UNIVERSITAT DE BARCELONA

Trabajo de Final de Grado

GRADO DE INGENIERÍA INFORMÁTICA

Facultad de Matemáticas e Informática
Universidad de Barcelona

Desarrollo de una aplicación móvil para sistemas operativos Android dirigido a mejorar la gestión diaria de asociaciones juveniles

---

Beatriz Bachs de Lacoma

Director: Simone Balocco
Realizado en: Departamento de Matemátiques e Informática

Barcelona, 15 de enero de 2024

Resumen

Este proyecto de final de carrera se enfoca en abordar las deficiencias de gestión y comunicación en asociaciones juveniles, con énfasis en la Asociación Cultural Familiar Carena. La carencia de herramientas eficientes se manifiesta en la utilización de grupos de WhatsApp, generando problemas como la falta de información completa para los padres y la dificultad de los monitores para planificar actividades. El objetivo principal de este trabajo consiste en desarrollar un prototipo de aplicación móvil que cubra las necesidades de monitores, personal administrativo y padres. Para ello, se ha realizado un análisis de mercado en el que se ha detectado que las funcionalidades clave de esta aplicación es que está dirigida a 3 tipos de usuarios distintos y que contiene un chat que permite la comunicación entre ellos. Gracias a este análisis y al estudio de requisitos se han detectado otras funcionalidades que deben cubrir las necesidades de los distintos agentes. Una vez detectados estos requisitos, se ha hecho el diseño de la aplicación utilizando la herramienta Moqups y se ha desarrollado utilizando Android Studio, siguiendo en todo momento una arquitectura que sigue los principios de “Clean Architecture”. Este trabajo destaca la importancia de soluciones tecnológicas que optimicen la gestión y la comunicación en asociaciones juveniles.

Resum

Aquest projecte de final de carrera s'enfoca a abordar les deficiències de gestió i comunicació en associacions juvenils, amb èmfasi a l'Associació Cultural Familiar Carena. La manca d'eines eficients es manifesta en la utilització de grups de WhatsApp, generant problemes com ara la manca d'informació completa per als pares i la dificultat dels monitors per planificar activitats. L'objectiu principal d'aquest treball és desenvolupar un prototip d'aplicació mòbil que cobreixi les necessitats de monitors, personal administratiu i pares. Per fer-ho, s'ha realitzat una anàlisi de mercat on s'ha detectat que les funcionalitats clau d'aquesta aplicació és que està dirigida a 3 tipus d'usuaris diferents i que conté un xat que permet la comunicació entre ells. Gràcies a aquesta anàlisi i a l’estudi de requisits s’han detectat altres funcionalitats que han de cobrir les necessitats dels diferents agents. Un cop detectats aquests requisits, s'ha fet el disseny de l'aplicació utilitzant l'eina Moqups i s'ha desenvolupat utilitzant Android Studio, tot seguint una arquitectura que segueix els principis de “Clean Architecture”. Aquest treball destaca la importància de solucions tecnològiques que optimitzin la gestió i la comunicació en associacions juvenils.

Abstract

This final degree project focuses on addressing management and communication deficiencies in youth associations, with an emphasis on the Carena Family Cultural Association. The lack of efficient tools is manifested in the use of WhatsApp groups, generating problems such as the lack of complete information for parents and the difficulty for monitors to plan activities. The main objective of this work is to develop a prototype of a mobile application that meets the needs of monitors, administrative staff and parents. To this end, a market analysis has been carried out in which it has been detected that the key functionalities of this application are that it is aimed at 3 different types of users and that it contains a chat that allows communication between them. Thanks to this analysis and the requirements study, other functionalities have been detected that should cover the needs of the different agents. Once these requirements were detected, the application was designed using the Moqups tool and developed using Android Studio, following at all times an architecture that follows the principles of "Clean Architecture". This work highlights the importance of technological solutions that optimise management and communication in youth associations.

Tabla de contenido

1 Introducción

1.1 Motivación

Después de varios años habiendo colaborado con diferentes casales y asociaciones juveniles que tratan con menores, he podido identificar que existe una falta de gestión y comunicación entre los monitores de la asociación, los padres de los asociados y el personal administrativo.

Actualmente no existe en el mercado ninguna aplicación al alcance de estas asociaciones que cubra las necesidades de todos los agentes implicados: padres, monitores y personal administrativo. La gestión diaria se realiza, en su mayoría, a través de grupos de WhatsApp en la que los monitores informan de las actividades planificadas y los padres indican la asistencia a estas actividades, con suerte, a través de encuestas de Whatssap.

Este modo de funcionamiento deriva en varios problemas. A continuación, se detallan algunos, ya que es objeto de este trabajo identificarlos todos:

- A los padres no les llega la información de las actividades planificadas para sus hijos al completo
- Los monitores no tienen claro qué niños van a participar de las actividades que han organizado, por lo que muchas veces falta material o espacio
- Las inscripciones a las actividades extras (campamentos, actividades de pago…) que no forman parte de la cuota, se realizan a través de la web. Esta información llega al equipo de administración y, estos, la traspasan a los monitores, por lo que los monitores no pueden consultar en tiempo real los datos que les interesan para planificar las actividades.

Es clara la necesidad que existe de encontrar una solución que permita gestionar el día a día de estas asociaciones y casales de forma ágil y segura.

En el apartado 3.1.1 se describirá en mayor detalle los problemas identificados y las necesidades de todos los agentes implicados (monitores, personal administrativo, padres).

El objetivo de la aplicación es que la pueda utilizar cualquier asociación o casal juvenil. Sin embargo, a lo largo de todo este proyecto se va a hablar de una asociación en concreto, la Asociación Cultural Familiar Carena, en la que participan 217 familias, 50 monitores voluntarios y más de 400 niños asociados. Esta asociación se dedica a organizar actividades de ocio y de estudio como herramientas de un proyecto educativo de menores.

1.2 Objetivos

La finalidad de este proyecto de final de grado es desarrollar un prototipo de aplicación que permita solucionar los problemas existentes en la gestión diaria de las asociaciones juveniles. Con este fin, se han definido los siguientes objetivos:

1. Identificar las necesidades diarias de los agentes implicados en las asociaciones juveniles
2. Llevar a cabo un análisis de mercado para detectar las necesidades no cubiertas de los usuarios objetivo
3. Diseñar, tanto visualmente como arquitectónicamente, un prototipo de aplicación
4. Desarrollar el prototipo para dispositivos con sistema operativo Android
5. Aprender a utilizar herramientas de diseño y nuevas soluciones de programación para resolver los problemas que se presenten

Para conseguir el objetivo final del trabajo, que es desarrollar el prototipo de una aplicación móvil, se seguirá el proceso que se sigue en cualquier desarrollo de aplicaciones móviles.

En primer lugar, se definirá la estrategia a seguir realizando un análisis de la situación y del mercado. Estos dos aspectos se abordan en los apartados "2. Análisis previo" y "3. Análisis de mercado", respectivamente. A continuación, en el apartado "4. Ingeniería de concepción", se definirán los requisitos y especificaciones del sistema. En esta sección también se llevará a cabo el diseño de la aplicación. Por último, y sólo después de haber dado todos los pasos anteriores, el desarrollo de la app se iniciará.

Cada vez que se desarrolla una nueva funcionalidad, se prueba toda la aplicación. Este proceso se repite de forma iterativa hasta que todas las funcionalidades están finalmente implementadas y la aplicación puede salir al mercado.

En este trabajo sólo se llevará a cabo la primera iteración del proceso, ya que el objetivo es estudiar la viabilidad funcional de la aplicación y no publicar la aplicación en el mercado para su libre uso por parte de los usuarios.

2 Análisis previo

En cualquier proyecto de desarrollo de software es esencial la gestión del trabajo y del equipo para poder llevar a cabo los objetivos del proyecto. En el año 2001, 17 expertos en desarrollo de software elaboraron un documento en el que se describía los principios básicos de la metodología que llamaron Agile[i].

La metodología Agile mantiene la dirección sin caer en la rigidez de los conocidos métodos en waterfall. Estos planean el trabajo desde el principio, sin lugar a imprevistos. De forma que cuando aparecen, resulta imposible reaccionar a tiempo. El agilismo, sin embargo, mantiene la capacidad de tomar la mejor opción en cada momento sin comprometer el proyecto. Los métodos Agile[ii] más populares del momento son Scrum y Kanban.

2.1 Metodología ágil Kanban

Durante el desarrollo de este proyecto se ha utilizado la metodología Kanban. Es un método visual que se utiliza para controlar las tareas a través de su división por fases, hasta su finalización[iii]. Su objetivo es mejorar la comunicación, en los proyectos y tener claro en todo momento, la fase en la que nos encontramos y el número de tareas que están pendientes. También permite establecer un límite en el número de tareas que se pueden tener en cada etapa del flujo de trabajo. Esto evita la sobrecarga y el bloqueo del trabajo, fomentando un flujo de trabajo constante[iv].

Este proyecto solamente se ha llevado a cabo por una persona, pero igualmente se ha utilizado esta metodología para organizar el trabajo. Para ello, se ha utilizado la herramienta Trello[v]. El tablero de Trello refleja las tareas a realizar ordenadas por prioridad. También dispone de una vista por columnas. Cada columna representa una etapa del trabajo. El tablero Kanban más básico puede presentar columnas como “trabajo pendiente”, “en progreso” y “terminado”. Las tareas individuales son representadas por tarjetas visuales en el tablero y avanzan a través de las diferentes columnas hasta que estén finalizadas. En este proyecto se han utilizado las siguientes columnas: 1) Backlog: corresponde a todas las tareas que se pretenden realizar a lo largo de todo el proyecto; 2) Doing: son las tareas en proceso que se están realizando; 3) Done: identifica las tareas realizadas

Para una mejor organización, en el tablero de este proyecto se han añadido todas las tareas necesarias para finalizar el proyecto. Esto implica tareas de desarrollo, pero también de documentación, configuración del proyecto y de la base de datos. Cada tipo de tarea se identifica a través de su etiqueta. Trabajar de esta manera ha sido útil para la gestión del proyecto porque permite definir mejor el tiempo de dedicación de cada tarea. A continuación, se explica la diferencia entre cada tipo de tarea:

- Configuración: identifica las tareas de configuración del proyecto. Por ejemplo, conexión de la app con la base de datos, modificaciones en el fichero .gradle de la app, etc. A estas tareas se les ha asignado la etiqueta “Configuración AND” de color grojo.

- Base de datos: identifica las tareas que implican trabajar con la base de datos. Por ejemplo, creación de la base de datos, modificación de algún campo de entrada, configuración de seguridad, etc. Se les ha asignado la etiqueta “Firebase” de color verde.

- Desarrollo: se refiere a las tareas que implica un desarrollo de código utilizando Android Studio. Por ejemplo, desarrollo de una nueva pantalla de la aplicación, de un fichero java con la lógica de negocio, etc. A estas tareas se les ha asignado la etiqueta “Desarrollo” de color azul.

- Memoria: identifica las tareas de escritura de la memoria para que quede documentado todo el trabajo que se ha realizado. Se les ha asignado a estas tareas la etiqueta “Memoria” de color naranja.

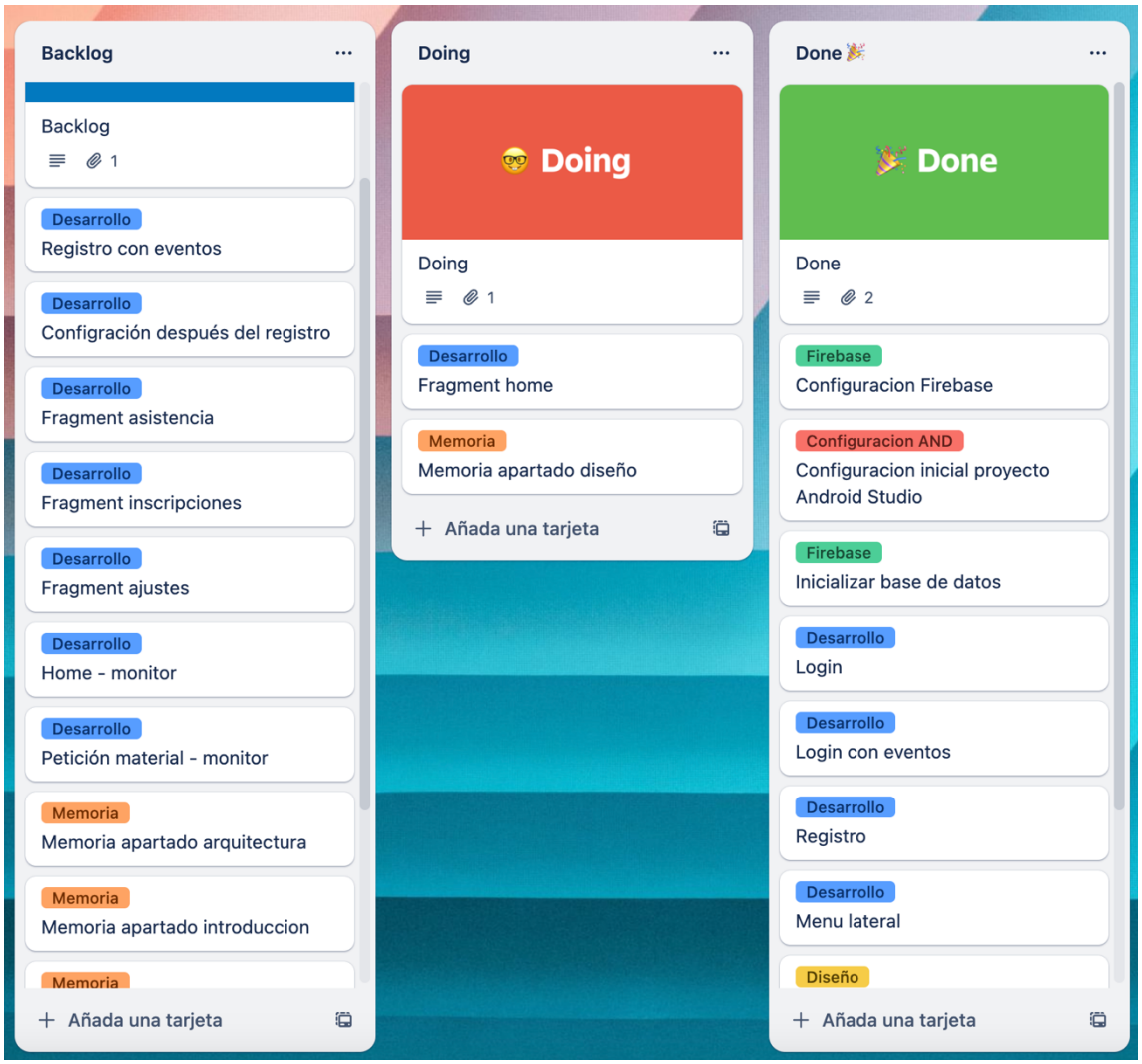


Figura 1. Tablero Trello del proyecto, formado por tres columnas que identifican las tareas del Backlog, las que se están realizando en ese momento y las realizadas.

2.2 GANTT

Un diagrama de GANTT es una forma práctica de mostrar las tareas a realizar en función del tiempo[vi]. En la figura siguiente se muestra el diagrama de Gantt de este proyecto[vii]. A la izquierda del gráfico se muestra la lista de actividades y, en la parte superior, la escala de tiempo. Cada actividad está representada por una barra, cuya posición y longitud reflejan el inicio, la duración y el final de la actividad.

Se ha planificado el proyecto en 9 grupos de tareas, que se han dividido a lo largo de los 4 sprints. Cada sprint tiene un mes de duración.

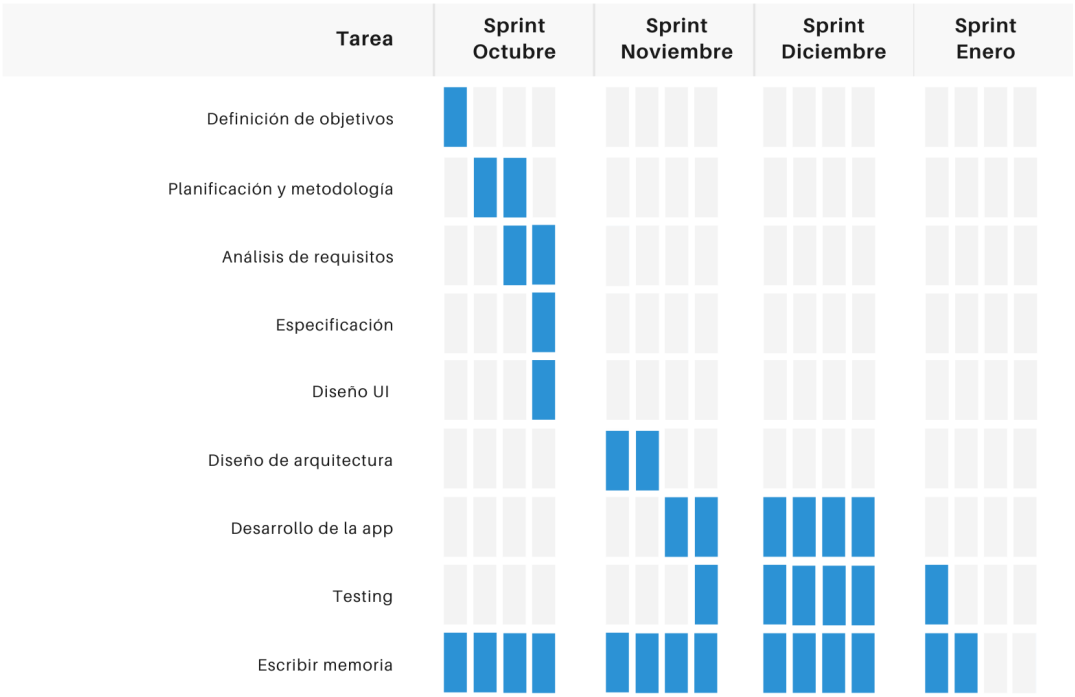


Figura 2. Diagrama de GANTT del proyecto. A la izquierda del diagrama aparecen todas las tareas que forman parte del proyecto. Cada casilla coloreada representa la semana del mes en que se va a trabajar en esa tarea.

