ILERNA

CFGS DESARROLLO DE APLICACIONES MULTIPLATAFORMA
Proyecto Final de Ciclo

TIDYTASK

APLICACIÓN ANDROID PARA LA GESTIÓN DE TAREAS PERSONALES.

Autor: Andrea Catalán Menacho

Tutor: Javier Martin Rivero

Fecha de entrega: 28/05/2025

Convocatoria: 2024 2025

Índice

1. Introducción……..2
2. Motivación…….3
3. Abstract…….3
4. Objetivo general……..4
5. Objetivos específicos……4
6. Metodología……4
7. Estimación de recursos y planificación……6
8. Requisitos funcionales……9
9. Requisitos no funcionales……..9
10. Modelo Entidad-Relación……10
11. Casos de uso…….11
    - 11.1 Registrarse
    - 11.2 Iniciar sesión
    - 11.3 Crear tarea
    - 11.4 Editar tarea
    - 11.5 Eliminar tarea
    - 11.6 Visualizar tareas
    - 11.7 Programar recordatorio
    - 11.8 Recibir notificación
12. Diseño……..15
    - 12.1 Pantalla de registro de usuario
    - 12.2 Pantalla de inicio / lista de tareas
    - 12.3 Pantalla de añadir / editar tarea
    - 12.4 Pantalla de confirmación de borrado
13. Prototipos de código……17
    - 13.1 Clase Usuario
    - 13.2 Clase Tarea
    - 13.3 Clase Categoría
    - 13.4 Clase Recordatorio
14. Pruebas unitarias con JUnit……20
    - 14.1 Prueba creación de tarea
    - 14.2 Prueba inicio de sesión de usuario

Introducción

Actualmente, la organización personal es un aspecto fundamental para el bienestar y la productividad, tanto en el ámbito académico como en el profesional. A lo largo del ciclo formativo de Desarrollo de Aplicaciones Multiplataforma he tenido la oportunidad de adquirir conocimientos en programación, diseño de interfaces, bases de datos y gestión de proyectos, los cuales me han permitido abordar el desarrollo de una aplicación funcional y útil en el día a día.

El proyecto que presento, titulado **TidyTask**, consiste en el desarrollo de una aplicación Android destinada a la gestión de tareas personales. Su objetivo es ayudar a los usuarios a planificar sus actividades, recibir recordatorios y priorizar responsabilidades mediante una interfaz sencilla y accesible.

Esta aplicación ha sido concebida desde una necesidad real de mejorar mi organización diaria, y representa además una forma de integrar y aplicar de forma práctica los conocimientos adquiridos durante los dos años de formación. A lo largo de este documento se detallarán los objetivos planteados, la metodología empleada, las herramientas utilizadas, así como el análisis, diseño, desarrollo y pruebas del proyecto.

Motivación

La idea de este proyecto surgió por una necesidad personal: a lo largo de estos dos años de ciclo he tenido dificultades para organizarme, recordar fechas importantes o tareas pendientes, tanto en el ámbito académico como en el personal. Aunque existen muchas aplicaciones de listas de tareas, la mayoría me resultaban complicadas, poco visuales o demasiado cargadas de funciones que no necesitaba.

Por eso decidí crear una aplicación sencilla, intuitiva y que se adaptara a lo que yo, como estudiante y persona ocupada, realmente necesitaba: apuntar tareas rápidamente, organizarlas por categorías o prioridad, y recibir recordatorios en el momento justo.

Además, este proyecto me ha permitido aplicar de forma práctica todo lo que he aprendido durante el ciclo: desde el diseño de la base de datos hasta el uso de notificaciones y diseño de interfaces en Android. Ha sido una forma de cerrar el ciclo con un proyecto útil, funcional y con el que me siento identificada.

Abstract

TidyTask is an Android application developed as a final project for the Multiplatform Application Development course. The app allows users to create, organize, and manage their daily tasks through a simple and intuitive interface. It also includes reminders using AlarmManager to help users stay on top of their priorities.

The project was built using Kotlin, XML, and SQLite for local data storage. All the logic runs locally, with no internet connection required. The goal is to provide a lightweight and practical tool for personal organization, especially useful for students and professionals with tight schedules.

Objetivo general

Desarrollar una aplicación Android que permita gestionar tareas personales de forma sencilla, rápida y sin necesidad de conexión a internet, incorporando recordatorios y organización por categorías.

Objetivos específicos

- Diseñar una interfaz clara e intuitiva en XML, adaptada a dispositivos móviles.
- Implementar un sistema de creación, edición y eliminación de tareas.
- Almacenar los datos de forma local utilizando SQLite.
- Organizar las tareas por prioridad y/o categoría.
- Incorporar recordatorios mediante AlarmManager.
- Garantizar que la aplicación funcione de forma completamente offline.
- Aplicar buenas prácticas de desarrollo y usabilidad móvil.

Metodología

Para llevar a cabo este proyecto he seguido una metodología ágil basada en Scrum, adaptada a un trabajo individual. Esta metodología me ha permitido organizar las tareas en fases semanales, priorizar funcionalidades y adaptarme a los cambios o dificultades que han surgido durante el desarrollo.

