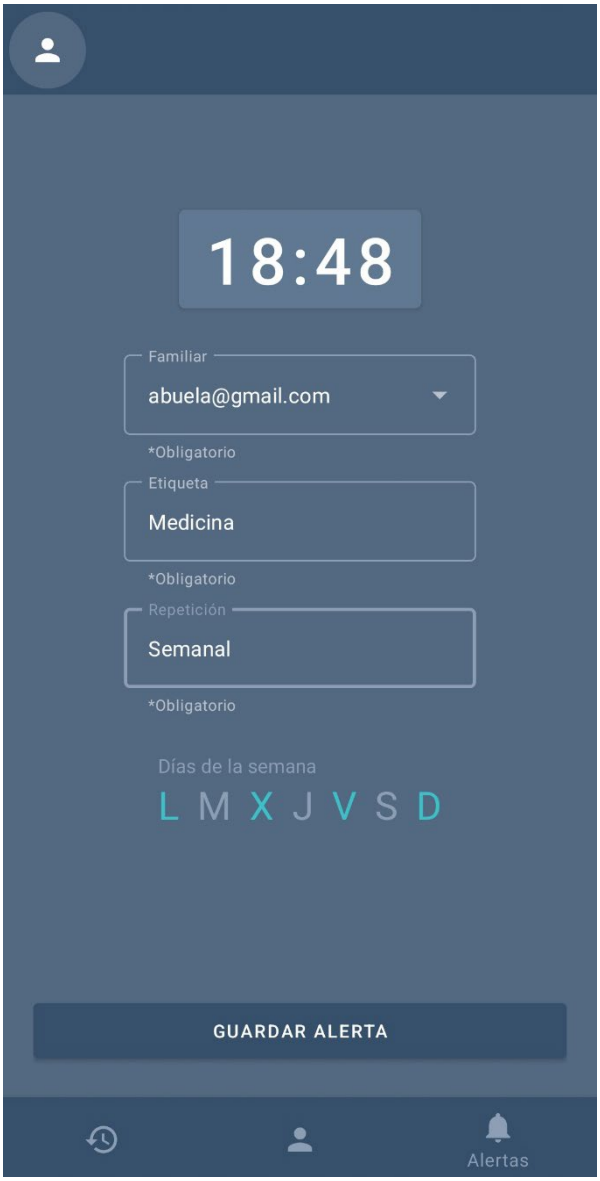


Figura 22: Alerta semanal

Si se quiere borrar alguna de las alertas establecidas, hay que ir a la ventana general de alertas y hacer clic sobre la cruz roja situada a la derecha de cada alerta. Se mostrará un panel que pregunta si realmente se quiere eliminar dicha alerta, como se puede ver en la Figura 23.