3 Análisis de mercado

Un punto importante de este proyecto es conocer las soluciones de mercado que proporcionan un servicio similar a la aplicación que se quiere desarrollar en este trabajo final de grado. De este modo, se pueden detectar los puntos fuertes y débiles de estas aplicaciones, describir las funcionalidades clave que la aplicación de este proyecto puede aportar a los usuarios y definir mejor lo que la diferencia de sus competidores.

3.1 Soluciones similares

La mayoría de las soluciones que existen en el mercado para gestionar actividades están dirigidas a las Asociaciones de Madres y Padres de Alumnos (AMPA) de los colegios. De entre las que existe, se han seleccionado las siguientes para hacer un análisis de la situación actual del mercado.

- MiAmpa[viii]: aplicación enfocada a gestionar el AMPA de los colegios. Ofrece una gran variedad de funcionalidades (inscripción, extraescolares, eventos, tienda, contabilidad, estadísticas, comedor, noticias, documentación y foro…).

  Está disponible para cualquier dispositivo móvil y también ofrece una interfaz web. Ofrece tres tipos de cuotas para sus socios. El plan gratuito es muy limitado (solo permite publicar una actividad, un evento y un máximo de 100 familias).

  Solamente hay dos tipos de usuarios: administrador y familias. El administrador es el que gestiona todas las actividades, campamentos, contabilidad… Tampoco existe un chat directo entre el administrador y las familias.

- TokApp[ix]: solución digital que conecta centros educativos con las familias a través de una plataforma web y App móvil. Permite la gestión de cobros, comunicación con las familias, control de asistencia y gestión de actividades (matriculación, horarios, pagos y asistencia) y de viajes.

  Toda esta gestión la familia la recibe a través de un chat. Lo que lo hace fácil de cómodo de utilizar para los profesores y administradores, pero la gestión para las familias es menos intuitiva y visual.

  Ofrece tres tipos de tarifas. La tarifa gratuita no permite el control de asistencia ni mensajería con las familias.

- EasyManager[x]: ofrece una solución web para gestionar actividades de empresas y colegios. Una interfaz fácil de usar para el administrador, que permite publicar campamentos y actividades de forma sencilla. Desde la misma plataforma el administrador gestiona la contabilidad y los cobros.

Sin embargo, no existe un canal de comunicación directo entre la familia y el administrador. Además, solo permite dos tipos de usuarios y las actividades y los campamentos no están separados por cursos.

Se puede obtener a partir de 45€ al mes, pero la cuota básica es muy reducida: solamente acepta 2 administradores y un módulo a elegir entre actividades, extraescolares y viajes.

3.2 Benchmarking

El benchmarking[xi] es un proceso en el que el éxito de una empresa/aplicación/proyecto se compara con el de otras empresas/aplicaciones/proyectos similares para descubrir si existe una brecha en el rendimiento que pueda cerrarse mejorando el rendimiento de la aplicación. El estudio de otras empresas puede poner de relieve lo que se necesita para mejorar la eficacia de la aplicación y convertirse en un actor más importante del sector.

Esta sección presenta una evaluación comparativa de la competencia. Este estudio se ha realizado con el objetivo de descubrir los puntos fuertes de la competencia y detectar las necesidades del mercado que la aplicación que se desarrolla en este proyecto puede satisfacer.

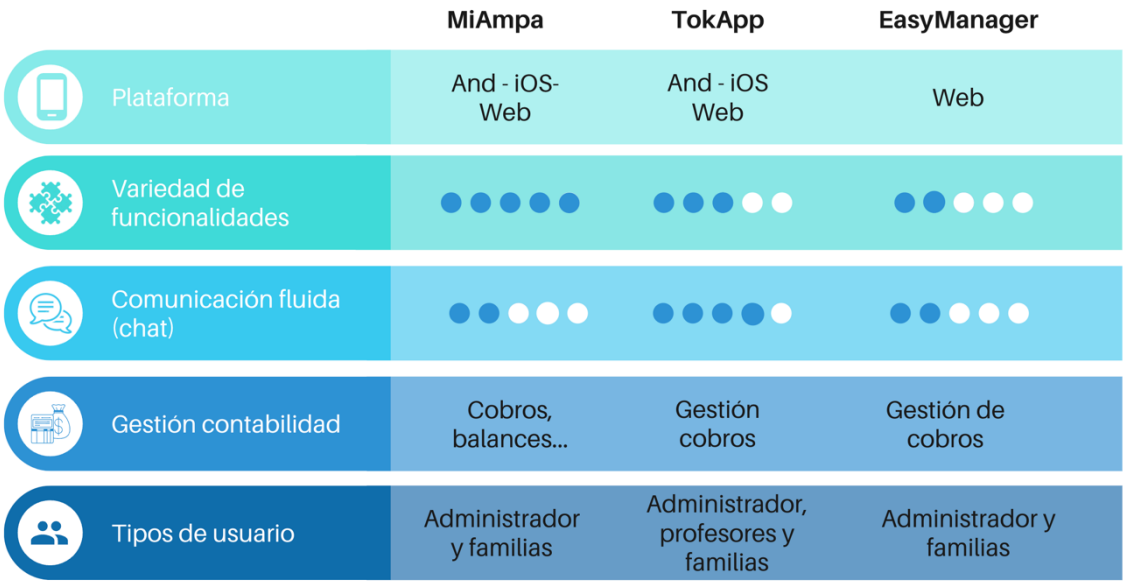


Figura 3. Benchmarking realizado en 3 aplicaciones móviles distintas. Cada fila es una característica clave de la aplicación que se valora.

Las aplicaciones existentes en el mercado ofrecen una mayor facilidad de acceso que la que se está desarrollando en este proyecto, ya que están disponibles para aplicaciones móviles y web y en este proyecto se está desarrollando una aplicación para dispositivos Android.

Otro punto débil de la aplicación de este proyecto es que no posee un módulo de contabilidad.

Sin embargo, todas las aplicaciones del mercado están muy enfocadas a gestionar actividades de los colegios, lo que hace que solo haya dos tipos de usuarios que puedan acceder a las apps: los administradores y las familias. TokApp es la única aplicación que ofrece otro tipo de usuarios, los profesores. Esta aplicación proporciona un chat grupal e individual profesor – familia, que hace que la comunicación sea fluida, aunque toda la gestión también se hace a través de cuestionarios y encuestas en ese chat. Lo que hace que se diferencie poco de plataformas como Whatssap.

Por tanto, los dos puntos fuertes de este proyecto son:

- Variedad de usuarios: la gestión se reparte entre los administradores y los monitores.
- Chat con padres: un chat que funciona para comunicarse, no para gestionar.

4 Ingeniería de concepción

En esta sección pasamos al siguiente paso del proceso, necesario en cualquier proyecto de software. En este paso el objetivo es responder qué sistema hay que construir, qué tiene que hacer el sistema y cómo tiene que hacerlo. Estas tres preguntas se responderán en las secciones "análisis de requisitos", "diseño" y "arquitectura", respectivamente.

4.1 Análisis de requisitos

Los requisitos son capacidades y condiciones a las que debe ajustarse el sistema. Uno de los principales retos de esta parte del trabajo es encontrar, comunicar y recordar lo que realmente se necesita, de una forma que hable claramente al cliente y a los miembros del equipo de desarrollo. Los requisitos pueden servir para definir qué sistema hay que construir[xii].

En esta línea, la famosa empresa estadounidense fundada por William Hewlett y David Packard, más conocida como HP, publicó un modelo para clasificar los atributos de calidad del software, tanto los requisitos funcionales como los no funcionales, conocido como modelo FURPS.

En ingeniería de software e ingeniería de sistemas, un requisito funcional[xiii] define una función de un sistema o su componente, donde una función se describe como una especificación de comportamiento entre entradas y salidas. Los requisitos funcionales pueden incluir cálculos, detalles técnicos, manipulación y procesamiento de datos y otras funcionalidades específicas que definen lo que se supone que debe lograr un sistema. El plan de implementación de este tipo de requisitos se detalla en el diseño del sistema. Por otro lado, un requisito no funcional es un requisito que especifica criterios que pueden utilizarse para juzgar el funcionamiento de un sistema, en lugar de comportamientos específicos.

FURPS[xiv] significa:

- Funcionalidad: especificación del comportamiento.
- Usabilidad (UX): especificación diseñada para garantizar que un producto sea fácil de usar. Por ejemplo, el sistema deberá poder manejarse con los dedos.
- Fiabilidad: especificaciones técnicas. Por ejemplo, el sistema estará disponible 24 horas al día, 7 días a la semana.
- Rendimiento: define lo bien que el sistema de software realiza determinadas funciones en condiciones específicas. Algunos ejemplos son la velocidad de respuesta del software, el rendimiento, el tiempo de ejecución y la capacidad de almacenamiento.
- Soportabilidad: tiene que ver con características como la mantenibilidad y escalabilidad de la solución. Por ejemplo, la interfaz de usuario se implementará en un navegador web con distintos idiomas.

La definición de estos requisitos es un punto esencial en el diseño y desarrollo de una aplicación. Haciendo un estudio de las necesidades de los distintos agentes que utilizarán

la aplicación (3.1.1), junto con el análisis de mercado realizado en el apartado 2.2, se han podido definir los requisitos funcionales, que se pueden consultar en el apartado 3.1.2.

4.1.1 Necesidades de los distintos agentes de la aplicación

Como se ha descrito en la introducción, la aplicación que se ha desarrollado en este proyecto de final de carrera está orientada a cubrir las necesidades de la gestión diaria de la Asociación Cultural Familiar Carena.

En el día a día de esta asociación están implicados varios agentes: los padres de los asociados, los monitores encargados de cada curso, el personal administrativo y los niños asociados. Se han realizado varias entrevistas a estas personas para poder detectar las necesidades que es necesario cubrir y que son susceptibles de ser solventadas por la aplicación que se va a desarrollar en este proyecto.

- Necesidades de los padres:
    - Asociar a un hijo
    - Dar de baja a un hijo
    - Obtener información de la programación:
        - Del plan semanal para cada hijo
        - De las actividades extras de los viernes y de los sábados
        - De los campamentos y convivencias
    - Inscribir a los hijos a cada actividad:
        - Indicar qué días de la semana va a ir cada hijo a la asociación
        - Indicar asistencia de los hijos a las actividades de los sábados
        - Inscripción a actividades que requieran un pago especial (campamentos, convivencias…)
    - Chat colectivo con los padres de cada curso
-
- Necesidades de los monitores encargados de cada curso:
    - Informar a los padres de:
        - Horario semanal
        - Actividades de los viernes
        - Actividades de los sábados
    - Conocer asistencia a cada actividad
    - Chat colectivo con los padres del curso
    - Petición de material para que el equipo administrativo lo compre

- Necesidades del personal administrativo:
    - Conocer los datos personales de los asociados y sus padres
    - Proporcionar una herramienta de pago de las actividades extras a todos los padres de los asociados
    - Conocer necesidades de compra de material con previsión
    - Publicar datos de campamentos

4.1.2 Definición de requisitos funcionales

En este apartado se enumeran los requisitos funcionales de la aplicación, que se han podido describir gracias a las necesidades detectadas en el apartado anterior.

1. Registro y autentificación de los usuarios
    a. Los usuarios se podrán registrar con un email y contraseña
    b. En el registro, el sistema pedirá distinta información según el tipo de usuario que sea (padre, monitor, personal administrativo)
    c. El sistema enviará un correo electrónico de confirmación al usuario después de que se haya registrado con éxito
    d. Los usuarios podrán iniciar sesión si ya se han registrado previamente
    e. Los usuarios tendrán la opción de recuperar la contraseña en caso de que no la recuerden
    f. La sesión se mantendrá encendida mientras no se cierre sesión

2. Funcionamiento dentro de la app
    a. El sistema detectará qué tipo de usuario ha iniciado sesión
    b. El sistema permitirá modificar los datos personales en todo momento
    c. El sistema mostrará las funciones disponibles para el tipo de usuario que ha iniciado sesión
    d. El sistema dispondrá de un menú lateral con las funcionalidades disponibles para el usuario que ha iniciado sesión
    e. El sistema mostrará la opción de chat en el menú lateral para los usuarios padres y los usuarios monitores
    f. Si el usuario es padre el sistema mostrará todos los espacios de trabajo de los hijos que tiene asociados
    g. El sistema informará de las actividades y horario de cada curso a los padres asociados
    h. Si el usuario es padre, el sistema permitirá marcar la asistencia a las actividades programadas de los hijos asociados
    i. Si el usuario es padre, el sistema permitirá la inscripción a las actividades extras
    j. Los usuarios monitores podrán publicar información sobre las actividades que han organizado para cada curso
    k. Los usuarios monitores podrán conocer la asistencia a las actividades que han organizado
    l. Los usuarios monitores podrán hacer peticiones de material
    m. El usuario administrador podrá publicar información sobre los campamentos y actividades extras
    n. El usuario administrador recibirá las peticiones de material que han hecho los monitores

4.1.3 Historias de usuario

COMO	QUIERO	PARA
Usuario no registrado	registrarme	Acceder a la aplicación
Usuario registrado	Iniciar sesión	
	Recuperar mi contraseña en caso de que no la recuerde	
	Cerrar sesión	Proteger mis datos
Padre	Asociar a uno o varios hijos	Que puedan disfrutar de las actividades del club
Padre de un asociado	Obtener información de las actividades programadas	Que puedan disfrutar de las actividades
	Inscribir a mis hijos a los campamentos y actividades extras	
	Conocer el nombre y datos de contacto de los monitores encargados	Poder comunicarme con ellos en caso de que sea necesario
	Comunicarme con los padres de los otros asociados	Comentar cualquier información o duda respecto a las actividades de la asociación
Monitor	Tener acceso a la lista de todos los asociados de un curso	Planificar mejor las actividades
	Conocer la asistencia a una actividad programada	
Monitor	Publicar información relevante sobre una actividad	Que los padres tengan los datos que necesitan
	Comunicarme con los padres de los niños asociados	Informar sobre las actividades
	Hacer peticiones de compra de material al equipo administrativo	Realizar las actividades planificadas
Personal administrativo	Publicar información sobre campamentos	Que los padres tengan los datos y puedan inscribir a los hijos

	Consultar las solicitudes de alta de los padres y sus hijos	Aceptar o rechazar la inscripción en función del número de plazas disponibles
	Conocer las necesidades de material	Comprar el material para las actividades planificadas con el dinero de la asociación

Tabla 1. Historias de usuario de la aplicación en formato "COMO - QUIERO - PARA".

4.2 Diseño visual

La diferencia entre un buen diseño de aplicación y uno deficiente suele estar en la calidad de su experiencia de usuario[xv]. Los tiempos de carga rápidos, la facilidad de uso y la satisfacción general del cliente durante una interacción deben ser partes integrales de su diseño. Un buen diseño de aplicación es claro, eficaz y estéticamente agradable.

El diseño de aplicaciones combina la interfaz de usuario[xvi] (UI) y la experiencia de usuario (UX). Mientras que la interfaz de usuario se refiere al estilo general de la aplicación (incluidos los colores, las fuentes y el aspecto general), la experiencia de usuario se centra en la funcionalidad y usabilidad reales.

Muchos usuarios abandonan una aplicación después de usarla por primera vez. Dado que los usuarios son muy exigentes con las aplicaciones que utilizan y abandonan rápidamente las que no les gustan, es esencial invertir tiempo y esfuerzo en crear una gran experiencia de usuario. Cuanto mejor sea el diseño, mayores serán las posibilidades de que el usuario se enganche a ella y, por tanto, siga utilizándola.

Dada su importancia, se ha dedicado un apartado específico a este punto en este proyecto final. Aquí se presenta el logotipo y los colores elegidos para la aplicación, así como el diseño de las pantallas.

4.2.1 Paleta de colores y tipografía

Como se ha comentado en la introducción, la aplicación que se está desarrollando en este proyecto está dirigida a todas las asociaciones juveniles que les pueda ser útil. Por tanto, no existirá unos colores predefinidos en la aplicación, sino que el tema se ajustará a los colores de la asociación que contrate la app.

Para hacer el prototipo, se ha escogido la Asociación Cultural y Familiar Carena[xvii], cuyo logotipo y colores primarios son los siguientes:

Figura 4. Logotipo de la Asociación Cultural Familiar Carena y código hexadecimal de los colores que forman el logo.

