# TIMEFLOW

# Indice

# I. Introducción

## Descripción

TimeFlow es una aplicación diseñada para gestionar tu tiempo de manera sencilla y eficiente. Incluye un calendario de eventos y un bloc de notas integrado.

El objetivo del proyecto era combinar las funcionalidades de diversas aplicaciones de gestión del tiempo en una sola, eliminando la necesidad de alternar entre múltiples herramientas.

TimeFlow ofrece:

- Una forma cómoda de crear y visualizar eventos.
- Un bloc de notas simple y fácil de usar.

## Motivación

Este proyecto final del curso es una oportunidad para demostrar lo que has aprendido y tu capacidad de desarrollo.

Siempre me ha gustado la idea de crear una aplicación móvil para gestionar el tiempo, con una interfaz atractiva y personalizada a mi estilo.

Objetivos del proyecto:

- Desarrollar una aplicación móvil con una interfaz estética y atractiva.
- Crear una herramienta fácil de usar y útil para la gestión del tiempo.

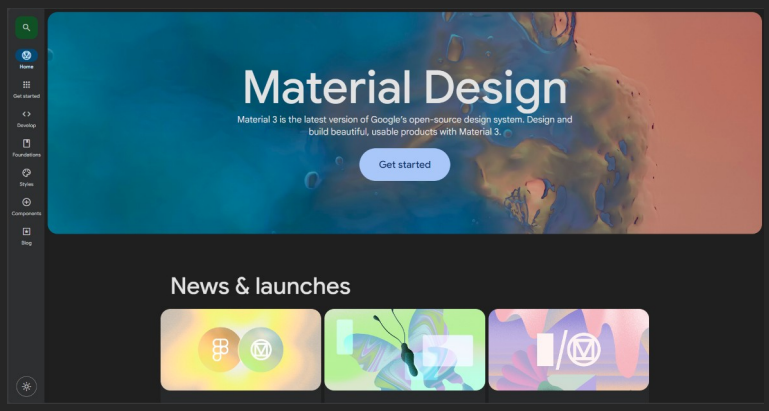
# II. Análisis e Investigación

## Investigación

### Diseño

Para la creación de mi diseño, utilicé una página recomendada por Android Studio en su documentación oficial. En esta página se pueden encontrar botones, menús, ejemplos y otros recursos útiles.

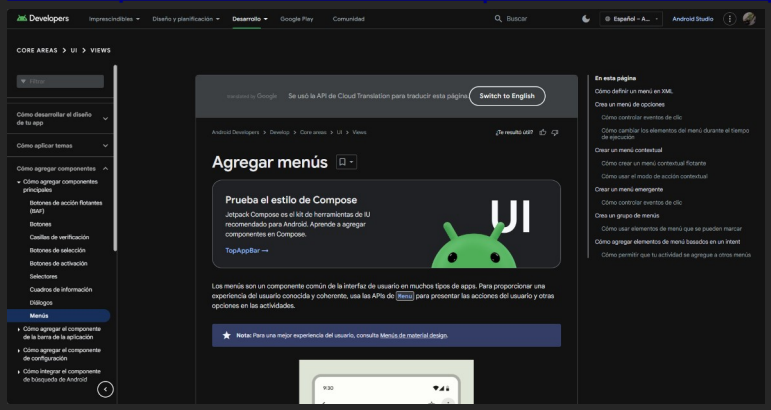
[Material Design]




### Programación

Aunque no he usado esta página extensivamente, en ocasiones he consultado la documentación oficial de Android Studio para realizar cambios específicos en algunos componentes.

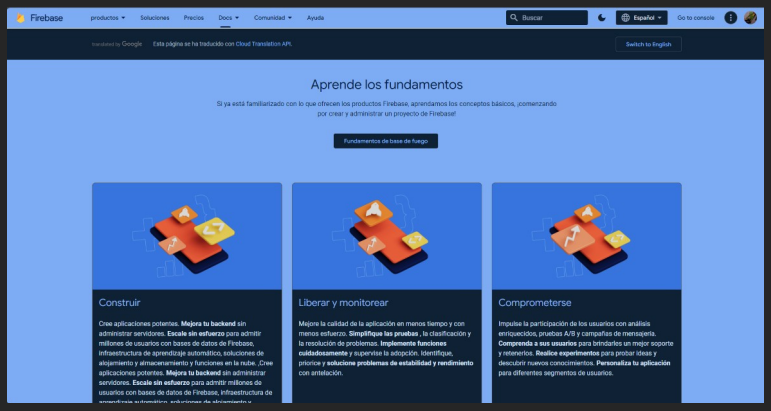
[Guías para desarrolladores | Android Developers]



## Servicios

Cuando necesitaba saber cómo conectar el proyecto a Firebase, así como borrar o usar objetos de la base de datos en la aplicación, recurrí a la documentación oficial de Firebase.

[Documentación de Firebase (google.com)]




## Otros

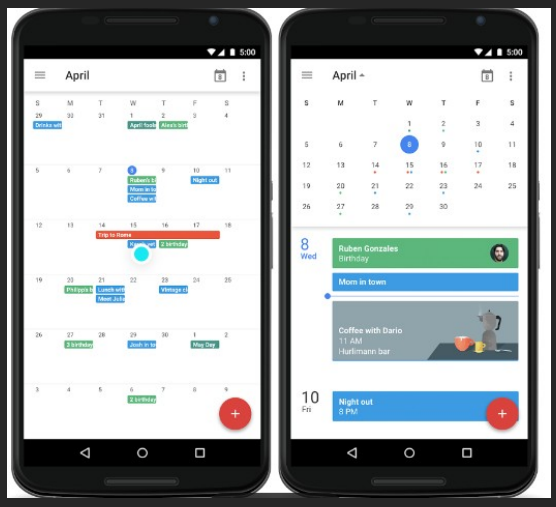
Además de la documentación oficial, he recurrido a videos instructivos y foros especializados para obtener información adicional y solucionar problemas específicos durante el desarrollo del proyecto.

## Análisis

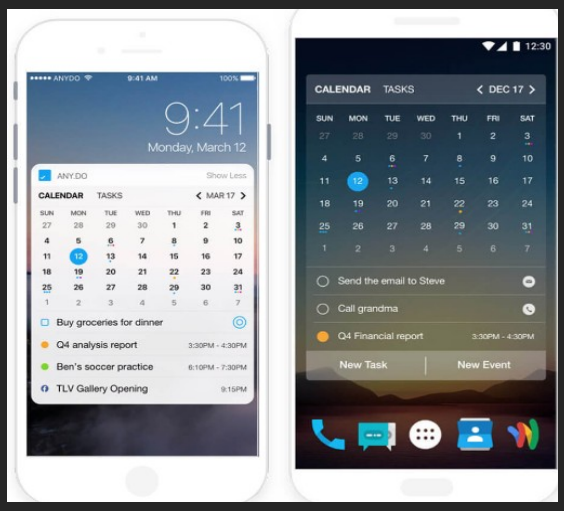
Este tipo de aplicaciones no está dirigido a todos los públicos; su objetivo principal son adultos ocupados que desean mantener un registro de sus tareas para no olvidarlas. Sin embargo, la aplicación también es lo suficientemente intuitiva como para ser utilizada por niños a partir de 6 años.

Antes de comenzar a desarrollar la aplicación, revisé las aplicaciones más instaladas y utilizadas por los usuarios para crear la mía siguiendo un formato similar.

### Google Calendar




### Any.do Calendar



# III. Desarrollo

## Módulos Utilizados

### Sistemas Informáticos:

- Instalación y configuración de Windows 10.
- Instalación y configuración de Android Studio.

### Base de Datos:

- Creación y manejo de Firebase Realtime Database.

### Programación:

- Uso del lenguaje JAVA.

### Lenguaje de marcas:

- Uso de XML.

### Desarrollo de interfaces:

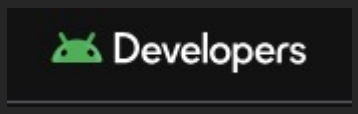
- Construcción de la aplicación siguiendo los conceptos SOLID y las reglas de usabilidad.

### Programación multimedia y dispositivos móviles:

- Uso de Android Studio.

## Softwares usados para la creación de la aplicación

### Android Studio:




Lo voy a usar para crear las funcionalidades del back y para crear el diseño del front.

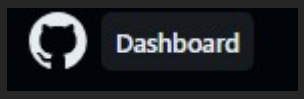
Descargar: Descarga Android Studio y App Tools - Android Developers

### Firebase:


Voy a usar Realtime Database para almacenar los datos de mi aplicación y poder acceder a ellos dependiendo de la circunstancia.

Acceder: Firebase | Google's Mobile and Web App Development Platform

### Github:




Lo estoy usando para publicar los avances del proyecto y ver las versiones y cambios introducidos.

Acceder: GitHub

Descargar: GitHub Desktop | Simple collaboration from your desktop

## Librerías utilizadas

1. implementation ("com.google.android.material:material:1.4.0") // Biblioteca de componentes de interfaz de usuario de Material Design.