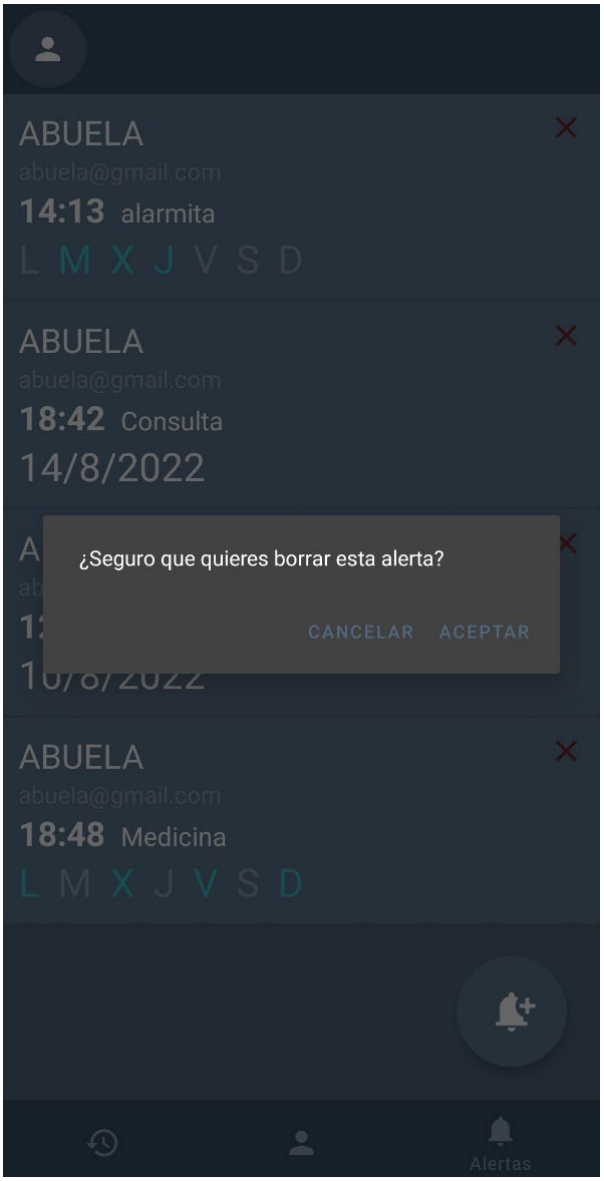


Figura 23: Eliminar alerta

Si ahora se hace clic sobre el icono de la esquina superior izquierda, se navega hasta la pantalla del perfil del usuario, como se puede ver en la Figura 24, donde se pueden ver los mismos datos que en el perfil de cualquier usuario, pero el usuario también puede editar su foto y su nombre, y cerrar sesión con su cuenta.