A partir de los colores primarios y utilizando la herramienta Color Palette Generator[xviii] de Canva se ha decidido que esta es la paleta de colores que se va a utilizar en la aplicación:

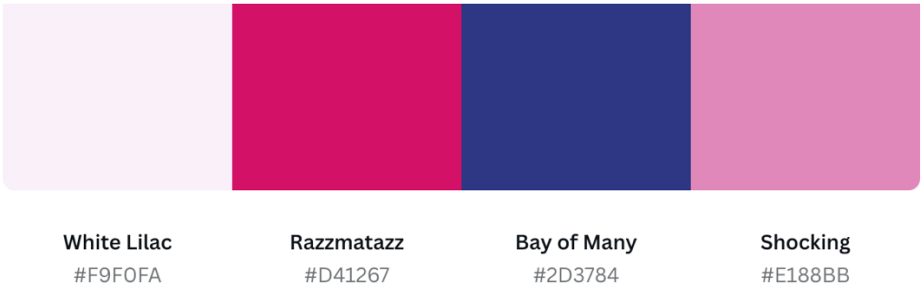


Figura 5. Paleta de colores de la aplicación. Se puede visualizar el color y el código hexadecimal del color.

La tipografía que se va a utilizar es Quicksand. Es una fuente sans-serif de Google[xix]. Se creó sobre una base de formas geométricas para dar impresión de amabilidad[xx]. Por sus extremos redondeados y su amplio espaciado es muy adecuada para una aplicación dirigida a adultos, pero cuyo objeto son los menores.

Los colores de la tipografía serán los siguientes:

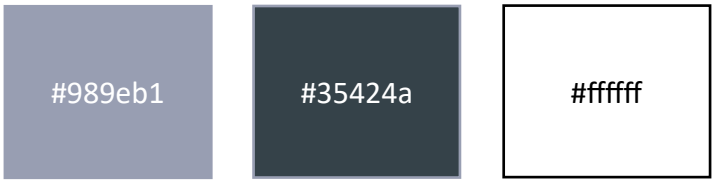


Figura 6. Color y código hexadecimal de la tipografía de la aplicación.

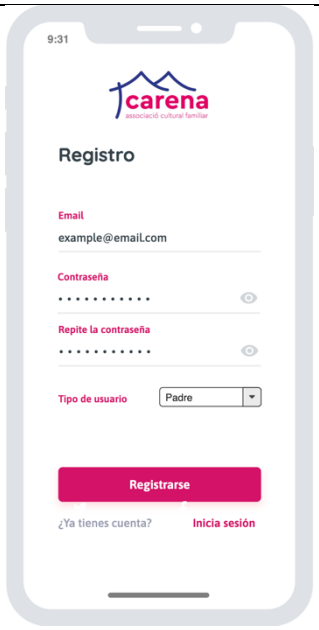
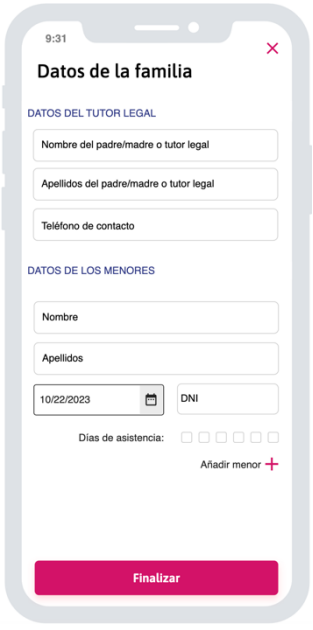
4.2.2 Mock ups

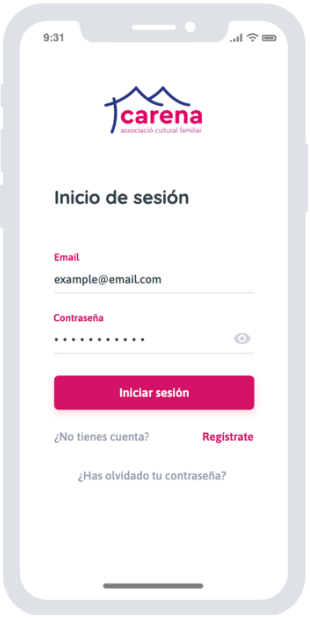
Con el logotipo diseñado y la paleta de colores de la aplicación elegida, el equipo de diseñadores gráficos pasaría a diseñar las pantallas de la aplicación en un editor gráfico.

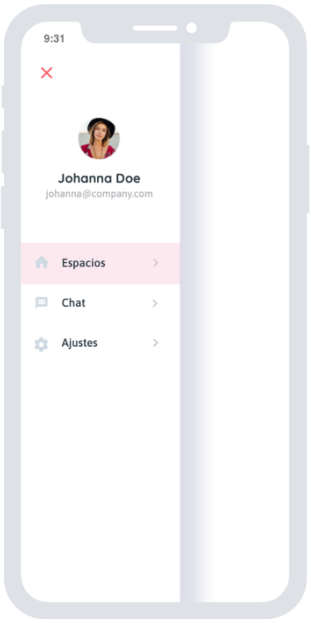
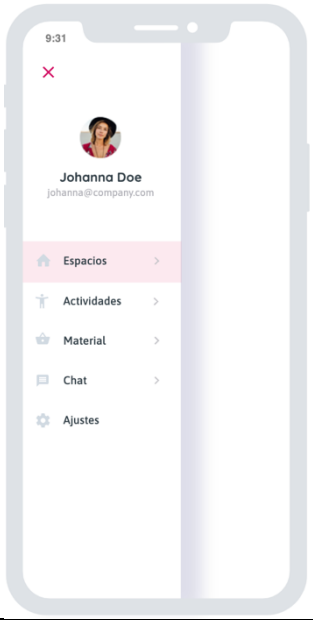
En este apartado se presenta el diseño gráfico de la aplicación hecho a partir de Moqups[xxi], una aplicación web que permite a los usuarios crear y colaborar en tiempo real en maquetas, wireframes, prototipos y diagramas.

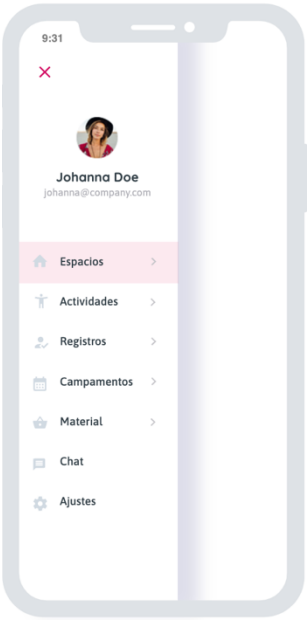
La siguiente tabla muestra una descripción y el diseño de cada pantalla de manera esquemática.

Identificador de pantalla	Tipo usuario	Mock-up	Descripción
Registro	Todos		Primera pantalla del registro, en la que se le pide al usuario el email y la contraseña.

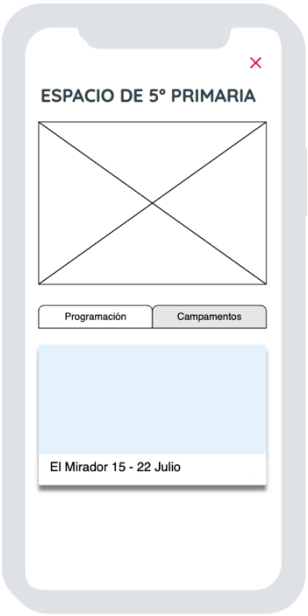
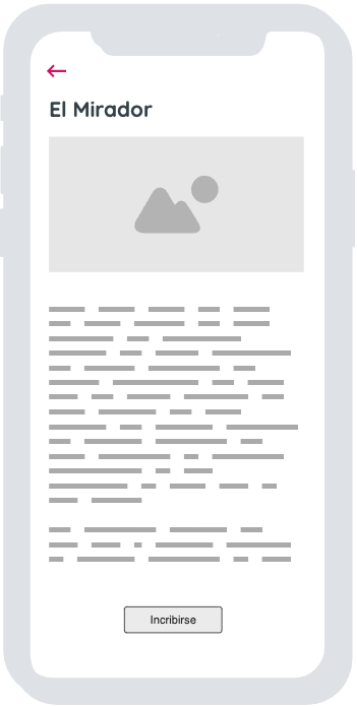
			En el caso de que el usuario ya tenga cuenta, con el botón “Inicia sesión” puede navegar a la pantalla de inicio de sesión. En esta pantalla también se pide al usuario que indique si se está registrando como monitor, o padre. Este dato es esencial para navegar a la siguiente pantalla, ya que a cada tipo de usuario se le pedirán unos datos distintos para finalizar el registro.
Configuración después del registro	Padre		Si el usuario ha indicado en el registro que es un usuario del tipo padre se le mostrará esta pantalla después de registrarse. No podrá acceder al resto de la aplicación hasta que rellene estos datos. Por tanto, si pulsa el botón de salir, se le cerrará la aplicación. Si inicia sesión, se le mostrará esta pantalla también. Una vez registrado, tendrá que esperar a que el administrador acepte su inscripción para poder acceder a la aplicación.

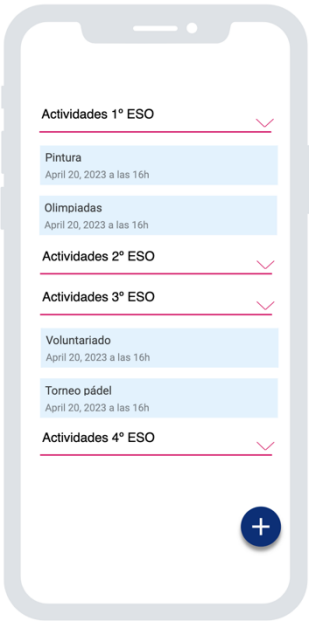
	Monitor		Pantalla que se muestra después del registro de un monitor. Aquí indicará sus datos personales y seleccionará los cursos de los cuales es monitor. Es esencial que el usuario rellene estos datos para poder acceder a la aplicación. Si no los rellena, cuando inicie sesión siempre verá esta pantalla. Una vez registrado, tendrá que esperar a que el administrador acepte su inscripción para poder acceder a la aplicación.
Inicio de sesión	Todos		Inicio de sesión común para todos los usuarios. Los campos necesarios para el inicio de sesión son el correo electrónico y la contraseña. Esta es la primera pantalla que el usuario ve al acceder a la aplicación. Permite la navegación a la pantalla de registro a través del botón “Regístrate”. En caso de que el usuario haya olvidado la contraseña, podrá restaurarla clicando sobre el botón “¿Has

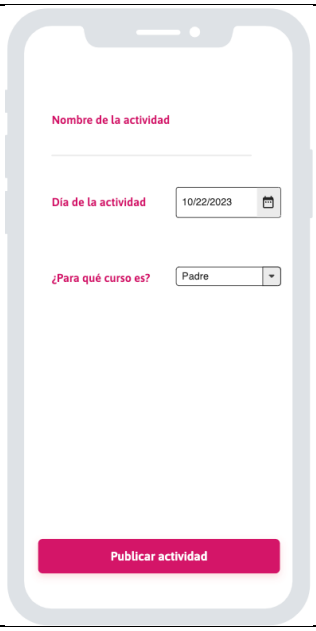
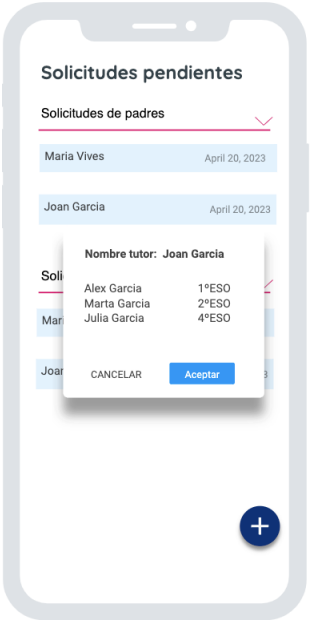
			olvidado tu contraseña?
Menú lateral	Padre		Menú lateral del usuario padre, al que se puede acceder desde cualquiera de las pantallas del menú a través de un movimiento lateral. Las opciones del menú para los usuarios padres son Espacios, Chat y ajustes.
	Monitor		Menú lateral del usuario monitor, al que se puede acceder desde cualquiera de las pantallas del menú a través de un movimiento lateral. Las opciones del menú lateral para los monitores son: Espacios, Actividades, Material, Chat y Ajustes.

	Administrador		Menú lateral del usuario administrador, al que se puede acceder desde cualquiera de las pantallas del menú a través de un movimiento lateral. Los administradores pueden acceder a todas las opciones de los usuarios anteriores, además de Registros y Campamentos.
Espacios	Todos		En esta pantalla el usuario visualizará todos los cursos que tiene asignados. Los padres visualizarán los cursos de los hijos asociados, los monitores los cursos de los que son encargados y el personal administrativo puede acceder a todos los cursos.

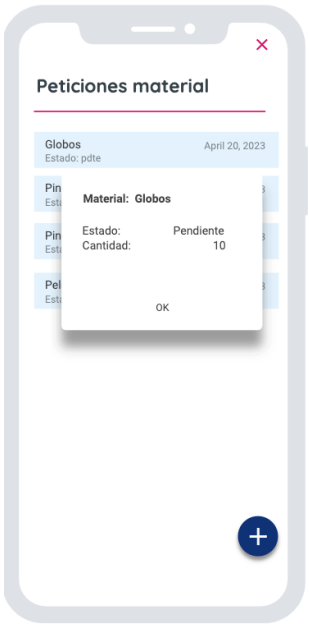
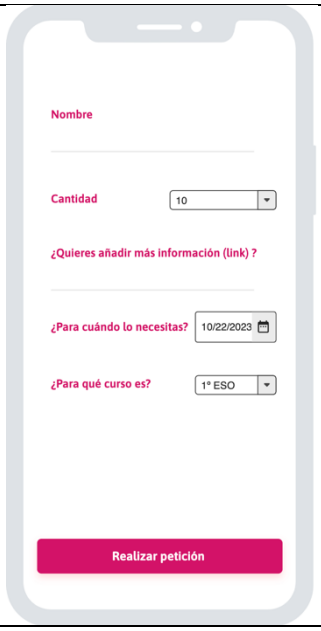
Curso - Vista programación	Todos		A esta pantalla se accede al clicar un curso de la lista de la opción del menú “Espacios”. Todos los usuarios visualizarán los mismos elementos. Sin embargo, los monitores y los administradores tendrán permisos de edición de los apartados. Cada curso tiene unos monitores encargados. Esta información aparecerá en la parte superior de la pantalla. Seguidamente, el horario semanal del curso en cuestión. En el apartado “Programación” aparecerán las actividades extras programadas para ese curso y los padres podrán marcar la asistencia. Esta información es importante porque ya se ha visto en el apartado 4.1.1 que es una de las necesidades de los padres y los monitores. Si un padre ya ha marcado que su hijo asistirá a esa

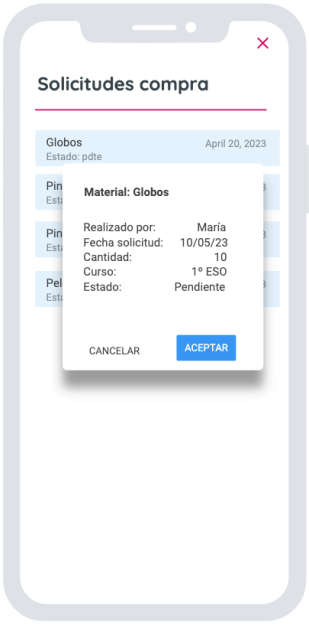
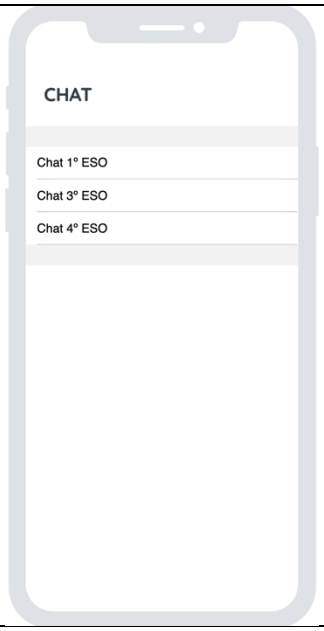
			actividad, al acceder a esta pantalla visualizará el checkBox de esa actividad marcado.
Curso – Vista campamentos	Todos		Misma pantalla que la captura anterior, pero habiendo clicado sobre la opción “Campamentos” de las tabs. En este apartado se visualizarán los campamentos de verano y convivencias de fin de semana programadas.
Curso – Detalle campamento	Todos		Al clicar sobre un campamento de la vista anterior se abrirá esta pantalla en la que se encontrará información de las actividades. Los padres visualizarán un botón para inscribir al hijo de ese curso en el campamento. Los administradores visualizarán un botón que al ser clicado podrán consultar el listado de inscritos a ese campamento. Los monitores no visualizarán ningún botón, solamente la

			información relativa al campamento.
Listado inscritos campamentos	Administrador		A esta pantalla se accede cuando el usuario administrador clica sobre el botón “Consultar inscripciones” de la descripción del campamento. El administrador podrá consultar el listado de asociados inscritos a un campamento determinado.
Listado de actividades	Monitores & Administrador		Listado de actividades por curso. Los administradores tendrán acceso a todos los cursos. Los monitores solamente visualizan las actividades de los cursos que tienen bajo su cargo. Al clicar sobre una actividad se accede al listado de inscritos. Al clicar sobre el botón flotante se navega al formulario para crear una actividad.
Formulario publicar actividad	Monitores & Administrador		En esta pantalla se piden los datos necesarios para crear una actividad. Después de publicarla, se navegará automáticamente al

			listado de actividades y se verá la nueva actividad en el curso correspondiente.
Registros pendientes	Administrador		El administrador tiene que aceptar cualquier registro que se haya hecho en la aplicación para que los usuarios puedan acceder. A esta pantalla se accede desde la opción del menú “Registros” del administrador. En esta pantalla se encontrarán todas las solicitudes de registro pendientes. El administrador podrá aceptarlas al clicar al botón “aceptar” del diálogo que se abre al clicar sobre un elemento de la lista.

