MentalCare

Jose María de Puelles Gutiérrez

Desarrollo de Interfaces

Índice

1 Introducción y descripción del proyecto

El ritmo de vida actual provoca que a muchas personas se enfrentan a altos niveles de estés, falta de organización y gran dificultad para crear hábitos saludables y llevarlos a cabo. O incluso organizarse objetivos a futuro.
Este proyecto consiste en el desarrollo de una aplicación Android orientada a fomentar la salud y el bienestar mediante la combinación de organización personal, seguimiento emocional y la motivación diaria.

1.1 Objetivos de la aplicación

La creación de esta aplicación ha sido con el objetivo de ayudar al usuario a mejorar su bienestar personal facilitando la creación y programación de tareas y hábitos diarios. Así como el registro de su estado emocional, promoviendo la constancia, el autocuidado y el logro de objetivos personales.

1.2 Problema social abordado

El proyecto aborda un problema social relacionado con la salud mental y el bienestar emocional ya que muchas personas no disponen de herramientas sencillas para:

- Expresar como se sienten a diario.
- Mantener hábitos saludables de forma constante.
- Organizar sus tareas sin generar más estrés.

La aplicación es una solución tecnológica accesible que integra la forma de llevar estos problemas en una única plataforma, ayudando al usuario a tomar consciencia de su estado emocional y sus rutinas diarias.

1.3 Público objetivo

Esta aplicación está dirigida principalmente a:

- Estudiantes,
- Personas con rutinas diarias exigentes.
- Usuarios que desean mejorar su organización personal y bienestar emocional.

No se requieren conocimientos técnicos previos. La aplicación está diseñada para ser intuitiva, clara y fácil de usar.

1.4 Funcionalidades principales

La aplicación cuenta con las siguientes funcionalidades principales:

- Diario Emocional: permite al usuario registrar diariamente su estado de ánimo mediante emojis y un texto descriptivo..
- Gestión de hábitos: creación y seguimiento de hábitos diarios. Permite marcarlos como completados cada día.
- Gestión de objetivos: creación y organización de tareas con fecha y estado de completado.
- Frases motivacionales: visualización de frases inspiradoras.
- Resumen diario: vista general del progreso del usuario, mostrando hábitos completados, tareas pendientes y estado emocional.

2 Análisis de herramientas y tecnologías

La mayor parte de las herramientas usadas ya se han usado anteriormente en el módulo de Desarrollo de Interfaces. Así que su uso es también por la cercanía que tiene el estudiante con las mismas.

2.1 Android Studio

Es el entorno de desarrollo integrado oficial para el desarrollo de aplicaciones Android. Proporciona herramientas para escribir código, diseñar interfaces, depurar y ejecutar aplicaciones en distintos dispositivos.

Se ha elegido el uso de esta app por ser la herramienta oficial recomendada por Google, usada ampliamente en entornos educativos y profesionales, además es completamente compatible con Jetpack Compose.

2.2 Kotlin

Es un lenguaje de programación moderno, conciso y seguro oficialmente soportado por Google para el desarrollo de aplicaciones Android.

Se ha elegido por su sintaxis clara, reducción de código repetitivo y su buena integración con Android Studio y Jetpack Compose.

2.3 Jetpack Compose

Es el framework moderno de Android para la creación de interfaces de usuario de manera declarativa.

Se ha elegido Compose porque permite crear interfaces dinámicas y reactivas con poco código, facilitando con esto la reutilización de componentes y la gestión de estado.

2.4 Material 3

Es el sistema de diseño de Google que proporciona componentes visuales, estilos y patrones de interacción coherentes para aplicaciones Android.

Se utiliza Material 3 para garantizar una interfaz moderna, accesible y consistente, siguiendo las recomendaciones de diseño actuales de Android.

2.5 Room

Room es una librería de persistencia de datos que forma parte de Android Jetpack, basada en SQLite, y facilita el almacenamiento local de datos.

Se utiliza Room para guardar de forma persistente los hábitos, tareas y registros emocionales del usuario, evitando la pérdida de información al cerrar la aplicación.

2.6 Arquitectura MVVM

MVVM (Model–View–ViewModel) es un patrón de arquitectura que separa la lógica de negocio de la interfaz de usuario.

Se ha elegido MVVM para mantener el código organizado, escalable y fácil de mantener, especialmente al trabajar con Jetpack Compose.

3 Diseño conceptual y experiencia de usuario (NUI)

La interfaz de MentalCare se ha diseñado bajo los principios de NUI (Natural User Interface), buscando que la interacción entre el usuario y el dispositivo sea lo más intuitiva y fluida posible, minimizando la curva de aprendizaje.

3.1 Diseño General y Flujo de Pantallas

El flujo de navegación se ha diseñado siguiendo un esquema de jerarquía plana para evitar que el usuario se pierda en menús profundos, algo vital en una app de salud mental.

- Flujo principal:
    - Pantalla de Login: Punto de entrada. El usuario introduce credenciales o se registra.
    - Pantalla de Seguridad (Condicional): Si el usuario tiene activado el PIN o la biometría en "Ajustes", el sistema intercepta la navegación y requiere validación antes de mostrar datos privados.
    - Home (Dashboard): Acción principal donde el usuario selecciona su emoción, actividades y añade una nota.

    - Calendario/Objetivos/Settings (Secundario): Consulta de histórico, gestión de metas y opciones mediante el menú lateral. Por cuestion de seguridad comprueba si el usuario está autenticado antes de acceder a este apartado.
- Navegación lateral (Drawer): Permite saltos rápidos a secciones secundarias como "Mis Objetivos", "Calendario Histórico" y "Ajustes".

3.2 Interacción Táctica y Gestos

Se han implementado gestos estándar de Android para asegurar la consistencia del sistema:

- Tap (Toque): Para selección de emoticonos de ánimo y navegación por botones.
- Scroll (Desplazamiento): Fluidez vertical en la pantalla principal para revisar el historial de registros del usuario.
- Swipe (Deslizar): Implementado en el menú lateral (Drawer) y en la navegación de calendarios. Permite una sensación de "manejo físico" de la interfaz.

3.3 Interacción por Voz - Propuesta Conceptual

Objetivo: Facilitar el uso de la app en momentos de crisis de ansiedad o baja movilidad.

- Funcionalidad: Implementación de un asistente de voz integrado (ej. "Añadir nota de voz").
- Caso de uso: El usuario dice: "Añadir objetivo: Meditar 10 minutos" o "Registrar que me siento feliz". El sistema procesaría el lenguaje natural para rellenar los campos automáticamente, eliminando la fricción de escribir.

3.4 Detección Facial y Análisis de Ánimo - Propuesta Conceptual

Objetivo: Automatizar la detección del estado emocional para ofrecer apoyo proactivo.

- Funcionalidad: Mediante el uso de la cámara frontal y APIs de reconocimiento facial (como Google ML Kit), la app podría analizar expresiones al abrirse.
- Justificación: Si la cámara detecta signos de fatiga o tristeza extrema, la app sugeriría automáticamente una frase motivadora o un ejercicio de respiración, personalizando la experiencia según el contexto biométrico del usuario.

3.5 Realidad Aumentada (AR) - Propuesta de Futuro

Objetivo: Crear un entorno de relajación inmersivo dentro de casa.

- Funcionalidad: "ZenSpace AR". El usuario podría proyectar mediante la cámara del móvil elementos relajantes en su habitación (como un árbol zen, lluvia virtual o una mascota virtual de apoyo). Se propone el uso de ARCore para el anclaje de objetos 3D en superficies planas del hogar.
- Justificación: La AR permite al usuario "salir" mentalmente de su entorno estresante sin abandonar su espacio físico, proporcionando una herramienta de "mindfulness" visualmente rica.

3.6 Adaptabilidad y Accesibilidad

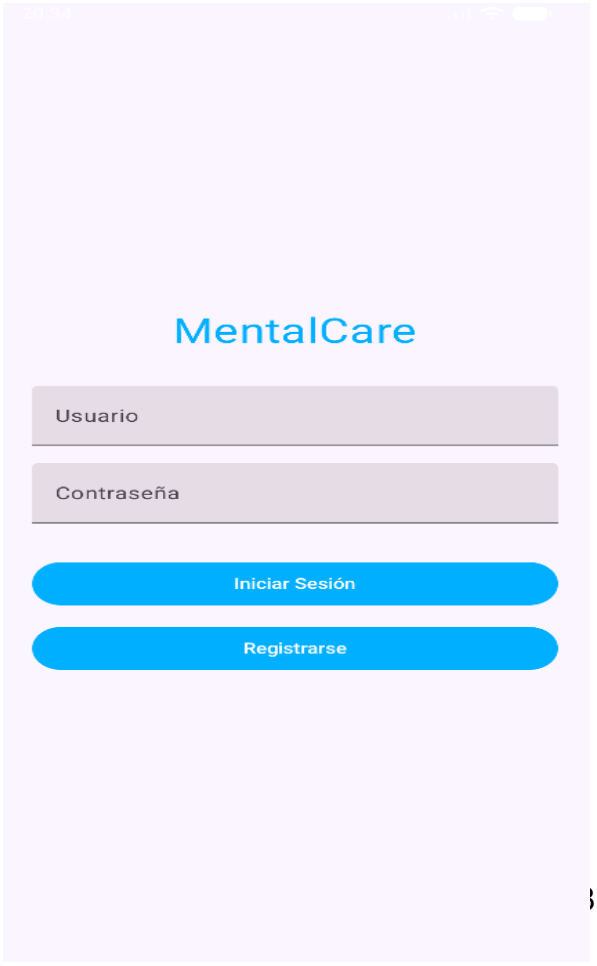
- Modo Oscuro/Claro: Implementado para reducir la fatiga visual según el entorno de iluminación del usuario.
- Escalado de Fuente: El usuario puede modificar el tamaño de la tipografía desde los ajustes, asegurando que personas con dificultades visuales puedan leer los registros sin esfuerzo.