En primer lugar, realicé una fase de análisis para definir los objetivos del proyecto, los requisitos principales y las tecnologías necesarias. A partir de ahí, dividí el trabajo en bloques semanales donde cada uno se centraba en una parte del desarrollo: diseño de la interfaz, creación de la base de datos, implementación de la lógica principal, pruebas, etc.

Para organizarme utilicé Trello, una herramienta que permite gestionar tareas mediante tableros visuales. En él dividí las tareas en columnas como “Por hacer”,

"En proceso" y "Finalizado", lo que me ayudó a llevar un control visual del progreso y saber en todo momento en qué punto me encontraba.

Durante el desarrollo también hice pequeñas validaciones funcionales para asegurar que lo que iba implementando cumplía los objetivos previstos, realizando ajustes cuando fue necesario. Esta flexibilidad ha sido clave para poder avanzar de forma progresiva y ordenada.

A continuación se detallan las principales tecnologías y herramientas utilizadas durante el desarrollo del proyecto:

- **Kotlin**

Lenguaje principal de programación con el que se ha desarrollado toda la lógica de la aplicación. Kotlin permite escribir código claro, conciso y está plenamente integrado en el desarrollo Android actual.

- **Android Studio**

Entorno de desarrollo oficial de Android, utilizado para diseñar las interfaces, programar el comportamiento de la app, ejecutar pruebas y generar los APK de instalación.

- **XML**

Lenguaje de marcado utilizado para el diseño de las interfaces gráficas (layouts). Permite estructurar y definir visualmente cada pantalla de la aplicación.

- **SQLite**

Sistema de base de datos local integrado en Android, utilizado para almacenar y gestionar las tareas del usuario sin necesidad de conexión a internet. La estructura se ha definido a través de sentencias SQL y clases auxiliares para su gestión.

- **AlarmManager**

Componente de Android que permite programar notificaciones automáticas. Se ha utilizado para que el usuario reciba recordatorios de sus tareas en el momento indicado.

- **Figma**

Herramienta de diseño utilizada para crear los mockups de la aplicación y planificar visualmente la estructura de las pantallas antes de implementarlas.

- **Trello**

Herramienta de gestión de tareas utilizada para organizar el proyecto siguiendo una metodología ágil. Se han definido las fases del desarrollo y se ha hecho seguimiento semanal del progreso.

- **Canva**

Herramienta de diseño gráfico que ha servido como inspiración para algunos elementos visuales de la interfaz, especialmente para seleccionar ideas de fondos, colores y estilos suaves.

- **Draw.io**

Aplicación online utilizada para realizar el modelo entidad-relación y los diagramas de casos de uso, ya que permite generar esquemas visuales de forma rápida y organizada.

- **GitHub**

Plataforma utilizada para almacenar el código fuente del proyecto, mantener un control de versiones y disponer de una copia de respaldo del trabajo realizado.

- **Google y YouTube**

Fuentes principales de documentación, resolución de dudas y aprendizaje de conceptos técnicos. Han sido clave para avanzar en el desarrollo y solucionar errores.

Estimación de recursos y planificación

La planificación del proyecto se llevó a cabo a lo largo de varias semanas, organizando el trabajo por fases según las funcionalidades principales a desarrollar. Desde el inicio, se definieron los bloques de tareas más importantes, asignando tiempos aproximados a cada uno de ellos para asegurar una buena distribución del trabajo a realizar.