Listado campamentos	Administrador		Listado de campamentos de todos los cursos. A esta pantalla se accede desde la opción “Campamentos” del menú del administrador. Al clicar sobre un elemento se accede al detalle del campamento (ya se ha comentado este diseño anteriormente). Al clicar sobre el botón flotante se accede al formulario para publicar un campamento.
Formulario publicar campamento	Administrador		Formulario del administrador para publicar un campamento. Al publicarla se navega automáticamente al listado de campamentos, en el que se podrá ver el nuevo campamento creado.

Listado peticiones material	Monitor		Los monitores, al acceder a la opción “Material” del menú visualizarán esta pantalla. En ella encontrarán un listado de las peticiones de compra de material que han hecho y el estado. Si se clica sobre un elemento de la lista se abre un diálogo con información detalla de la petición. Al clicar sobre el botón flotante se navega al formulario de petición de material.
Formulario petición material	Monitores		Formulario para hacer una solicitud de compra de material a los administradores. Al clicar sobre el botón “Realizar petición” se navegará automáticamente al listado, en el que se podrá ver la nueva solicitud.

Solicitudes compra material	Administrador		El administrador visualizará esta pantalla al acceder a la opción “Material” del menú lateral. Verá un listado de solicitudes de material y el estado. Al clicar sobre un elemento se abrirá un diálogo con la información de esa petición. El administrador podrá aceptar la petición o rechazarla.
Chat	Todos		Funcionalidad disponible para todos los usuarios. Al acceder a la pantalla de chats, encontrarán un listado de los grupos a los que tienen acceso.

Ajustes	Todos		Todos los usuarios podrán modificar sus datos personales desde la opción "Ajustes" del menú. Además, los padres podrán dar de baja / alta a sus hijos. Los monitores podrán asignarse / desasignarse cursos.

Tabla 2. Diseño de todas las pantallas de la aplicación para cada usuario y su descripción.

4.3 Arquitectura

En este apartado se va a describir la arquitectura, un punto esencial en cualquier proyecto de software, ya que desempeña un papel crucial en el éxito y la sostenibilidad de un proyecto[xxii].

4.3.1 Base de datos

Hay varias bases de datos que se pueden utilizar en una aplicación móvil, y la elección depende de varios factores, como la complejidad de los datos, los requisitos de rendimiento y la escalabilidad.

Algunas de las bases de datos que se pueden utilizar son[xxiii] [xxiv]:

- MySQL[xxv]: es una de las bases de datos SQL más conocidas del mercado. Es muy adecuada para aplicaciones con grandes cantidades de datos y usuarios simúlatenos. Ofrece una gran escalabilidad vertical, pero no siempre es adecuado para aplicaciones que requieren flexibilidad y movilidad cuando se trata de la cantidad y el tipo de datos que se almacenan. Google Cloud SQL, Amazon RDS y Microsoft Azure ofrecen servicios de bases de datos que incluyen soporte para este tipo de base de datos.

- SQLite[xxvi]: es un motor de base de datos SQL ligero, autónomo y sin servidor que almacena datos directamente en un único archivo. Es ampliamente utilizado en el desarrollo de aplicaciones móviles debido a su simplicidad, pequeño tamaño y portabilidad. SQLite es adecuado para aplicaciones de menor escala que requieren una base de datos local.

- PostgreSQL[xxvii]: también es una base de datos SQL, por lo que maneja el mismo modelo de filas y columnas que otros sistemas de gestión de bases de datos relacionales. A diferencia de otras bases de datos SQL, PostgreSQL también es compatible con JSON, por lo que admite más tipos de datos además de los estructurados, como datos geoespaciales. PostgreSQL es idóneo para aplicaciones móviles complejas que requieren sólidas capacidades de gestión de datos.

- Firebase Realtime Database[xxviii]: es una base de datos en la nube que ofrece Firebase, una plataforma de Google. Utiliza un modelo de datos basado en árboles JSON. Los datos se organizan en una estructura de árbol de nodos clave-valor. Ofrece sincronización en tiempo real, pero una escalabilidad limitada. También tiene limitación en el tipo de consultas que se pueden hacer.

- Cloud Firestore[xxix]: es la otra base de datos en la nube que ofrece Firebase. Utiliza un modelo de datos basado en documentos y colecciones, por tanto, es una base de datos NoSQL. Los datos se organizan en documentos que contienen campos clave-valor, y los documentos se agrupan en colecciones. Ofrece sincronización en tiempo real, similar a Realtime Database. Sin embargo, Cloud Firestore permite consultas más avanzadas y granularidad en la sincronización de datos. Ofrece mayor escalabilidad que Realtime Database, lo que significa que puede manejar mejor grandes conjuntos de datos y cargas de escritura intensivas. Permite consultas más potentes y flexibles, incluyendo filtrado, ordenación y combinación de múltiples campos.

- MongoDB[xxx]: es una popular base de datos NoSQL orientada a documentos conocida por su flexibilidad y escalabilidad. Almacena datos en documentos flexibles de tipo JSON y admite esquemas dinámicos. MongoDB es idónea para aplicaciones móviles que requieren una gran escalabilidad, análisis en tiempo real y gestión de datos no estructurados o semiestructurados.

- Couchbase[xxxi]: es otra base de datos NoSQL y basado en la nube. Combina la flexibilidad del modelado de datos JSON con la escalabilidad y el rendimiento de una arquitectura distribuida. Couchbase es también una excelente opción para aplicaciones móviles que requieren una base de datos escalable horizontalmente y susceptible de crecer y cambiar con el tiempo.

Estas son algunas de las posibles bases de datos que se pueden utilizar. De entre las mencionadas anteriormente, SQLite claramente se descarta porque almacena los datos en un fichero local y la aplicación que se está desarrollando en este proyecto requiere una base de datos compartida entre todos los usuarios.

Para el desarrollo de este proyecto, Firestore es una opción ideal por las siguientes razones[xxxii]:

- Firestore es parte de Firebase, una plataforma completa de desarrollo móvil proporcionada por Google. Firebase proporciona bibliotecas y SDK específicos para Android, facilitando la integración con Android Studio, lo que simplifica mucho configuración y la consulta de la base de datos a través de la aplicación.

- Ofrece sincronización en tiempo real, lo que significa que los datos se actualizan automáticamente en todos los clientes conectados cuando hay cambios. Esto es fundamental para proporcionar una experiencia de usuario en tiempo real y mantener los datos de la aplicación actualizados.

- Admite la persistencia de datos en el dispositivo, lo que permite a la aplicación funcionar offline. Los datos se sincronizarán automáticamente cuando la conexión esté disponible nuevamente. Esto es crucial para proporcionar una experiencia de usuario sin problemas, incluso en condiciones de conectividad intermitente.

- Proporciona un sistema de seguridad integrado con reglas de seguridad que se pueden definir y gestionar fácilmente. Se puede controlar el acceso a los datos y garantizar que solo los usuarios autorizados puedan realizar operaciones.

- Firebase ofrece herramientas de desarrollo y monitoreo que facilitan el seguimiento del rendimiento de la aplicación, la depuración y la mejora continua. Se puede acceder a la base de datos a través de la web de Firebase.

- Firebase ofrece una gran variedad de extensiones fáciles de integrar en la aplicación.

Para el tipo de aplicación que se está desarrollando, Realtime Database también es una buena opción. Sin embargo, se ha decido utilizar Firestore por la familiaridad que se tiene con esta base de datos porque ya se ha trabajado con ella en anteriores proyectos.

Firestore funciona con el plan “pago por uso”, todas las funcionalidades son gratuitas hasta que se supera un límite de uso o de almacenamiento. Además, para las consultas complejas es necesario crear índices. Según la cantidad de índices creados, Firestore también puede cobrar por ellos.

Por otro lado, Firestore tiene límites en la cantidad de operaciones de escritura y lectura por segundo que puede manejar. Superar estos límites puede resultar en operaciones fallidas o en la necesidad de escalar el plan de facturación. También tiene restricciones en la estructura de los documentos, cada documento no puede exceder los 1 MB.

Para la aplicación de este proyecto no se han superado en ningún caso los límites de uso y de almacenamiento, por eso también ha sido una buena opción utilizar esta base de datos.

4.3.1.1 Estructura de la base de datos

Dentro del conjunto de herramientas de Firebase, FirebaseAuth[xxxiii] proporciona servicios de autentificación para aplicaciones móviles y web. Permite agregar un sistema de autentificación robusta a cualquier tipo de aplicación, sin tener que gestionar complejidades como la gestión de usuarios, la verificación de identidad y la seguridad de contraseñas.

Se ha decidido utilizar FirebaseAuth[xxxiv] para la gestión de usuarios de este proyecto porque proporciona varios proveedores y métodos de autentificación, lo que brinda una gran flexibilidad en cómo los usuarios pueden registrarse e iniciar sesión. En la aplicación que se va a desarrollar en este proyecto solamente se va a permitir la autentificación a través de correo y contraseña, pero utilizar FirebaseAuth permitirá ampliar los métodos de autentificación sin mucho esfuerzo.

Otra de las razones por las que se utiliza FirebaseAuth es porque proporciona funciones de verificación de cuenta y restablecimiento de contraseña.

FirebaseAuth registra a los usuarios a través de su correo electrónico y les asocia un identificador UID. Este identificador se utilizará en las otras colecciones de la base de datos para identificar a los usuarios. En FirebaseAuth también se almacenará la foto de perfil y el nombre del usuario.

El resto de datos de la aplicación se almacenan en Firestore, que utiliza un modelo de datos basado en documentos y colecciones[xxxv]:

Un documento contiene un conjunto de pares clave-valor, donde la clave es el nombre del campo y el valor puede ser de varios tipos, incluyendo maps, números, booleanos, objetos anidados, arrays y strings.

Una colección es un grupo de documentos. Se pueden organizar documentos en colecciones para facilitar la consulta y la recuperación de datos relacionados.

Las colecciones de Firestore necesarias para el correcto funcionamiento de la aplicación son las siguientes:

- Usuarios: el sistema mostrará unas funcionalidades u otras según el tipo de usuario que esté utilizando la app. Por ejemplo, un usuario monitor podrá editar información acerca de una actividad a través de la app, pero un usuario padre solo podrá consultar esa información. Para ello ha sido necesario crear esta colección que se consultará nada más iniciar sesión. Esta colección contiene está formada por documentos que se identifican a través del UID. Cada documento tiene un atributo que indica el tipo de usuario. Este atributo puede tomar los valores monitor, progenitor o administrativo.

- Progenitores: colección que contiene todos los padres de los menores asociados. Los padres pueden tener varios hijos asociados, por tanto, esta colección

almacenará en el atributo “listado hijos” el UID de los menores. También guardará información del nombre del padre.

- Asociados: colección que almacena todos los menores asociados. Cada documento de la colección se identifica con el UID de un usuario. Los atributos de cada documento son: nombre, año nacimiento, UID del padre, id del curso al que pertenece y días de asistencia.

- Monitores: los monitores pueden tener bajo su cargo uno o varios cursos. Cada documento de la colección se identifica con el UID de un usuario. Los atributos de cada documento son: nombre y listado de cursos, que es un array de ids de la colección “Cursos”.

- Cursos: esta colección almacena los cursos. Se identifica por el ID y sus atributos son: nombre del curso, horario, listado de campamentos (que es un array de ids de la colección “Campamentos”), listado de monitores (que es un array de UID de usuarios) y listado de actividades (que es un array de ids de la colección “Programación”).

- Personal administrativo: colección reservada al personal administrativo de la asociación. Se identifica cada documento a partir del UID del usuario. Tiene como único atributo el nombre.

- Campamentos: almacena información sobre los campamentos de verano y los que se realizan durante el curso. Sus atributos son: nombre, fecha, foto, descripción e inscripciones, que es un array de UID de asociados.

- Programación: almacena información sobre las actividades extras que se organizan para el fin de semana. Sus atributos son: nombre, fecha, descripción e inscripciones, que es un array de UID de asociados.

Se ha considerado guardar en una misma colección los campamentos y las actividades que se almacenan en la colección “Programación”, ya que tienen prácticamente los mismos atributos. Sin embargo, se ha considerado oportuno separar esta información en dos colecciones distintas para posibles futuras extensiones en los atributos de la base de datos y por eficiencia por la parte del código.
Todas estas colecciones se van a transformar en clases a nivel de Java. Cada clase contendrá los mismos atributos que los que existen en Firestore.

También se ha utilizado Firebase Storage, un servicio de almacenamiento en la nube proporcionado por Firebase. Firebase Storage se utiliza para almacenar y recuperar archivos, como imágenes, videos, archivos de audio y otros tipos de archivos binarios en la nube. En este proyecto se ha utilizado para almacenar las imágenes de perfil de los usuarios, las imágenes de portada de los campamentos y las fotos que los monitores suben en los formularios de solicitud de material.

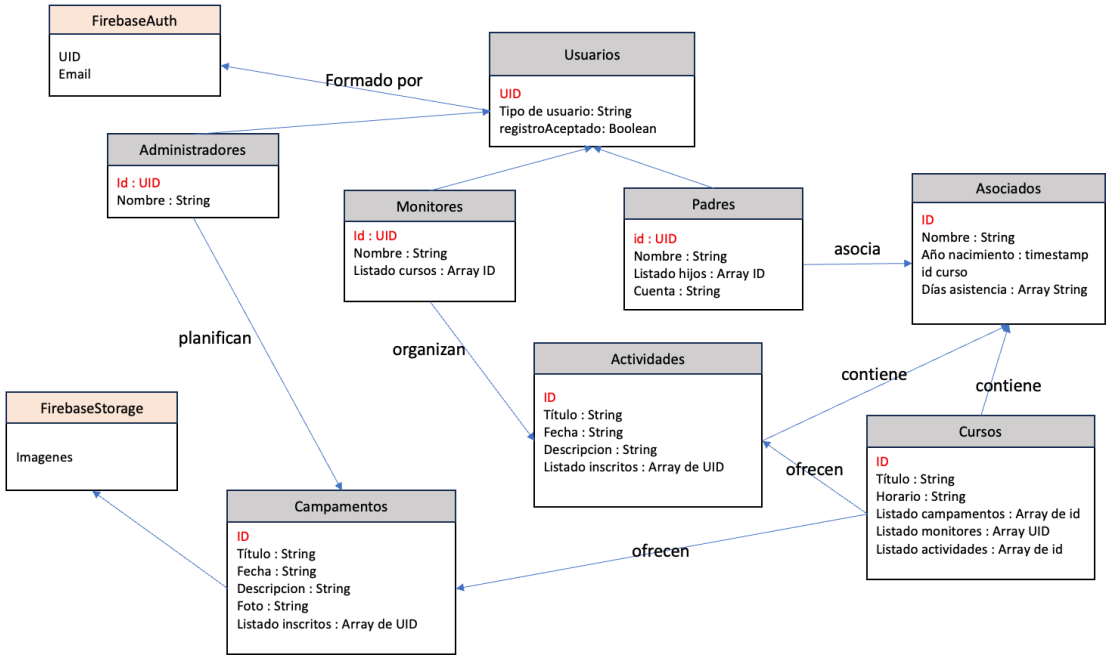


Figura 7. Esquema de las colecciones que forman la base de datos y sus relaciones.

4.3.2 Diseño de software

El diseño de las clases y activities se ha hecho teniendo en cuenta el modelo de capas[xxxvi] y Clean Architecture[xxxvii]. De esta forma, se estructura y organiza el código de manera clara, buscando la modularidad, la reutilización y la mantenibilidad.

Cada capa en el modelo de capas tiene un propósito específico y tiene unas responsabilidades determinadas. Siguiendo los principios de Clean Architecture[xxxviii], las capas se organizan de manera concéntrica, donde las capas externas dependen de las capas internas, pero las internas no dependen de las capas externas.

La capa de presentación se encarga de presentar la información al usuario y de recibir las interacciones del usuario. En esta capa se utilizará el patrón Modelo-Vista-Presentador. La vista está formada por todas las activities y ficheros xml de Android Studio que interactúan con el usuario. El presenter recoge las acciones del usuario y almacena esos datos en modelos. Es el encargado, también, de comunicarse con los usecases en el caso de que el sistema requiera realizar una consulta a la base de datos.

La capa de dominio contiene toda la lógica de negocio. Esta capa procesa la entrada del usuario desde la capa de presentación, realiza las operaciones de negocio y coordina la interacción entre las diferentes partes del sistema. En el caso de que se requiera una consulta a la base de datos, el presenter se comunicará de manera directa con el usecase. Cada operación o tarea que realiza la aplicación estará definido en un usecase, por tanto, habrá un usecase para iniciar sesión, otro para registrarse, otro para consultar los cursos asociados a cada usuario, etc.

Cuando se haga una llamada a un usecase, se creará un hilo secundario para no bloquear la aplicación. Cuando la consulta a la base de datos finalice, el usecase lo comunicará al presenter a través de eventos o de callbacks, según sea necesario en cada consulta.