4 Diseño de la interfaz gráfica

En este apartado mostraremos las distintas pantallas de la aplicación,y para que sirve cada una.

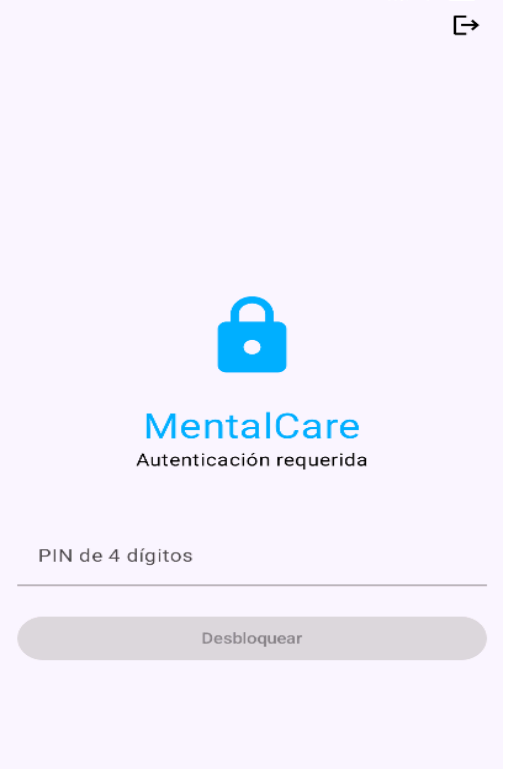
4.1 Pantalla de Inicio

Nada más abrir la app por primera vez nos solicitara habilitar notificaciones. En esta pantalla de Inicio podremos iniciar sesión con una cuenta que ya habremos creado anteriormente o podremos darle al botón de registrarse y se nos abrirá un Dialog que nos permitirá crearnos una cuenta de usuario.



4.2 Pantalla de Autenticación

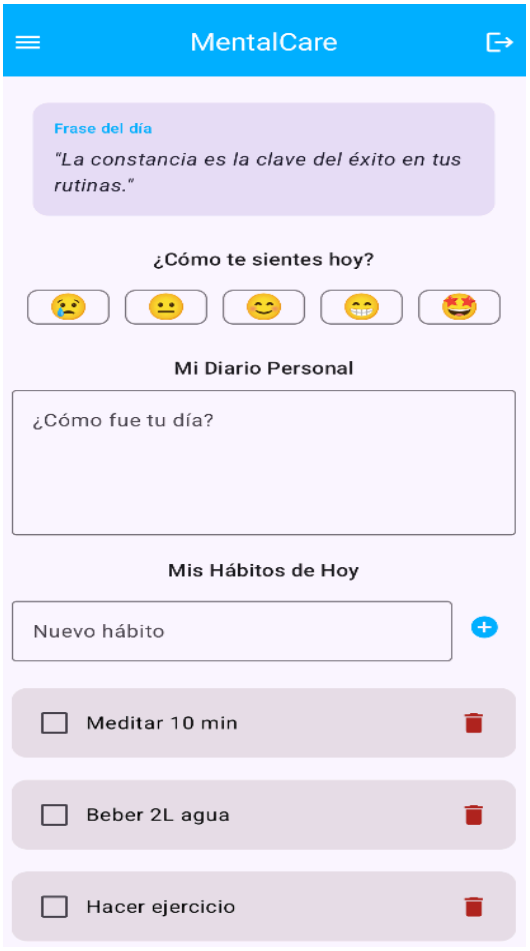
En caso de que el usuario con el que hayamos iniciado sesión haya introducido un PIN u otra forma de autenticación nos saldrá esta pantalla en particular. En ella tendremos que introducir el PIN o autenticarnos de la manera que el usuario haya definido en las opciones. También podremos darle a logout e iniciar sesión con otro usuario en caso de que no nos acordemos del PIN..



4.3 Pantalla Home

Esta es la pantalla inicial de la aplicación. Tendrá posibilidad de scroll en la Home, de abrir la barra lateral y de cerrar sesión.
En esta pantalla tendremos:

- Frase motivacional: Apartado donde saldrá una frase motivacional del día de hoy. Esta frase saldrá de un registro de frases ya insertado anteriormente.
- Emoji: Apartado donde el usuario que resumirá nuestro estado de ánimo del día de hoy.
- Diario Personal: Apartado que consistirá en un diario personal. Ahí el usuario escribirá lo que sienta ese día. Puede escribir o no hacerlo eso es voluntad del mismo.
- Hábitos: Aqui el usuario podrá llevar un control de los hábitos que quiere realizar diariamente. Puede añadir y

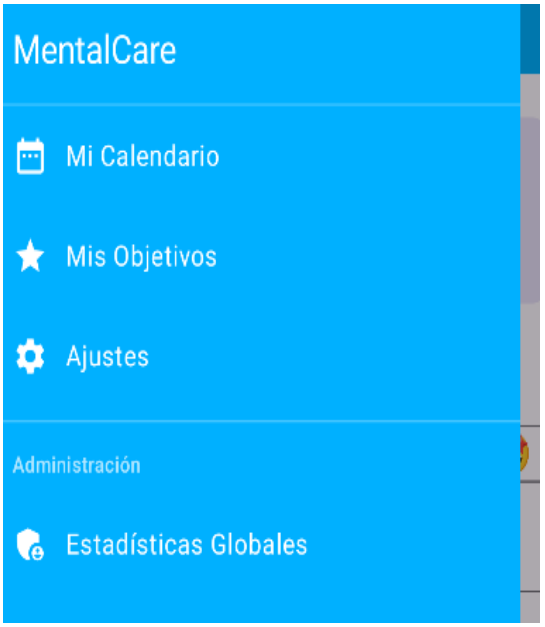


eliminar hábitos nuevos en cualquier momento.

4.4 Barra lateral

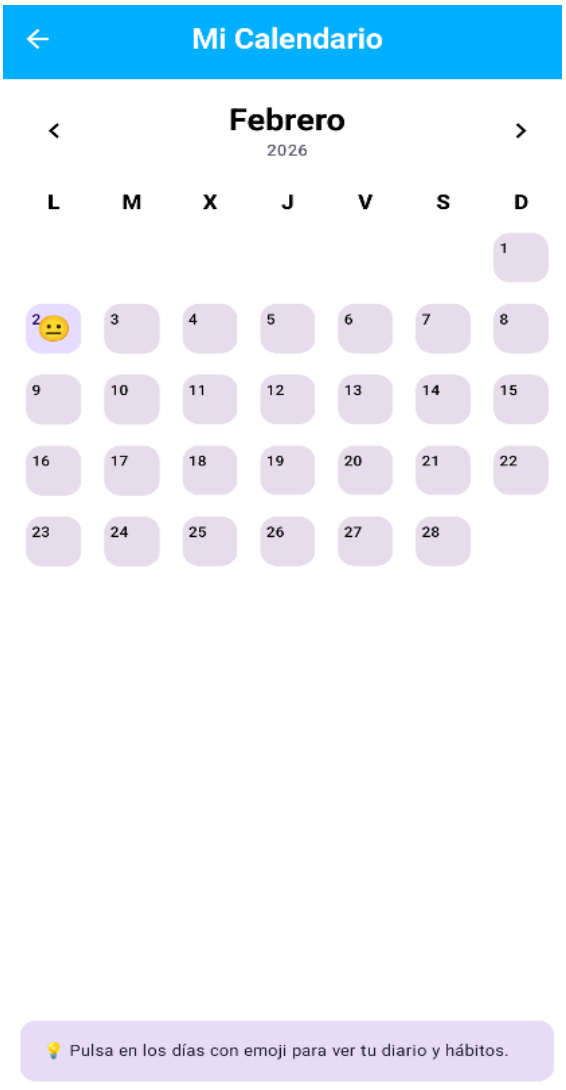
Desde este apartado accederemos al resto de la aplicación.Tendremos 4 lugares a los que podemos ir.