2. implementation("androidx.core:core-ktx:1.9.0") // Extensiones de Kotlin para facilitar el uso de la biblioteca core de Android.

3. implementation("androidx.appcompat:appcompat:1.6.1") // Soporte para acciones de compatibilidad con versiones anteriores de Android.

4. implementation("com.google.android.material:material:1.11.0") // Versión más reciente de la biblioteca de Material Design, que puede incluir nuevas funcionalidades y correcciones.

5. implementation("androidx.constraintlayout:constraintlayout:2.1.4") // Biblioteca para crear interfaces de usuario complejas con menos anidaciones de vistas.

6. implementation("com.google.firebase:firebase-database:20.3.1") // Biblioteca para interactuar con la base de datos en tiempo real de Firebase.

7. implementation("androidx.lifecycle:lifecycle-runtime-ktx:2.6.1") // Extensiones de Kotlin para la gestión del ciclo de vida en Android.

8. implementation("androidx.activity:activity-compose:1.7.2") // Integración de actividades con Jetpack Compose.

9. implementation(platform("androidx.compose:compose-bom:2023.03.00")) // Plataforma BOM para manejar versiones compatibles de Jetpack Compose.

10. implementation("androidx.compose.ui:ui") // Componentes básicos de la interfaz de usuario de Jetpack Compose.

11. implementation("androidx.compose.ui:ui-graphics") // Utilidades gráficas para Jetpack Compose.

12. implementation("androidx.compose.ui:ui-tooling-preview") // Herramientas de previsualización para el diseño en Jetpack Compose.

13. implementation("androidx.compose.material3:material3") // Componentes de Material Design 3 para Jetpack Compose.

14. testImplementation("junit:junit:4.13.2") // Biblioteca para realizar pruebas unitarias con JUnit.

15. androidTestImplementation("androidx.test.ext:junit:1.1.5") // Extensiones de JUnit para pruebas instrumentadas.

16. androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1") // Biblioteca para realizar pruebas de la interfaz de usuario con Espresso.

17. androidTestImplementation(platform("androidx.compose:compose-bom:2023.03.00")) // Plataforma BOM para manejar versiones compatibles de Jetpack Compose en pruebas.

18. androidTestImplementation("androidx.compose.ui:ui-test-junit4") // Biblioteca para realizar pruebas de Jetpack Compose con JUnit4.

19. debugImplementation("androidx.compose.ui:ui-tooling") // Herramientas adicionales para el desarrollo y depuración con Jetpack Compose.

20. debugImplementation("androidx.compose.ui:ui-test-manifest") // Utilidades para gestionar el manifiesto durante las pruebas con Jetpack Compose.

# IV. Diseño y Planificación

## Planificación

| | Preproyecto | Seguimiento 1 | Seguimiento 2 | Seguimiento 3 |
|---|---|---|---|---|
| Planificación de la aplicación vistas base datos... | 5h | | | |
| - Creación del login<br>- Creación de la vista para crear eventos | | 12h | | |
| - Crear servicio de notificaciones | | | 15h | |
| - Creación de la vista “notepad”<br>- Creación de la vista perfil | | | | 16h |

## Diseño

### Principios de usabilidad:

1. Se usan iconos reconocibles para comodidad del usuario
2. Se utiliza la barra inferior de google para que los usuarios se sientan familiarizados
3. La aplicación tiene un nombre bonito y elegante
4. Colores simples para que la aplicación no se sienta cargada
5. La aplicación cumple los principios SOLID

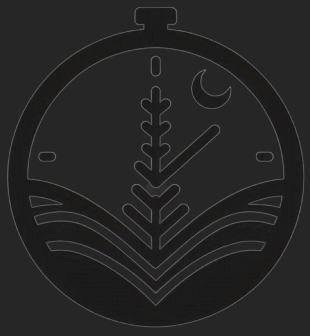
### Inspirado:

Al crear mi aplicación me di cuenta de que no podía usar el formato que quisiera porque si no mi aplicación quedaría muy cutre y seria incomoda de usar para los usuarios teniendo esto en cuenta tome la decisión de inspirarme con aplicaciones

El diseño de mi aplicación está bastante inspirado con el de Whatsapp en modo oscuro

### Logotipo:

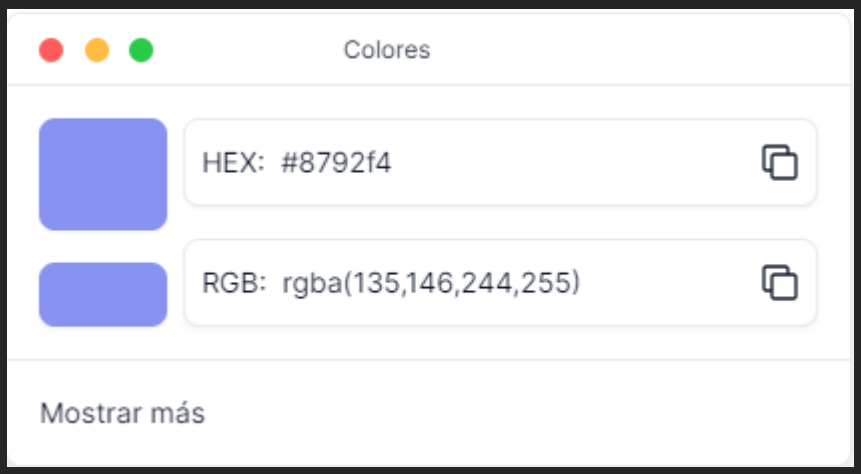
El icono ha sido creado usando las nuevas tecnologías de inteligencia artificias

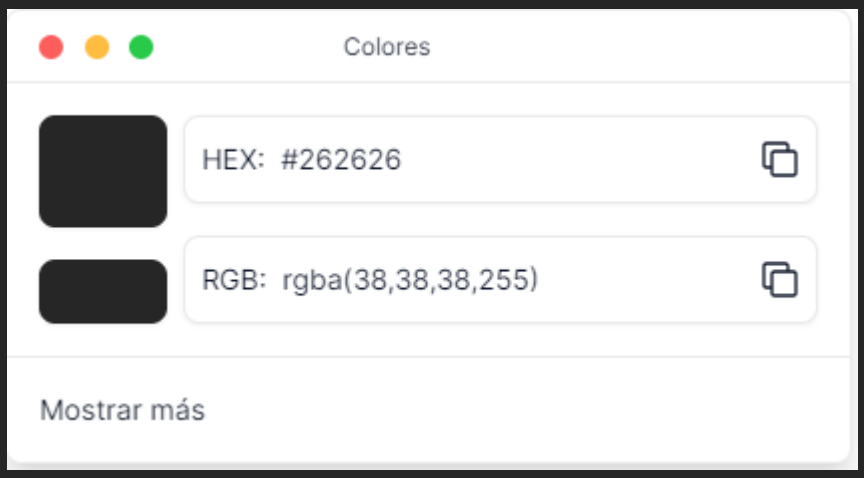


### Colores:

Mi aplicación al estar basada en Whatsapp en modo oscuro usa muy pocos colores porque sigue la teoría de la simplicidad para que el usuario se sienta más cómodo y no tan abrumado

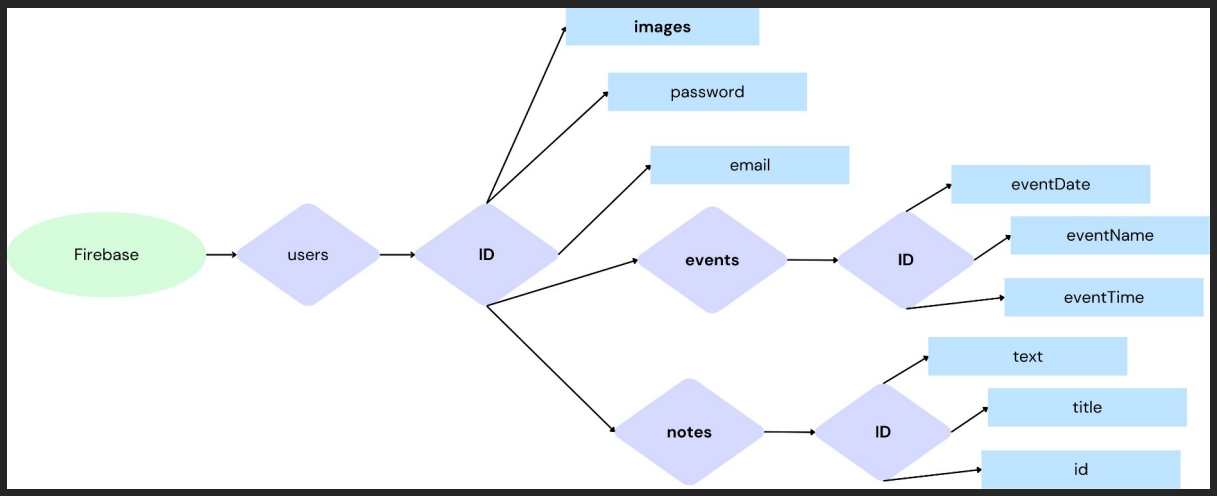
Estos son los dos colores principales:



