
# MEMORIA DE PRÁCTICAS

## App Taskshare

### Descripción breve

TaskShare es una app para gestionar tareas compartidas

*IES GONZALO CHACÓN*

*6 de junio de 2025*

*Adrian Carretero Alcazar*

*adrian.carretero16@educa.madrid.org*

# 1 INDICE

# 1. INTRODUCCIÓN

## 1.1 JUSTIFICACION DEL PROYECTO (DEFINICION DEL PROBLEMA)

En la vida cotidiana, especialmente en hogares o grupos de convivencia, la gestión de tareas puede volverse complicada debido a la falta de organización y comunicación. Esta situación puede causar olvidos, acumulación de responsabilidades e incluso conflictos. Para abordar este problema, surge **TaskShare**, una aplicación móvil que busca optimizar la administración de tareas compartidas, facilitando la coordinación entre los miembros del hogar o grupo mediante tecnología accesible, funcional y motivadora.

## 1.2 OBJETIVOS GENERALES Y ESPECIFICOS

### 1.2.1 OBJETIVO GENERAL

- Desarrollar una aplicación móvil eficiente y fácil de usar que permita la gestión, asignación y seguimiento de tareas compartidas en entornos domésticos o colaborativos.

### 1.2.2 OBJETIVOS ESPECÍFICOS

- Permitir a los usuarios crear y asignar tareas para realizar.
- Implementar recordatorios automáticos para evitar olvidos.
- Incluir un sistema de incentivos mediante puntos para fomentar el cumplimiento de tareas.
- Poder contactar con los usuarios agregados por WhatsApp, teléfono y/o mensaje(sms).
- Ofrecer opciones de personalización como modo oscuro/claro.
- Incluir funcionalidades de recuperación de contraseña mediante correo electrónico.
- Utilizar Kotlin para el desarrollo de la app y Firebase para la gestión de autenticación, almacenamiento y notificaciones.

## 1.3 PUBLICO OBJETIVO

**TaskShare** está dirigida a:

- Familias o personas que viven juntas y necesitan coordinar tareas del hogar.
- Grupos de compañeros de piso, residencias estudiantiles o espacios compartidos.
- Usuarios que buscan una solución práctica, motivadora y centralizada para organizar tareas cotidianas entre varias personas.

## 1.4 VISIÓN GENERAL | APP A DESARROLLAR

**TaskShare** es una aplicación móvil desarrollada en Kotlin con backend en Firebase, pensada para facilitar la administración de tareas compartidas. Permitirá a los usuarios crear, asignar y seguir tareas entre contactos, incorporando recordatorios automáticos, notificaciones por mensajería externa, personalización de la interfaz, sistema de puntos como incentivo, etc. El objetivo es ofrecer una herramienta práctica, moderna y efectiva para mejorar la convivencia y la organización en grupos.

# 2 ESTUDIO PREVIO

Existen apps populares para la gestión de tareas como Todoist, Google Tasks y Trello. Todoist ofrece muchas funciones, pero requiere pago para opciones avanzadas. Google Tasks es muy simple pero limitada. Trello es excelente para trabajo en equipo, aunque más compleja y enfocada a proyectos.


*Todoist*

*Google Task*

*Trello*

Mi aplicación busca un punto intermedio, ofreciendo una herramienta sencilla, gratuita y colaborativa, centrada en la creación y asignación de tareas con fecha y hora.

## 2.1 TECNOLOGÍAS UTILIZADAS Y ELECCIÓN

Existen varias opciones para desarrollar apps:

- Nativo (Kotlin, Swift): máximo rendimiento.
- Multiplataforma (React Native, Flutter): un solo código para Android e iOS.

Hay múltiples opciones para poder desarrollar la app, pero se tenía que buscar la más acorde al funcionamiento de la aplicación.

Estuve investigando y todas las vías me llevaban a un sitio, Java.


*2-1Esquema lenguajes*

Java es una buena herramienta para el desarrollo de una aplicación que gestiona tareas.

Dí un paso más, seguí investigando y me encontré con Kotlin.