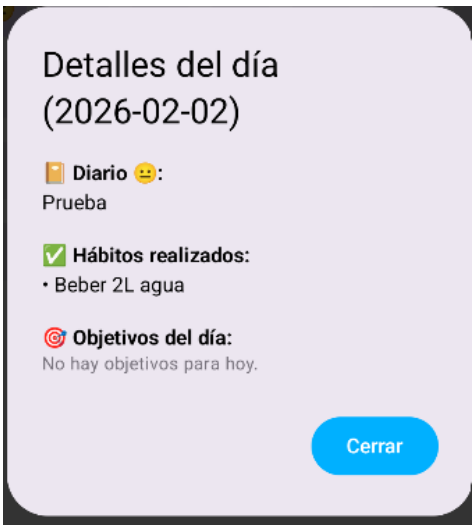
- Calendario
- Objetivos
- Ajustes
- Estadísticas Globales (saldrá esta opción en caso de ser admin)



4.5 Mi Calendario

En esta pestaña tendremos el calendario. Aquí podremos revisar que días hemos cumplido con un hábito que nos hemos propuesto. Que escribimos ese día. Y que objetivos nos pusimos para el mismo y si este lo cumplimos. Al seleccionar un día específico, el sistema busca en la base de datos el dateKey correspondiente (ej: '2026-02-03'). Si el filtro devuelve un resultado positivo, se muestran los detalles; de lo contrario no se mostraría nada.



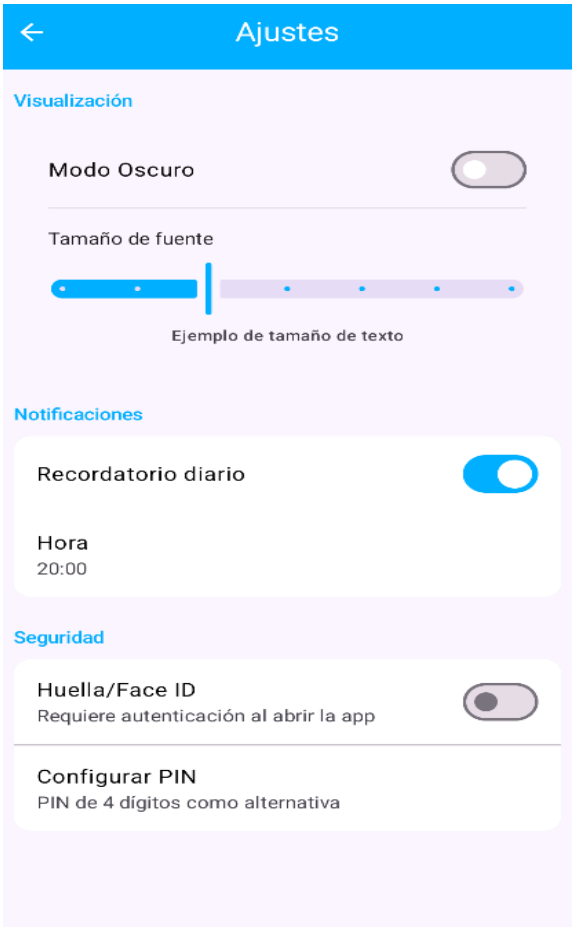


4.6 Mis Objetivos

En esta pestaña el usuario podrá registrar objetivos mediante un Dialog que sale al pulsar el Floating Button. Al añadir el mismo deberá fijar un nombre y una fecha para cumplir ese objetivo. Al añadirlo le saldrá en la lista de pendientes. Y en caso de cumplirlo podrá pulsarlo y este se moverá a la lista de Mis Logros. Al seleccionar la fecha se abrirá otro Dialog que mostrará un calendario donde podrá seleccionar el día para el que quiere cumplir ese objetivo.

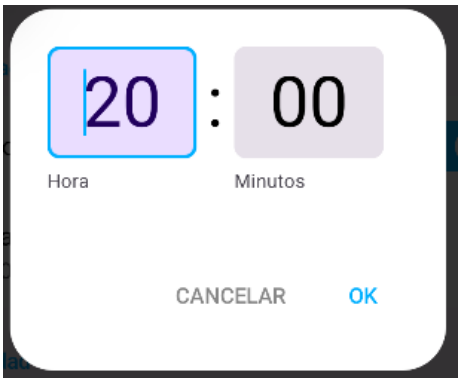
4.7 Ajustes

En esta página tendremos varias cosas divididas en apartados según su utilidad. En este caso son 3 apartados que cada uno tendrá distintas opciones.



- Visualización: Estas opciones adaptarán la app y la información que queremos mostrar al usuario.
    - Modo Oscuro: Para cuando el usuario quiera que se vea la app con tonos más oscuros.
    - Tamaño de Fuente: Por si el usuario quiere ampliar o disminuir la fuente de letra.

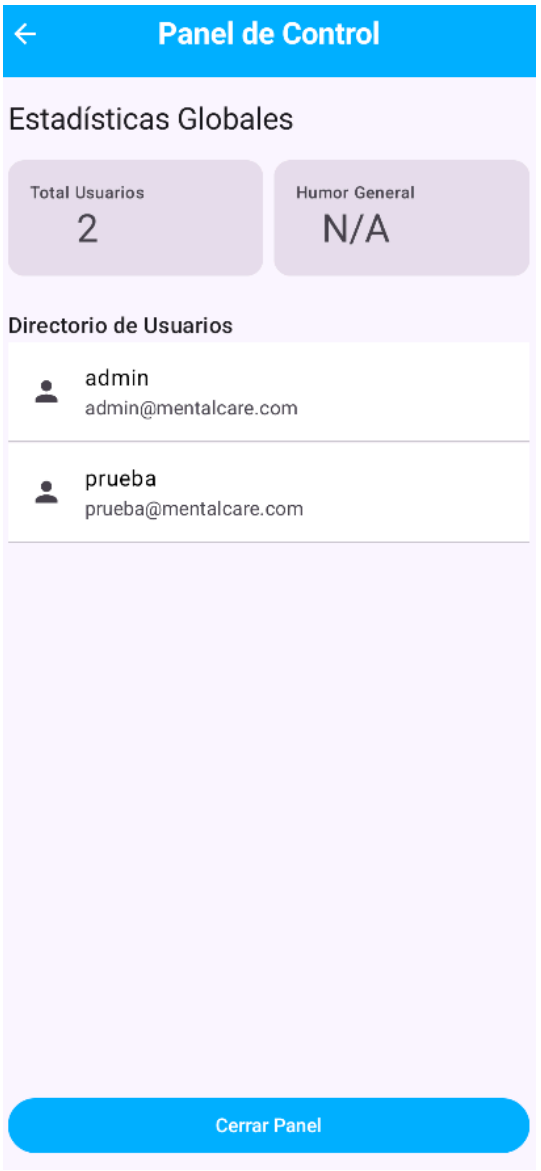
- Notificaciones: Estas opciones permitirá el envio de un recordatorio al usuario para que recuerde usar la app.
    - Switch: Permitirá activar o desactivar el recordatorio.
    - Hora: Permitirá mediante un Dialog poner la hora a la que queremos el mismo.



- Seguridad: Aquí tendremos diversas opciones de seguridad para proteger la app y la información que contiene.
    - Huella/FaceID: Para una futura implementación hay un switch para activar o desactivar el uso de esta función.
    - PIN: Podrás configurar una contraseña de 4 dígitos.Se ha implementado un sistema de rutas protegidas que bloquea el acceso al Home y resto de funcionalidades si el usuario no supera el desafío de seguridad.