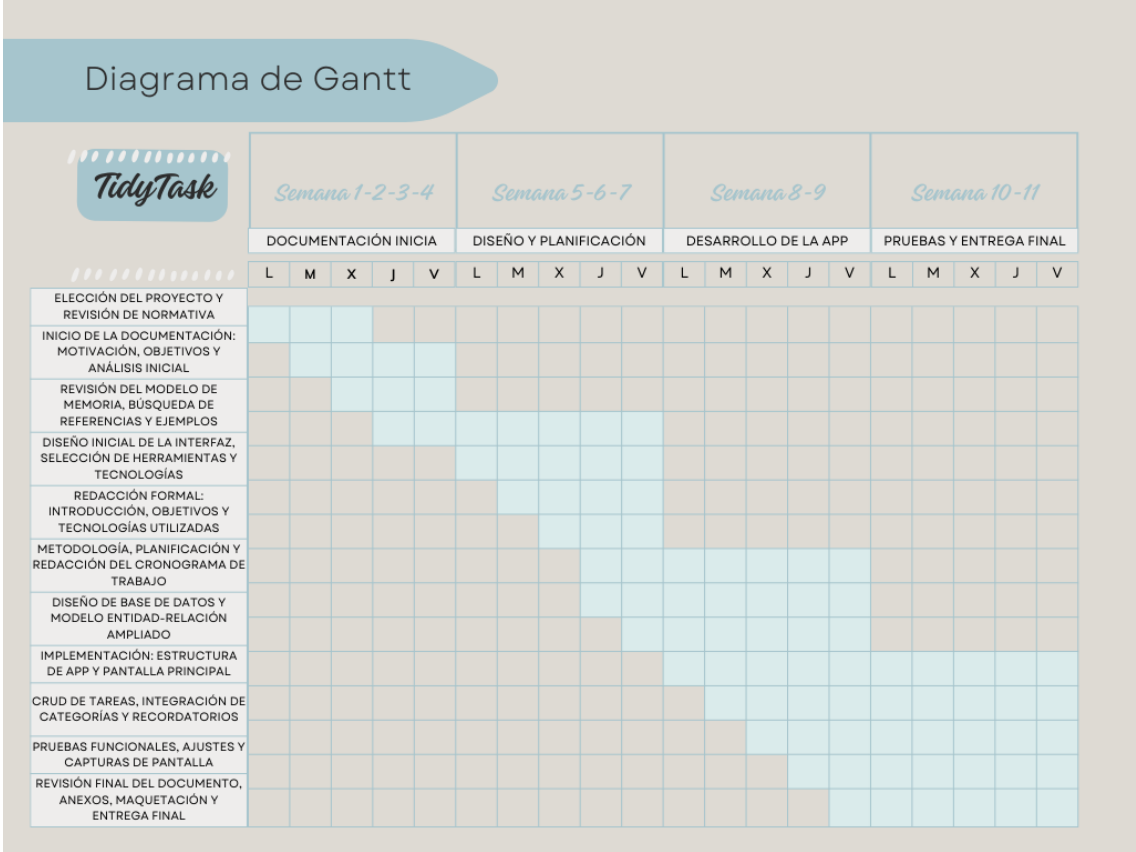
El desarrollo comenzó con una fase inicial de análisis y diseño, en la que se definieron los requisitos, se diseñó la base de datos y se realizaron los primeros bocetos de la interfaz. A continuación, se pasó a la implementación progresiva de las pantallas, la lógica de negocio y la base de datos en SQLite. Finalmente, se integraron las notificaciones con AlarmManager y se realizaron las pruebas y ajustes finales.

Para controlar el avance del proyecto, se elaboró un **diagrama de Gantt** con la estimación inicial de tiempos, el cual se fue actualizando a medida que el desarrollo progresaba. En este diagrama se reflejan las principales fases del proyecto y su duración.

Además, se utilizó Trello como herramienta de organización para dividir las tareas en columnas y seguir el flujo de trabajo. Esta combinación me permitió mantener una visión clara del progreso y poder adaptarme a los cambios necesarios a lo largo del desarrollo.

Semana	**Fechas**	**Tareas principales**
1	**11 – 17 de marzo**	Elección del proyecto, primera reunión con el tutor y revisión de normativa
2	**18 – 24 de marzo**	Inicio de la documentación: motivación, objetivos y análisis inicial
3	**25 – 31 de marzo**	Revisión del modelo de memoria, búsqueda de referencias y ejemplos
4	**1 – 7 de abril**	Diseño inicial de la interfaz, selección de herramientas y tecnologías
5	**8 – 14 de abril**	Redacción formal: introducción, objetivos y tecnologías utilizadas
6	**15 – 21 de abril**	Metodología, planificación y redacción del cronograma de trabajo
7	**22 – 28 de abril**	Diseño de base de datos y modelo entidad-relación ampliado

8	**29 abril – 5 mayo**	Implementación: estructura de app y pantalla principal
9	**6 – 12 de mayo**	CRUD de tareas, integración de categorías y recordatorios
10	**13 – 17 de mayo**	Pruebas funcionales, ajustes y capturas de pantalla
11	**18 – 20 de mayo**	Revisión final del documento, anexos, maquetación y entrega final



Requisitos funcionales

• La aplicación debe permitir crear, editar y eliminar tareas.

• La aplicación debe permitir clasificar las tareas por categoría o prioridad.

• La aplicación debe permitir establecer una fecha y hora para cada tarea.

• La aplicación debe notificar al usuario cuando se acerque la fecha límite de una tarea.

• La aplicación debe guardar los datos de forma persistente usando una base de datos local (SQLite).

• La aplicación debe mostrar un listado de tareas pendientes, ordenadas por fecha o prioridad.

• La aplicación debe permitir marcar una tarea como completada.

Requisitos no funcionales

• La aplicación debe funcionar sin conexión a internet.

• La interfaz debe ser clara, intuitiva y fácil de usar.

• El tiempo de respuesta de la aplicación debe ser inferior a 2 segundos por acción.