Opté por Kotlin con Jetpack Compose para crear una app nativa Android moderna y fluida, siguiendo las recomendaciones oficiales de Google. Además, usé Firebase (Firestore y Auth) por su fácil integración, escalabilidad y porque evita mantener un servidor propio.

## 2.2 ELECCIÓN DE COLORES

El color es uno de los elementos más influyentes en el diseño de una aplicación, ya que tiene un impacto directo en la percepción del usuario, su experiencia y la conexión emocional con la interfaz. A través del color, es posible transmitir valores clave como la confianza, la productividad o la creatividad, fundamentales para el posicionamiento de una app como es **Taskshare**, enfocada en la colaboración y la gestión eficiente del tiempo.

Los colores no solo influyen en la estética, sino también en la **usabilidad** y el **comportamiento del usuario**. Estudios de diseño de interfaz han demostrado que las paletas cromáticas adecuadas pueden mejorar la retención de usuarios y la claridad en la navegación.

*2-2Colores disponibles*

En **Taskshare**, la elección de la paleta de colores toma como referencia estos principios. Se prioriza una combinación que inspire **confianza**, **productividad** y una interfaz **limpia y moderna**. Por ejemplo, el uso de tonos azules y morados suaves puede reforzar la idea de colaboración eficiente y progreso, mientras que los acentos en colores cálidos (como el rojo) ayudan a resaltar elementos interactivos sin saturar la vista.

La correcta aplicación del color refuerza así los valores de la marca, mejora la experiencia de usuario y contribuye a una interfaz visualmente coherente y profesional.

### 2.2.1 LOGO

El logotipo de la aplicación **TaskShare** ha sido diseñado utilizando la herramienta en línea **Canva**, que permite crear diseños visuales de manera intuitiva y profesional. El logo ha sido concebido con una estética simple, moderna y fácilmente reconocible, haciendo uso de **tonalidades rojas muy claras**, que aportan una sensación de dinamismo, energía y enfoque.

La elección del color rojo claro no ha sido aleatoria. El rojo es un color asociado comúnmente con la **acción**, la **productividad** y la **urgencia**, características que están estrechamente relacionadas con el propósito de la aplicación: facilitar la organización y compartición de tareas entre usuarios. Al utilizar un tono más claro, se transmite también una sensación de cercanía y accesibilidad, evitando que el rojo resulte agresivo o invasivo.

Existen numerosas aplicaciones populares que utilizan el rojo en sus logotipos con fines similares. Por ejemplo:

- **YouTube**, que emplea el rojo para captar la atención y transmitir dinamismo en el ámbito audiovisual.
- **Pinterest**, que también utiliza el rojo para evocar creatividad, inspiración y acción.
- **Todoist**, una app de gestión de tareas, usa tonos rojizos para expresar eficiencia y control.

De manera similar, **TaskShare** utiliza este código visual para posicionarse como una herramienta intuitiva y orientada a mejorar la productividad diaria, transmitiendo con su identidad visual una propuesta clara y coherente.

El logotipo incluye también una tipografía moderna y legible, con un estilo limpio que representa la simplicidad de uso de la app. Se ha buscado que tanto el icono como el nombre de la aplicación puedan reconocerse fácilmente en distintos tamaños y plataformas (pantallas móviles, tiendas de apps, redes sociales, etc.).

En conjunto, el logo cumple con su función de representar visualmente la esencia de **TaskShare**: colaboración, organización y eficiencia.


*2-3Logo App Taskshare*

# 3 HERRAMIENTAS UTILIZADAS

Para el desarrollo de la aplicación **TaskShare**, se han empleado diversas herramientas, tecnologías y servicios que han permitido construir una solución moderna, robusta y alineada con las prácticas actuales del desarrollo móvil multiplataforma.

## 3.1 ANDROID STUDIO (IDE)

El entorno principal de desarrollo ha sido **Android Studio**, la plataforma oficial proporcionada por Google para el desarrollo de aplicaciones Android. Este IDE ofrece una amplia gama de herramientas integradas que facilitan tareas como la edición de código, el diseño de interfaces, la depuración, la gestión de dependencias, la emulación de dispositivos y el análisis de rendimiento.