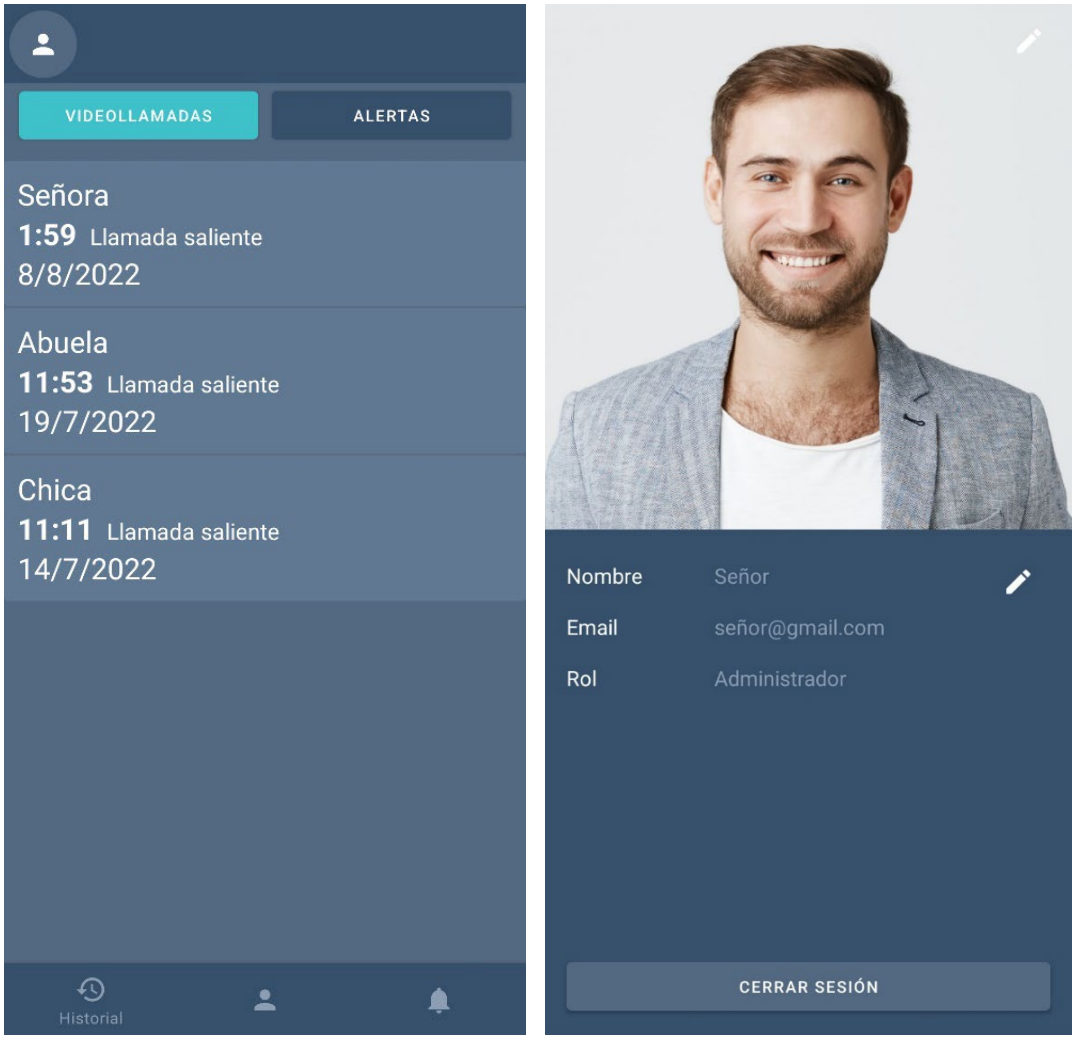


Figura 24: Perfil de usuario

Si se hace clic sobre el icono del lapicero de la imagen, se abre la galería donde se da la opción de seleccionar una nueva foto y modificarla, como se puede ver en la Figura 25.

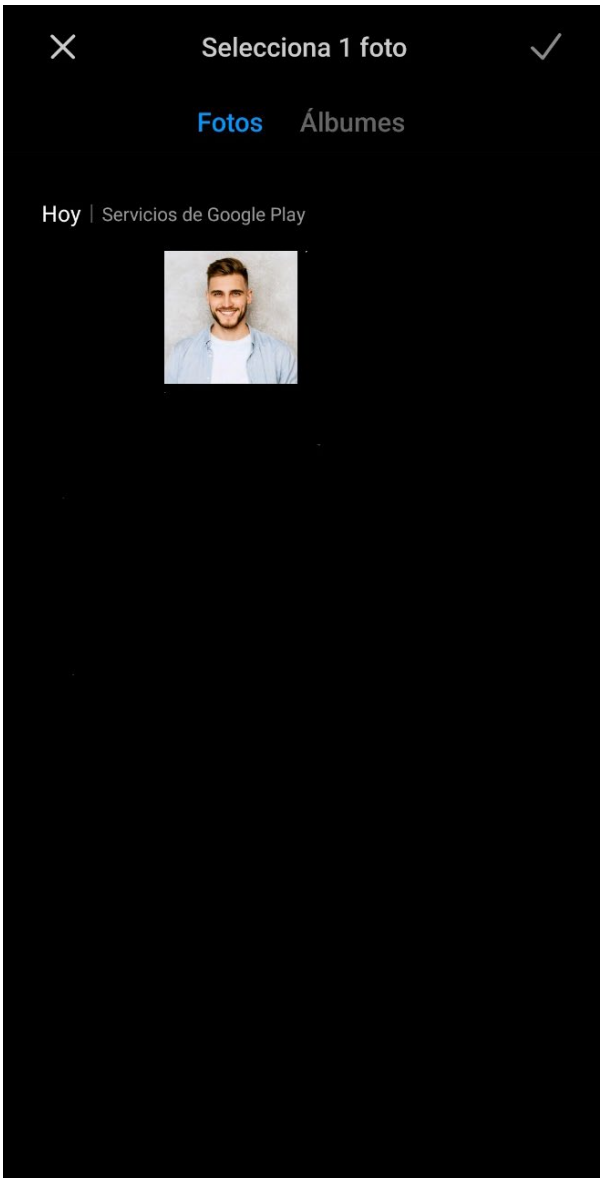


Figura 25: Modificar imagen de usuario

Si en cambio, se hace clic sobre el icono del lapicero del nombre, se abre un panel donde se da la opción de introducir un nuevo nombre y se modificará, como se puede ver en la Figura 26.