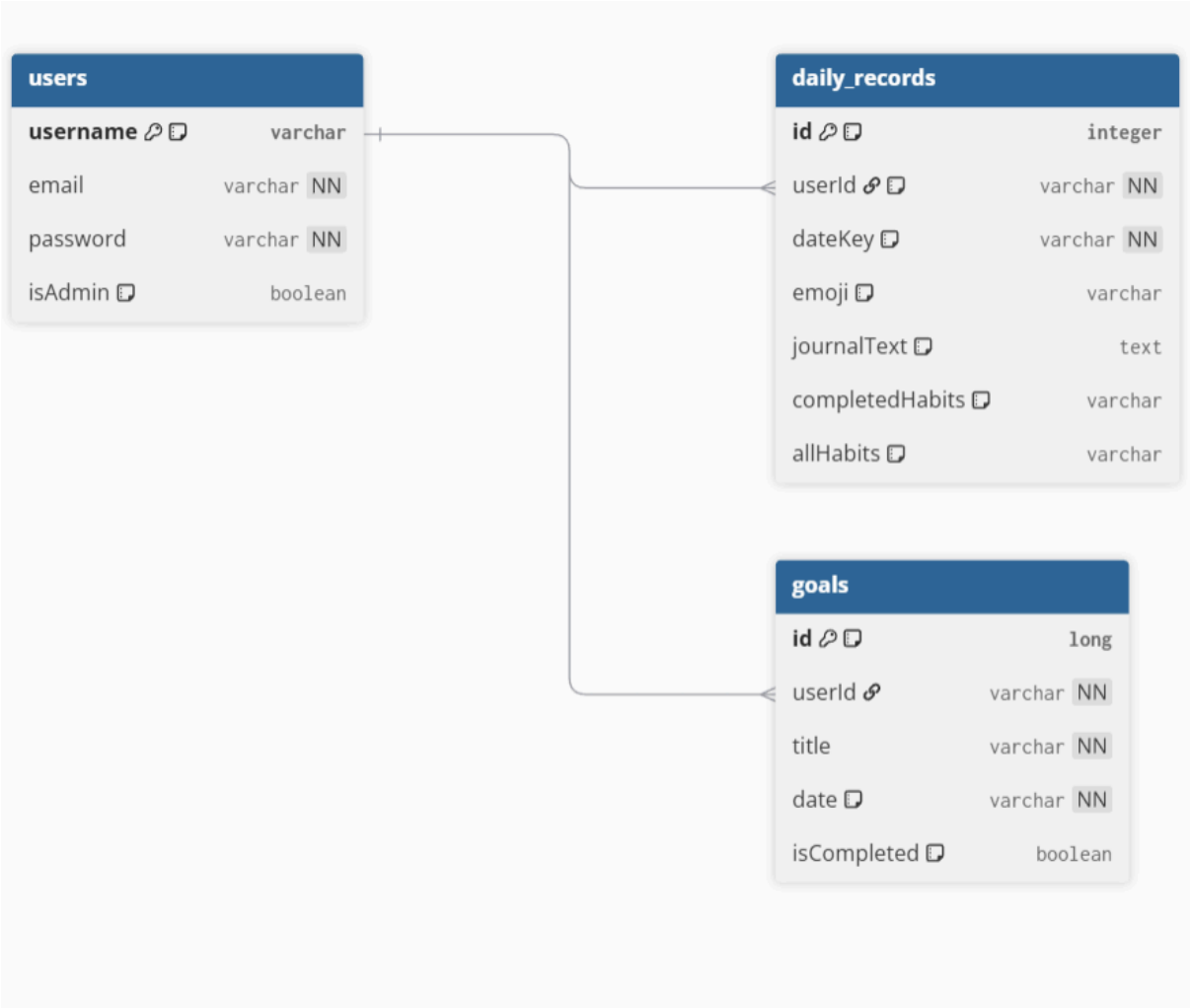
4.8 Pantalla de Estadísticas (Exclusiva de Admin)

En esta pantalla tendremos las Estadísticas Globales (solo esta implementado localmente) el número de usuarios registrados. Se aplican filtros de agregación sobre la totalidad de los registros para calcular el Humor General. El sistema filtra todos los DayRecord existentes, agrupa los emojis y extrae el valor predominante, transformando datos brutos en información estadística útil. Aparte de eso tendremos el directorio de usuarios donde veremos todos los usuarios con una cuenta en la app.



5 Arquitectura de Base de Datos

La base de datos está compuesta por 3 entidades principales relacionadas entre sí mediante claves foráneas (Foreign Keys), siguiendo el patrón de integridad referencial. El diagrama de relaciones es el siguiente:



- Entidad Central (users): A diferencia de otros sistemas que usan IDs numéricos, este diseño utiliza el username como Clave Primaria y Clave Ajena.
- Persistencia Relacional:
    - daily_records almacena la carga emocional y notas del diario. La relación con users asegura que al filtrar por currentUser.username, el calendario solo muestre datos privados.
    - goals gestiona las metas personales. El campo isCompleted permite realizar el recuento de totales necesario para los informes de progreso.

Relaciones:

- UserEntity -> DayRecord: Relación 1:N (un usuario puede tener múltiples registros diarios)
- UserEntity -> GoalEntity: Relación 1:N (un usuario puede tener múltiples objetivos)

5.1 UserEntity

Almacena la información de autenticación y privilegios de cada usuario registrado en la aplicación.

Campo	Tipo	Descripción	Restricciones
username	String	Nombre de usuario. Identificador único.	PRIMARY KEY, NOT NULL
email	String	Correo electrónico del usuario	NOT NULL
password	String	Contraseña del usuario. Almacenada de forma segura	NOT NULL
isAdmin	Boolean	Indica si el usuario tiene permisos de administrador	Default: false

Decisiones de diseño:

- Se utiliza username como clave primaria en lugar de un ID numérico para simplificar la lógica de autenticación y reducir la complejidad en las consultas. relacionadas.
- El campo isAdmin permite diferenciar entre usuarios estándar y administradores, habilitando la funcionalidad de Estadísticas Globales.

5.2 DayRecord

Registra el estado emocional y las actividades realizadas por el usuario cada día. Es la entidad central para el Diario Emocional y el seguimiento de hábitos.

Campo	Tipo	Descripción	Restricciones
id	Int	Identificador único del registro	PRIMARY KEY, AUTO_INCREMENT
dateKey	String	Fecha del registro en formato String (ej: "2026-02-03")	NOT NULL
userId	String	Usuario al que pertenece este registro	FOREIGN KEY -> UserEntity.username, NOT NULL
emoji	String	Emoticono que representa el estado de ánimo	DEFAULT: "" (vacío)
journalText	String	Nota personal del día (diario textual)	DEFAULT: "" (vacío)
completedHabits	List<String>	Lista de hábitos completados ese día	DEFAULT: [] (lista vacía)
allHabits	List<String>	Lista de todos los hábitos disponibles ese día	DEFAULT: [] (lista vacía)

Decisiones de diseño:

- El campo dateKey se almacena como String (ej: "2026-02-03") en lugar de timestamp para facilitar las consultas por fecha exacta y mejorar la legibilidad en el código.
- emoji almacena el emoticono como String para facilitar la visualización directa en la interfaz.
- completedHabits y allHabits permiten rastrear qué hábitos estaban disponibles y cuáles fueron completados. Esto es útil para:
    - Mantener historial incluso si el usuario elimina hábitos en el futuro

Diferencia entre completedHabits y allHabits:

- allHabits: ["Meditar", "Ejercicio", "Leer", "Beber agua"]
- completedHabits: ["Meditar", "Ejercicio"] ← Solo los que el usuario marcó como completados ese día

5.3 GoalEntity

Almacena los objetivos o metas personales que el usuario desea alcanzar, permitiendo un seguimiento organizado de su progreso.

Campo	Tipo	Descripción	Restricciones
id	Long	Identificador único del objetivo	PRIMARY KEY, AUTO_INCREMENT
userId	String	Usuario al que pertenece este registro	FOREIGN KEY -> UserEntity.username, NOT NULL
tittle	String	Descripción del objetivo (ej: "Correr 5km")	NOT NULL
date	String	Fecha límite para cumplir el objetivo (formato String)	NOT NULL
isCompleted	Boolean	Indica si el objetivo ha sido completado	DEFAULT: false

Decisiones de diseño:

- Se utiliza Long como tipo de clave primaria en lugar de Int para soportar un mayor número de objetivos sin riesgo de desbordamiento.
- isCompleted es un booleano que permite clasificar los objetivos en dos categorías visuales: "Pendientes" (false) y "Mis Logros" (true).

5.4 Operaciones de Base de Datos (DAOs)

Room utiliza Data Access Objects (DAOs) para gestionar las operaciones CRUD (Create, Read, Update, Delete) de forma eficiente y type-safe. A continuación se describen las operaciones implementadas para cada entidad:

5.4.1 UserDao - Gestión de Usuarios

Responsable de las operaciones de autenticación, registro y administración de usuarios.

Método	**Descripción**	**Retorno**
getUserByUsername(username: String)	Obtiene un usuario específico por su nombre de usuario (usado en login)	UserEntity? (nullable)
registerUser(user: UserEntity)	Registra un nuevo usuario. Si el username ya existe, aborta la operación	suspend fun
getUserCount()	Cuenta el total de usuarios registrados (usado en estadísticas admin)	Int
getAllUsers()	Obtiene todos los usuarios (solo para administradores)	Flow<List<UserEntity>>

Código completo (Documentado en código):

```
@Dao
interface UserDao {
    @Query("SELECT * FROM users WHERE username = :username LIMIT 1")
    suspend fun getUserByUsername(username: String): UserEntity?

    // Aborta si el usuario existe (evita duplicados)
    @Insert(onConflict = OnConflictStrategy.ABORT)
    suspend fun registerUser(user: UserEntity)

    @Query("SELECT COUNT(*) FROM users")
    suspend fun getUserCount(): Int

    @Query("SELECT * FROM users")
    fun getAllUsers(): Flow<List<UserEntity>>
}
```

Uso en la aplicación:

- Eventos: getUserByUsername() se ejecuta al hacer login para validar credenciales.
- Valores calculados: getUserCount() se muestra en la pantalla de "Estadísticas Globales" (solo admin).