Gracias a su integración con Gradle, Android Studio permite automatizar tareas de compilación y gestionar las diferentes versiones de la aplicación de forma sencilla. Además, ofrece compatibilidad total con Jetpack Compose, facilitando así el diseño de interfaces declarativas.

*3-1Android Studio*

## 3.2 LENGUAJE DE PROGRAMACIÓN: KOTLIN

La lógica de la aplicación ha sido desarrollada utilizando **Kotlin**, un lenguaje moderno, conciso y seguro, oficialmente recomendado por Google para el desarrollo en Android. Kotlin permite escribir menos código con mayor expresividad y seguridad frente a errores comunes, como los *null pointer exceptions*, gracias a su sistema de tipos null-safe.

La elección de Kotlin ha permitido aprovechar las ventajas de la programación funcional, la interoperabilidad con Java, y la compatibilidad total con librerías modernas del ecosistema Android.


*3-2Kotlin*

## 3.3 JETPACK COMPOSE (UI)

Para el diseño de la interfaz de usuario se ha utilizado **Jetpack Compose**, el toolkit declarativo de Google para el desarrollo de UI en Android. Compose simplifica el proceso de creación de interfaces, eliminando la necesidad de usar XML tradicional y permitiendo describir la interfaz directamente desde el código Kotlin.

Este enfoque no solo mejora la productividad, sino que también facilita la reutilización de componentes, el manejo del estado y la integración con otras APIs modernas. Gracias a Jetpack Compose, la aplicación cuenta con una interfaz limpia, responsive y fácilmente mantenible.

*3-3Jetpack Compose*

## 3.4 FIREBASE (BACKEND Y SERVICIOS EN LA NUBE)

La aplicación se apoya en la plataforma **Firebase** como backend en la nube, lo que proporciona una infraestructura escalable y segura sin necesidad de implementar servidores propios. Se han utilizado específicamente los siguientes servicios:

- **Firebase Firestore**: Base de datos NoSQL en tiempo real, utilizada para almacenar y sincronizar las tareas, listas y usuarios entre dispositivos. Firestore permite estructurar los datos en colecciones y documentos, facilitando la consulta eficiente y el manejo de permisos por usuario.
- **Firebase Authentication**: Sistema de autenticación que permite gestionar el inicio de sesión de usuarios mediante correo electrónico y contraseña. Proporciona una capa de seguridad y validación robusta con integración directa en Android.

Firebase ha sido clave para asegurar que los datos estén sincronizados en tiempo real y almacenados de forma segura, lo que mejora la experiencia de usuario y facilita la escalabilidad de la aplicación.


*3-4Firebase*

## 3.5 GITHUB (CONTROL DE VERSIONES)

Durante todo el proceso de desarrollo se ha utilizado **GitHub** como sistema de control de versiones. Gracias al uso de Git, ha sido posible mantener un historial detallado de los cambios realizados, crear ramas para nuevas funcionalidades, corregir errores sin afectar a la versión principal y colaborar de forma ordenada en el desarrollo.

Además, GitHub ha servido como respaldo en la nube del proyecto, garantizando la integridad del código fuente y facilitando el acceso desde diferentes dispositivos de desarrollo.


*3-5Logo GitHub*

## 3.6 DISPOSITIVO FÍSICO PARA LAS PRUEBAS

Para garantizar el correcto funcionamiento de la aplicación en un entorno real, se ha utilizado un **dispositivo físico Huawei** como plataforma de prueba principal. Esto ha permitido identificar comportamientos específicos del hardware, optimizar el rendimiento y verificar aspectos como la fluidez de las animaciones, la respuesta táctil y la persistencia de datos.

Aunque también se ha utilizado el emulador de Android Studio para pruebas preliminares, las pruebas en un dispositivo real han sido clave para asegurar una experiencia de usuario satisfactoria.