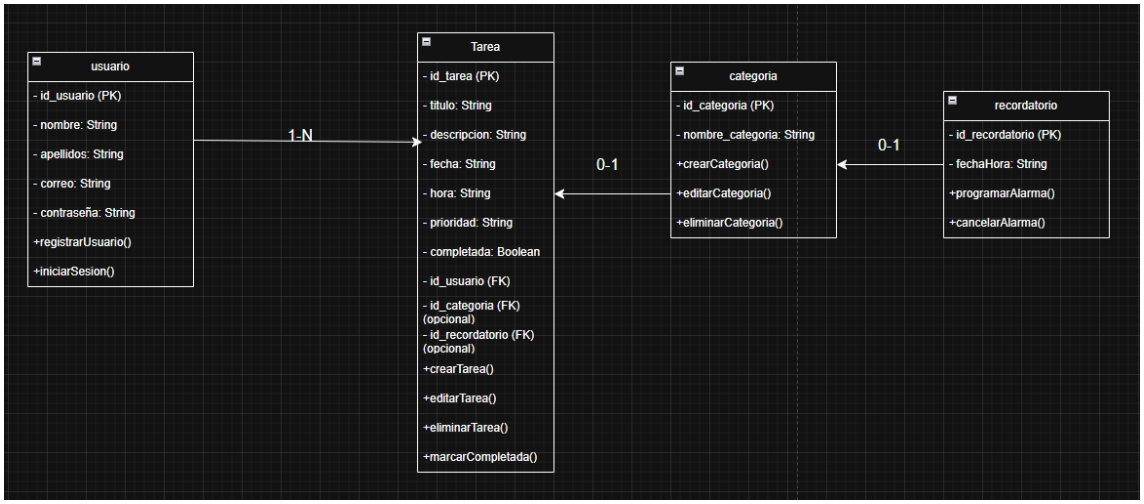
• La aplicación debe estar diseñada exclusivamente para dispositivos Android.

• Los datos deben guardarse de forma segura en la base de datos local del dispositivo.

• La aplicación debe tener un consumo de batería y recursos optimizado.

• El diseño debe adaptarse a distintos tamaños de pantalla.

Modelo Entidad-Relación



El modelo E-R de la aplicación TidyTask se compone de cuatro entidades principales: Usuario, Tarea, Categoría y Recordatorio.

La entidad Usuario almacena los datos necesarios para gestionar el acceso y las tareas de forma personalizada.

La entidad Tarea contiene toda la información relativa a cada actividad: título, descripción, fecha, hora, prioridad, estado de completado y los identificadores de categoría y recordatorio asociados.

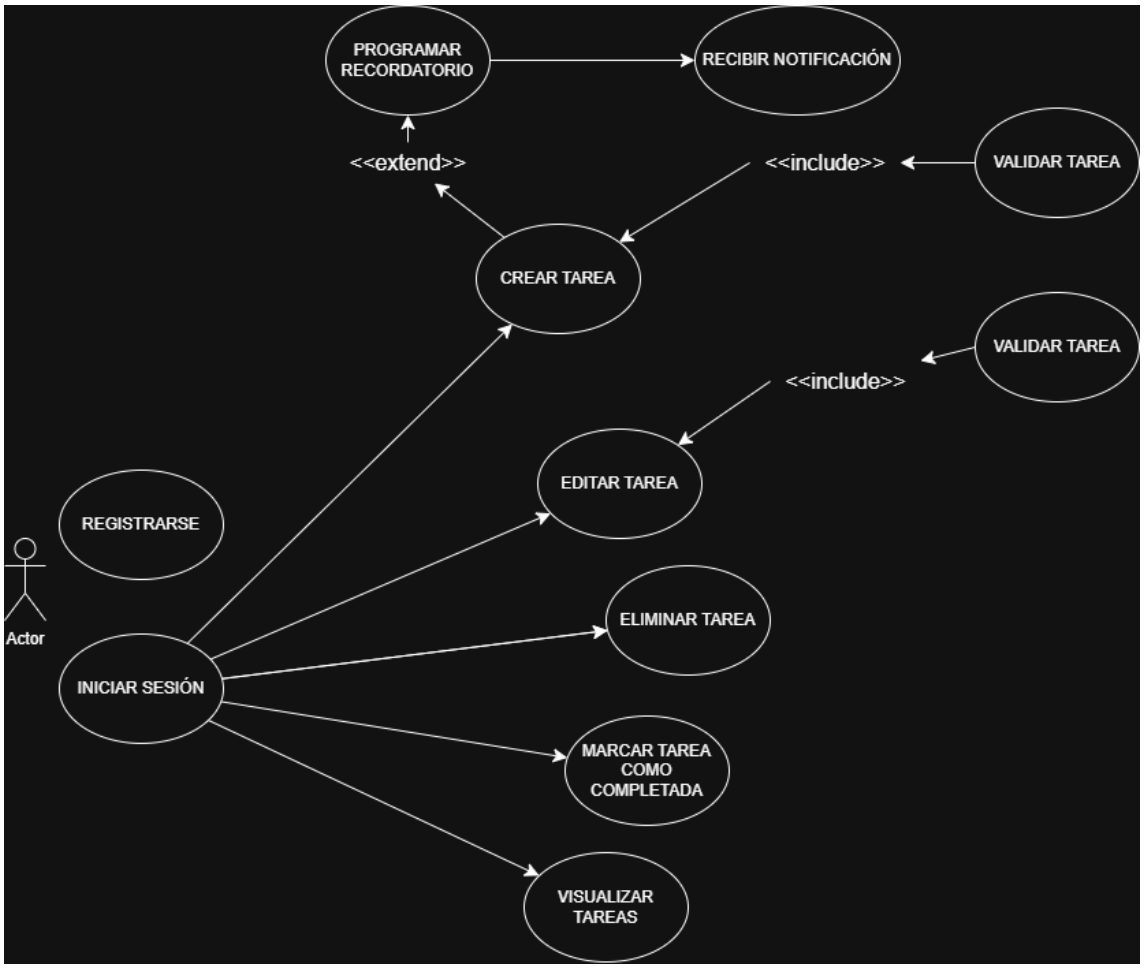
La entidad Categoría permite clasificar las tareas según su tipo o nivel de importancia, mejorando la organización para el usuario.