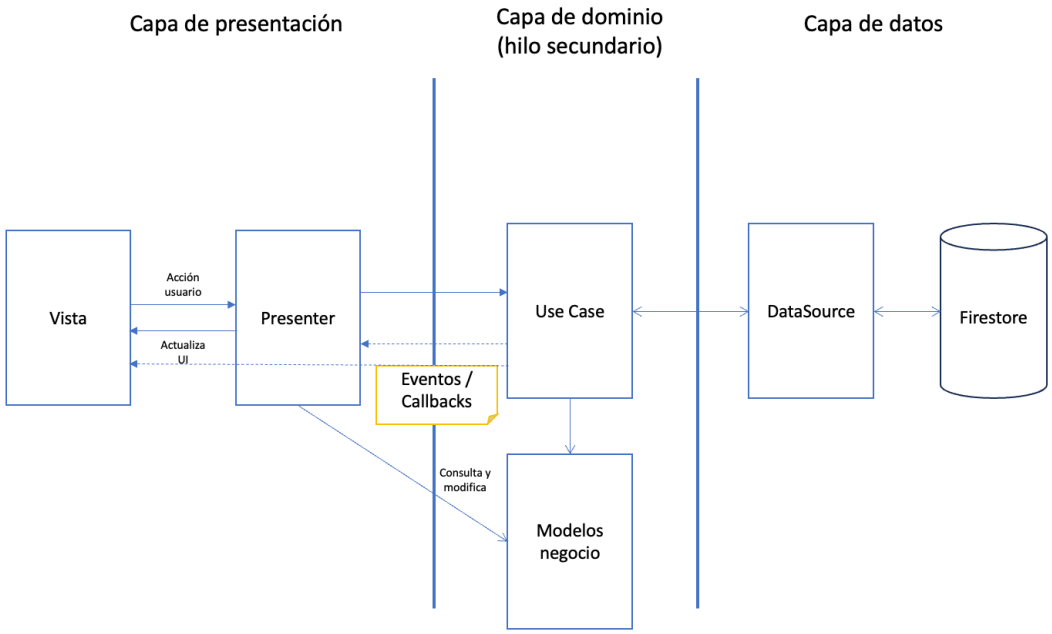


Figura 8. Esquema de la arquitectura de la aplicación. En él se pueden ver todas las clases que formaran cada funcionalidad de la app.

4.3.2.1 Callbacks vs Eventos

Un callback es una función que se pasa como argumento a otra función y que luego se invoca dentro de la función externa para completar algún tipo de rutina o acción[xxxix].

Dado que no es necesario utilizar ninguna librería externa, las callbacks[xl] son más fáciles de implementar que los eventos y proporcionan una estructura de código más lineal, que en algunos casos puede ser útil. Es decir, la respuesta de la llamada asíncrona se ejecuta en el mismo sitio desde el que se ha realizado la llamada. En este sentido se habla de linealidad.

Los callbacks poseen un defecto importante y es que la gestión puede ser complicada si se tienen múltiples llamadas asíncronas anidadas, ya que resulta en un código muy difícil de leer y mantener y es complicado el manejo de errores. Este problema es conocido como “callback hell”.

Un evento[xli], en cambio, es una señal que comunica a la aplicación que ha sucedido algo. Es necesario que haya un emisor del evento y que haya uno o varios receptores que estén escuchando ese evento. A diferencia de los callbacks, para implementar un evento es necesario utilizar alguna librería externa.

El uso de eventos puede conducir a una arquitectura más desacoplada y modular, ya que no es necesario seguir una estructura lineal. Es decir, el evento se emitirá justo después de la respuesta del servicio, pero los receptores del evento puede ser cualquier parte de la aplicación. Esto es muy útil cuando una misma llamada a servicio puede actualizar datos de distintos puntos de la aplicación.

Por el contrario, implementar un sistema de eventos puede ser más complicado que simplemente usar callbacks. También es más difícil rastrear el flujo, ya que las respuestas pueden provenir de varias fuentes.

En este proyecto se ha tomado la decisión de utilizar callbacks en llamadas lineales en las que la respuesta del servicio solo se utiliza en una parte de la aplicación y en la que no se encadenan múltiples llamadas a servicio. Un ejemplo es el inicio de sesión y el registro. En cambio, en las partes de código en las que es necesaria múltiples llamadas encadenadas o esa información que se carga se utiliza en distintas partes de la app, se han utilizado eventos.

5 Ingeniería de detalle

Con el diseño hecho, ya es posible empezar a desarrollar el código de la aplicación. Los programadores intentan implementar las pantallas que han diseñado los diseñadores gráficos. Sin embargo, a veces es necesario introducir pequeñas modificaciones porque algunos puntos a nivel técnico no se pueden realizar.

En el Anexo se pueden consultar las credenciales de los usuarios existentes hasta la fecha para poder acceder a la aplicación si se desea.

5.1 Tecnologías utilizadas

En la realización de este trabajo de final de grado se han empleado diversas tecnologías que han desempeñado un papel fundamental en el desarrollo y éxito del proyecto. La plataforma de desarrollo móvil Android Studio ha servido como el entorno de desarrollo principal, proporcionando un conjunto robusto de herramientas para la creación y depuración de aplicaciones Android. Gracias a su interfaz intuitiva y su integración con el lenguaje de programación Java, se ha logrado una eficiencia significativa en la codificación y la implementación de características clave.

Para la gestión de la base de datos, se ha optado por Firestore, una base de datos NoSQL ofrecida por Firebase. Firestore ha permitido almacenar y recuperar datos de manera eficiente, siguiendo un modelo de datos flexible y escalable. La integración fluida entre Android Studio y Firestore ha facilitado la sincronización de datos en tiempo real, garantizando una experiencia de usuario dinámica y actualizada. En el apartado 4.3.1 se explica con más detalle por qué se ha utilizado esta base de datos y las colecciones que la conforman para este proyecto.

En el proceso de diseño de la interfaz de usuario se ha utilizado la herramienta Moqups para crear prototipos visuales y esquematizar las pantallas de la aplicación. Moqups ha proporcionado una plataforma para el diseño, permitiendo la creación rápida de wireframes y la visualización de la estructura de la aplicación antes de la implementación. En el apartado 4.2.2 presenta el diseño de las pantallas realizado con Moqups.

5.2 Implementación y resultado

En este apartado se detalla los pasos seguidos para implementar cada una de las funcionalidades de la aplicación.

En el proyecto de Android Studio, cada una de las funcionalidades está contenida en un package. Cada uno de los packages contiene la arquitectura descrita en el apartado *4.3.2.*

Para mayor flexibilidad y consistencia en el código, se ha creado una BaseActivity y BaseFragment de las que heredan todas las activities y fragments de la aplicación. Estas clases actúan como plantillas que encapsulan funcionalidades comunes. En estas clases base se han añadido los siguientes métodos:

- *getLayoutRes():* se sobreescribe en todas las clases para devolver el layout. La clase base, y no las clases hijas, es la encargada de recoger el layout e inflar la vista.

- *configureToolbar():* se sobreescribe este método en aquellas clases hija en las que se quiera esconder el botón de retroceso o de cerrar, o cambiar el título de la toolbar.

- *bindViews():* este método se utiliza para encontrar una vista en la jerarquía de la interfaz de usuario (UI) de una actividad o un fragmento. Aquí se encontrarán todos los *findViewByID* de los componentes de la UI.

- *configureUI():* se llama a este método justo después del método *bindViews()*. Tal y como indica el nombre, configura la vista. Por ejemplo, aquí se encontrará el código para setear el título de una pantalla o configurar un recyclerView o añadir un listener a un botón. Para mayor encapsulamiento, este método llamará a su vez a otros métodos que configuren distintas partes de la vista.

- *showProgress() / hideProgress():* los xml de las clase base contienen un componente *ProgressBar* que conforma el loader de la aplicación. Se llamará a estos métodos cuando se requiera mostrar o esconder el loader.

Otro punto importe en varios de las endpoints que se han implementado para realizar las consultas a la base de datos es la utilización de índices. En Firebase Firestore, un índice es una estructura que ayuda a acelerar las consultas en la base de datos. Firestore utiliza índices para realizar búsquedas eficientes y ordenadas de los documentos que cumplen con ciertos criterios de consulta. Si no se habilitan estos índices no es posible realizar alguna de las consultas, Firestore las bloquea para evitar operaciones costosas y mejorar la eficiencia en la recuperación de datos. Sin embargo, cuando Firestore bloquea una consulta también proporciona el enlace necesario para habilitar ese índice de manera automática. Esto se ha utilizado, por ejemplo, en el endpoint para cargar los mensajes o las peticiones de material.

5.2.1 Autentificación

Dos actividades con un solo xml conforman la pantalla de registro y de inicio de sesión de la aplicación. Estas pantallas están formadas por EditTexts, que recogen el correo electrónico y la contraseña del usuario, y un botón, entre otros componentes.

En el caso de que el usuario se encuentre en la pantalla de registro, cuando clique el botón de registro se comprueba si la contraseña cumple con los requisitos necesarios de seguridad. En caso afirmativo, se recogen los valores de tipo de usuario, email y contraseña y se inicia una cascada de llamadas, pasando por el presenter y por el usecase, creando los modelos necesarios, hasta utilizar la API proporcionada por FirebaseAuth para crear un usuario.

Esta API comprueba de manera automática si ya existe un usuario en la base de datos con el mismo email. En caso negativo, registra el usuario en la base de datos y lo guarda en la colección “Usuarios” de Firestore. Una vez creado, se navegará a la pantalla de configuración correspondiente para continuar con el registro.

En la pantalla de configuración se piden los datos personales al usuario: nombre, datos de los hijos, en el caso de que sea padre; cursos de los que se es encargado, en el caso de los monitores. Cuando el usuario clique el botón “Finalizar” de esta pantalla, se recogerán los datos y se guardarán en la colección ““Monitores” o “Padres” y “Asociados”, según corresponda.

En el caso de los monitores, además, se guardará en la colección “Cursos” el monitor, para que quede almacenado en la base de datos el curso del que es encargado ese monitor.

Una vez se hayan almacenado todos estos datos en la base de datos, le aparecerá al usuario el mensaje “Estamos validando tu solicitud, en breves recibirás un mail con la confirmación.” y no podrá acceder a la aplicación hasta que alguno de los administradores valide la solicitud de registro.

Registro	
EndPoint	**Descripción**
`createUserWithEmailAndPassword (String email, String pwd)`	Se utiliza la API de FirebaseAuth para crear un usuario utilizando el servicio Authentication de Firebase. Son necesarios el email y la contraseña como parámetros.
`addUserToUserCollection(String UID, UserType tipoUsuario)`	Se crea un nuevo documento en la colección “Usuarios”. El id de este documento corresponde al UID del usuario registrado.
`saveParentInCollection(ParentModel)`	Guarda los datos personales del usuario padres en la colección “Padres”.
`saveChildrenInCollection(ChildrenModel)`	Guarda los datos personales de los asociados en la colección “Asociados”.
`saveMonitorInCollection(MonitorModel)`	Guarda los datos personales de los monitores en la colección “Monitores”.
`saveMonitorInCourseCollection(String uidMonitor)`	Guarda en el monitor en el curso/s de los que es encargado.

Tabla 3. Endpoints utilizados en la funcionalidad de registro y configuración de la aplicación.

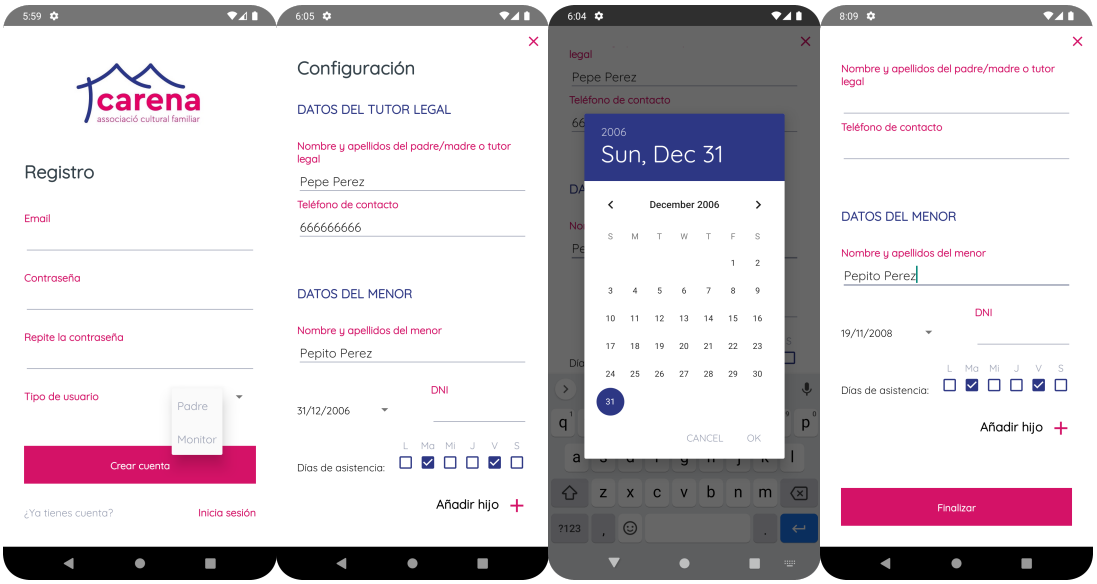


Figura 9. Capturas de pantalla reales de la funcionalidad de registro y configuración de la aplicación.

En el caso del inicio de sesión, se le pide al usuario que indique su correo electrónico y contraseña. Al clicar al botón, se utiliza la API de FirebaseAuth para comprobar si ese usuario existe en la base de datos y si ha introducido la contraseña correctamente. En caso afirmativo, se consulta la colección "Usuarios" para comprobar qué tipo de usuario ha iniciado sesión (Administrador, monitor, padre). Luego se consulta la colección "Administradores", "Monitores" o "Padres", según corresponda, para obtener los datos del usuario en cuestión.

Si el usuario es de tipo Padre, se hace otra llamada a la base de datos para consultar de la colección "Asociados" los datos de los hijos que ha inscrito a la asociación.

Inicio de sesión	
EndPoint	**Descripción**
`signInWithEmailAndPassword (String email, String pwd)`	Se utiliza la API de FirebaseAuth para autenticar al usuario. Son necesarios el email y la contraseña como parámetros.
`loadUserFromDBUseCase(String uid)`	Se consulta la colección "Usuarios" para comprobar qué tipo de usuario ha iniciado sesión.
`loadParentFromCollection(String uid)`	Se consulta la colección "Padres" para obtener los datos personales del usuario padre que ha iniciado sesión.
`loadChildrenFromCollection(String uid)`	Se consulta la colección "Asociados" para obtener todos los asociados cuyo idPadre coincida con el UID que se pasa como parámetro.
`loadMonitorFromCollection(String uid)`	Se consulta la colección "Monitores" para obtener los datos

	personales del usuario monitor que ha iniciado sesión.
`loadAdministradorFromColecction(String uid)`	Se consulta la colección "Administradores" para obtener los datos personales del usuario administrador que ha iniciado sesión.

Tabla 4. Endpoints utilizados para permitir al usuario iniciar sesión en la aplicación.

Figura 10. Capturas de la pantalla de inicio de sesión y de la pantalla después del inicio de sesión, en el caso de que la solicitud de registro del usuario esté pendiente.

5.2.2 Menú lateral

En el diseño de la interfaz de mi aplicación, el menú lateral desempeña un papel crucial al proporcionar una navegación intuitiva y eficiente. Este menú está compuesto principalmente por una actividad (Activity) central y un conjunto de fragmentos que representan las diversas opciones de navegación. La implementación del menú lateral se ha llevado a cabo utilizando componentes clave de Android, como AppBarConfiguration, DrawerLayout, NavigationView, NavController y NavigationUI.

La Activity central actúa como el contenedor principal, albergando el menú lateral y gestionando la transición entre los distintos fragmentos asociados a cada opción. DrawerLayout se ha empleado para crear el diseño deslizante del menú lateral, mientras que NavigationView proporciona la interfaz visual para las opciones de navegación. El componente NavController se encarga de la navegación entre fragmentos de manera coherente, asegurando una experiencia de usuario fluida y consistente.

La configuración del menú lateral se ha simplificado y optimizado mediante el uso de AppBarConfiguration, que define las opciones del menú y especifica la actividad principal a la que se debe regresar al seleccionar una opción. Además, NavigationUI facilita la

integración del menú lateral con el control de navegación, simplificando el manejo de las transiciones entre fragmentos.

En la activity principal se comprueba qué tipo de usuario ha iniciado sesión para mostrar / esconder algunas opciones del menú. Los usuarios padres solamente tienen habilitada la opción “Espacios”, “Chat” y “Ajustes”. Los usuarios monitores se les suma a las anteriores las opciones “Actividades” y “Material”. Los administradores, además, pueden acceder a “Solicitudes pendientes” y “Campamentos”.

El menú lateral también contiene, a parte de las opciones del menú, un componente con la imagen de perfil del usuario, el nombre y el email. Estos datos se cargan por primera vez cuando el usuario accede a la aplicación, pero, si en tiempo de ejecución, el usuario cambia estos valores en la base de datos desde los ajustes de la aplicación los valores que se muestran aquí deberían cambiar.

Aquí se añade una dificultad: el menú forma parte de la main activity y los ajustes de la app es un fragment que se inicia desde esa mainActivity. Por tanto, el fragment no tiene acceso a los valores de la main activity por la jerarquía de vistas de Android Studio. Para afrontar este problema se encontraron dos soluciones.

La primera es utilizar una interfaz de comunicación entre el fragment y la activity. La interfaz la implementa la activity y se llama desde el fragment, una vez los valores se han modificado.

Otra aproximación, que es la que he decidido implementar para aprender nuevas soluciones de programación, es la utilización de eventos. La implementación es más sencilla, incluso, que la utilización de interfaces de comunicación. Simplemente se ha tenido que crear una clase *ProfileDataChangedEvent,* lanzar el evento cuando los datos han cambiado y suscribirse a ese evento desde la MenuActivity.