# 4 DISEÑO DEL SISTEMA

## 4.1 ARQUITECTURA GENERAL DE LA APP

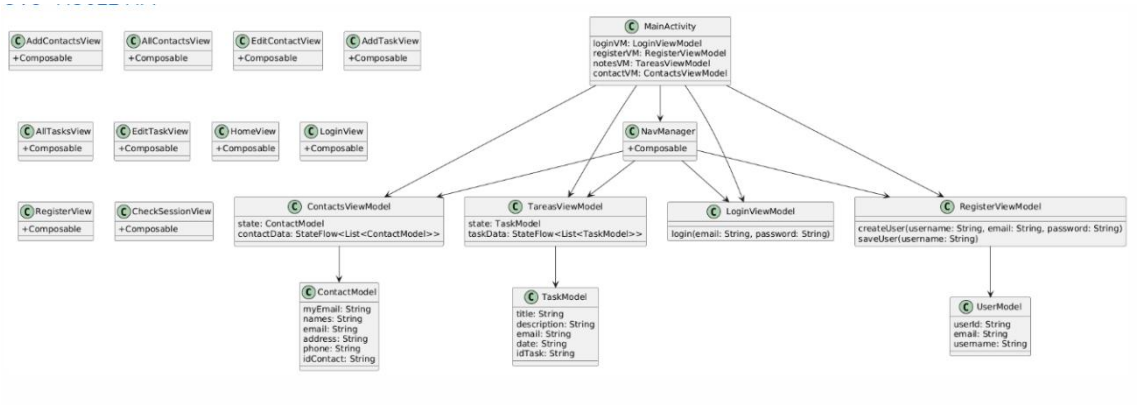
La aplicación TaskShare está diseñada siguiendo la arquitectura MVVM (Modelo-Vista-ViewModel), la cual permite una separación clara entre la lógica de negocio y la interfaz de usuario. Esta arquitectura ayuda a que el código sea más mantenible, testable y escalable.

- Modelo: Representa los datos y lógica del negocio. Se encarga de comunicarse con Firebase para operaciones como crear, leer, actualizar y eliminar tareas o usuarios.
- Vista: Son las pantallas desarrolladas con Jetpack Compose. Reciben datos del ViewModel y muestran la información al usuario.
- ViewModel: Actúa como intermediario entre la vista y el modelo. Gestiona el estado de la UI y expone los datos a través de variables observables.

## 4.2 DIAGRAMA UML

Muestro a continuación la imagen de mi UML realizada a partir del proyecto.

Como podemos ver, la estructura MVVM deja muchas clases sin enlazar, ya que son las vistas de la app y no hay una conexión directa entre ellas.




*4-1Diagrama UML App Taskshare*

## 4.3 DISEÑO DE LA BASE DE DATOS (ESTRUCTURA DE FIRESTORE)

Para el almacenamiento de datos se ha utilizado Firebase Firestore, una base de datos NoSQL en tiempo real. A diferencia de una base de datos relacional, Firestore almacena la información en forma de colecciones y documentos, lo que facilita el escalado y el acceso rápido desde aplicaciones móviles.

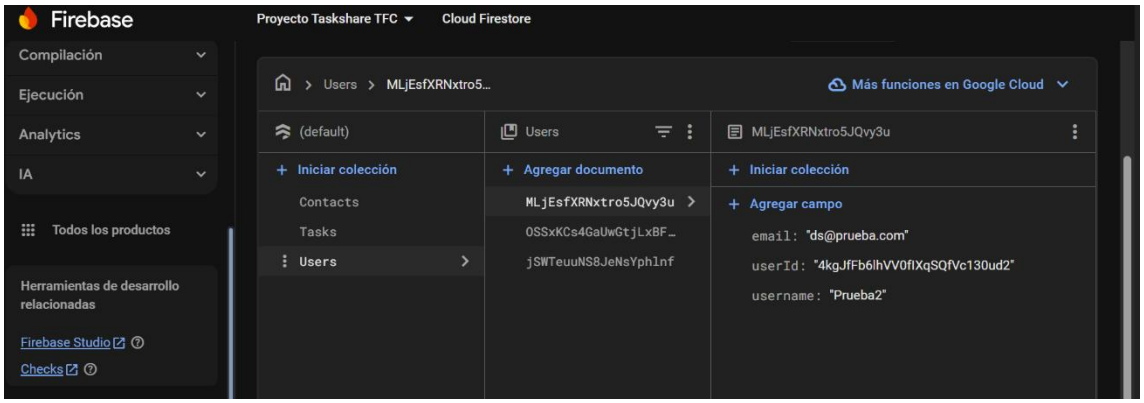
La estructura general de la base de datos utilizada en la aplicación es la siguiente:

```
Users (colección)
├── userId_001 (documento)
│   ├── userId: "userId_001"
│   ├── email: "usuario@example.com"
│   └── username: "nombreUsuario"

Tasks (colección)
├── taskId_001 (documento)
│   ├── idTask: "taskId_001"
│   ├── title: "Título de la tarea"
│   ├── description: "Descripción de la tarea"
│   ├── email: "usuario@example.com"   ← usuario que la creó
│   └── date: "2025-05-01"

Contacts (colección)
├── contactId_001 (documento)
│   ├── myEmail: "usuario@example.com"  ← usuario que guarda el contacto
│   ├── names: "Nombre del contacto"
│   ├── email: "contacto@example.com"
│   ├── address: "Calle Ejemplo 123"
│   └── phone: "123456789"
```