La entidad Recordatorio gestiona la fecha y hora en la que el sistema debe notificar al usuario acerca de una tarea pendiente.

Cada usuario puede tener varias tareas asociadas (relación 1:N). Cada tarea pertenece a una única categoría y puede tener asociado un recordatorio que envíe una notificación.

Con esta estructura garantizamos una correcta gestión de las tareas, facilitamos su clasificación, y mantenemos la información de forma persistente en una base de datos

local (SQLite).



Caso de Uso: Registrarse

- **Nombre:** Registrarse
- **Actor principal:** Usuario
- **Precondiciones:** El usuario no debe estar registrado previamente.
- **Descripción breve:** Permite a un nuevo usuario crear una cuenta en la aplicación proporcionando su nombre, correo electrónico y contraseña.
- **Flujo principal:**
    1. El usuario accede a la pantalla de registro.
    2. Rellena los campos requeridos: nombre, correo y contraseña.
    3. El sistema valida los datos.
    4. El sistema crea una nueva cuenta y muestra un mensaje de éxito.
- **Postcondiciones:** El usuario queda registrado en la base de datos.

Caso de Uso: Iniciar sesión

- **Nombre:** Iniciar sesión
- **Actor principal:** Usuario
- **Precondiciones:** El usuario debe estar registrado.
- **Descripción breve:** Permite al usuario acceder a la aplicación mediante su correo electrónico y contraseña.
- **Flujo principal:**
    1. El usuario introduce su correo electrónico y contraseña en la pantalla de inicio de sesión.
    2. El sistema valida las credenciales.
    3. Si son correctas, se inicia sesión y se redirige al usuario a la pantalla principal.
- **Postcondiciones:** El usuario accede a su área de gestión de tareas.

Caso de Uso: Crear tarea

- **Nombre:** Crear tarea
- **Actor principal:** Usuario
- **Precondiciones:** El usuario debe haber iniciado sesión.
- **Descripción breve:** Permite al usuario crear una nueva tarea introduciendo un título, descripción, fecha, hora y prioridad.
- **Flujo principal:**
    1. El usuario accede a la opción "Crear tarea".
    2. Introduce los datos requeridos en el formulario.
    3. El sistema incluye la acción de validar campos.
    4. Si los datos son correctos, se guarda la tarea en la base de datos.
- **Postcondiciones:** La nueva tarea queda registrada en el sistema.

Caso de Uso: Editar tarea

- **Nombre:** Editar tarea
- **Actor principal:** Usuario
- **Precondiciones:** El usuario debe haber iniciado sesión y tener tareas existentes.
- **Descripción breve:** Permite modificar los datos de una tarea existente.
- **Flujo principal:**
    1. El usuario selecciona una tarea de su lista.
    2. Accede a la opción "Editar".
    3. Modifica los campos deseados.
    4. El sistema incluye la acción de validar campos.
    5. Se guardan los cambios.
- **Postcondiciones:** La tarea se actualiza con la nueva información.

Caso de Uso: Eliminar tarea

- **Nombre:** Eliminar tarea
- **Actor principal:** Usuario
- **Precondiciones:** El usuario debe haber iniciado sesión.
- **Descripción breve:** Permite eliminar una tarea de la base de datos.
- **Flujo principal:**
    1. El usuario selecciona una tarea de la lista.
    2. Pulsa la opción "Eliminar".
    3. El sistema solicita confirmación.
    4. Si confirma, la tarea se elimina.
- **Postcondiciones:** La tarea desaparece de la lista de tareas.

Caso de Uso 6: Visualizar tareas

- **Nombre:** Visualizar tareas
- **Actor principal:** Usuario
- **Precondiciones:** El usuario debe haber iniciado sesión.
- **Descripción breve:** Muestra una lista de todas las tareas del usuario.
- **Flujo principal:**
    1. El usuario accede a la pantalla de inicio.
    2. El sistema carga y muestra la lista de tareas ordenadas.
- **Postcondiciones:** Las tareas se visualizan en pantalla.