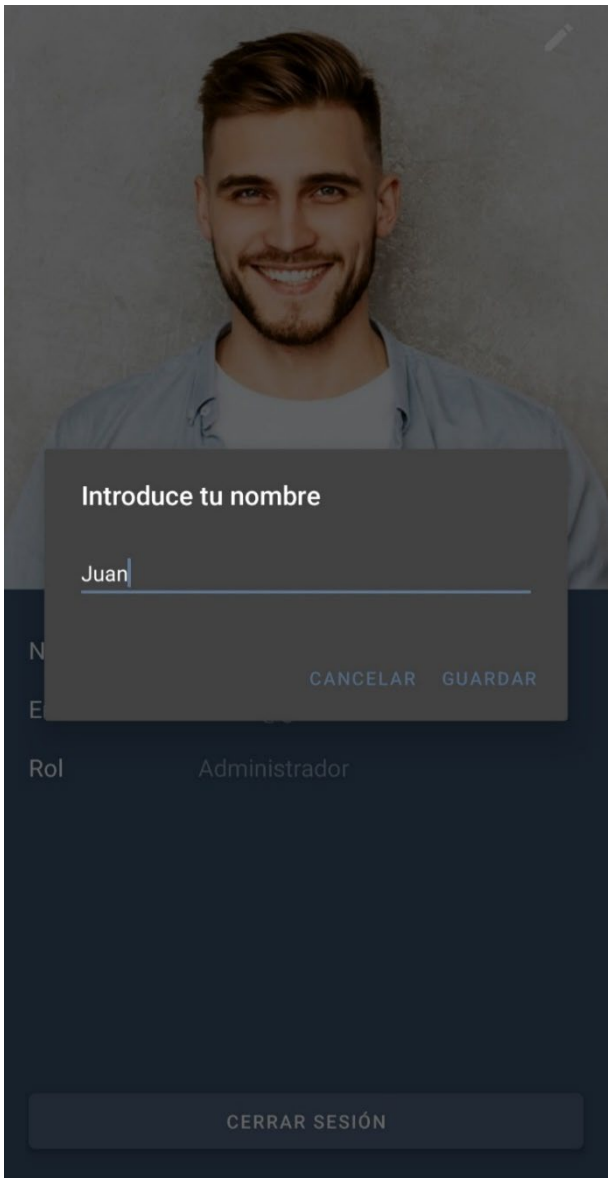


Figura 26: Modificar nombre de usuario

Ahora ya se podrían ver todos los datos modificados. Si ahora se hace clic sobre el botón de cerrar sesión, se envía al usuario a la pantalla de inicio y cerrará sesión en su cuenta de usuario, como se puede ver en la Figura 27.