*4-2Estructura BBDD*

**Relaciones y estructura:**

- **USERS** contiene información básica de cada usuario, autenticado previamente con Firebase Auth.
- **TASKS** almacena las tareas relacionadas con cada usuario.
- **CONTACTS** representa contactos que tenemos agregados en nuestra app. Cada contacto que agregamos está relacionado con el contacto que lo ha creado.




*4-3Captura Firebase*

Esto permite filtrar fácilmente por tareas asignadas, por usuario, y obtener relaciones en tiempo real gracias a la sincronización automática de Firestore.

**Ventajas de este diseño:**

- Estructura simple y escalable.
- Fácil acceso y consulta con Firestore.
- Evita redundancia y permite mantener relaciones lógicas con identificadores.

# 5 DESARROLLO E IMPLEMENTACIÓN

La aplicación **Taskshare** fue desarrollada utilizando tecnologías modernas del ecosistema Android, priorizando la mantenibilidad del código, la experiencia de usuario y la integración con servicios en la nube. A continuación, se detallan los principales componentes técnicos implementados:

## 5.1 DESARROLLO DE LA INTERFAZ CON JETPACK COMPOSE

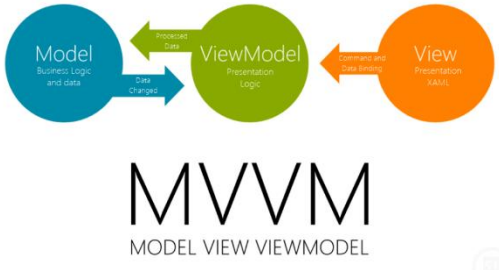
Se empleó **Jetpack Compose**, la herramienta declarativa de UI de Android, para construir una interfaz limpia, modular y altamente responsiva. Esta tecnología permite definir la apariencia de la aplicación mediante funciones denominadas *Composables*, lo que facilita la reutilización de componentes visuales y la adaptación a distintos tamaños de pantalla.

- Las pantallas principales como **login**, **registro**, **listado de tareas**, y **creación/edición de tareas** fueron desarrolladas de forma independiente, siguiendo principios de diseño limpio.
- Se aplicaron buenas prácticas de navegación con `NavController`, integrando rutas entre Composables de manera fluida.
- El diseño responde a criterios de usabilidad, con elementos intuitivos, accesibles y estéticamente coherentes con la identidad visual definida para Taskshare.

## 5.2 GESTIÓN DE ESTADO CON VIEWMODELS

Para mantener una separación clara entre la lógica de negocio y la interfaz, se utilizó el patrón **MVVM (Model-View-ViewModel)** con `ViewModel` y `mutableStateOf`.

- Cada pantalla cuenta con su propio ViewModel, responsable de manejar los **estados UI**, validaciones, interacciones del usuario y navegación.
- Se implementaron `mutableStateOf` y `LiveData` para reflejar los cambios de estado en tiempo real, como campos del formulario, visibilidad de errores o estados de carga.
- Esta arquitectura permitió una experiencia fluida, sin bloqueos, y una gestión eficaz del ciclo de vida de los componentes.