Al acceder al menú principal, además, se realiza una consulta a la base de datos para cargar la imagen de perfil del usuario. En caso de que exista, se mostrará en el menú lateral.

Para poder almacenar imágenes se ha utilizado Firebase Storage, el servicio de almacenamiento en la nube proporcionado por Google. Las imágenes de perfil se almacenan dentro de la carpeta *profile_images* utilizando como nombre del archivo el UID del usuario.

Menú lateral	
EndPoint	**Descripción**
`loadProfileImage (String UID)`	Accede a Firestore para cargar la imagen de perfil del usuario.

Tabla 5. Enpoint utilizado para mostrar el menú lateral.

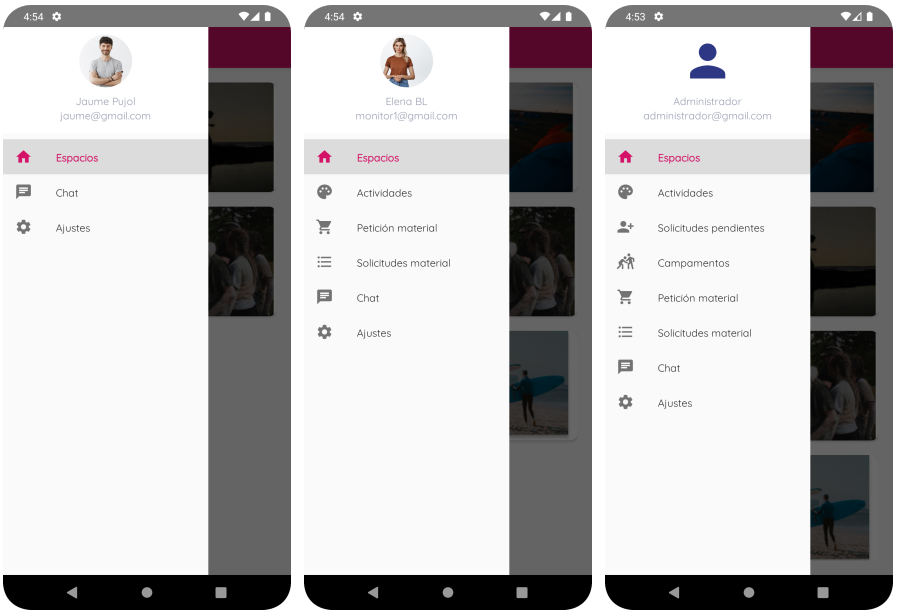


Figura 11. Capturas reales del menú lateral de la aplicación para cada tipo de usuario.

5.2.3 Opción menú: Espacios

Esta funcionalidad forma la primera opción del menú. Aquí todos los usuarios pueden consultar información referente a los cursos que tienen asignados. Los padres podrán acceder a la información de los cursos de sus hijos, los monitores a los cursos de los que son encargados, los administradores a todos los cursos. En ningún caso los padres y los monitores podrán acceder a un espacio de un curso que no tienen asignado.

Está formada por un conjunto de fragments y activities. La primera pantalla que se visualiza es un listado de los cursos. Para poder mostrar este listado es necesario consultar la base de datos para cargar los datos de los cursos.

Si el usuario clica sobre alguno de los cursos accederá a una pantalla en la que podrá encontrar información relativa al curso en cuestión, tal como el nombre de los monitores, el horario semanal y el listado de actividades y campamentos programados.

Si el usuario es un padre, podrá inscribir/desinscribir a los hijos a las actividades y a los campamentos. En este caso se llamará al endpoint *updateInscritoActividad o* updateInscritoCampamento, respectivamente.

Los padres solamente tendrán permisos de consulta en estas pantallas, pero los monitores y los administradores podrán editar los monitores que forman parte de ese curso, así como el horario semanal.

Espacios	
EndPoint	**Descripción**
`loadActivities(CursoModel curso)`	Devuelve el listado de actividades para ese curso. Cada actividad contiene el nombre, descripción y fecha, además

	del listado de inscritos. Esta información se encuentra en la colección “Programación de la base de datos”
`updateInscritoActivity(Actividad, Asociado)`	Actualiza la lista de inscritos a la actividad en la base de datos y añade al asociado.
`loadCamps(CursoModel curso)`	Devuelve el listado de campamentos de ese curso y los datos de cada campamento, que están almacenados en la colección “Campamentos” de la base de datos.
`loadCampImage(String urlImage)`	Puesto que las imágenes de la aplicación se almacenan en storage y no en firestore, es necesario realizar otra consulta para recuperar la imagen de cada campamento.
`updateInscritoCamp(CampamentoModel)`	Actualiza la lista de inscritos al campamento en la base de datos y añade al asociado.

Tabla 6. Endpoints utilizados para mostrar la pantalla el listado de cursos y la pantalla de actividades y campamentos del apartado "Espacios".

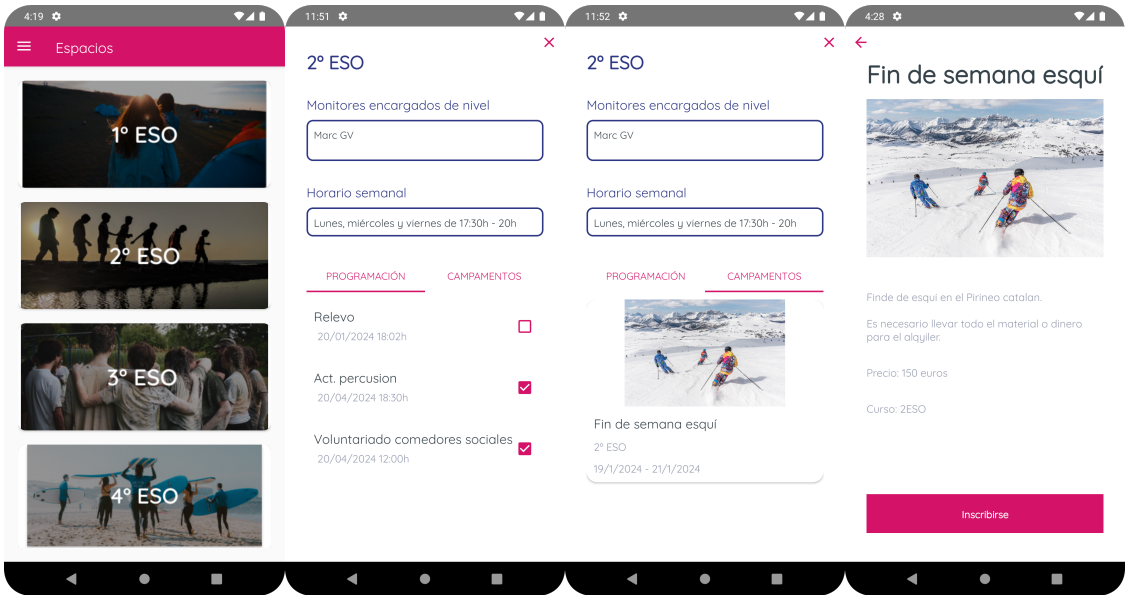


Figura 12. Capturas de pantalla reales del apartado "Espacios" de la aplicación.

5.2.4 Opción menú: Actividades

Esta funcionalidad solamente está disponible para los monitores y los administradores. Desde aquí se puede consultar el listado de actividades programadas por curso, además del listado de inscritos a cada actividad. Los monitores solamente tendrán listados los cursos de los cuales son encargados, los administradores podrán acceder a todos los cursos.

Para poder mostrar este listado es necesario acceder a la base de datos, por eso se llama al endpoint *loadActivities.* Este endpoint ya se está utilizando en la opción del menú "Espacios", por tanto, no ha sido necesario volver a implementar la llamada, dado que se reaprovecha el usecase que utiliza este endpoint. Es una de las ventajas de haber utilizado la arquitectura descrita en el apartado 4.3.2.

Esta pantalla es difícil de gestionar porque está formada por un recyclerView de recyclerViews. El primer recycler contiene el listado de cursos del usuario y los otros contienen el listado de actividades por curso.

En la parte inferior derecha de la pantalla se muestra un botón flotante. Al clicarlo se navega a una pantalla que contiene un formulario en el que se piden los datos necesarios para poder publicar una actividad. Cuando el monitor o administrador rellenen el título de la actividad, el día, el curso al que está dirigido y cliquen sobre el botón "Publicar actividad", se añadirá esa actividad al listado de actividades del curso seleccionado y los padres podrán inscribir a sus hijos desde el apartado "Programación".

Actividades	
EndPoint	**Descripción**
`loadActivities(CursoModel curso)`	Devuelve el listado de actividades para ese curso. Cada actividad contiene el nombre, descripción y fecha, además del listado de inscritos.
`publiActivity(ActividadModel act)`	Añade un nuevo documento a la colección "Actividades" con la información relativa a esa actividad.

Tabla 7. Endpoints utilizados en la funcionalidad "Actividades" de los monitores y administradores.

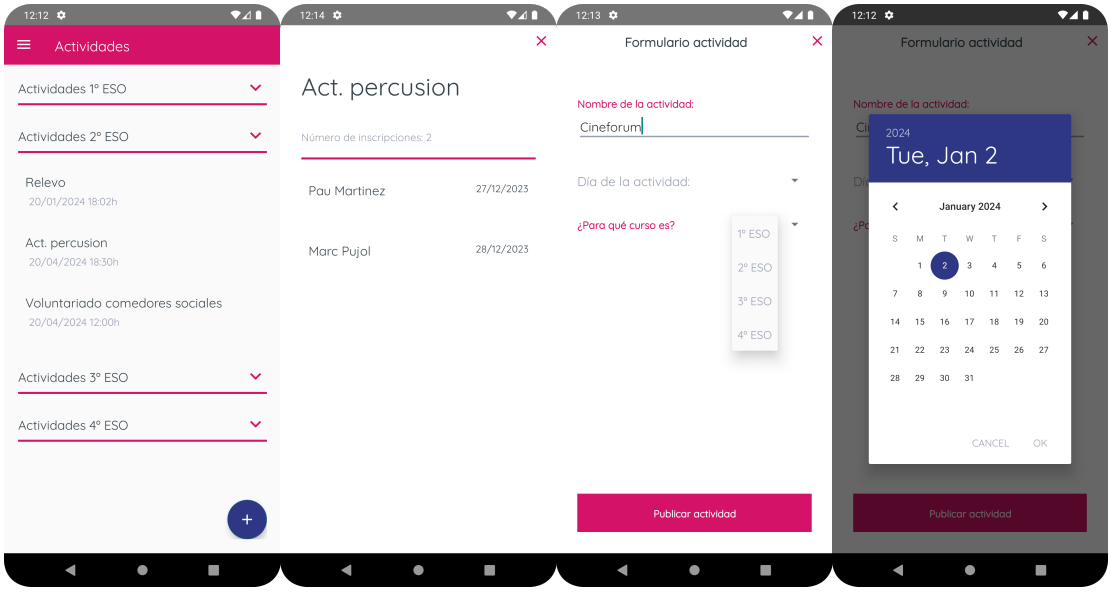


Figura 13. Capturas de pantalla reales de la funcionalidad "Actividades" disponible para los monitores y administradores.

5.2.5 Opción menú: Solicitudes pendientes

Desde esta pantalla los administradores pueden consultar todos los usuarios que se han registrado y que están pendientes de ser validados. Está formado por dos recyclerViews que contienen el listado de solicitudes pendientes de padres y de monitores. Para rellenar estas listas es necesario consultar la colección “Usuarios”.

Cuando el administrador clica sobre uno de los elementos de la lista, se le abre un cuadro de diálogo en el que le aparece el nombre del usuario y los cursos a los que ha pedido acceso. Hasta que el administrador no acepte la solicitud, el usuario padre/monitor podrá iniciar sesión, pero no podrá acceder a ninguna otra funcionalidad de la aplicación.

Solicitudes pendientes	
EndPoint	**Descripción**
`loadRegistrationRequests()`	Consulta la colección “Usuarios” de firestore y devuelve un listado con todos los usuarios cuyo parámetro registroAceptado sea false.
`acceptRegistrationRequest(String uid)`	Accede a la colección “Usuarios” para modificar el parámetro registroAceptado del usuario que se indica en el parámetro de entrada del endPoint.

Tabla 8. Endpoints utilizados en la funcionalidad "Solicitudes pendientes" de los administradores.

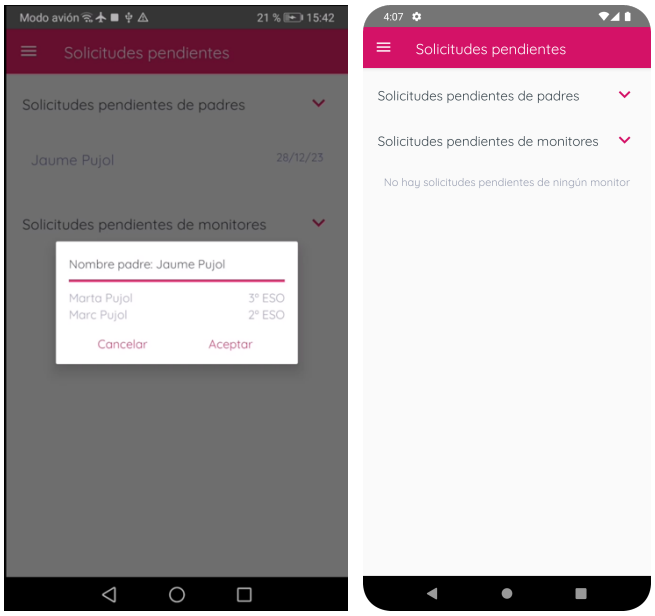


Figura 14. Capturas de pantalla reales de la funcionalidad "Solicitudes pendientes" dispoble para los administradores.

5.2.6 Opción menú: Campamentos

Desde esta funcionalidad los administradores pueden consultar todos los campamentos publicados de todos los cursos, consultar las inscripciones a un campamento determinado, publicar un nuevo campamento o eliminarlo.
Nada más clicar sobre esta opción del menú se accede a una pantalla formada por un recyclerView, cuyo contenido es el listado de campamentos publicados. Para poder cargar esta pantalla es necesario, por un lado, consultar la colección "Campamentos" de la base de datos para cargar la información de los campamentos y, por otro lado, acceder al storage de Firebase para cargar la imagen asociada a ese campamento.

Cuando el usuario clica sobre un elemento de la lista, accede al detalle de ese campamento. El administrador en el detalle se encuentra con dos botones, uno para consultar el listado de inscritos a ese campamento, el otro para eliminar el campamento de la oferta de campamentos.

En la pantalla del listado de campamentos se puede visualizar un botón flotante en la parte inferior derecha de la pantalla. Si el administrador clica sobre ese botón, navega a otra pantalla en la que se encuentra con un formulario con todos los campos necesarios para publicar un nuevo campamento.

Campamentos	
EndPoint	**Descripción**
`loadAllCamps()`	Consulta la colección "Campamentos" de firestore y devuelve un listado de todos los campamentos.
`loadCampImage(String urlImage)`	Puesto que las imágenes de la aplicación se almacenan en storage y no en firestore, es necesario realizar otra consulta para recuperar la imagen de cada campamento.
`publiCamp(CampamentoModel)`	Crea un nuevo documento en la colección "Campamentos" con la información relativa al campamento en cuestión.
`removeCamp(CampamentoModel)`	Elimina el campamento en cuestión de la colección "Campamentos".

Figura 15. Endpoints utilizados en la funcionalidad "Campamentos" de los administradores.

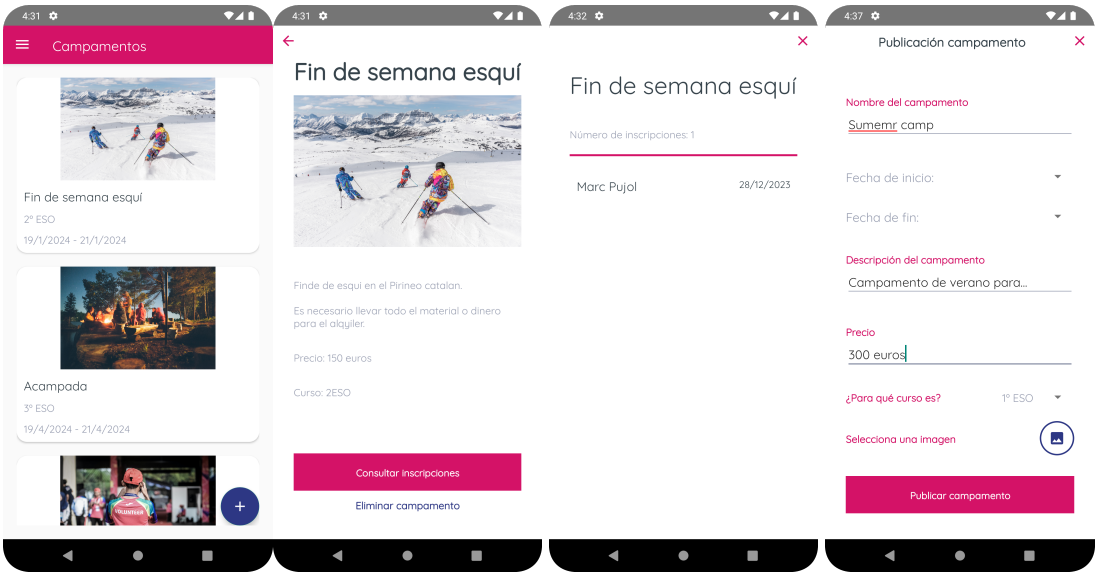


Figura 16. Capturas de pantalla reales de la funcionalidad "Campamentos" dispoble para los administradores.