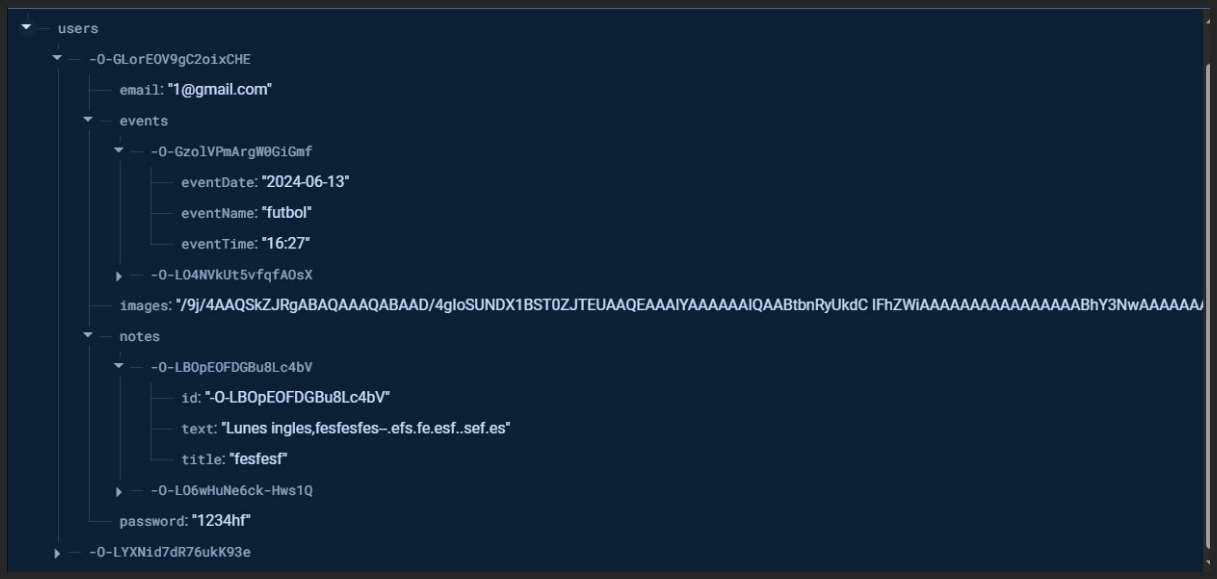



## Base de datos

Este es el diagrama de mi base de datos los “ID” son unos id únicos




Estoy usando Realtime Database para almacenar mis datos:




## Users (Usuarios):

- Este es el nodo principal que contiene la información de cada usuario registrado en tu sistema.
- Cada usuario tiene un "id único" que lo identifica de manera exclusiva en la base de datos.
- Dentro de cada "id único" de usuario se almacenan los siguientes atributos:
- email: La dirección de correo electrónico asociada al usuario.
- password: La contraseña del usuario (probablemente almacenada de manera segura, por ejemplo, encriptada o hash).

- images: Imágenes asociadas al perfil del usuario, si es aplicable.

## Events (Eventos):

- Este nodo almacena información sobre los eventos registrados en tu sistema.
- Cada evento tiene un "id único" que lo distingue de otros eventos en la base de datos.
- Dentro de cada "id único" de evento se encuentran los siguientes atributos:
- eventDate: La fecha en la que ocurre el evento.
- eventName: El nombre o título del evento.
- eventTime: La hora en la que el evento tiene lugar.

## Notes (Notas):

- Aquí se guardan las notas creadas en tu sistema.
- Cada nota tiene un "id único" que la identifica de manera única dentro de la base de datos.
- Dentro de cada "id único" de nota se almacenan los siguientes atributos:
- text: El contenido principal de la nota, el texto que contiene la información que el usuario ha ingresado.
- title: El título o encabezado de la nota para identificarla.
- id: Es el mismo id se usa para algunas consultas.

# V. Pruebas, Permisos y Control de Errores

## Pruebas

### Descripción de Monkey

Monkey es una herramienta de prueba incluida en el Android SDK que genera eventos aleatorios (toques, deslizamientos, presiones de teclas, etc.) en la interfaz de usuario de la aplicación. Es utilizada para simular un uso intensivo y detectar posibles fallos, cuelgues o comportamientos inesperados bajo condiciones extremas.

### Resultados de las Pruebas

Las pruebas de estrés se realizaron en la versión 1.0.0 de la aplicación usando un Samsung Galaxy S21 con Android 11. Se ejecutaron 5000 eventos aleatorios, evaluando la estabilidad y robustez de la aplicación. La aplicación respondió adecuadamente, sin fallos críticos ni comportamientos anómalos. Estas pruebas aseguran que la aplicación puede manejar un uso intensivo sin problemas significativos.

## Permisos

El usuario necesita los siguientes permisos en el Manifest:

```
<uses-permission
android:name="android.permission.POST_NOTIFICATIONS" />
<uses-permission
android:name="android.permission.FOREGROUND_SERVICE" />
<uses-permission android:name="android.permission.INTERNET" />
```

También dentro del Manifest se declara el servicio que está funcionando el servicio en segundo plano:

```
<service android:name=".Service.EventNotificationService" />
```

## Control de errores

El proyecto implementa un sistema integral para gestionar errores y excepciones que puedan surgir durante la ejecución. A continuación, se describen las políticas y prácticas adoptadas para este propósito.

### Políticas de Manejo de Errores

1. Detección Temprana: El código está diseñado para detectar errores tan pronto como ocurren, mediante la validación rigurosa de entradas y condiciones previas.

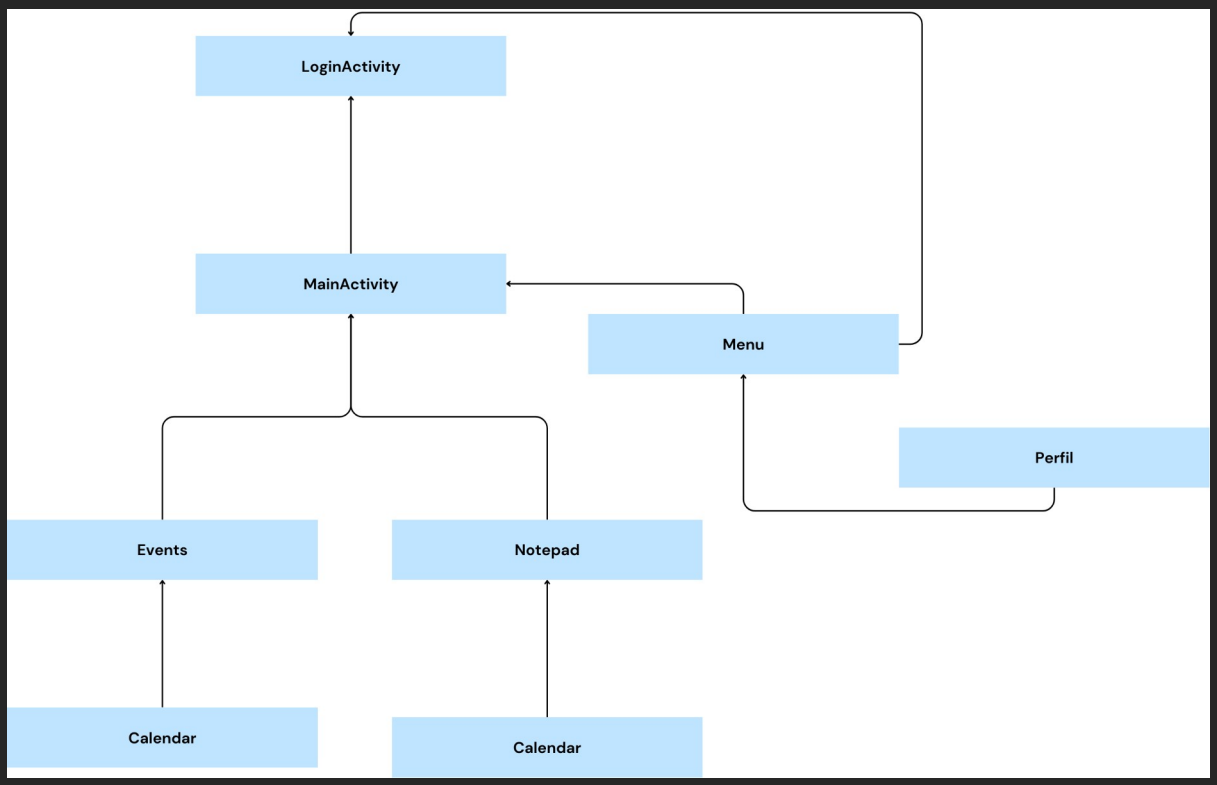
2. Consistencia del Estado: En caso de errores, se intenta mantener la consistencia del estado del sistema, evitando que se propaguen inconsistencias.

3. Notificación y Registro: Todos los errores y excepciones se registran en un archivo de logs para su análisis posterior. Los usuarios son notificados con mensajes claros y concisos.