*5-1Estructura MVVM*

## 5.3 CONEXIÓN CON FIREBASE (AUTENTICACIÓN, GUARDADO DE TAREAS)

La integración con **Firebase** permitió ofrecer funcionalidades clave sin necesidad de un backend personalizado, acelerando el desarrollo y mejorando la escalabilidad:

- **Firebase Authentication** se utilizó para permitir el **inicio de sesión y registro de usuarios mediante correo electrónico y contraseña**, con control de errores y retroalimentación visual.
- **Cloud Firestore** fue la base de datos utilizada para **almacenar y sincronizar en tiempo real las tareas, datos de usuario y contactos compartidos**.
- Las operaciones CRUD (crear, leer, actualizar, eliminar) fueron gestionadas con colecciones y documentos jerárquicos en Firestore, garantizando rendimiento y consistencia.

## 5.4 VALIDACIONES Y LÓGICA (AÑADIR TAREAS, EDITAR, ELIMINAR)

La lógica interna de Taskshare contempla validaciones estrictas para asegurar la integridad de los datos introducidos por el usuario.

- Al **crear una tarea**, se verifica que los campos obligatorios como título, descripción y fecha estén completos. En caso contrario, se muestra un mensaje de error y se impide su envío.
- El usuario puede **editar tareas existentes** desde el listado. Los cambios se sincronizan automáticamente en Firestore.
- Las tareas pueden marcarse como **completadas** (mejora), y posteriormente ser **eliminadas** si ya no se desean conservar.
- Se aplicó un control de errores robusto para gestionar posibles fallos de conexión, autenticación o conflictos de escritura en la base de datos.

## 5.5 GESTIÓN DE FECHAS Y HORAS

Para la planificación efectiva de tareas, se integró un sistema intuitivo de selección de fechas y horarios:

- Se utilizaron los componentes DatePicker y TimePicker de Jetpack Compose, adaptados a Material Design, para permitir al usuario seleccionar la fecha y hora de ejecución de cada tarea.

- Estos valores se convierten y almacenan como **Timestamp** en Firestore, permitiendo su ordenación cronológica y uso futuro para recordatorios y notificaciones.

# 6 PRUEBAS Y VALIDACIÓN

Durante el desarrollo de **Taskshare**, se realizaron diferentes fases de prueba para asegurar la funcionalidad, usabilidad y estabilidad de la aplicación en diversos dispositivos. Las pruebas incluyeron tanto validaciones técnicas como recolección de feedback de usuarios reales, permitiendo corregir errores, ajustar la experiencia de usuario y mejorar la calidad general del producto.


*6-1Pruebas y validación*

## 6.1 ¿CÓMO PROBASTE LA APP?

Las pruebas de la aplicación se realizaron de forma **manual**, verificando cada componente y flujo funcional en distintos entornos. Se optó por un enfoque de prueba exploratoria e iterativa, en el que se evaluaron cada una de las funcionalidades de manera individual:

- Registro e inicio de sesión de usuarios
- Creación, edición, eliminación y marcaje de tareas
- Validaciones de formularios
- Persistencia de datos en Firestore
- Navegación entre pantallas
- Comportamiento visual en diferentes modos (claro/oscuro) (mejora)

Además, se realizaron pruebas de conexión y sincronización en tiempo real con Firebase para validar que los datos se almacenaran y actualizaran correctamente.

## 6.2 ¿QUÉ DISPOSITIVOS/EMULADORES USASTE?

Para garantizar la compatibilidad en distintos tipos de hardware y versiones de Android, se realizaron pruebas en los siguientes entornos:

- **Dispositivo físico:** Huawei P30 Lite (Android 10)
- **Emulador:** Pixel 4 API 30 (Android 11) a través de Android Studio

El uso combinado de dispositivo real y emulador permitió identificar diferencias de comportamiento y rendimiento que podrían surgir en entornos reales de uso.