Caso de Uso: Programar recordatorio

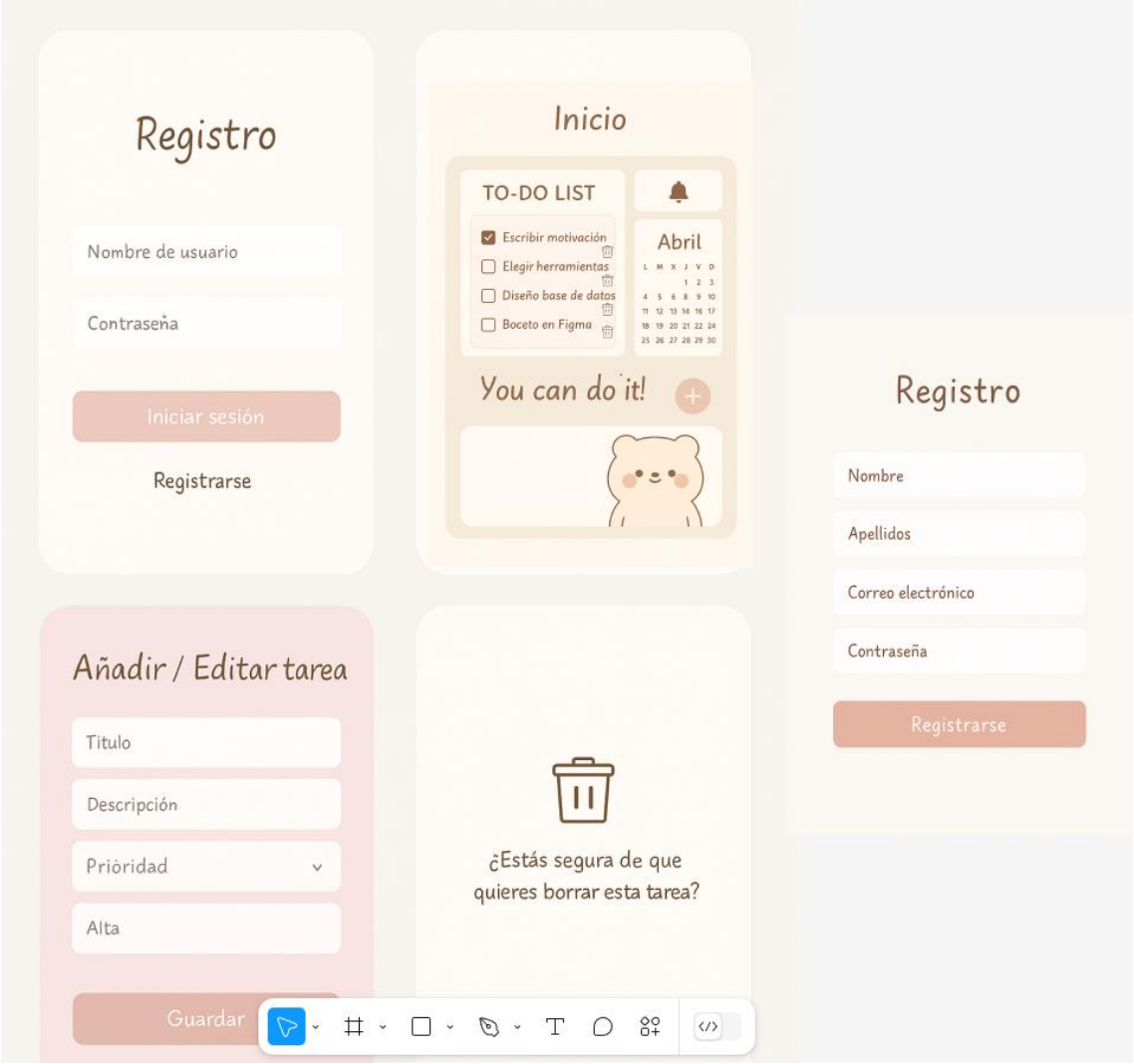
- **Nombre:** Programar recordatorio
- **Actor principal:** Usuario
- **Precondiciones:** El usuario debe haber iniciado sesión y estar creando o editando una tarea.
- **Descripción breve:** Permite asociar una notificación futura a una tarea.
- **Flujo principal:**
    1. Durante la creación o edición de una tarea, el usuario activa la opción de recordatorio.
    2. Introduce la fecha y hora para la notificación.
    3. El sistema guarda esta programación.
- **Postcondiciones:** El sistema lanzará una notificación en la fecha y hora indicada.

Caso de Uso: Recibir notificación

- **Nombre:** Recibir notificación
- **Actor principal:** Sistema

- **Precondiciones:** Debe existir un recordatorio programado.
- **Descripción breve:** El sistema lanza una notificación en la fecha y hora indicada por el usuario.
- **Flujo principal:**
    1. Llega la fecha y hora programada para una tarea.
    2. El sistema genera automáticamente una notificación para el usuario.
- **Postcondiciones:** El usuario recibe una alerta en su dispositivo.

DISEÑO



Pantalla de registro de usuario

Esta es la pantalla inicial donde el usuario puede registrarse. He incluido campos básicos como nombre, apellidos, correo y contraseña. El diseño sigue la línea estética del resto de la app, con colores suaves y un estilo limpio.

Pantalla de inicio / lista de tareas

Esta pantalla es el centro principal de la aplicación. Aquí el usuario puede ver sus tareas, organizadas por fecha, con opción para marcarlas como completadas (check) o eliminarlas (icono de papelera). También se muestra un calendario y una campanita para representar los recordatorios. Todo está integrado dentro de un contenedor tipo “planner visual” que refuerza la idea de organización y armonía.