- Mensajes claros: Si registerUser() falla por OnConflictStrategy.ABORT, se muestra un mensaje: "El nombre de usuario ya está en uso".

5.4.2 DailyRecordDao - Gestión de Registros Diarios

Responsable del almacenamiento y recuperación de los estados emocionales y hábitos diarios del usuario.

Método	**Descripción**	**Retorno**
getRecordsByUser(userId: String)	Obtiene todos los registros de un usuario específico (usado en calendario histórico)	Flow<List<DayRecord>>
insertRecord(record: DayRecord)	Guarda o actualiza un registro diario. Si ya existe para esa fecha, lo reemplaza	suspend fun
getAllRecords()	Obtiene todos los registros de todos los usuarios (solo admin, para estadísticas globales)	Flow<List<DayRecord>>

Código completo (Documentado en código):

```
@Dao
interface DailyRecordDao {
    @Query("SELECT * FROM daily_records WHERE userId = :userId")
    fun getRecordsByUser(userId: String): Flow<List<DayRecord>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertRecord(record: DayRecord)

    @Query("SELECT * FROM daily_records")
    fun getAllRecords(): Flow<List<DayRecord>>
}
```

Uso en la aplicación:

- App integrada: insertRecord() se ejecuta al final del día cuando el usuario guarda su estado emocional y hábitos completados.
- Se implementa un filtro mediante la cláusula WHERE userId = :userId en el método getRecordsByUser. Esto garantiza la privacidad de los datos, asegurando que un usuario solo pueda visualizar sus propios registros emocionales y hábitos, filtrando cualquier información perteneciente a otras cuentas almacenadas en el mismo dispositivo.

- Gráficos: Los datos de getAllRecords() se procesan para generar el gráfico de "Humor General" en la pantalla de admin.
- OnConflictStrategy.REPLACE: Si el usuario edita su registro del día, sobrescribe el anterior en lugar de crear un duplicado.

5.4.3 GoalDao - Gestión de Objetivos

Responsable de la creación, actualización y eliminación de objetivos personales del usuario.

Método	**Descripción**	**Retorno**
getGoalsByUser(userId: String)	Obtiene todos los objetivos de un usuario (pendientes y completados)	Flow<List<GoalEntity>>
insertGoal(goal: GoalEntity)	Crea un nuevo objetivo o actualiza uno existente	suspend fun
deleteGoal(goal: GoalEntity)	Elimina permanentemente un objetivo	suspend fun
updateGoal(goal: GoalEntity)	Actualiza los datos de un objetivo (título, fecha o estado)	suspend fun

Código completo (Documentado en código):

```
@Dao
interface GoalDao {
    @Query("SELECT * FROM goals WHERE userId = :userId")
    fun getGoalsByUser(userId: String): Flow<List<GoalEntity>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertGoal(goal: GoalEntity)

    @Delete
    suspend fun deleteGoal(goal: GoalEntity)

    @Update
    suspend fun updateGoal(goal: GoalEntity)
}
```

Uso en la aplicación:

- Asociación de eventos: Al pulsar un objetivo en "Pendientes", se llama a updateGoal() cambiando isCompleted = true, lo que lo mueve visualmente a "Mis Logros".
- Distribución de acciones: El FloatingActionButton de la pantalla "Mis Objetivos" ejecuta insertGoal() al crear un nuevo objetivo.
- Valores calculados: Se filtran los objetivos con isCompleted = true para calcular el porcentaje de logros.

5.4.4 AppDatabase - Configuración de Room

Clase principal de la persistencia de datos.

Código completo (Documentado en código):

```
@Database(entities = [UserEntity::class, DayRecord::class, GoalEntity::class],
version = 2)
@TypeConverters(Converters::class)
abstract class AppDatabase : RoomDatabase() {
   abstract fun userDao(): UserDao
   abstract fun dailyRecordDao(): DailyRecordDao
   abstract fun goalDao(): GoalDao
}
```

Configuración:

- version = 2: Indica que esta es la segunda versión del esquema de base de datos (probablemente tras añadir el campo allHabits a DayRecord).
- @TypeConverters(Converters::class): Habilita la conversión automática de List<String> a formato almacenable en SQLite.
- fallbackToDestructiveMigration(): En caso de cambio de versión incompatible, elimina la base de datos anterior y crea una nueva (útil en desarrollo, en producción se usarían migraciones).

5.4.5 TypeConverters - Conversión de Tipos Complejos

Clase Converters: Permite a Room persistir tipos de datos no primitivos.
Room solo entiende tipos básicos (Int, String, etc.). Para guardar Listas o Fechas, necesitamos convertir estos objetos en Strings para guardarlos y viceversa al leerlos.

```
class Converters {
   private val gson = Gson()

   @TypeConverter
   fun fromStringList(value: List<String>?): String {
```

```
        return gson.toJson(value)
    }

    @TypeConverter
    fun toStringList(value: String): List<String>? {
        val listType = object : TypeToken<List<String>>() {}.type
        return gson.fromJson(value, listType)
    }

    // Example for LocalDate if you are using it
    @RequiresApi(Build.VERSION_CODES.O)
    @TypeConverter
    fun fromTimestamp(value: String?): java.time.LocalDate? {
        return value?.let { java.time.LocalDate.parse(it) }
    }

    @TypeConverter
    fun dateToTimestamp(date: java.time.LocalDate?): String? {
        return date?.toString()
    }

}
```

5.4.6 Reactividad (Flow) y Estrategia de Filtrado

La mayoría de métodos de lectura retornan Flow<List<T> en lugar de valores directos. Esto proporciona:

Ventaja	**Descripción**
Reactividad	La UI se actualiza automáticamente cuando cambian los datos en la BD
Eficiencia	No requiere polling manual para detectar cambios
Integración con Compose	Se puede usar directamente con collectAsState() en ViewModels

Además de la reactividad, el uso de filtros en las consultas SQL (cláusula WHERE) es fundamental para el rendimiento. Al filtrar por userId directamente en el DAO, la aplicación realiza una carga selectiva de datos. Esto significa que si la base de datos tiene miles de registros de diferentes usuarios, Room solo procesa y envía al Flow aquellos que coinciden con la sesión activa, optimizando drásticamente el uso de memoria RAM y la batería del dispositivo.

Aparte de filtros mencionados anteriormente tenemos filtros propios de kotlin. Algunos de estos son:

- val goalsForDay = goals.filter { it.date == selectedDate }

Filtro de visualización temporal que extrae los objetivos específicos de una fecha seleccionada por el usuario.

- val filteredGoals = if (selectedTab == 0) {
  goals.filter { !it.isCompleted }
  } else {
  goals.filter { it.isCompleted }
  }

Filtro de estado que segrega la información entre tareas pendientes y logros conseguidos, mejorando la organización visual (NUI)

6. Organización de Paquetes

En este apartado hablare de como están organizados los paquetes en el código. La documentación técnica del mismo estará en el propio código.

6.1 Carpeta Raíz (com.example.mentalcare)

MainActivity.kt: Es el punto de entrada de la aplicación. Configura el canal de notificaciones, establece el idioma español y lanza la navegación principal (AppNavigation).

6.2 Carpeta de Modelos (com.example.mentalcare.model)

Define la estructura de los datos (tablas de la base de datos).

- UserEntity.kt: Define la tabla de Usuarios (nombre, email, contraseña y si es administrador).
- DayRecord.kt: Define la tabla de Registros Diarios. Guarda el emoji, el texto del diario y las listas de hábitos de cada día.
- GoalEntity.kt: Define la tabla de Objetivos. Guarda el título de la meta, la fecha límite y si ya se ha cumplido.
- UserSession.kt: Una clase ligera (Parcelable) que se usa para pasar los datos del usuario logueado entre pantallas de forma segura.

6.3 Carpeta de Datos y Persistencia (com.example.mentalcare.data)

Gestiona el acceso a la base de datos y la lógica de datos.

- UserDAO.kt: Interfaz con las consultas SQL para la tabla de usuarios (login, registro, conteo).
- DailyRecordDao.kt: Interfaz para gestionar los registros diarios. Permite guardar el progreso y leer el historial del calendario.
- GoalDao.kt: Interfaz para las operaciones de los objetivos (crear, marcar como completado, borrar).
- AppDatabase.kt: Clase principal de Room que une las entidades con los DAOs. Incluye la configuración de la base de datos y la migración.
- Converter.kt: (Ubicado técnicamente en data o model). Traduce tipos complejos como las List<String> de los hábitos a un formato que la base de datos pueda guardar (JSON).
- MotivationManager.kt: Contiene la lógica para seleccionar automáticamente la "Frase del Día" basándose en la fecha actual.