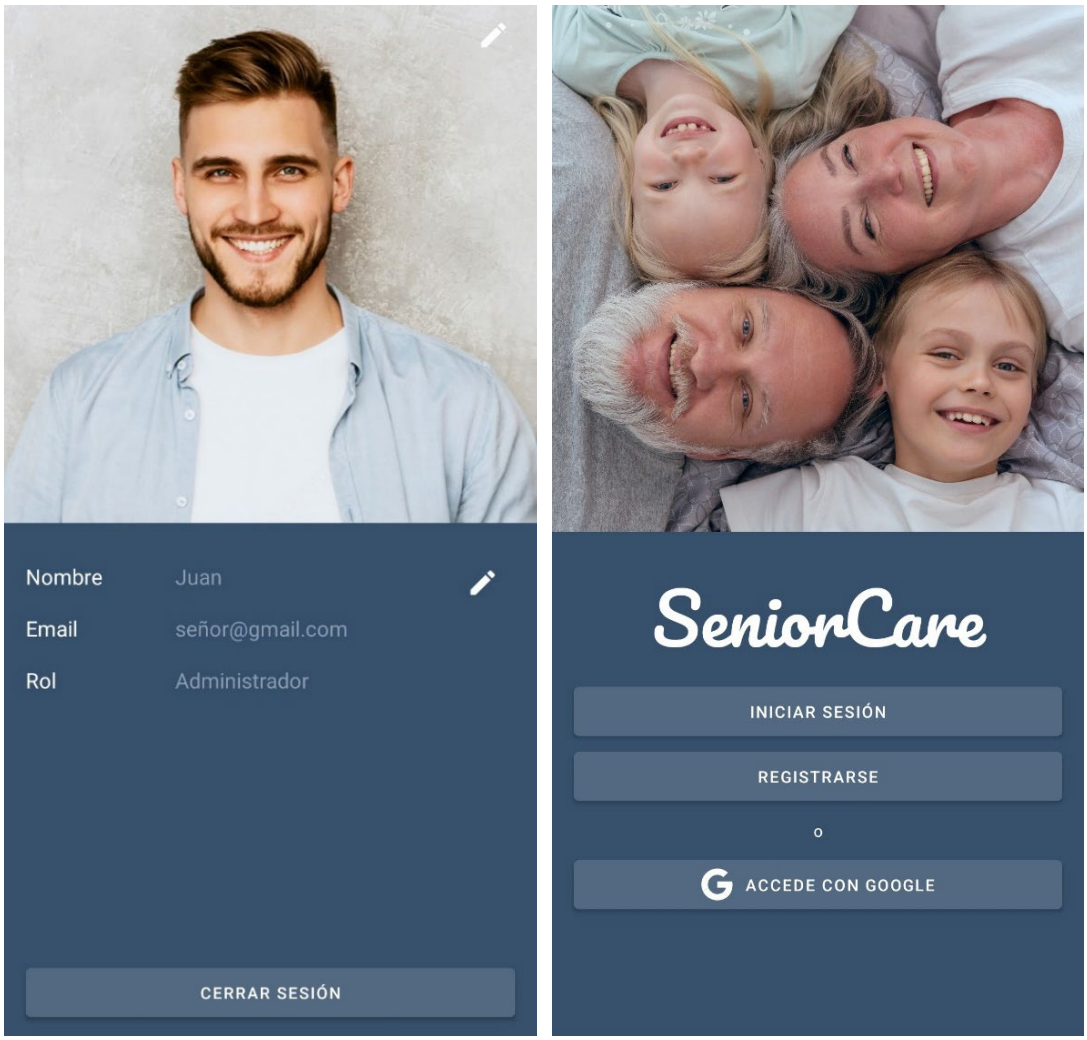


Figura 27: Cerrar Sesión

2.3. Android TV

Al iniciar la aplicación en Android TV se ve la misma pantalla, pero con otra disposición adaptada a este formato. También están adaptados los controles para que se pueda desplazar por el pad direccional del mando de Android o de la televisión si esta tiene el protocolo CEC. Podemos verlo en la Figura 28.

Figura 28: Mando de Android TV

Esta parte de la aplicación está pensada para los usuarios familiares o personas mayores, aunque pueden acceder también desde el dispositivo móvil y los usuarios normales también pueden acceder a través de la televisión. Se puede acceder de la misma forma que en el caso del dispositivo móvil con tres métodos diferentes, como se puede ver en la Figura 29.