5.2.7 Opción menú: Material

Esta funcionalidad está disponible para los monitores y los administradores. Con esta funcionalidad se pretende aportar una solución limpia a la compra de material. Desde aquí los monitores pueden pedir al equipo de administración que compren el material necesario para las actividades.

En el apartado *4.2.2* se muestra el diseño de esta pantalla. Sin embargo, a la hora de implementarla se ha realizado un cambio en el diseño. En vez de añadir un botón flotante en el listado de solicitudes de compra de material, que permite navegar al formulario de solicitud, se ha decidido que el formulario de solicitud y el listado de peticiones aparezcan como dos opciones del menú independientes. Esta decisión se ha tomado después de un test de experiencia realizado a un usuario real.

En la opción del menú “Petición material” los usuarios rellenan los datos del formulario para registrar la solicitud de compra de material. Cuando el usuario clica el botón y la petición se ha registrado correctamente, se navega automáticamente al listado de solicitudes.

En la opción del menú “Solicitudes de material” los usuarios encuentran un listado de peticiones de compra ordenadas cronológicamente. Solamente los administradores tendrán visibilidad de todas las solicitudes realizadas por todos los usuarios. Los usuarios monitores solamente visualizarán las solicitudes emitidas por ellos mismos. Para realizar este tipo de query ha sido necesario habilitar en firestore un índice sobre la colección “Material”.

Cada elemento de la lista muestra el material, la fecha en que se va a realizar la actividad y el estado de la solicitud. Si el usuario desea más información acerca de una solicitud en concreto, puede clicar sobre un elemento de la lista y se le abrirá un cuadro de diálogo. En este cuadro encontrará todos los datos referentes a la solicitud en cuestión.

Los administradores, además de la información de la solicitud, encontrarán en este cuadro de diálogo dos botones. Uno para rechazar la solicitud y otro para aceptarla. Cuando el administrador clique alguno de estos dos botones, cambiará el estado de la solicitud.

Para poder actualizar la lista de solicitudes después de cerrar el diálogo, ha sido necesario lanzar un evento desde el diálogo. Este evento lo registra el fragment de solicitudes, que al recibirlo volverá a llamar al endpoint *loadSolicitudes().*

Material	
EndPoint	**Descripción**
`loadSolicitudes()`	Devuelve un listado de solicitudes de compra de material. Para ello consulta la colección “Material” y devuelve el listado ordenado cronológicamente.
`aceptarSolicitud(boolean aceptar)`	Cambia el estado de la solicitud de compra de material a “Aceptada” o “Rechazada”, dependiendo del parámetro de entrada.
`makeMaterialRequest(MaterialModel)`	Añade un nuevo documento a la colección “Material” con los datos correspondientes a la solicitud de compra de material.

Tabla 9. Endpoints utilizados en la funcionalidad "Material" de los administradores y monitores.

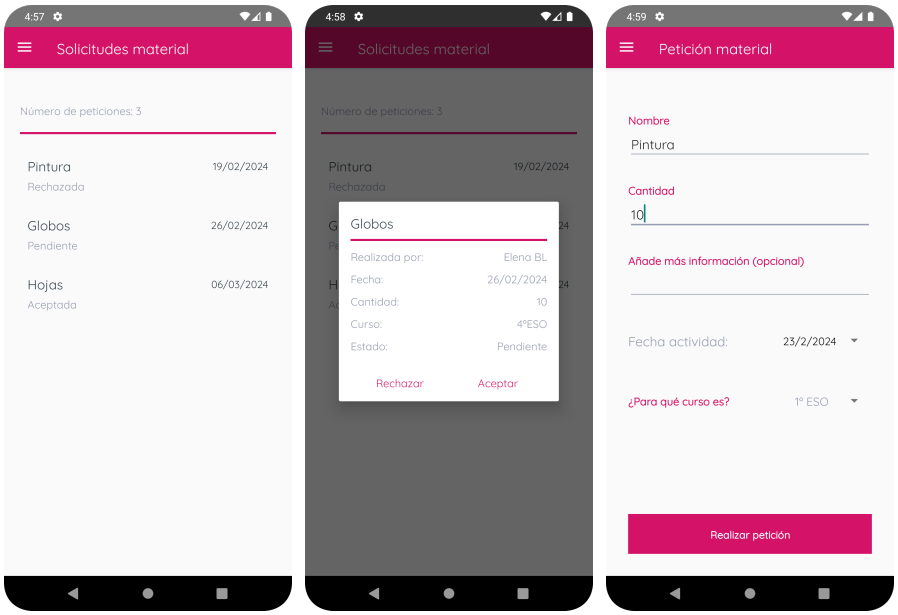


Figura 17. Capturas de pantalla reales de la funcionalidad "Material" disponible para los administradores y monitores.

5.2.8 Opción menú: Chat

Esta funcionalidad es una de las características clave de esta aplicación. Dispone de un chat grupal con todos los padres y monitores de cada curso. Además, un chat individual con todos los usuarios de la aplicación. Está disponible para todos los usuarios y permite enviar y recibir mensajes en tiempo real.

La primera pantalla de esta funcionalidad del menú está formada por un listado de chats grupales de los cursos al que el usuario está asociado y, debajo, una lista de todos los chats individuales en el que ese usuario participa (Fig.18). Al clicar sobre cualquiera de los chats, se navega al espacio en el que se pueden ver todos los mensajes enviados anteriormente y enviar nuevos.

Para poder implementar esta funcionalidad ha sido necesario crear la colección “Chats”, que se usa tanto en los chats grupales como en los individuales. Esta colección almacena todos los chats que se han iniciados, es decir, tantos los grupales como los individuales. Cada documento tendrá los siguientes atributos, según si ese chat corresponde a un grupo a un usuario:

Atributos documento chat grupal	**Atributos documento chat individual**
- idChat - isPrivate = false - nombreChat - ultimeMensaje	- isPrivate = true - uidParticipantes - ultimoMensaje

Tabla 10. Atributos de cada documento de la colección "Chats" de Firestore

Además, “Mensajes” es la colección en la que se almacenan los mensajes enviados a los grupos. Cada documento de esa colección representa un mensaje y está formado por los siguientes atributos:

- Sender
- Text
- Timestamp
- idChat

Esta es la información que se necesita para mostrar los mensajes en los grupos. El atributo timestamp guarda la fecha y hora en la que ese mensaje ha sido enviado, dato que también se muestra en el chat y que es necesario para mostrar los mensajes en orden cronológico. Por tanto, una vez se accede a un chat grupal, se accede a la colección “Mensajes” y se cargan todos los mensajes pertenecientes a ese chat ordenados cronológicamente. Los mensajes que han sido enviados por el propio usuario se mostrarán alineados a la derecha, en vez de a la izquierda.

Por otro lado, la colección “MensajesPrivados” almacena los chats individuales que se han iniciado. Cada uno de los documentos de esta colección está formado por un array “participantes” formado por los 2 UID de los usuarios que participan de ese chat.

Además, cada documento contiene una subcolección “mensajes”, en la que se almacenan todos los mensajes enviados en ese chat.

Para actualizar en tiempo real los mensajes de textos enviados ha sido necesario, en primer lugar, obtener con una consulta de Firestore los documentos que pertenecen a ese chat, ya sea grupal o individual, y, en segundo lugar, añadir un “listener” a esa colección para comprobar si ha habido algún cambio de tipo adición. En el fragmento de código 1 del Anexo se puede ver el fragmento de código que corresponde a esta consulta.

Cada vez que el usuario actual clique el botón “Enviar” del chat también se accederá a la base de datos para añadir un nuevo documento con ese mensaje. Esto se hace a través del endpoint *sendMessage(String text).*

Chat	
EndPoint	**Descripción**
`loadMessages()`	Carga todos los mensajes pertenecientes a un grupo y los ordena cronológicamente. Accede a la colección “Mensajes” de Firestore.
`sendMessage(String text)`	Crea un nuevo documento en la colección “Mensajes” que representa el nuevo mensaje que se ha enviado.

Tabla 11. Endpoints utilizados en la funcionalidad "Chat".

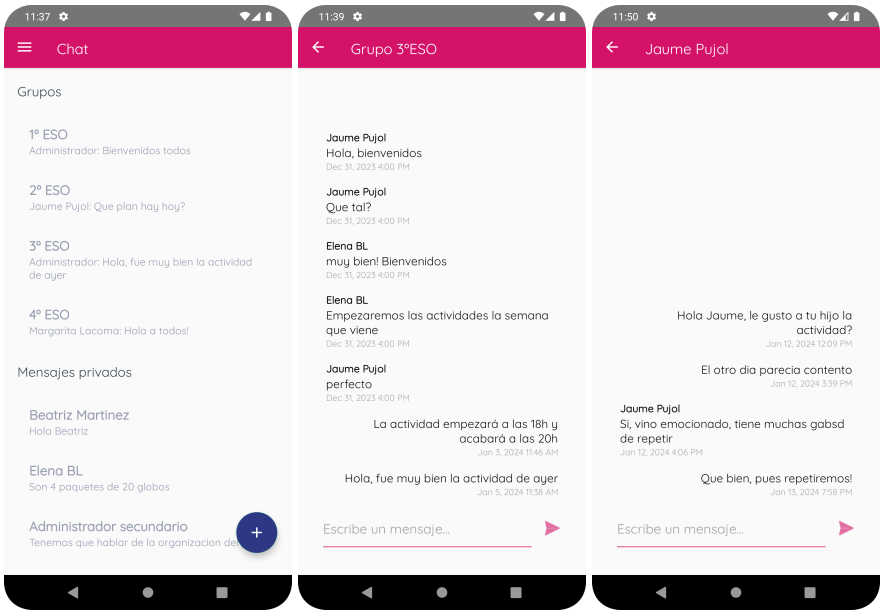


Figura 18. Capturas de pantalla reales de la funcionalidad "Chat".

5.2.9 Opción menú: Ajustes

Esta es la última opción del menú y está disponible para todos los usuarios. Es una funcionalidad básica en cualquier aplicación. Desde aquí el usuario puede cambiar la foto de perfil, el nombre, el correo electrónico o la contraseña de acceso.
Esta pantalla, aunque parece sencilla, requiere mucha lógica por detrás porque cada uno de los datos que se pueden modificar están almacenados en una parte de Firebase distinta.

Para poder modificar la foto de perfil es necesario llamar al endpoint *saveImageInDB(),* que accederá a Storage de Firebase para guardar la nueva imagen de perfil o sustituirla por la que ya existía.

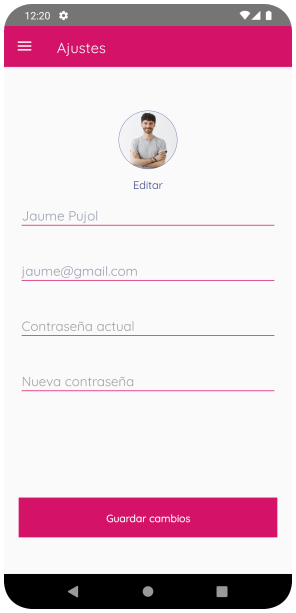


Figura 19. Captura de pantalla real de la funcionalidad "Ajustes".

Para modificar el nombre del usuario, la contraseña o la dirección de correo electrónico es necesario llamar al endpoint *saveDataInDB()* , que accede a FirebaseAuth para actualizar estos datos.

Una vez se han modificado estos datos en la base de datos es necesario actualizarlos también en el menú lateral de la aplicación. En este punto existe una dificultad: el menú lateral forma parte de la activity principal de la aplicación, por lo que el fragment desde el que se actualizan los datos no tiene visibilidad sobre esa activity. Para solucionar este problema existen dos aproximaciones, la utilización de interfaces de comunicación o la utilización de eventos.

En este proyecto he decidido utilizar eventos para solucionar este problema ya que es una aproximación que conozco poco y que he querido aprender. Para ello he creado 3 eventos distintos, que se lanzan desde el fragment de ajustes y los recoge la MenuActivity.

En el anexo, en el fragmento de código 2, se muestra cómo se lanza el evento en el momento que se ha actualizado el nombre de usuario. Y en el fragmento de código 3, la MenuActcity se suscribe al evento ProfileDataChangedEvent y actualiza el nombre del usuario en el menú lateral.

Ajustes	
EndPoint	**Descripción**
`saveDataInDB()`	Actualiza el nombre de usuario.
`changeAuthData()`	Reautentica al usuario actual para poder acceder a FirebaseAuth y cambiar la contraseña o el correo electrónico.
`saveImageInDB()`	Accede a Storage para almacenar la nueva foto de perfil.

Tabla 12. Endpoints utilizados en la funcionalidad "Ajustes".

6 Estrategia QA

En el desarrollo de la aplicación, se ha adoptado una estrategia QA centrada en la realización de pruebas de regresión manuales. Para llevar a cabo este enfoque, se ha colaborado con un grupo de tres testers voluntarios, quienes se han encargado de evaluar la aplicación de forma manual cada vez que se implementaba una nueva funcionalidad y se integraba en la rama de desarrollo develop. Su tarea ha consistido en identificar cualquier defecto o comportamiento inesperado que pudiera surgir con las nuevas incorporaciones. En caso de encontrar algún problema, los testers notificaban inmediatamente el defecto, permitiendo así una corrección inmediata para mantener la estabilidad y calidad de la aplicación.

Este seguimiento se ha llevado a cabo a través de un formulario de Google. Los testers voluntarios rellenaban el formulario cada vez que encontraban un defecto en la aplicación. En el anexo se adjunta el formulario y las respuestas.

Esta estrategia de QA, basada en pruebas manuales de regresión[xlii], ha proporcionado una evaluación continua de la aplicación a medida que se introducían cambios en el código. La participación activa de los testers ha garantizado una revisión exhaustiva de las nuevas funcionalidades y ha contribuido a la detección temprana y resolución eficiente de posibles problemas, mejorando la calidad del software.

Una vez desarrolladas todas las funcionalidades de la aplicación, se ha llevado a cabo un test de rendimiento. El test ha consistido en la conexión simultánea de 12 usuarios distintos a la aplicación. Se reunieron estas 12 personas el mismo día a la misma hora y se les pidió que navegaran de manera controlada. Es decir, todos accedieron en el mismo momento a la aplicación y a las distintas opciones del menú. De esta manera la aplicación realizaba la misma consulta a la base de datos, pero en 12 usuarios distintos.

Con este test se ha querido comprobar el tiempo de respuesta de las distintas consultas que la aplicación hace a la base de datos. Para ello se ha comparado el tiempo medio de respuesta cuando solamente un usuario ha iniciado sesión en la base de datos, con el tiempo de respuesta cuando varios usuarios han iniciado sesión y están haciendo la misma consulta.

Para ello me he ayudado de Firebase Performance Monitoring[xliii], una herramienta de Firebase que mide y analiza métricas clave de rendimiento, como tiempos de inicio, tiempos de red, tiempos de respuesta de las solicitudes de base de datos, etc. A través de la consola que proporciona la propia herramienta se pueden visualizar informes detallados sobre el rendimiento de la aplicación, identificar áreas de mejora y tomar decisiones informadas para optimizar el rendimiento.

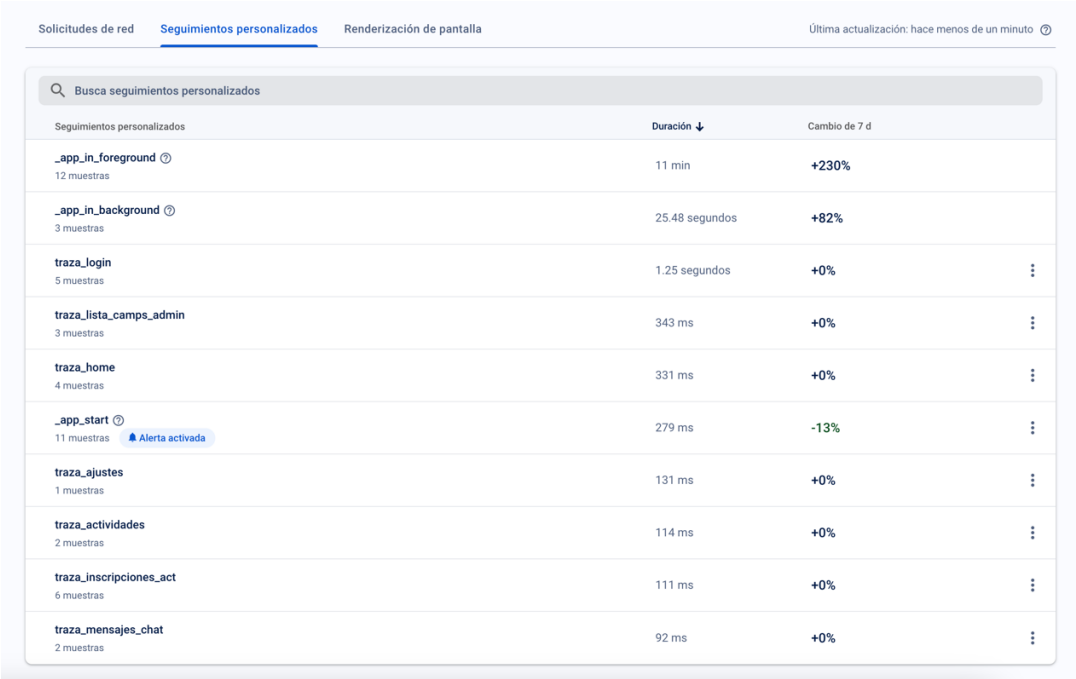


Figura 20. Vista de parte de la consola de Firebase Performance.