# VI. Organización del proyecto, Clases y Casos de uso

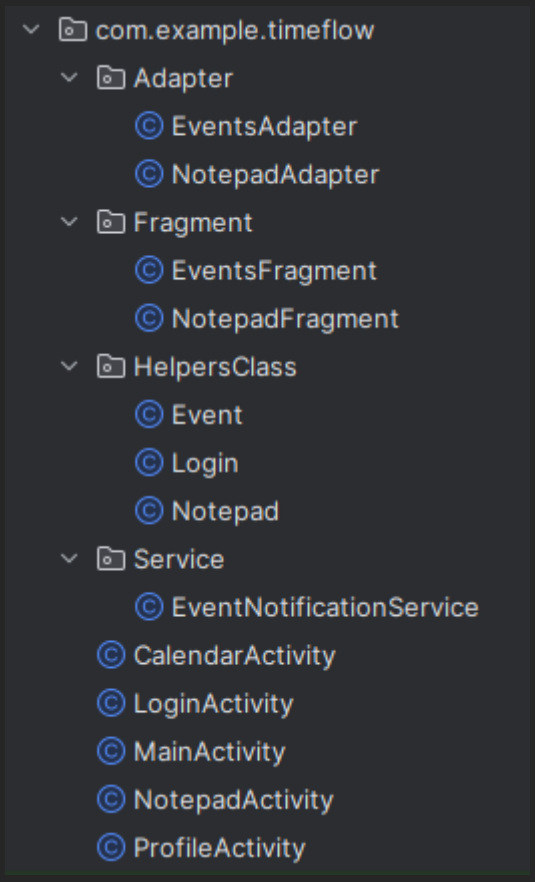
## Organización del proyecto

Diagrama de clases:

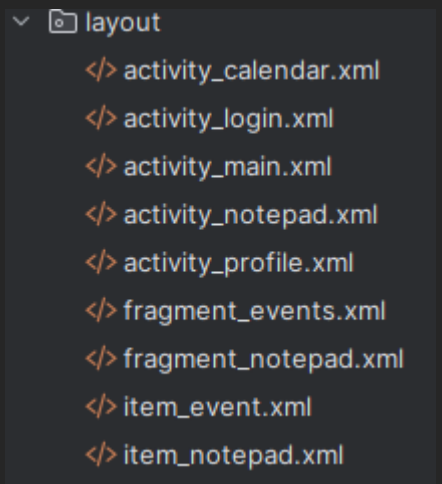


Organizacion de las clases:

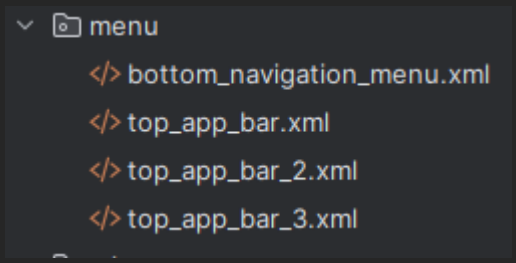
- Lógica




- Front

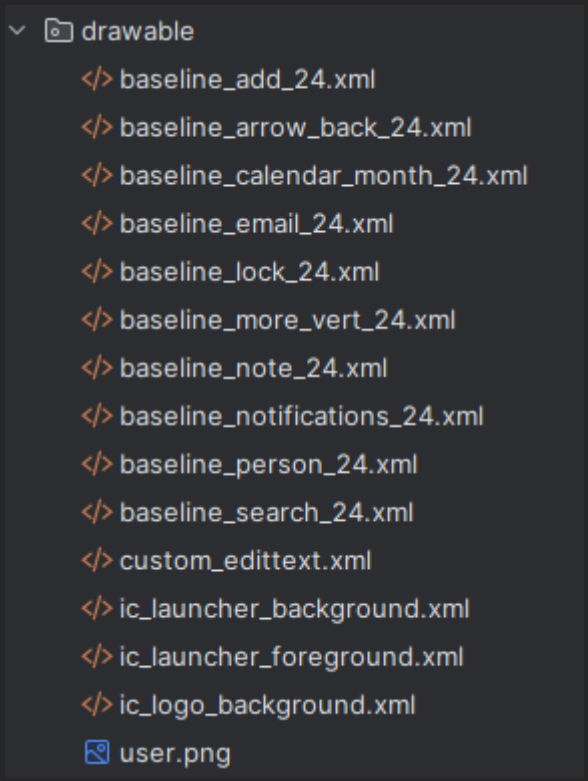


- Menús

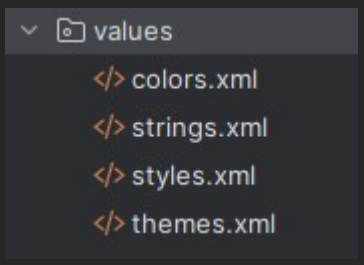



Hay que tener en cuenta que estos activity también usan estilos o iconos:

- Iconos




- Estilos



## Clases

En este apartado quiero explicar de que esta encargada cada una de las clases mostradas.

### EventsAdapter

1. Variables de instancia:

- private Context mContext: Almacena el contexto de la aplicación para poder utilizarlo en diferentes partes de la clase.

- private List<Event> mEvents: Contiene una lista de objetos de tipo Event, que representan los eventos que se mostrarán en el RecyclerView.

- private OnItemClickListener mListener: Almacena una instancia de la interfaz OnItemClickListener, que se utiliza para manejar los eventos de clic en los elementos del RecyclerView.

2. Interfaz OnItemClickListener:

- Define un método onItemClick(int position) que será implementado por quien instancie EventsAdapter. Este método se invocará cuando se haga clic en un elemento del RecyclerView.

3. Constructor EventsAdapter:

- Recibe como parámetros:

  - Context context: Contexto de la aplicación.

  - List<Event> events: Lista de eventos que se mostrarán en el RecyclerView.

  - OnItemClickListener listener: Instancia de la interfaz OnItemClickListener para manejar eventos de clic en los elementos del RecyclerView.

- Inicializa las variables de instancia mContext, mEvents y mListener con los valores recibidos.

4. Método onCreateViewHolder:

- Método obligatorio de RecyclerView.Adapter que infla el diseño de cada elemento de la lista.

- Crea y devuelve una instancia de EventsViewHolder, pasándole el diseño inflado y el listener de clics.

5. Método onBindViewHolder:

- Método obligatorio de RecyclerView.Adapter que actualiza el contenido de cada elemento de la lista.

- Obtiene el evento correspondiente de la lista mEvents según la posición proporcionada.

- Actualiza las vistas (eventName, eventDate y eventTime) del EventsViewHolder con la información del evento.

6. Método getItemCount:

- Método obligatorio de RecyclerView.Adapter que devuelve el número de elementos en la lista mEvents.

7. Método scrollToBottom:

- Método personalizado que permite hacer scroll hasta el último elemento del RecyclerView.

- Verifica si hay elementos en la lista (getItemCount() > 0) y, de ser así, hace un scroll suave hasta la posición del último elemento.

8. Clase estática EventsViewHolder:

- Clase interna estática que extiende RecyclerView.ViewHolder.

- Contiene referencias a las vistas (eventName, eventDate, eventTime) dentro de cada elemento de la lista.

- Configura un OnClickListener para el itemView`(la vista raíz de cada elemento) que invoca el método onItemClick  del listener cuando se hace clic en un elemento del RecyclerView.

## NotepadAdapter

1. Variables de instancia:

- mContext: Almacena el contexto de la aplicación para utilizar en diferentes partes de la clase.

- mNotepads: Lista de objetos Notepad que se mostrarán en el RecyclerView.

- mListener: Interfaz OnItemClickListener para manejar eventos de clic en los elementos del RecyclerView.

2. Interfaz OnItemClickListener:

- Define un método onItemClick(int position) que será implementado por quien instancie NotepadAdapter. Este método se invocará cuando se haga clic en un elemento del RecyclerView.

3. Constructor NotepadAdapter:

- Recibe como parámetros:

  - context: Contexto de la aplicación.

  - notepads: Lista de objetos Notepad que se mostrarán en el RecyclerView.

  - listener: Instancia de la interfaz OnItemClickListener para manejar eventos de clic en los elementos del RecyclerView.

- Inicializa las variables de instancia mContext, mNotepads y mListener con los valores recibidos.

4. Método onCreateViewHolder:

- Método obligatorio de RecyclerView.Adapter que infla el diseño de cada elemento de la lista (item_notepad.xml).

- Crea y devuelve una instancia de NotepadViewHolder, pasándole el diseño inflado y el listener de clics.

5. Método onBindViewHolder:

- Método obligatorio de RecyclerView.Adapter que actualiza el contenido de cada elemento de la lista.

- Obtiene el objeto Notepad correspondiente de la lista mNotepads según la posición proporcionada.

- Actualiza las vistas (title y text) del NotepadViewHolder con la información del objeto Notepad.

6. Método getItemCount:

- Método obligatorio de RecyclerView.Adapter que devuelve el número de elementos en la lista mNotepads.

7. Clase estática NotepadViewHolder:

- Clase interna estática que extiende RecyclerView.ViewHolder.

- Contiene referencias a las vistas (title y text) dentro de cada elemento de la lista.