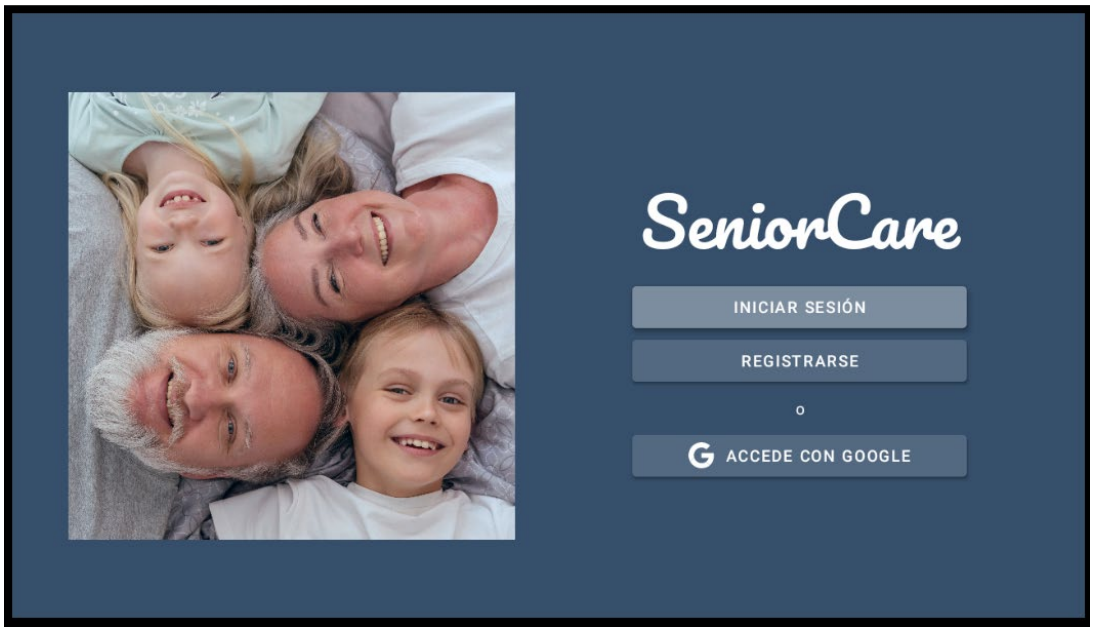


Figura 29: Pantalla inicial TV

Para iniciar sesión y realizar el registro, se muestra el mismo panel que en el dispositivo móvil pero colocado en horizontal de forma que se vea correctamente, como se puede ver en la Figura 30. También aparece un teclado flotante para poder introducir los datos.

Figura 30: Inicio de sesión TV

Al acceder se tiene una vista más simplificada y adaptada a una televisión y con menos interfaces. Se puede ver la lista de familiares que están agregados al igual que antes, pero distribuidos de otra forma, como se puede ver en la Figura 31.

Figura 31: Ver familiares añadidos TV

Se pueden realizar búsquedas de usuarios de forma similar al dispositivo móvil, y también está el teclado flotante que ayuda en la interacción con la televisión, como se puede ver en las Figuras 32 y 33.