Pantalla de añadir / editar tarea

Esta pantalla permite crear una nueva tarea o modificar una existente. El usuario podrá introducir el título, descripción, fecha, hora y prioridad. He querido que esta vista sea lo más intuitiva posible, dejando el foco en los campos importantes botones que llevan a cabo la acción.

Pantalla de confirmación de borrado

Esta pequeña ventana aparece cuando el usuario intenta borrar una tarea. Sirve como medida de seguridad para evitar eliminaciones accidentales

PROTOTIPOS DE CÓDIGO

activity_main.xml | Usuario.kt | Tarea.kt | Categori

```kotlin
package com.example.tidytask

class Usuario(
    var idUsuario: Int,
    var nombre: String,
    var correo: String,
    var contrasena: String
) {
    fun registrarUsuario(): Boolean {

        return true
    }

    fun iniciarSesion(): Boolean {

        return true
    }
}
```

La clase Usuario define los atributos principales de un usuario registrado, y proporciona funciones básicas para registrar nuevos usuarios e iniciar sesión en la aplicación.

activity_main.xml | Usuario.kt | Tarea.kt | Categoria.kt | Recordatorio.kt | Mai

```kotlin
package com.example.tidytask

class Tarea(
    var idTarea: Int,
    var titulo: String,
    var descripcion: String,
    var fecha: String,
    var hora: String,
    var prioridad: String,
    var completada: Boolean = false,
    var idUsuario: Int,
    var idCategoria: Int? = null,
    var idRecordatorio: Int? = null
) {
    fun crearTarea(): Boolean {

        return true
    }

    fun editarTarea(nuevoTitulo: String, nuevaDescripcion: String): Boolean {
        titulo = nuevoTitulo
        descripcion = nuevaDescripcion
        return true
    }

    fun eliminarTarea(): Boolean {

        return true
    }

    fun marcarCompletada(): Boolean {
        completada = true
        return true
    }
}
```

La clase Tarea contiene la estructura básica de una tarea, incluyendo atributos como título, descripción, fecha y prioridad. La tarea, podrá ser categorizada o recordada si el usuario así lo desea pero no es obligatorio. Además, implementa funciones para crear, editar, eliminar y marcar tareas como completadas.

activity_main.xml | Usuario.kt | Tarea.kt | Categoria.kt

```kotlin
package com.example.tidytask

class Categoria(
    var idCategoria: Int,
    var nombreCategoria: String
) {
    fun crearCategoria(): Boolean {

        return true
    }

    fun eliminarCategoria(): Boolean {

        return true
    }
}
```

La clase Categoría permitirá agrupar tareas bajo diferentes categorías creadas por el usuario, facilitando su organización. Incluye funciones para crear y eliminar categorías.

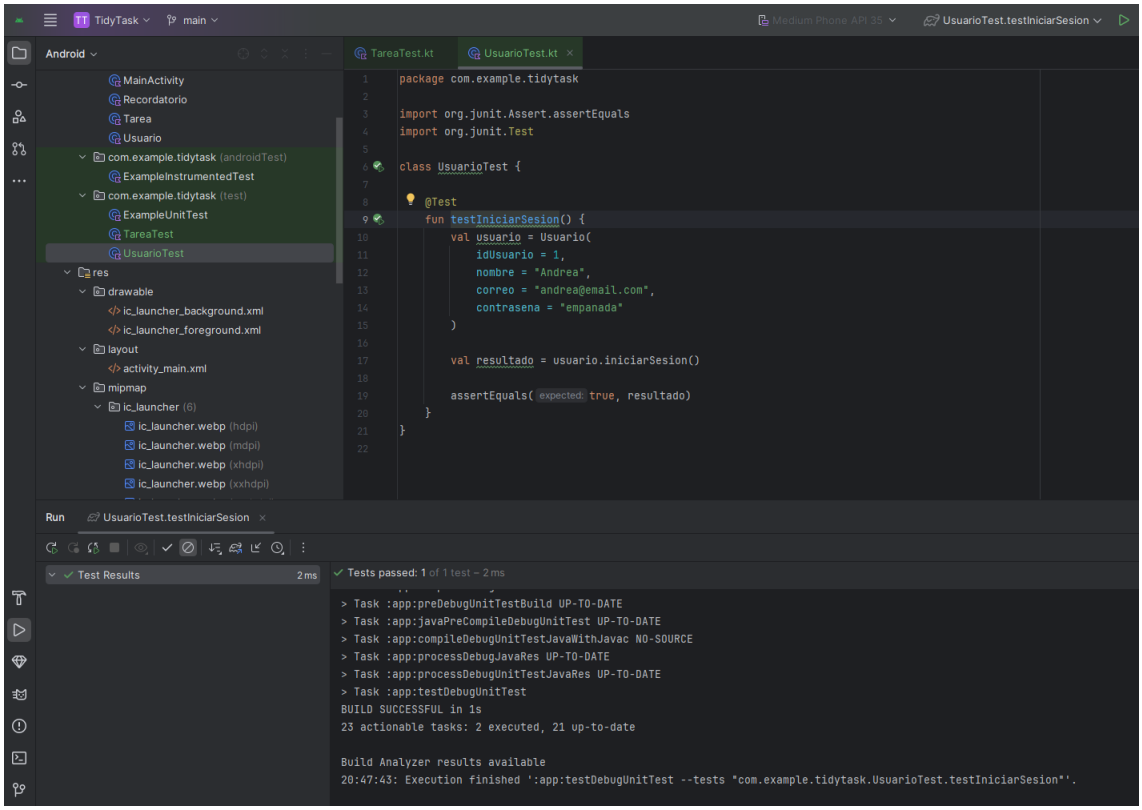
activity_main.xml | Usuario.kt | Tarea.kt | Categoria.kt | Recordatorio.kt

```kotlin
package com.example.tidytask

class Recordatorio(
    var idRecordatorio: Int,
    var fechaHora: String
) {
    fun programarAlarma(): Boolean {

        return true
    }

    fun cancelarAlarma(): Boolean {

        return true
    }
}