## 6.3 ¿QUÉ ERRORES ENCONTRASTE Y SOLUCIONASTE?

Durante el proceso de pruebas, se detectaron varios errores funcionales y visuales, los cuales fueron corregidos con ajustes en el código y validaciones adicionales:

- **Validaciones de campos vacíos:**
  En versiones iniciales, era posible crear tareas sin completar todos los campos requeridos. Se implementaron validaciones explícitas con mensajes de error amigables para evitar este comportamiento.
- **Acceso a documentos inexistentes en Firestore:**
  Al intentar acceder a documentos que ya habían sido eliminados o mal referenciados, se producían errores de lectura. Se añadieron comprobaciones previas de existencia (document.exists()) para manejar estos casos de forma segura.
- **Problemas con el cambio de tema (modo oscuro):**
  Algunos componentes no se adaptaban correctamente al modo oscuro, dificultando la lectura. Se revisaron los estilos y se ajustaron colores para asegurar la compatibilidad visual en ambos modos. (No solucionado. Marcado como mejora).
- **Problemas de permisos en Firestore:**

  Cada vez que se quería acceder a alguna colección había que otorgarle permisos para la creación, lectura, actualización y borrado de datos.
- **Problemas con la puntuación:**

  Se intentó implementar la funcionalidad de los puntos, pero complicaba demasiado la app y no había tiempo para poder desarrollarlo. Se optó por tener una app simple y sencilla en el que funciona todo correctamente. Esta app está encaminada al desarrollo de múltiples mejoras, una de ellas es la puntuación.


*6-2Error*

## 6.4 FEEDBACK DE USUARIOS

Se realizó una prueba con tres usuarios (compañeros del entorno doméstico), quienes proporcionaron sugerencias útiles:

- Añadir recordatorios.
- Simplificar la navegación entre pantallas.
- Mejorar la visibilidad del botón de crear tarea.
- Poder comunicarse por Whatsapp.
- Mejorar la imagen de la app.
- Implementar un chat propio dentro de la propia app.

Estas son algunas de las ideas que me dieron los usuarios. Naturalmente no podía poner todas, ya que sería bastante pesado. Decidí seleccionar las que mejor encajaban.

# 7 ENTREGA Y DOCUMENTACIÓN

A continuación, se detallan los materiales que se entregan junto con esta memoria, los cuales constituyen el conjunto completo de documentación y productos desarrollados en el marco del Trabajo de Fin de Grado (TFG) titulado **“TaskShare”**.

*7-1Documentación*

## 7.1 MEMORIA DEL PROYECTO

Se entrega este documento en formato PDF, que recoge de forma detallada todas las fases del desarrollo del proyecto. La memoria incluye el análisis del problema, los objetivos planteados, el estudio del estado del arte, la metodología utilizada, el proceso de diseño e implementación, así como las conclusiones alcanzadas. También se incluyen diagramas, capturas de pantalla, tablas y referencias bibliográficas que enriquecen el contenido y aportan una visión completa del proyecto.

## 7.2 DIARIO DE DESARROLLO

Junto a la memoria, se adjunta un documento en formato PDF que constituye un **diario de desarrollo**. En él se han ido registrando cronológicamente todas las actividades llevadas a cabo durante el proceso de trabajo, incluyendo:

- Fechas de trabajo.
- Número de horas dedicadas por jornada.
- Actividades realizadas (programación, diseño, pruebas, etc...).
- Reflexiones sobre los avances y obstáculos encontrados.

Este diario permite mostrar de forma transparente el esfuerzo, la planificación y la evolución del proyecto desde sus primeras ideas hasta su finalización.

## 7.3 CÓDIGO FUENTE

Se entrega un archivo comprimido en formato .zip que contiene la totalidad del **código fuente de la aplicación TaskShare**, incluyendo:

- Código del frontend
- Código del backend

- Estructura de la base de datos

El código está organizado en carpetas según buenas prácticas de desarrollo y cuenta con comentarios explicativos en las secciones más relevantes, para facilitar su comprensión y reutilización.

## 7.4 PRESENTACIÓN POWER POINT

Se entrega un archivo power point en el que se puede ver la presentación de este proyecto para el día de la defensa.