- Configura un OnClickListener para itemView (la vista raíz de cada elemento) que invoca el método onItemClick del listener cuando se hace clic en un elemento del RecyclerView.

## EventsFragment

1. Declaraciones de variables:

- mRecyclerView: Instancia de RecyclerView para mostrar la lista de eventos.

- mAdapter: Instancia de EventsAdapter para gestionar los datos del RecyclerView.

- mEvents: Lista de eventos que se mostrarán en el RecyclerView.

- progressBar: Instancia de ProgressBar para mostrar el progreso de carga.

- textView: Instancia de TextView para mostrar un mensaje cuando no hay eventos.

- userId: Almacena el ID del usuario actual.

2. onCreateView:

- Método del ciclo de vida de Fragment donde se infla el diseño de la interfaz de usuario (fragment_events.xml).

- Se inicializan los elementos de la interfaz de usuario, se configuran SharedPreferences para obtener userId, se inicia un servicio de notificaciones de eventos, se configura el RecyclerView, se configuran los botones y se inicia la escucha de la base de datos si userId no está vacío. Finalmente, se manejan los argumentos que podrían haber sido pasados al fragmento.

3. initializeUI:

- Método para inicializar textView obtenido del layout del fragmento.

4. setupSharedPreferences:

- Método para obtener userId desde SharedPreferences. Si no se encuentra en SharedPreferences, se intenta obtenerlo de los extras de la actividad que inició el fragmento.

5. startEventNotificationService:

- Método para iniciar el servicio EventNotificationService, que gestionará las notificaciones de eventos.

6. setupRecyclerView:

- Método para configurar el RecyclerView.

- Se obtiene la referencia al RecyclerView del layout, se establece un LayoutManager y se crea un nuevo EventsAdapter con el contexto actual, la lista de eventos vacía y el fragmento como listener de clics.

7. setupCalendarButton:

- Método para configurar un OnClickListener en un botón de calendario (calendarButton) para navegar a la actividad de calendario (CalendarActivity).

8. setupProgressBar:

- Método para configurar la visibilidad inicial y obtener la referencia a un ProgressBar (progressBar) del layout del fragmento.

9. setupDatabaseListener:

- Método para configurar un ValueEventListener en la base de datos Firebase.

- Se escucha los eventos del usuario actual y se filtran los eventos que coincidan con la fecha actual. Luego, se actualiza la lista de eventos y la interfaz de usuario.

10. handleArguments:

- Método para manejar los argumentos pasados al fragmento.

- Se obtienen y manejan los datos relacionados con consultas de palabras clave, actualizaciones y fechas seleccionadas desde la base de datos.

11. updateEventList (versión DataSnapshot):

- Método para actualizar la lista de eventos a partir de un DataSnapshot de Firebase.

- Se ordenan los eventos por fecha y se actualiza la interfaz de usuario.

12. updateEventList (versión List<Event>):

- Método para actualizar la lista de eventos a partir de una lista de objetos Event.

- Se ordenan los eventos por fecha y se actualiza la interfaz de usuario.

13. updateEventListFromMap:

- Método para actualizar la lista de eventos a partir de un TreeMap de eventos ordenados por fecha.

- Se actualiza la lista de eventos y se notifica al adapter para reflejar los cambios en el RecyclerView.

14. updateUI:

- Método para actualizar la interfaz de usuario después de cambios en la lista de eventos.

- Se desplaza el RecyclerView hasta la posición más baja, se actualiza la visibilidad de progressBar y textView según si hay eventos para mostrar.

15. fetchDataFromDatabase:

- Método para recuperar datos de la base de datos Firebase filtrados por una palabra clave de consulta.

- Se configura una consulta Firebase para buscar eventos que coincidan con la palabra clave y se actualiza la lista de eventos y la interfaz de usuario.

16. fetchDataByDateFromDatabase:

- Método para recuperar datos de la base de datos Firebase filtrados por una fecha específica.

- Se configura una consulta Firebase para buscar eventos que coincidan con la fecha seleccionada y se actualiza la lista de eventos y la interfaz de usuario.

17. handleDatabaseError:

- Método para manejar errores de la base de datos Firebase.

- Se muestra un mensaje de error en el registro y se oculta el progressBar.

18. goToCalendarActivity:

- Método para iniciar la actividad CalendarActivity y pasar userId como extra.

19. onItemClick:

- Implementación del método onItemClick de la interfaz EventsAdapter.OnItemClickListener.

## NotepadFragment

1. Declaraciones de variables:

- mRecyclerView: Instancia de RecyclerView para mostrar la lista de notas.

- mAdapter: Instancia de NotepadAdapter para gestionar los datos del RecyclerView.

- mNotepad: Lista de notas que se mostrarán en el RecyclerView.

- progressBar: Instancia de ProgressBar para mostrar el progreso de carga.

- textView: Instancia de TextView para mostrar un mensaje cuando no hay notas.

- userId: Almacena el ID del usuario actual.

2. onCreateView:

- Método del ciclo de vida de Fragment donde se infla el diseño de la interfaz de usuario (fragment_notepad.xml).

- Se inicializan los elementos de la interfaz de usuario, se configuran SharedPreferences para obtener userId, se configura el RecyclerView con un GridLayoutManager de 2 columnas, se configuran los botones y se inicia la escucha de la base de datos si userId no está vacío.

3. initializeUI:

- Método para inicializar textView obtenido del layout del fragmento.

4. setupSharedPreferences:

- Método para obtener userId desde SharedPreferences. Si no se encuentra en SharedPreferences, se intenta obtenerlo de los extras de la actividad que inició el fragmento.

5. setupRecyclerView:

- Método para configurar el RecyclerView.

- Se obtiene la referencia al RecyclerView del layout, se establece un GridLayoutManager con 2 columnas y se crea un nuevo NotepadAdapter con el contexto actual, la lista de notas vacía y el fragmento como listener de clics.

6. setupCalendarButton:

- Método para configurar un OnClickListener en un botón de calendario (calendarButton) para navegar a la actividad de notas (NotepadActivity).

7. setupProgressBar:

- Método para configurar la visibilidad inicial y obtener la referencia a un ProgressBar (progressBar) del layout del fragmento.

8. setupDatabaseListener:

- Método para configurar un ValueEventListener en la base de datos Firebase.

- Se escuchan las notas del usuario actual y se actualiza la lista de notas y la interfaz de usuario en consecuencia.

9. updateNoteList:

- Método para actualizar la lista de notas con una nueva lista de notas recibida como parámetro.

- Se limpia la lista actual mNotepad, se añaden todas las notas nuevas y se notifica al adapter para reflejar los cambios en el RecyclerView.

10. updateUI:

- Método para actualizar la interfaz de usuario después de cambios en la lista de notas.

- Se actualiza la visibilidad de progressBar y textView según si hay notas para mostrar.

11. goToNotepadActivity:

- Método para iniciar la actividad NotepadActivity y pasar userId como extra.

12. onItemClick:

- Implementación del método onItemClick de la interfaz NotepadAdapter.OnItemClickListener.

- Se obtiene la nota clickeada en la posición especificada y se inicia la actividad NotepadActivity pasando la nota como extra.

13. handleDatabaseError:

- Método para manejar errores de la base de datos Firebase.

- Se muestra un mensaje de error en el registro y se oculta el progressBar.

## Event

- Variables de instancia:

  - eventName: Almacena el nombre del evento.

  - eventTime: Guarda la hora del evento.

  - eventDate: Contiene la fecha del evento.

- Constructores:

  - Event(): Constructor vacío requerido para la deserialización por Firebase.

  - Event(eventDate, eventTime, eventName): Constructor que inicializa las variables eventName, eventTime y eventDate al crear un nuevo objeto Event.

- Métodos:

  - getEventName(): Retorna el nombre del evento almacenado.

  - setEventName(eventName): Establece el nombre del evento con el valor proporcionado.

  - getEventTime(): Retorna la hora del evento guardada.

  - setEventTime(eventTime): Define la hora del evento con el valor especificado.

  - getEventDate(): Retorna la fecha del evento.

  - setEventDate(eventDate): Establece la fecha del evento con el valor dado.

## Login

- Variables de instancia:

  - email: Almacena la dirección de correo electrónico del usuario.

  - password: Guarda la contraseña del usuario.

- Constructores:

  - Login(): Constructor vacío que es requerido para ciertas operaciones de deserialización.

  - Login(email, password): Constructor que inicializa las variables email y password al crear un nuevo objeto Login.

- Métodos:

  - getEmail(): Retorna la dirección de correo electrónico almacenada.

  - setEmail(email): Establece la dirección de correo electrónico con el valor proporcionado.

  - getPassword(): Retorna la contraseña guardada.

  - setPassword(password): Define la contraseña con el valor especificado.

## Notepad

- Variables de instancia:

  - id: Almacena un identificador único para el bloc de notas.

  - title: Guarda el título del bloc de notas.

  - text: Contiene el texto del bloc de notas.

- Constructores:

  - Constructor vacío: Requerido para ciertas operaciones de serialización/deserialización.

  - Constructor con parámetros: Inicializa las variables id, title y text al crear un nuevo objeto Notepad.