```

La clase Recordatorio permitirá asociar una fecha y hora a una tarea para programar una notificación futura. Simula las funciones de programar y cancelar alarmas.

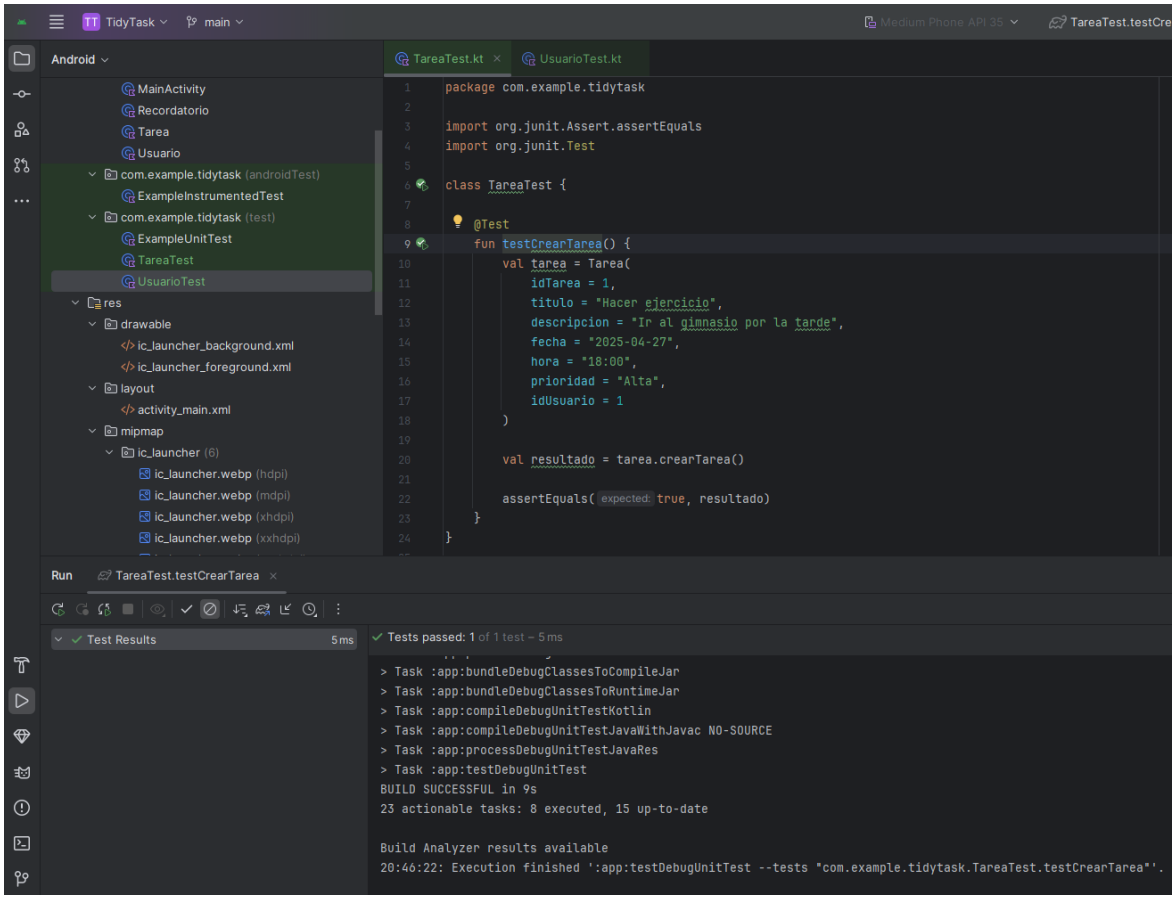
PRUEBAS UNITARIAS CON JUNIT

Prueba unitaria de inicio de sesión de usuario en TidyTask



Se comprueba que el método iniciarSesion() permite acceder correctamente cuando el usuario tiene los datos completos.

Prueba unitaria de creación de tarea en TidyTask.



Se comprueba que el método crearTarea() funciona correctamente, devolviendo true al crear una tarea con todos sus datos completos.