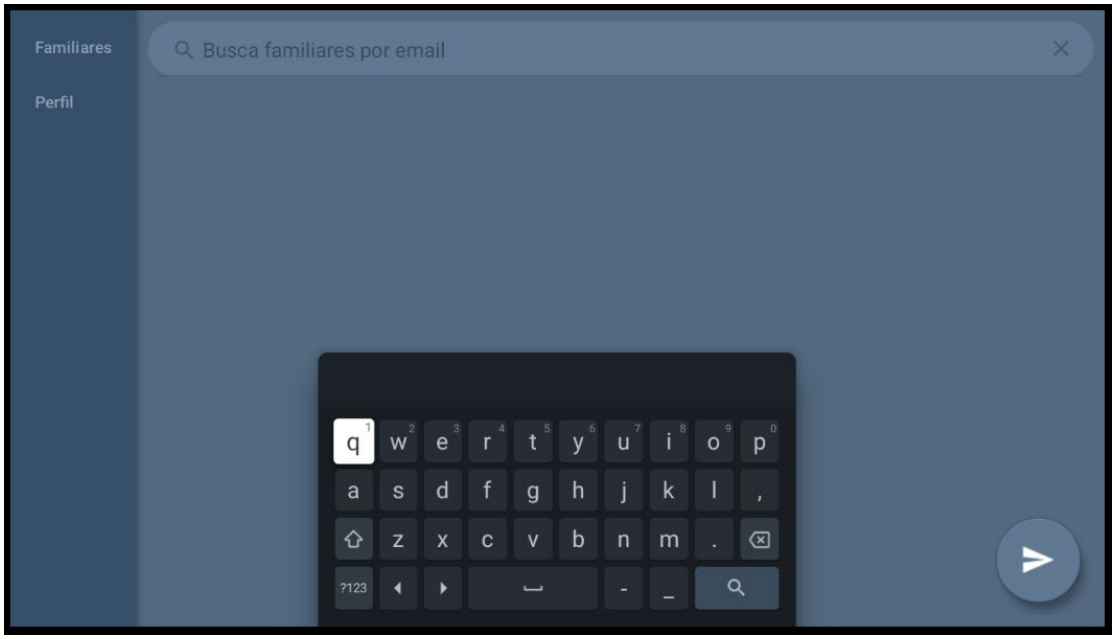


Figura 32: Búsqueda de usuarios TV

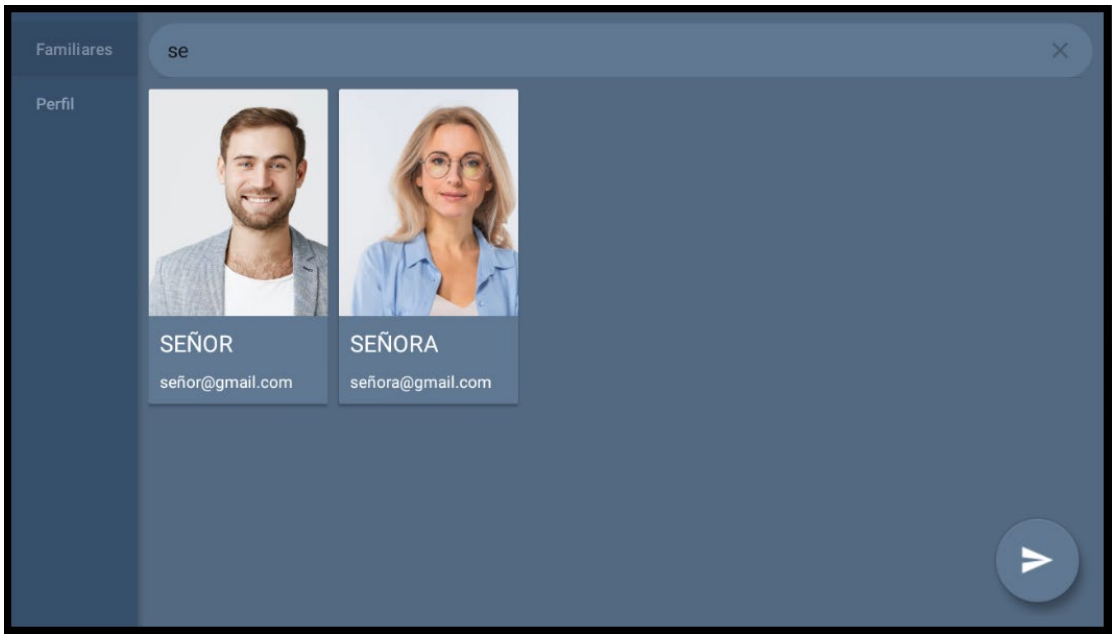


Figura 33: Resultados de la búsqueda de usuarios TV

Si se hace clic sobre una carta de usuario, se muestra el perfil detallado, y en este caso, a este familiar lo tenemos agregado, como se puede ver en la Figura 34. Aquí se puede ver la función de establecer a un familiar como “Gestor de alertas” de modo que pueda crear y modificar sus alertas.