- Métodos:

  - Métodos getter y setter: Para acceder y modificar las variables privadas (id, title, text).

## EventNotificationService

- Variables de instancia:

  - CHANNEL_ID: Identificador del canal de notificación.

  - NOTIFICATION_ID: Identificador único para las notificaciones.

  - TAG: Etiqueta para registros de mensajes de depuración.

  - userId: Identificador del usuario actual.

  - eventsRef: Referencia a la base de datos Firebase para eventos del usuario.

  - handler: Manejador utilizado para ejecutar tareas periódicas.

  - runnable: Tarea que se ejecuta periódicamente para verificar eventos.

- Métodos principales:

  - onCreate(): Inicializa el servicio.

  - onStartCommand(): Se llama cuando se inicia el servicio.

  - onDestroy(): Se llama cuando se destruye el servicio.

  - initializeService(): Inicializa el servicio, configura el canal de notificación y comienza el servicio en primer plano.

- createNotificationChannel(): Crea el canal de notificación para versiones de Android Oreo y superiores.

- createNotification(): Crea una notificación para mostrar en el servicio en primer plano.

- checkEvents(): Verifica los eventos del usuario y muestra notificaciones para eventos próximos.

- isEventNear(Event event): Verifica si un evento está próximo en el tiempo.

- showNotification(String title, String message, int notificationId): Muestra una notificación para un evento próximo.

## CalendarActivity

Variables:

- buttonHour: Botón para seleccionar la hora del evento.
- saveButton: Botón para guardar un evento nuevo o editado.
- deleteButton: Botón para eliminar un evento existente.
- editButton: Botón para editar un evento existente.
- hourText: Campo de texto para ingresar la hora del evento.
- dateText: Campo de texto para ingresar la fecha del evento.
- nameText: Campo de texto para ingresar el nombre del evento.
- progressBar: Barra de progreso utilizada para indicar la carga o procesamiento.
- topAppBar: Barra superior de material que contiene opciones de navegación y menú.

Métodos:

- onCreate(Bundle savedInstanceState): Método de inicio de la actividad. Configura la vista, inicializa vistas y botones, recupera el ID de usuario y configura el calendario.
- logout(): Cierra la sesión del usuario, eliminando el ID de usuario de las preferencias compartidas y deteniendo el servicio de notificaciones de eventos.
- stopEventNotificationService(): Detiene el servicio de notificaciones de eventos.
- initializeViews(): Inicializa las vistas de la interfaz de usuario y oculta la barra de progreso.
- setupButtons(): Configura los botones según si se está creando un nuevo evento o editando uno existente.
- setupForEditEvent(Intent intent): Prepara la interfaz para editar un evento existente, mostrando los detalles del evento seleccionado.
- setupForNewEvent(): Prepara la interfaz para crear un nuevo evento, ocultando el botón de eliminar y mostrando el botón de guardar.
- retrieveUserId(): Recupera el ID de usuario desde las preferencias compartidas o desde los extras del intent.

- getSelectedDate(): Obtiene la fecha seleccionada por el usuario desde el calendario y la muestra en el campo de texto correspondiente.
- onHourButtonClick(): Abre un diálogo para seleccionar la hora del evento y muestra la hora seleccionada en el campo de texto.
- deleteEvent(): Elimina el evento seleccionado de la base de datos Firebase Realtime Database.
- editEvent(): Edita la fecha y hora de un evento existente en la base de datos Firebase Realtime Database.
- onSaveButtonClick(): Guarda un nuevo evento en la base de datos Firebase Realtime Database o muestra un mensaje de error si algún campo está vacío o el nombre del evento es demasiado largo.
- showProgressBar(): Muestra la barra de progreso para indicar la carga o procesamiento.
- hideProgressBar(): Oculta la barra de progreso.

## LoginActivity

Variables:

- signupEmail: Campo de texto para ingresar el correo electrónico durante el registro o inicio de sesión.
- signupPassword: Campo de texto para ingresar la contraseña durante el registro o inicio de sesión.
- signupButton: Botón para registrar un nuevo usuario.
- loginButton: Botón para iniciar sesión con un usuario existente.
- progressBar: Barra de progreso utilizada para indicar la carga o procesamiento de operaciones.

Métodos:

- onCreate(Bundle savedInstanceState): Método de inicio de la actividad. Configura la vista, inicializa vistas y botones, y verifica si hay un usuario existente con sesión activa.
- initializeViews(): Inicializa las vistas de la interfaz de usuario y oculta la barra de progreso.
- checkForExistingUser(): Verifica si hay un usuario con sesión activa almacenado en las preferencias compartidas y redirige a la actividad principal si existe.
- setupButtonListeners(): Configura los listeners para los botones de registro e inicio de sesión.
- registerUser(): Registra un nuevo usuario en la base de datos Firebase Realtime Database si el correo electrónico no está en uso.
- createUser(DatabaseReference reference, String email, String password): Crea un nuevo usuario en la base de datos Firebase con el correo electrónico y contraseña proporcionados, y guarda el ID de usuario en las preferencias compartidas.
- loginUser(): Inicia sesión con el correo electrónico y contraseña proporcionados verificando su existencia en la base de datos Firebase.
- verifyUser(DataSnapshot snapshot, String userPassword): Verifica la contraseña del usuario comparándola con la almacenada en la base de datos Firebase.

- saveUserIdToPreferences(String userId): Guarda el ID de usuario en las preferencias compartidas para mantener la sesión activa.
- goToEventsActivity(String userId): Inicia la actividad principal (MainActivity) y pasa el ID de usuario como extra.
- validatePassword(): Valida que la contraseña ingresada durante el registro o inicio de sesión no esté vacía.
- validateEmail(): Valida que el correo electrónico ingresado durante el registro o inicio de sesión tenga un formato válido.
- showProgressBar(): Muestra la barra de progreso para indicar la carga o procesamiento.
- hideProgressBar(): Oculta la barra de progreso.
- showToastAndHideProgressBar(String message): Muestra un mensaje Toast con el mensaje proporcionado y luego oculta la barra de progreso.

## MainActivity

Variables:

- navigationBarView: Vista de barra de navegación inferior (NavigationBarView) utilizada para la navegación entre fragmentos.
- topAppBar: Barra de herramientas superior (MaterialToolbar) que contiene elementos de menú y opciones de búsqueda.
- searchView: Vista de búsqueda (SearchView) utilizada para realizar búsquedas dentro de los fragmentos.
- PDF_URL: URL del archivo PDF que se descargará y abrirá cuando se seleccione la opción manual desde el menú superior.
- userId: Identificador único del usuario actual, pasado como extra desde la actividad de inicio de sesión (LoginActivity).

Métodos:

- onCreate(Bundle savedInstanceState):
  Método de inicio de la actividad. Configura la vista (R.layout.activity_main), inicializa vistas y fragmento inicial (EventsFragment), y configura los listeners para la barra de navegación inferior y el menú superior.
- setupBottomNavigationBar():
  Configura el listener de selección de elementos de la barra de navegación inferior (NavigationBarView). Reemplaza el fragmento actual con EventsFragment o NotepadFragment según el elemento seleccionado, y controla la visibilidad de los elementos del menú superior.
- handleSearch():
  Configura la vista de búsqueda (SearchView) para permitir al usuario buscar eventos por nombre. Maneja la visibilidad de la barra de herramientas superior y la vista de búsqueda, y actualiza dinámicamente los resultados de búsqueda en EventsFragment.
- setupTopAppBarMenu():
  Configura los listeners para los elementos del menú de la barra de herramientas superior (MaterialToolbar). Maneja las acciones de búsqueda, cierre de sesión, búsqueda en calendario, visualización de perfil y descarga del manual PDF.

- downloadAndOpenPdf():
  Descarga y abre el archivo PDF especificado por PDF_URL utilizando una intención (Intent) de visualización. Muestra un mensaje Toast si no se encuentra un visor de PDF adecuado en el dispositivo.
- calendarSearch():
  Muestra un diálogo de selección de fecha (DatePickerDialog) para que el usuario seleccione una fecha específica. Actualiza EventsFragment con eventos programados para la fecha seleccionada.
- replaceFragment(Fragment fragment):
  Reemplaza el contenido del contenedor de fragmentos (R.id.frameLayout) con el fragmento proporcionado.
- logout():
  Cierra la sesión del usuario eliminando el ID de usuario de las preferencias compartidas, deteniendo el servicio de notificación de eventos (EventNotificationService) y redirigiendo al usuario a la actividad de inicio de sesión (LoginActivity).
- stopEventNotificationService():
  Detiene el servicio de notificación de eventos (EventNotificationService).
- goToProfileActivity(String userId):
  Inicia la actividad del perfil del usuario (ProfileActivity) pasando el ID de usuario como extra.

## NotepadActivity

Variables:

- topAppBar: Barra de herramientas superior (MaterialToolbar) utilizada para la navegación y acciones de menú.
- userId: Identificador único del usuario actual.
- id: Identificador de la nota actualmente mostrada o editada.
- textTitle: Campo de texto (EditText) para el título de la nota.
- textText: Campo de texto (EditText) para el contenido de la nota.
- clickedNote: Objeto de tipo Notepad que representa la nota seleccionada para visualización o edición desde la actividad anterior.