# 8 CONCLUSIONES Y MEJORAS FUTURAS

El desarrollo de **Taskshare** representó una experiencia valiosa tanto a nivel técnico como personal. A través del diseño e implementación de la aplicación, fue posible consolidar conocimientos, enfrentar desafíos reales del desarrollo móvil, y crear una solución funcional centrada en la organización y colaboración entre usuarios.


*8-1Conclusiones*

## 8.1 LECCIONES APRENDIDAS

Durante este proyecto se adquirieron aprendizajes significativos, tanto en el plano técnico como en el metodológico:

- Dominio de **Jetpack Compose** como herramienta moderna para el desarrollo de interfaces en Android.
- Aplicación del patrón **MVVM**, reforzando la separación de responsabilidades y facilitando el mantenimiento del código.
- Integración práctica de **servicios en la nube** mediante Firebase, aprendiendo a manejar autenticación, bases de datos en tiempo real y control de errores.
- Importancia del **feedback temprano de usuarios** como factor clave para mejorar la usabilidad.

También se reforzó la capacidad de **resolver errores de manera autónoma**, depurar de forma efectiva y documentar el avance del proyecto de forma clara y ordenada.

## 8.2 ELEMENTOS QUE FUNCIONARON BIEN

Entre los aspectos más exitosos del proyecto se destacan:

- **Estabilidad general de la aplicación**, con un funcionamiento fluido y sin cierres inesperados.
- **Gestión de tareas eficiente y sincronizada** en la nube, con tiempos de respuesta rápidos.
- **Interfaz intuitiva y moderna**, gracias al uso de Jetpack Compose y una paleta de colores adecuada.
- **Arquitectura bien estructurada**, permitiendo escalar funcionalidades futuras con facilidad.

## 8.3 ÁREAS DE MEJORA CON MÁS TIEMPO DISPONIBLE

Si se hubiese contado con más tiempo o recursos, habría sido posible mejorar y ampliar varios aspectos:

- **Refinar aún más la experiencia de usuario**, especialmente en transiciones, animaciones y personalización.
- **Automatizar pruebas (testing unitario e instrumental)** para garantizar mayor robustez ante posibles cambios.
- **Implementar control de errores más detallado**, incluyendo mensajes contextuales y manejo de estados offline.
- **Mejorar la gestión de usuarios y permisos**, especialmente en tareas compartidas.
- **Implementación de la puntuación,** en el que los usuarios puedan "jugar" y que la realización de las tareas sea un compromiso y así evitar la procrastinación.
- **Modo oscuro y modo claro,** para aquellos usuarios que son más exigentes para con su vista.

*8-2Tiempo*

## 8.4 POSIBLES FUNCIONALIDADES FUTURAS

Taskshare tiene un alto potencial de evolución. Algunas funcionalidades que podrían incorporarse en futuras versiones incluyen:

- **Notificaciones push**: para recordar tareas próximas a vencer o colaboraciones pendientes.
- **Colaboración entre usuarios**: permitir compartir tareas o listas con otros contactos registrados. Llevar seguimiento de las puntuaciones.
- **Etiquetas y categorías personalizadas**: para clasificar tareas por tema, prioridad o contexto.
- **Modo oscuro adaptativo**: basado en la preferencia del sistema o configurable por el usuario.
- **Sincronización con calendarios externos**: como Google Calendar, para una planificación más integral.
- **Autenticación biométrica**: para mejorar la seguridad y accesibilidad de la app.

- **Estadísticas de productividad**: visualización de tareas completadas, hábitos y metas alcanzadas.
- **Implementación de la puntuación:** en el que los usuarios puedan "jugar" y que la realización de las tareas sea un compromiso y así evitar la procrastinación.
- **Mejora de la imagen de la app.**

*8-3Funcionalidades futuras*

# 9 BIBLIOGRAFÍA


- https://firebase.google.com/ (Ayuda base de datos)
- https://stackoverflow.com/ (Ayuda con problemas)
- https://www.google.com/ (Imágenes)
- https://github.com/ (Control de versiones)
- https://lottiefiles.com/ (Animaciones)
- https://www.flaticon.es/ (Iconos)
- https://www.canva.com/ (Logo App)
- https://chatgpt.com/ (Resolución de errores)