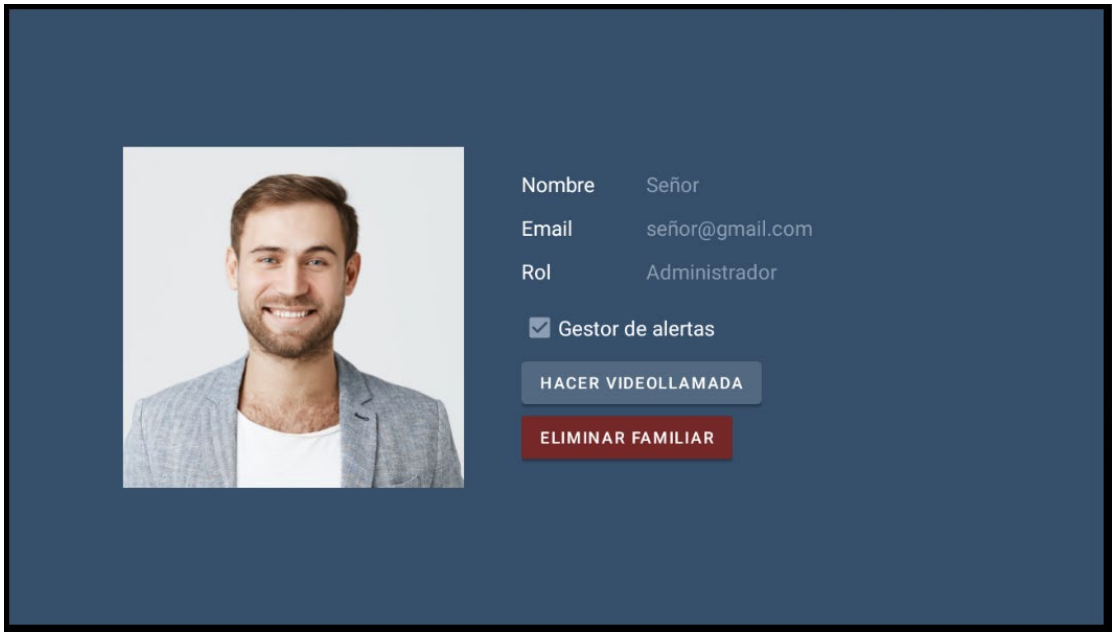


Figura 34: Perfil de familiar TV

Si se hace clic sobre el botón flotante de la pantalla principal, se muestran las solicitudes pendientes de forma similar, como se puede ver en la Figura 35.

Figura 35: Solicitudes pendientes TV

Por último, si se navega al apartado del perfil en el lado izquierdo, se puede ver el perfil del usuario de igual forma que en el dispositivo móvil, como se puede ver en la Figura 36.

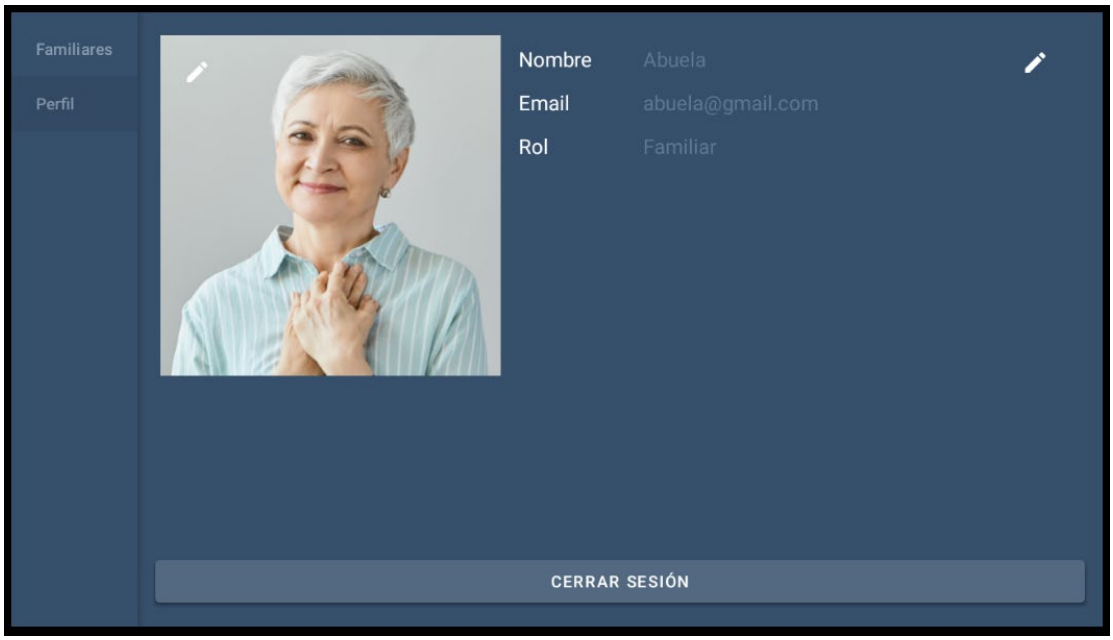


Figura 36: Perfil de usuario TV