6.4 Carpeta de Pantallas (com.example.mentalcare.screens)

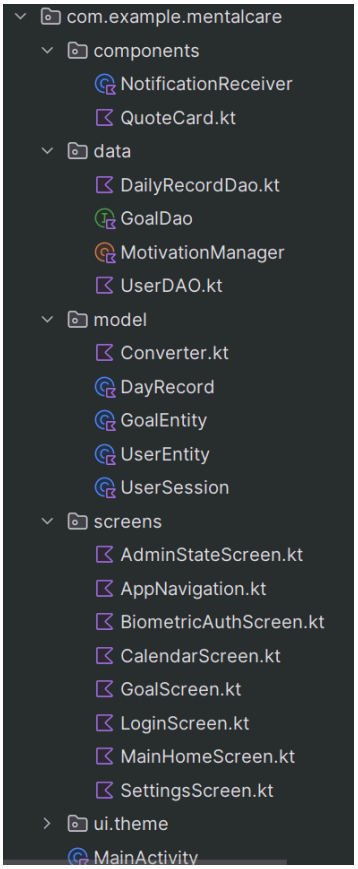
Contiene la interfaz de usuario (UI) creada con Jetpack Compose.

- AppNavigation.kt: El cerebro de la navegación. Gestiona el flujo entre pantallas, el modo oscuro, el tamaño de la fuente, la creación de la base de datos y los permisos de notificaciones.
- LoginScreen.kt: Pantalla de acceso y diálogo de registro de nuevos usuarios.
- MainHomeScreen.kt: El Dashboard principal. Aquí el usuario selecciona su emoji, escribe en el diario y marca sus hábitos diarios.
- CalendarScreen.kt: Muestra el historial mensual. Permite ver qué se registró en días pasados (hábitos, diario y metas).
- GoalScreen.kt: Gestión de metas. Permite separar los objetivos pendientes de los logros conseguidos.
- SettingsScreen.kt: Panel de configuración. Incluye ajustes visuales (tema/fuente), notificaciones y seguridad (PIN/Huella).
- BiometricAuthScreen.kt: Pantalla de bloqueo de seguridad que solicita el PIN o la huella digital antes de entrar a la app.
- AdminStateScreen.kt: Pantalla exclusiva para administradores. Muestra el total de usuarios, el humor general de la comunidad y el directorio de cuentas.

6.5 Carpeta de Componentes Reutilizables (com.example.mentalcare.components)

Piezas de interfaz o lógica que se usan en varias partes.

- QuoteCard.kt: El diseño visual de la tarjeta que muestra la frase motivacional en la Home.
- NotificationReceiver.kt: El componente que "despierta" a la app en segundo plano para lanzar la notificación de recordatorio al usuario a la hora programada.



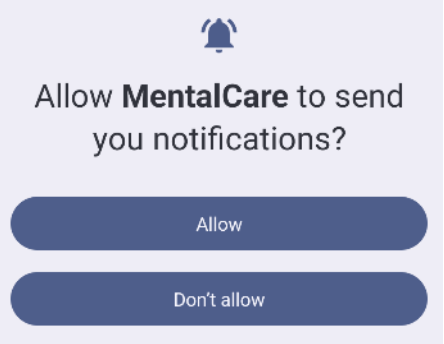
7. Manual de Usuario y Ayuda

7.1 Guía de Inicio Rápido

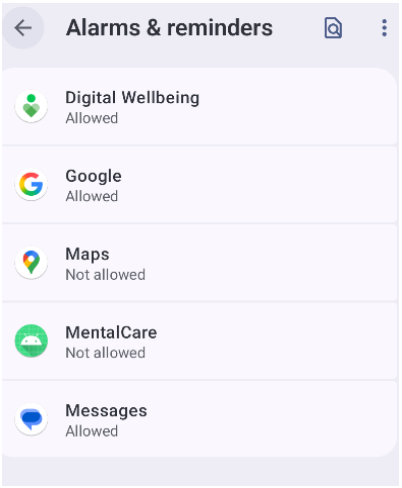
En este apartado mostraremos las funciones básicas de la app. Como como iniciar sesión, como añadir un hábito, como revisar nuestro registro de un día, como añadir un objetivo y como ver el panel de administración.

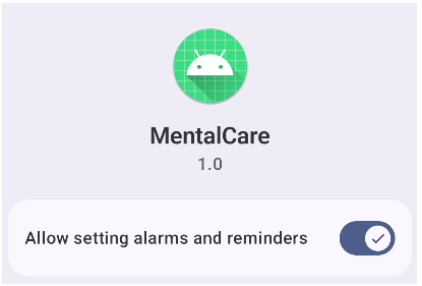
7.1.1 Registro

1. Nada más abrir la aplicación nos saldrá un Dialog preguntándonos si queremos habilitar las notificaciones. Es recomendable hacerlo para que la aplicación nos mande recordatorios de uso a la hora que queramos.

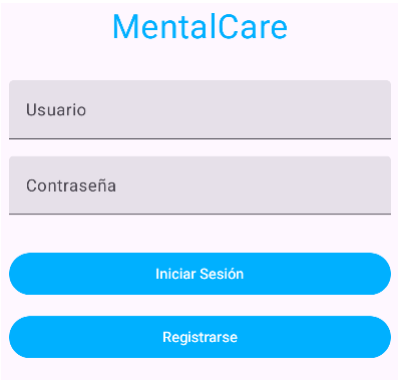


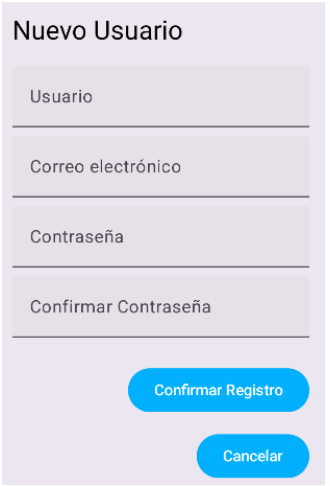
2. Para aceptarlas deberemos darle a Allow y nos saldrá una ventana de configuración de varias apps. Le damos a MentalCare. Y activamos el switch que nos permitirá recibir las notificaciones. Tras eso volvemos a la aplicación.

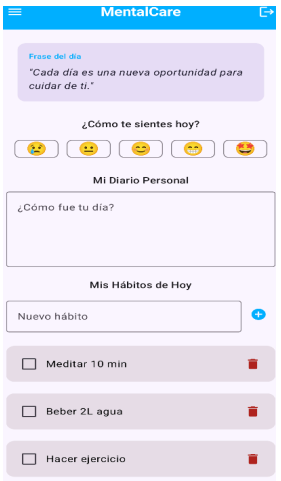




3. Al volver a la aplicación le damos a Registrarse y nos saldrá un Dialog de registro. Los campos que deberemos rellenar tienen varios requisitos.
    a. Nota: Todos los campos deben estar rellenos.
        i. Usuario: No debe ser un usuario ya registrado.
        ii. Email: No debe ser un email ya registrado.
        iii. Contraseña: Debe ser de 8 caracteres o más.
        iv. Confirmar Contraseña: Debe ser igual a Contraseña.
    b. Después de eso hay que darle a Confirmar Registro.



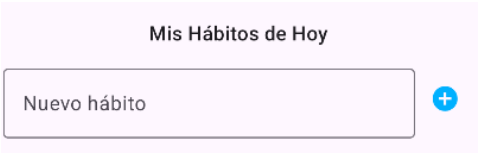




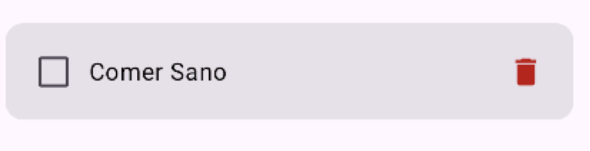
4. Luego de eso insertamos los datos insertados en la pestaña anterior y nos llevará a la pantalla principal.

7.1.2 Creación de Hábito nuevo

1. Para crear un hábito nuevo deberemos dirigirnos al apartado de Mis Hábitos de Hoy.



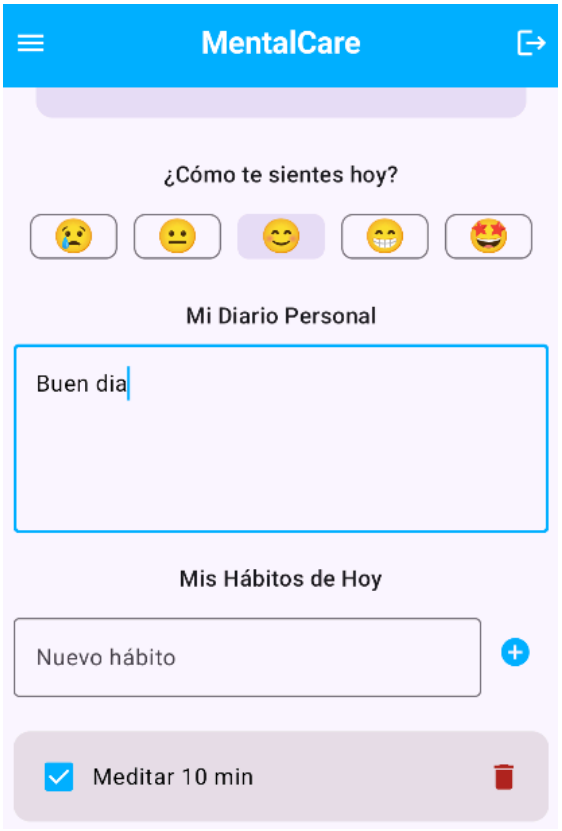
2. En el OutlinedTextField insertamos un Hábito que queramos empezar a hacer
3. Luego de eso le damos al + y este se añadirá a la lista de abajo.