Métodos:

- onCreate(Bundle savedInstanceState):

Método de inicio de la actividad. Configura la vista (R.layout.activity_notepad), inicializa vistas y configura la barra de herramientas superior (topAppBar). También carga la nota seleccionada desde el intent recibido.

- initialize():

Inicializa las variables userId, textTitle, textText y topAppBar. Lee el ID de usuario desde las preferencias compartidas o desde el intent si está vacío.

- setupTopAppBar():

Configura los listeners para la barra de herramientas superior (topAppBar). Maneja las acciones de navegación, cierre de sesión y eliminación de notas.

- loadNoteFromIntent():

Extrae la nota seleccionada del intent recibido. Si hay una nota válida, establece el texto del título y el contenido en los campos correspondientes (textTitle y textText).

- onBackPressed():

Sobreescritura del método onBackPressed() para guardar la nota y finalizar la actividad cuando se presiona el botón de retroceso.

- saveNoteAndFinish():

Guarda la nota actual llamando al método saveNote() y finaliza la actividad.

- logout():

Cierra la sesión del usuario eliminando el ID de usuario de las preferencias compartidas, deteniendo el servicio de notificación de eventos (EventNotificationService) y redirigiendo al usuario a la actividad de inicio de sesión (LoginActivity).

- stopEventNotificationService():

Detiene el servicio de notificación de eventos (EventNotificationService).

- saveNote():

Guarda la nota en la base de datos Firebase Realtime Database bajo el nodo correspondiente al usuario (users/{userId}/notes). Actualiza la nota existente o crea una nueva según sea necesario. También maneja la eliminación de la nota si se borran el título y el contenido.

- deleteCurrentNote():

Elimina la nota actualmente mostrada o editada de la base de datos Firebase Realtime Database bajo el nodo de notas del usuario (users/{userId}/notes).

## ProfileActivity

Variables:

- emailText: Campo de texto (EditText) para mostrar el correo electrónico del usuario.
- passwordText: Campo de texto (EditText) para mostrar y cambiar la contraseña del usuario.
- imageButton: Botón de imagen (ImageButton) utilizado para cargar y mostrar la imagen de perfil del usuario.
- deleteButton: Botón (Button) para eliminar la cuenta del usuario.
- changeButton: Botón (Button) para cambiar la contraseña del usuario.
- progressBar: Barra de progreso (ProgressBar) para mostrar el estado de las operaciones.
- PICK_IMAGE_REQUEST: Constante entera que representa el código de solicitud para seleccionar una imagen desde la galería.
- imageUri: URI de la imagen seleccionada desde la galería.
- imageRef: Referencia a la ubicación de almacenamiento de imágenes del usuario en la base de datos Firebase Realtime Database.
- userRef: Referencia a la ubicación del usuario en la base de datos Firebase Realtime Database.
- userId: Identificador único del usuario actual.
- topAppBar: Barra de herramientas superior (MaterialToolbar) utilizada para la navegación y acciones de menú.

Métodos:

- onCreate(Bundle savedInstanceState):

Método de inicio de la actividad. Configura la vista (R.layout.activity_profile), inicializa vistas y establece la barra de herramientas superior (topAppBar). También carga la información del perfil del usuario y configura los listeners de los botones.

- initializeViews():

Inicializa las vistas (emailText, passwordText, imageButton, deleteButton, changeButton, progressBar) y obtiene el userId del intent recibido o de las preferencias compartidas.

- setupTopAppBar():

Configura los listeners para la barra de herramientas superior (topAppBar). Maneja la navegación de vuelta y el cierre de sesión del usuario.

- setupUserProfile():

Obtiene y muestra la información del perfil del usuario (email y password) desde la base de datos Firebase Realtime Database. También carga y muestra la imagen de perfil del usuario si está disponible.

- setupButtons():

Configura los listeners para los botones deleteButton, changeButton y imageButton. Maneja las acciones de eliminar cuenta, cambiar contraseña y seleccionar imagen de perfil.

- logout():

Cierra la sesión del usuario eliminando el userId de las preferencias compartidas, deteniendo el servicio de notificación de eventos (EventNotificationService) y redirigiendo al usuario a la actividad de inicio de sesión (LoginActivity).

- stopEventNotificationService():

Detiene el servicio de notificación de eventos (EventNotificationService).

- deleteUser():

Muestra un cuadro de diálogo de confirmación para eliminar la cuenta del usuario. Elimina los datos del usuario de la base de datos Firebase Realtime Database y redirige al usuario a la actividad de inicio de sesión después de completar la operación.

- changePassword():

Cambia la contraseña del usuario en la base de datos Firebase Realtime Database. Valida que la nueva contraseña no esté vacía antes de realizar el cambio.

- openGallery():

Abre la galería del dispositivo para permitir al usuario seleccionar una imagen de perfil. Convierte la imagen seleccionada a formato base64 y la guarda en la base de datos Firebase Realtime Database.

- onActivityResult(int requestCode, int resultCode, Intent data):

Maneja el resultado de la selección de imagen desde la galería. Carga la imagen seleccionada en imageButton y la guarda en la base de datos Firebase Realtime Database en formato base64.

- bitmapToBase64(Bitmap bitmap):

Convierte un objeto Bitmap en una cadena codificada en base64 para su almacenamiento en la base de datos Firebase Realtime Database.

- base64ToBitmap(String base64String):

Convierte una cadena codificada en base64 de vuelta a un objeto Bitmap para mostrar la imagen de perfil almacenada en la base de datos Firebase Realtime Database.

- showProgressBar():

Muestra la barra de progreso (progressBar) para indicar que se están realizando operaciones en segundo plano.

- hideProgressBar():

Oculta la barra de progreso (progressBar) cuando las operaciones en segundo plano han finalizado.

## Otros

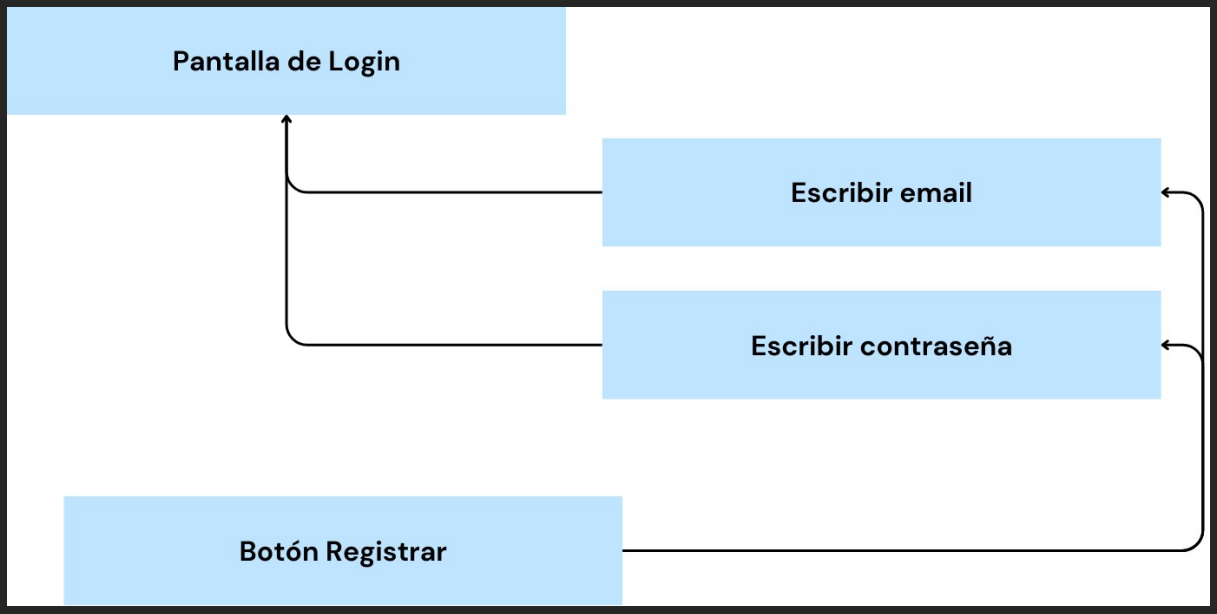
El directorio "drawable" es donde se alojan todos los recursos gráficos, como iconos e imágenes, utilizados en la aplicación. Por otro lado, en el directorio "layout" se encuentran los archivos XML que definen la estructura y el diseño de las distintas interfaces de usuario. En cuanto al directorio "menu", se encuentran dos archivos XML que definen los menús superiores e inferiores de la aplicación. Por último, en el directorio "values" se almacenan recursos comunes como temas, estilos, cadenas de texto y definiciones de color que se utilizan en toda la aplicacion

## Casos de uso

En este apartado se explican de manera sencilla y visual los casos a los que se dará soporte por parte de la aplicación