Una vez integrado el SDK de Firebase Performance en el proyecto se han podido añadir trazas de seguimiento de ciertas consultas clave. Performance es capaz de capturar el tiempo desde que se inicia la traza hasta que se detiene. De esta manera, cada vez que un usuario utiliza la aplicación y hace una consulta a la base de datos que tiene traza, se mide el tiempo de respuesta. Para añadir una traza que Performance pueda detectar se tiene que implementar las siguientes líneas de código:

```
// Iniciar el seguimiento
Trace trace =
FirebasePerformance.getInstance().newTrace("nombre_del_seguimiento");
trace.start();

// Realizar la consulta a Firestore
// Lógica de la consulta aquí

// Finalizar el seguimiento después de que la consulta haya finalizado
trace.stop();
```

Fragmento código 1. Líneas de código a implementar para añadir trazas de Firebase Performance en una aplicación de Android Studio.

En la siguiente tabla se muestra el tiempo de respuesta de las trazas que se han añadido a la aplicación, cuando solamente hay un usuario conectado y cuando los 12 usuarios se han conectado. Se han añadido estas trazas porque son las que se han considerado claves porque afectan a la experiencia de usuario.

Traza	Tiempo respuesta un solo usuario conectado	Tiempo respuesta múltiples usuarios conectados
Traza_login	1,25 segs	1,19 segs
Traza_home	331 msegs	347 msegs
Traza_actividades	114 msegs	91 msegs
Traza_inscripcion_act	111 msegs	133 msegs

Traza_lista_camps_admin	343 msegs	451 msegs
Traza_ajustes	231 msegs	170 msegs
Traza_mensajes_chat	92 msegs	147 msegs

Tabla 13. Comparativo del tiempo de respuesta de los mismos endpoints en un caso ideal en el que solo hay un usuario conectado y en un caso real, cuando hay 12 usuarios conectados.

Los datos obtenidos no demuestran ningún tipo de latencia superior cuando hay varios usuarios conectados a la aplicación. La mayoría de las trazas muestran un aumento en el tiempo de respuesta de la consulta a la base de datos. Sin embargo, esta diferencia de tiempo podría deberse a fluctuaciones en la red y no a una sobrecarga.

La traza correspondiente al chat sí que muestra un aumento muy significativo del tiempo. No obstante, el tiempo total en milisegundos sigue siendo imperceptible para el ser humano, por lo que podemos concluir que la experiencia de usuario sigue siendo buena cuando hay varios usuarios chateando.

Sería necesario, igualmente, realizar este test de rendimiento con muchos más usuarios para acercarse a una situación real. Sin embargo, la realización de esta prueba ha sido de gran provecho personal para aprender esta herramienta de Firebase y conocer en mayor profundidad el universo de soluciones de Google.

7 Conclusiones

En este trabajo de final de carrera se ha abordado la problemática existente en la gestión diaria de asociaciones juveniles, específicamente la Asociación Cultural Familiar Carena. La falta de una herramienta eficiente de comunicación entre monitores, padres y personal administrativo se identificó como un obstáculo significativo para llevar a buen fin el desarrollo de las actividades previstas. La ausencia de aplicaciones en el mercado que satisfagan las necesidades de todas las partes involucradas ha motivado la creación de un prototipo de aplicación para dispositivos con sistema operativo Android.

Para alcanzar los objetivos establecidos, se ha seguido un proceso estructurado que ha incluido un análisis previo de la situación y del mercado, que ha proporcionado información valiosa para orientar el desarrollo.

Le ha seguido el apartado ingeniería de concepción, en el que se definen los requisitos, especificaciones y el diseño visual y arquitectónico del prototipo, incluyendo la estructura de la base de datos. En este apartado también se ha abordado necesidades específicas de los distintos usuarios a través de historias de usuario.

Gracias a este análisis previo, en el que se estudiaron aplicaciones similares y se detectaron las necesidades de los usuarios, se pudo empezar con el desarrollo de la aplicación. Durante el desarrollo se siguió una estrategia de calidad basada en tests de regresión manuales. Cada vez que una nueva funcionalidad era desarrollada, se realizaba un testeo profundo de esa funcionalidad y cualquier defecto era notificado para su pronta corrección.

Si bien este proyecto abordó la primera iteración del desarrollo, su enfoque en la viabilidad funcional sienta las bases para futuras implementaciones y mejoras.

Ha sido muy gratificante poder colaborar en este proyecto y ver cómo soluciones tecnológicas pueden facilitar la tarea educativa de las asociaciones juveniles frente a los menores que asisten a sus actividades. Además de trabajar por cumplir los objetivos planteados, el trabajo ha supuesto una gran labor de aprendizaje en el que se han aprendido nuevos patrones de programación y herramientas tecnológicas.

Aunque todo el desarrollo del trabajo se ha enfocado a la Asociación Cultural Familiar Carena, esta aplicación es una herramienta muy útil para cualquier asociación similar, por lo que puede llegar a un público mucho más amplio y se pueden seguir ampliando las funcionalidades de acuerdo con las necesidades de todos los agentes involucrados.

Con el trabajo finalizado, se puede afirmar que este proyecto es viable y que se han alcanzado los objetivos planteados en la introducción. Como resultado, se ha obtenido una aplicación ejecutable que ayuda a todas las personas que tienen contacto con la asociación a obtener una comunicación más fluida y eficiente y facilita el trabajo de gestión.

8 Anexos

8.1 Usuarios

Tipo	Nombre	Correo	Contraseña
Administrador	Administrador	administrador@gmail.com	123456
Administrador	Administrador secundario	administrador1@gmail.com	123456
Monitor	Elena BL	monitor1@gmail.com	123456
Monitor	Marc GV	monitor2@gmail.com	123456
Padre	Beatriz Martinez	beatriz@gmail.com	123456
Padre	Jaume Pujol	jaume@gmail.com	123456

Tabla 14. Credenciales de los usuarios de prueba

8.2 Fragmentos de código

```
private void loadMessages() {
    //Se vacía la lista de mensajes para evitar duplicidades
    messages.clear();

    //Se obtiene de la colección "Mensajes" la lista de todos los
documentos pertenecientes al grupo idChat
    //y ordenados cronológicamente
    messagesCollection.whereEqualTo("idChat",
idChat).orderBy("timestamp", Query.Direction.ASCENDING)
            .addSnapshotListener((queryDocumentSnapshots, e) -> {
                //Se añade un listaner a la colección para obtener
una lista de cambios de tipo "ADDED"
                //de esa colección. Así se cargarán en tiempo real
los nuevos mensajes añadidos a la colección
                for (DocumentChange dc :
queryDocumentSnapshots.getDocumentChanges()) {
                    if (dc.getType() == DocumentChange.Type.ADDED) {
                        ChatMessageModel message =
dc.getDocument().toObject(ChatMessageModel.class);
                        messages.add(message);
                    }
                }
            //Se devuelve el listado de mensajes al presenter
                callback.onSuccess(messages)
            });
}
```

Fragmento código 2. Método al que se llama para cargar todos los mensajes de un chat y actualizar el listado en tiempo real.

```
private void saveDataInDB(){

FirebaseFirestore db = AppManager.getInstance().getFirestore();

//Se actualiza el nombre de usuario en la colección correspondiente
db.collection(getUserCollection())

.document(AppManager.getInstance().getUser().getFirebaseUser().getUi
d())
        .update("nombre", userFullNameUpdated)
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                //Se lanza un evento indicando el nuevo nombre de
usuario
                EventBus.getDefault().post(new
ProfileDataChangedEvent(userFullNameUpdated));
            }
        });
}
```

Fragmento código 3. Método al que se llama para actualizar la foto de perfil del menú lateral cuando se ha modificado en la pantalla de Ajustes. Se emite el evento "ProfileDataChangedEvent".

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void onAuthenticationSuccess(ProfileDataChangedEvent event) {
    nombreTextView.setText(event.getNombreUsuario());
}
```

Fragmento código 4. MenuActivity se suscribe al evento "ProfileDataChangedEvent" de esta manera

9 Bibliografía

[i] SERRADOR, P. y PINTO, J.K., 2015. Does Agile work? — A quantitative analysis of agile project success. International journal of project management, vol. 33, no. 5, pp. 1040-1051. ISSN 0263-7863. DOI 10.1016/j.ijproman.2015.01.006.

[ii] DINGSØYR, T., NERUR, S., BALIJEPALLY, V. y MOE, N.B., 2012. A decade of agile methodologies: Towards explaining agile software development. The Journal of systems and software, vol. 85, no. 6, pp. 1213-1221. ISSN 0164-1212. DOI 10.1016/j.jss.2012.02.033.

[iii] "¿Qué es un tablero kanban? | Atlassian". Atlassian. Accedido el 10 de octubre de 2023. [En línea]. Disponible: https://www.atlassian.com/es/agile/kanban/boards

[iv] "Tablero Kanban". Deloitte Spain. Accedido el 10 de octubre de 2023. [En línea]. Disponible: https://www2.deloitte.com/es/es/pages/technology/articles/que-es-tablero-kanban.html

[v] "What is Trello: Learn Features, Uses & More | Trello". Manage Your Team's Projects From Anywhere | Trello. Accedido el 12 de octubre de 2023. [En línea]. Disponible: https://trello.com/tour

[vi] "Gantt.com". Gantt.com. Accedido el 12 de octubre de 2022. [En línea]. Disponible: https://www.gantt.com/

[vii] "What Is a Gantt Chart? | Definition & Examples | APM". APM | Chartered Membership Organisation. Accedido el 12 octubre de 2023. [En línea]. Disponible: https://www.apm.org.uk/resources/find-a-resource/gantt-chart/

[viii] "Mi ampa". Sofwate de gestión ampa para tu colegio. Accedido el 20 de octubre de 2023. [En línea]. Disponible: https://miampa.com/

[ix] "Inicio - TokApp School". TokApp School. Accedido el 20 de octubre de 2023. [En línea]. Disponible: https://www.tokappschool.com/

[x] "EasyManager". Programa Gestión Pyme - EasyManager. Accedido el 21 de octubre de 2023. [En línea]. Disponible: https://www.easymanager.app/

[xi] CODLING, S., 2000. Benchmarking. Madrid: AENOR. ISBN 8481432520.

[xii] N Inukollu, V., Keshamon, D. D., Kang, T., & Inukollu, M. (2014). Factors Influncing Quality of Mobile Apps: Role of Mobile App Development Life Cycle. International Journal of Software Engineering & Applications, 5(5), 15–34. https://doi.org/10.5121/ijsea.2014.5502

[xiii] BERENBACH, B. y BERENBACH, Brian., 2009. Software & systems requirements engineering : in practice. First edition. New York: McGraw-Hill. ISBN 1-282-04859-7.

[xiv] Eeles, P. (2001). What Is an Architectural Requirement? The FURPS + System for Classifying Requirements. The Rational Edge. Retrieved from http://itculiacan.webs.com/descargas/z_FURPS Requisites.pdf

[xv] KAPROS, Evangelos. y KOUTSOMBOGERA, Maria., 2018. *Designing for the User Experience in Learning Systems*. 1st ed. 2018. Cham: Springer International Publishing. ISBN 3-319-94794-X.

[xvi] "UI vs UX: What's the Difference between UI & UX Design? | Figma". Figma. Accedido el 26 de noviembre de 2023. [En línea]. Disponible: https://www.figma.com/resource-library/difference-between-ui-and-ux/

[xvii] Accedido el 15 de septiembre de 2023. [En línea]. Disponible: https://www.carena.org/

[xviii] "Coolors - The super fast color palettes generator!" Coolors.co. Accedido el 18 de octubre de 2023. [En línea]. Disponible: https://coolors.co/

[xix] "Browse Fonts - Google Fonts". Google Fonts. Accedido el 18 de octubre de 2023. [En línea]. Disponible: https://fonts.google.com/

[xx] "Top 10 Fonts for Your Website Design". Bakata Solutions S.L.U. Accedido el 18 de octubre de 2023. [En línea]. Disponible: https://bakata.es/blog/blog-1/top-10-tipografias-para-el-diseno-de-tu-pagina-web-8#:~:text=5.,los%20contenidos%20de%20la%20web.

[xxi] "Herramienta de Maquetas, Esquemas & Prototipos UI En Línea · Moqups". Online Mockup, Wireframe & UI Prototyping Tool · Moqups. Accedido el 19 de octubre de 2023. [En línea]. Disponible: https://moqups.com/es/

[xxii] "Software Architecture for faster development & maintenance". Software intelligence Automated | CAST. Accedido el 25 de noviembre de 2023. [En línea]. Disponible: https://www.castsoftware.com/glossary/what-is-software-architecture-tools-design-definition-explanation-best#:~:text=Software%20architecture%20is,%20simply,%20the,the%20software%20into%20the%20future.

[xxiii] F. Day. "Top 5 Databases for Mobile Applications". Noble Desktop | Design & Coding Classes and Certificate Programs. Accedido el 8 de octubre de 2023. [En línea]. Disponible: https://www.nobledesktop.com/classes-near-me/blog/top-databases-for-mobile-applications

[xxiv] Jorge, L. (2023) Top 31 mobile app databases for your app in 2023: A comprehensive guide, Medium. Disponible en: https://medium.com/@livajorge7/top-31-mobile-app-databases-for-your-app-in-2023-a-comprehensive-guide-6d0d1af05412. Accedido el 8 de octubre de 2023.

[xxv] Oracle, "MySQL," *Mysql.com*, 2000. https://www.mysql.com/. Accedido el 8 de octubre de 2023.

[xxvi] Sqlite Home Page. Disponible en: https://www.sqlite.org/index.html. Accedido el 8 de octubre de 2023.

[xxvii] The PostgreSQL Global Development Group, "PostgreSQL: The world's most advanced open source database," *Postgresql.org*, 2019. https://www.postgresql.org/. Accedido el 8 de octubre de 2023.

[xxviii] "Cómo elegir tu base de datos: Cloud Firestore o Realtime Database | Firebase Realtime Database," *Firebase*. https://firebase.google.com/docs/database/rtdb-vs-firestore?hl=es-419. Accedido el 8 de octubre de 2023.

[xxix] J. G. G. Torres, "Diferencias entre Realtime Database vs Firestore," *Medium*, Nov. 27, 2018. https://jggomezt.medium.com/diferencias-entre-realtime-database-vs-firestore-94364a2cc09e. Accedido el 9 de octubre de 2023.

[xxx] "La base de datos líder del mercado para aplicaciones modernas," *MongoDB*, 2019. https://www.mongodb.com/es. Accedido el 9 de octubre de 2023.

[xxxi] "Couchbase NoSQL Database | Couchbase," *Couchbase.com*, 2019. https://www.couchbase.com/. Accedido el 9 de octubre de 2023.

[xxxii] "What is Firestore? Everything you Need to Know in 2024," Lido.app. [Online]. Disponible en: https://www.lido.app/firebase/what-is-firestore. Accedido el 10 de octubre de 2023.

[xxxiii] J. Clark, "What is Firebase authentication?," Back4App Blog, 30-Jan-2021. [Online]. Disponible en: https://blog.back4app.com/firebase-authentication/. Accedido el 17 de octubre de 2023.

[xxxiv] "Firebase Authentication," Firebase. [Online]. Disponible en: https://firebase.google.com/products/auth?hl=es-419. Accedido el 17 de octubre de 2023.

[xxxv] "Elige una estructura de datos," Firebase. [Online]. Disponible en: https://firebase.google.com/docs/firestore/manage-data/structure-data?hl=es-419. Accedido el 25 de octubre de 2023.

[xxxvi] Indeed.com. [Online]. Disponible en: https://www.indeed.com/career-advice/career-development/what-are-the-layers-in-software-architecture. Accedido el 3 de noviembre de 2023.

[xxxvii] LANO, K. y YASSIPOUR TEHRANI, S., 2023. *Introduction to Software Architecture Innovative Design using Clean Architecture and Model-Driven Engineering*. 1st ed. 2023. Cham: Springer Nature Switzerland. ISBN 3-031-44143-5.

[xxxviii] "Clean Coder Blog," Cleancoder.com. [Online]. Disponible en: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html. Accedido el 3 de noviembre de 2023.

[xxxix] “Callback function,” MDN Web Docs. [Online]. Disponible en: https://developer.mozilla.org/en-US/docs/Glossary/Callback_function. Accedido el 15 de noviembre de 2023.

[xl] Wikipedia contributors, “Callback (computer programming),” Wikipedia, The Free Encyclopedia, Accedido el 15 de noviembre de 2023. [Online]. Disponible en: https://en.wikipedia.org/w/index.php?title=Callback_(computer_programming)&oldid=1185233420.

[xli] Wikipedia contributors, “Event (computing),” Wikipedia, The Free Encyclopedia, Accedido el 16 de noviembre de 2023. [Online]. Disponible en: https://en.wikipedia.org/w/index.php?title=Event_(computing)&oldid=1185403427.

[xlii] BAUMGARTNER, M., KLONK, M., MASTNAK, C., PICHLER, H., SEIDL, R. y TANCZOS, S., 2021. Agile Testing: The Agile Way to Quality. 1st Edition 2021. Cham: Springer International Publishing AG. ISBN 9783030732080.

[xliii] “Firebase Performance Monitoring,” Firebase. [Online]. Disponible en: https://firebase.google.com/docs/perf-mon?hl=es. Accedido el 20 de diciembre de 2023.