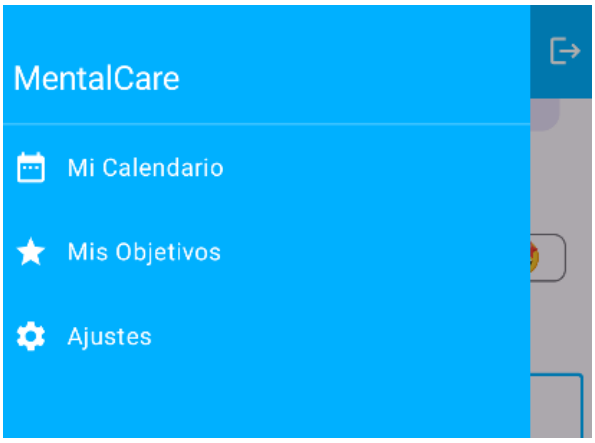
4. Si queremos borrar algún hábito deberemos darle al icono de la basura al lado del mismo.

7.1.3 Creación y Revisión de Registro de un día

1. Para añadir un registro deberemos hacer mínimo una de las 3 próxiimas cosas.
    a. Seleccionar un Emoji de la lista: Este emoji reflejará nuestro estado de ánimo el día de hoy.
    b. Escribir en Mi Diario Personal: Aquí escribiremos como nos sentimos el día de hoy.
    c. Seleccionar Hábitos: Aquí le darlemos a los Checkbox de los hábitos seleccionados.



2. Tras eso deslizaremos la pantalla hacia un lado o le daremos al botón de menú de la barra superior. Con eso se nos abrirá la barra lateral con las distintas páginas que podemos acceder. Seleccionaremos la de Mi Calendario.

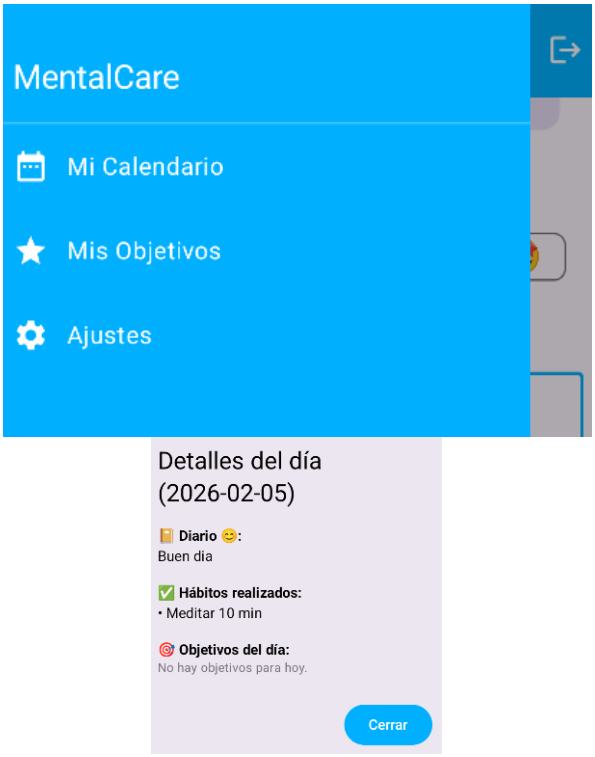


3. Nada más entrar en la App nos saldrá el calendario con el mes actual.

4. Tras eso seleccionamos el día que tenga registro. En caso de tener un emoji seleccionado se mostrará en el calendario. Al seleccionar el día que tenga registro nos saldrá un Dialog con el resumen del día.

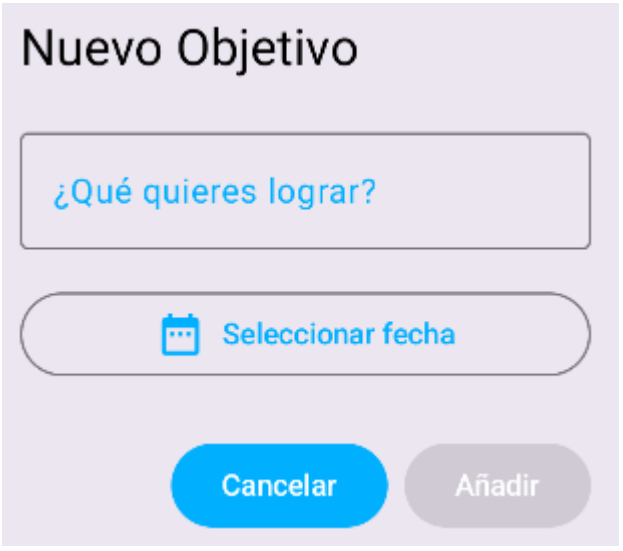
7.1.4 Gestión de Metas

1. Desde la pantalla inicial deslizamos la misma hacia un lado o le daremos al botón de menú de la barra superior. Con eso se nos abrirá la barra lateral con las distintas páginas que podemos acceder. Seleccionaremos la de Mis Objetivos.

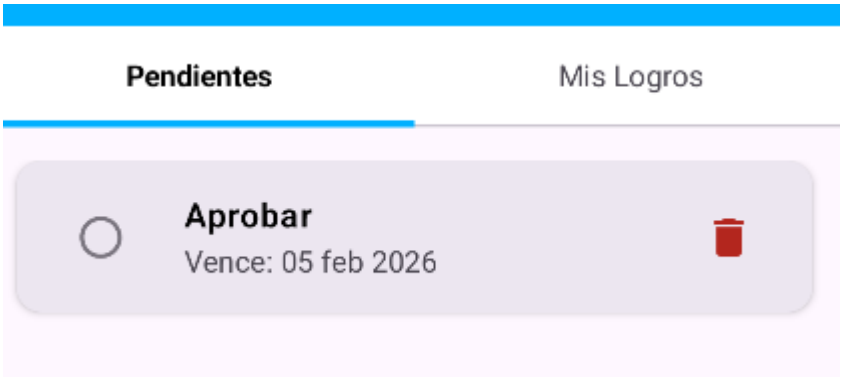


2. En este apartado tendremos los apartados de Pendientes y Mis Logros Al pulsar Mis Logros veremos los logros que hayamos completado. Al darle a Pendientes veremos los que no hemos completado. Para añadir un logro le daremos al FloatingActionButton de abajo a la derecha.

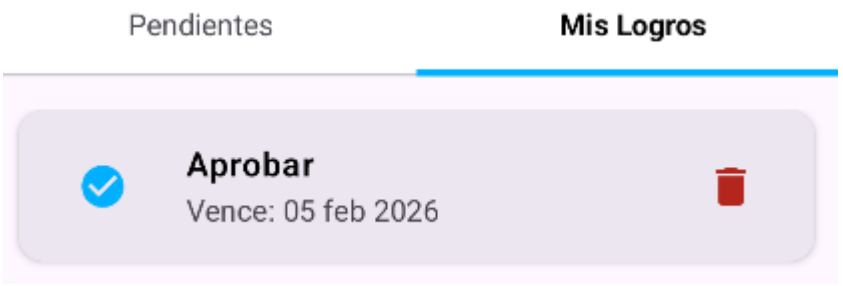
3. Al darle al botón nos saldrá un Dialog con dos apartados. En el primer apartado escribiremos el Objetivo a cumplir. En el segundo seleccionaremos una fecha en un Dialog que se nos abrirá.



4. Tras eso le damos a Ok. Volveremos a la pestaña anterior y le daremos a Añadir. Tras eso nos saldrá el objetivo en el apartado de Pendientes.



5. Si le damos al circulo al lado del objetivo este se moverá a Mis Logros y si vamos a la pestaña de Mis Logros podremos darle de nuevo y moverlo a Pendientes. En el Calendario nos saldrá además una marca con el objetivo.



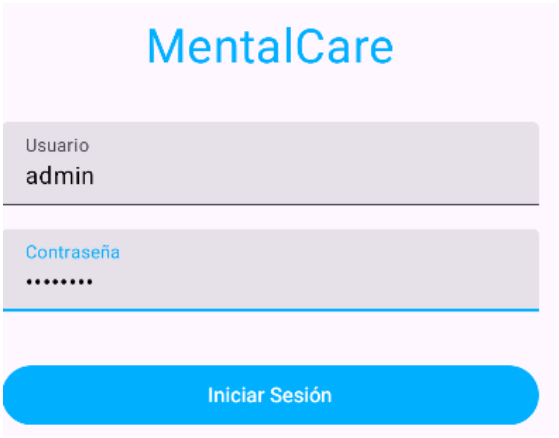
7.1.5 Panel de Administración

Para acceder al panel de administración deberemos loguearnos como administrador. Para eso deberemos cerrar sesión.

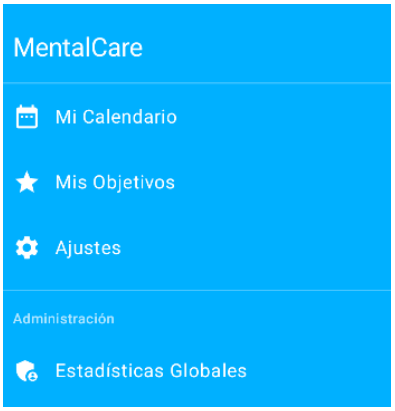
1. Para cerrar sesión deberemos ir a la pantalla inicial. Y darle al botón de la derecha que sale en la barra superior.

2. Tras eso cerraremos sesión. Y deberemos identificarnos como admin. Para identificarnos como admin deberemos insertar la siguiente información en la pantalla de inicio de sesión.
    a. Usuario: admin
    b. Contraseña: admin123

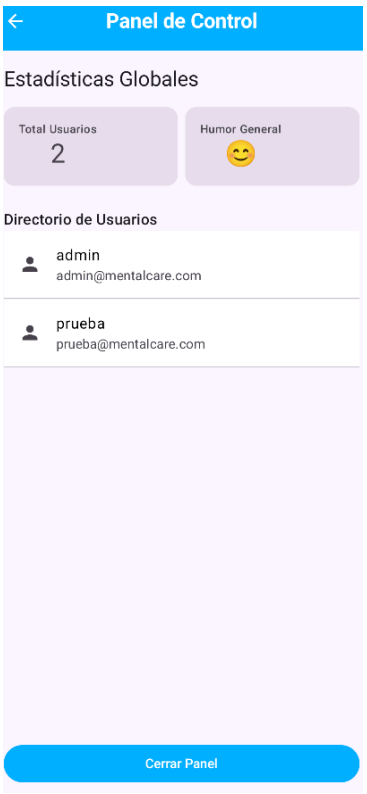
    Tras eso le daremos a Iniciar sesión.



3. En la pantalla de inicio de la app deslizaremos la pantalla y nos saldrá un apartado nuevo llamado Estadísticas Globales. Presionamos en él.



4. Al darle nos saldrá la pestaña del panel de administración. En este podremos revisar algunos datos:
    a. Número de Usuario
    b. Humor General (emoji más usado)
    c. Directorí de Usuarios



8. Estrategia de Pruebas

Aquí tenemos una tabla clasificando las pruebas que hemos hecho a la hora de probar la aplicación.

ID	Caso de prueba	Pasos	Resultado esperado	Estado
P01	Registro de usuario nuevo	1. Abrir App 2. Click en Registrarse 3. Introducir username: test1 4. Introducir email: test@gmail.com 5. password y confirmación: 12345678 6. Click en Confirmar Registro	Registro con éxito. Inicia sesión ahora	Exitoso
P02	Login con credenciales incorrectas	1. Introducir username="test1” 2. Introducir	Usuario o contraseña incorrectos	Exitoso

		password="incorrecta" 3. Click en "Iniciar sesión”		
P03	Guardar Registro Emocional	1. Login como test1 2. Seleccionar emoji 3. Escribir nota 4. Marcar Hábitos	Visible en Calendario	Exitoso
P04	Editar Registro del mismo día	1. Cambiar emoji 2. Modificar nota	Sin nota duplicada	Exitoso
P05	Crear Objetivo nuevo	1. Ir a Mis Objetivos 2. Click en FAB 3. Titulo: “Correr 10km” 4. Fecha: 2026-03-10 5. Guardar	Objetivo aparece en lista "Pendientes"	Exitoso
P06	Eliminar Objetivo	1. Pulsar en Papelera al lado de objetivo a eliminar	Objetivo desaparece de lista y de Calendario	Exitoso
P09	Configurar PIN	1. Acceder a Ajustes 2. Ir a apartado de Seguridad 3. Introducir PIN: 1234 4. Confirmar PIN: 1234 5. Reabrir App	Al reabrir la app solicita PIN	Exitoso
P10	Comprobar PIN sea único por usuario	1. Configurar PIN (P09) 2. Hacer Logout 3. Iniciar sesión como otro usuario 4. Volver a iniciar sesión con usuario con PIN	Al iniciar sesión como otro usuario no se comparte PIN. Al volver a iniciar sesión con el usuario con PIN se le solicita al usuario que lo introduzca.	Exitoso

9. Manual Técnico de Instalación

9.1 Requisitos

Mínimo absoluto:

- Android 8.0 (Oreo, API 26)
    - Justificación: En tu archivo MainActivity.kt y AppNavigation.kt utilizas @RequiresApi(Build.VERSION_CODES.O). Esto se debe a que el sistema de Canales de Notificaciones y la API de fechas que manejas (java.time) requieren como mínimo la API 26.
- Procesador: Arquitectura de 64 bits (ARMv8 o x86_64) para un rendimiento fluido de Jetpack Compose.
- Memoria RAM: Mínimo 2 GB (Recomendado 4 GB). Jetpack Compose es más exigente en memoria que el sistema antiguo de Views al renderizar la interfaz de forma declarativa.
- Seguridad: Sensor de Huella dactilar o Reconocimiento Facial compatible con BiometricPrompt (necesario para la funcionalidad de BiometricAuthScreen.kt).
- Almacenamiento: Mínimo 50 MB de espacio libre. La base de datos SQLite (Room) ocupa muy poco espacio, pero el almacenamiento crece ligeramente conforme el usuario añade fotos o muchos registros de texto en el diario.

9.2 Firma Digital

La aplicación ha sido sometida a un proceso de Firma Digital mediante una clave privada (Keystore) generada específicamente para este proyecto.

- Algoritmo de firma: V2 (Full APK Signature) y V1 (Jar Signature).
- Propósito: Garantizar que el código no ha sido alterado por terceros (Integridad) y verificar la identidad del desarrollador ante el sistema operativo Android.

9.3 Instalación

1. Descargar el archivo MentalCare_v1.0.apk desde el canal de distribución (GitHub Releases).
2. Ejecutar el archivo en el explorador de archivos del dispositivo.
3. Aceptar los diálogos de seguridad de Android.
4. Una vez finalizado, el icono personalizado (ic_launcher) aparecerá en el menú de aplicaciones.

9.4 Configuración Post-Instalación

Tras la primera ejecución, la aplicación solicitará los siguientes permisos críticos:

- POST_NOTIFICATIONS: Para el envío de recordatorios diarios de bienestar.

- SCHEDULE_EXACT_ALARM: Para garantizar que las notificaciones lleguen a la hora exacta configurada en Ajustes (Android 12+).

9.5 Proceso de Desinstalación

La aplicación está configurada para realizar una limpieza completa tras su eliminación:

1. Eliminación de Binarios: El sistema borra el archivo ejecutable.
2. Borrado de Datos Locales: La base de datos SQLite gestionada por Room (mentalcare-db) es eliminada permanentemente.
3. Limpieza de Preferencias: Los ajustes de usuario y configuraciones de seguridad almacenados en SharedPreferences se borran para proteger la privacidad del usuario saliente.

9.6 Canal de Distribución y Control de Versiones

Se ha utilizado Git como sistema de control de versiones y GitHub como plataforma de despliegue. El artefacto final (APK) está disponible en la sección de 'Releases' del repositorio, facilitando la entrega continua y el acceso al ejecutable firmado."

10. Conclusiones

El desarrollo de esta App ha supuesto un reto personal. Creo que es el proyecto que más me ha costado. Siento que tiene mucho margen de mejora pero haber conseguido hacer funcionar todo me ha parecido sorprendente. Siento que esta es una propuesta interesante sobre la que podría trabajar en el futuro.

10.1 Líneas de Trabajo Futuro

Aunque la aplicación es plenamente funcional en esta versión, el diseño está realizado de forma escalable para permitir futuras ampliaciones y mejoras, entre las que destacan:

- Análisis de tendencias: Implementación de gráficos avanzados (líneas o barras) que permitan al usuario visualizar su evolución emocional mensual.
- Sincronización en la nube: Migración parcial de la persistencia local a Firebase, facilitando el acceso a la información desde múltiples dispositivos.
- Integración con wearables: Conexión con relojes inteligentes para automatizar la recogida de datos biométricos (frecuencia cardíaca, sueño) y relacionarlos con el estado emocional del usuario.