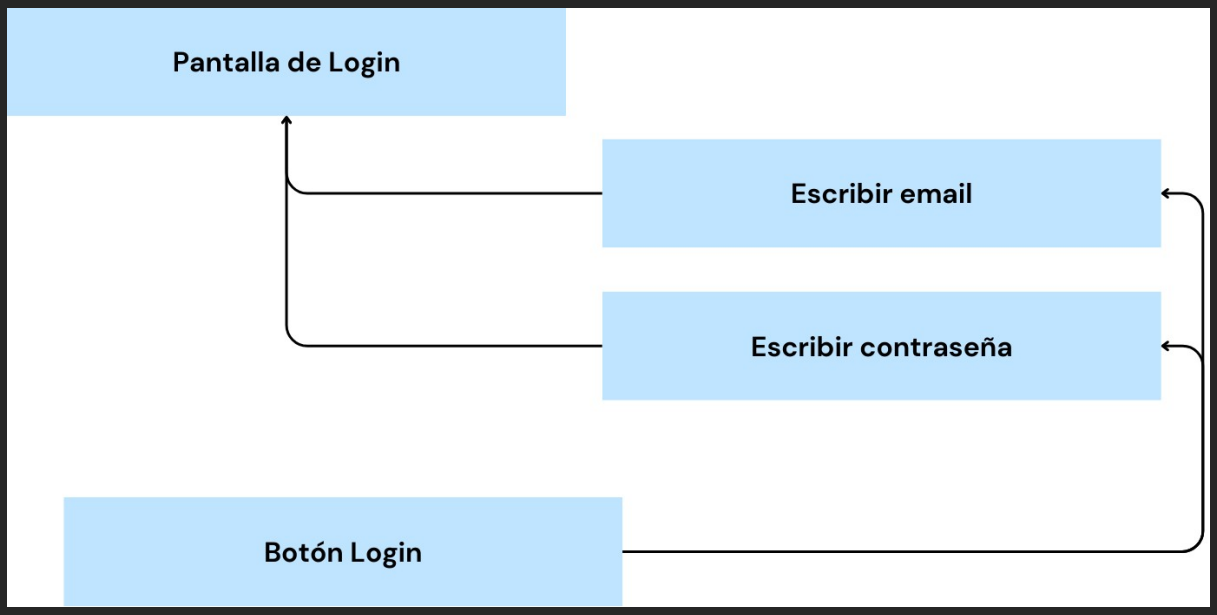
### Registrar Usuario

Al abrir la aplicación te saldrá una ventana en la que tendrás que rellenar tanto el campo email como contraseña siguiendo unos requisitos después de esto le pulsar el botón registrar.



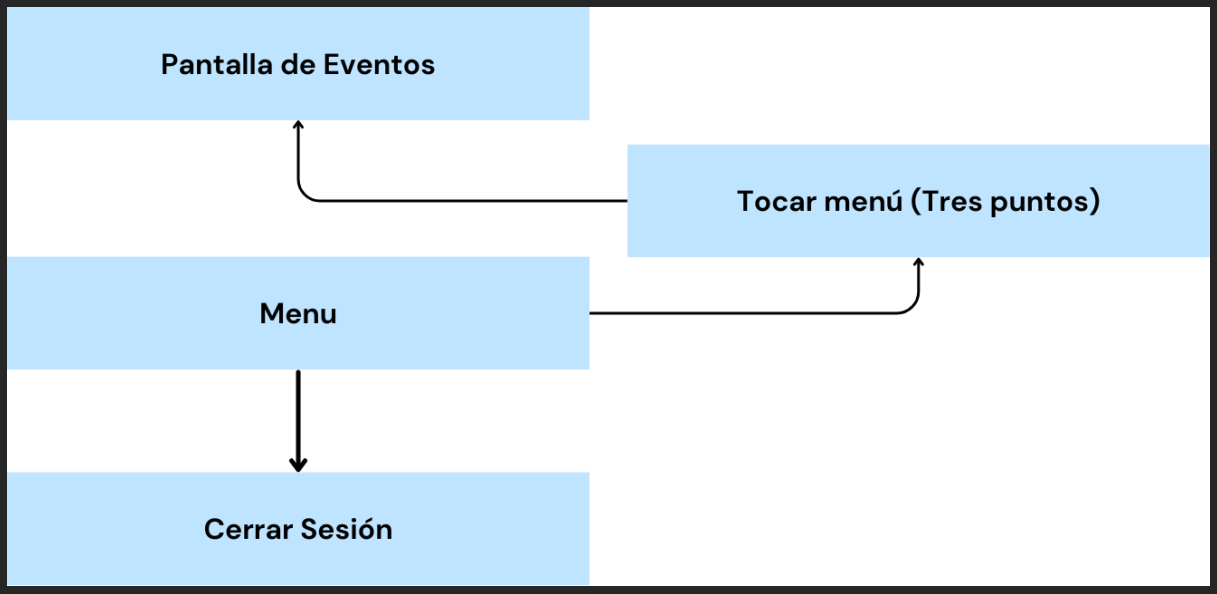
## Iniciar Sesión

Al abrir la aplicación te saldrá una ventana en la que tendrás que rellenar tanto el campo email como contraseña (Este usuario debe estar previamente registrado) después de esto le pulsar el botón login.



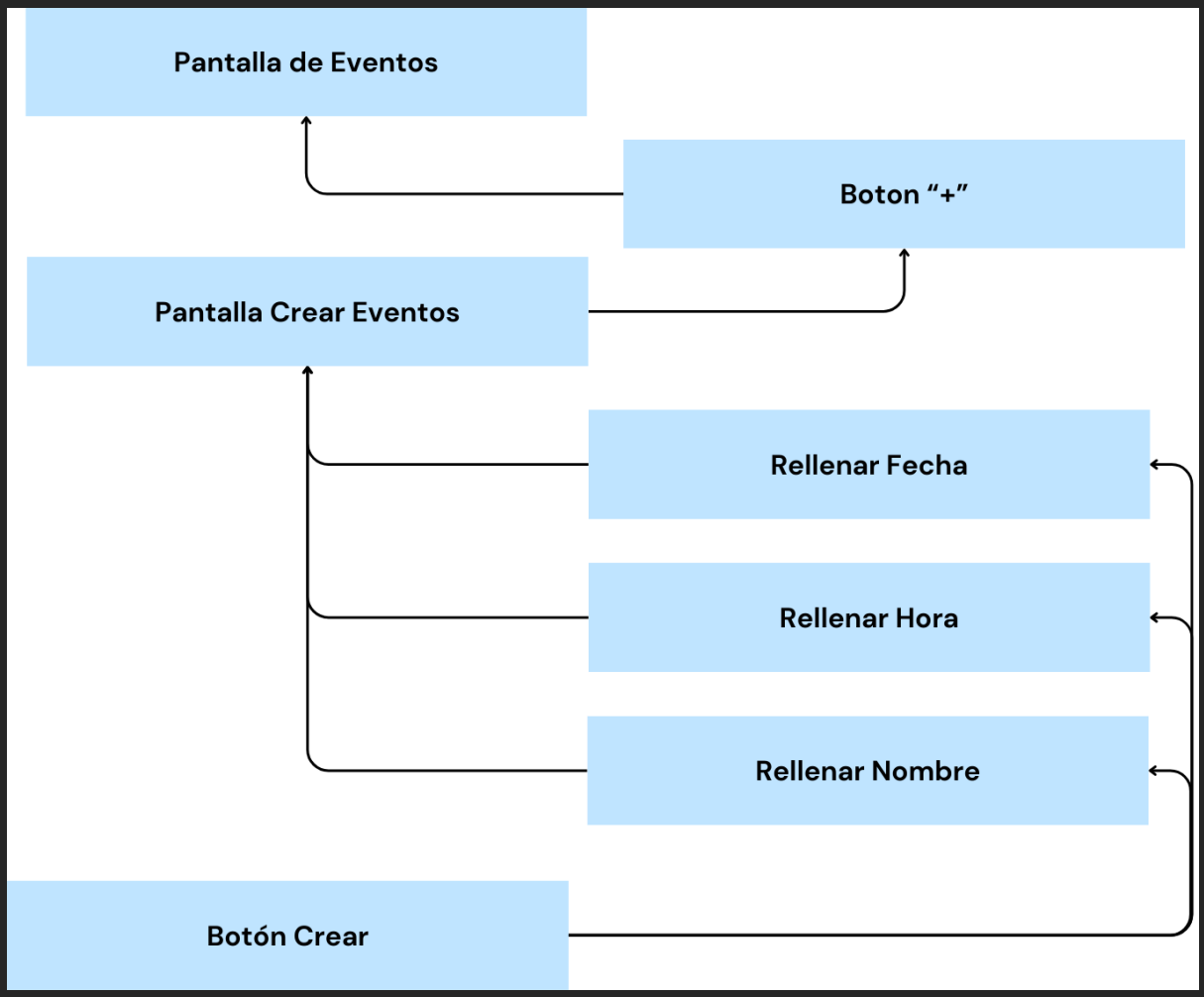

## Cerrar sesión

Inicias sesión deberás pulsar el Menú (tres botones) de arriba a la derecha se abrirá el menú hay toca cerrar sesión.



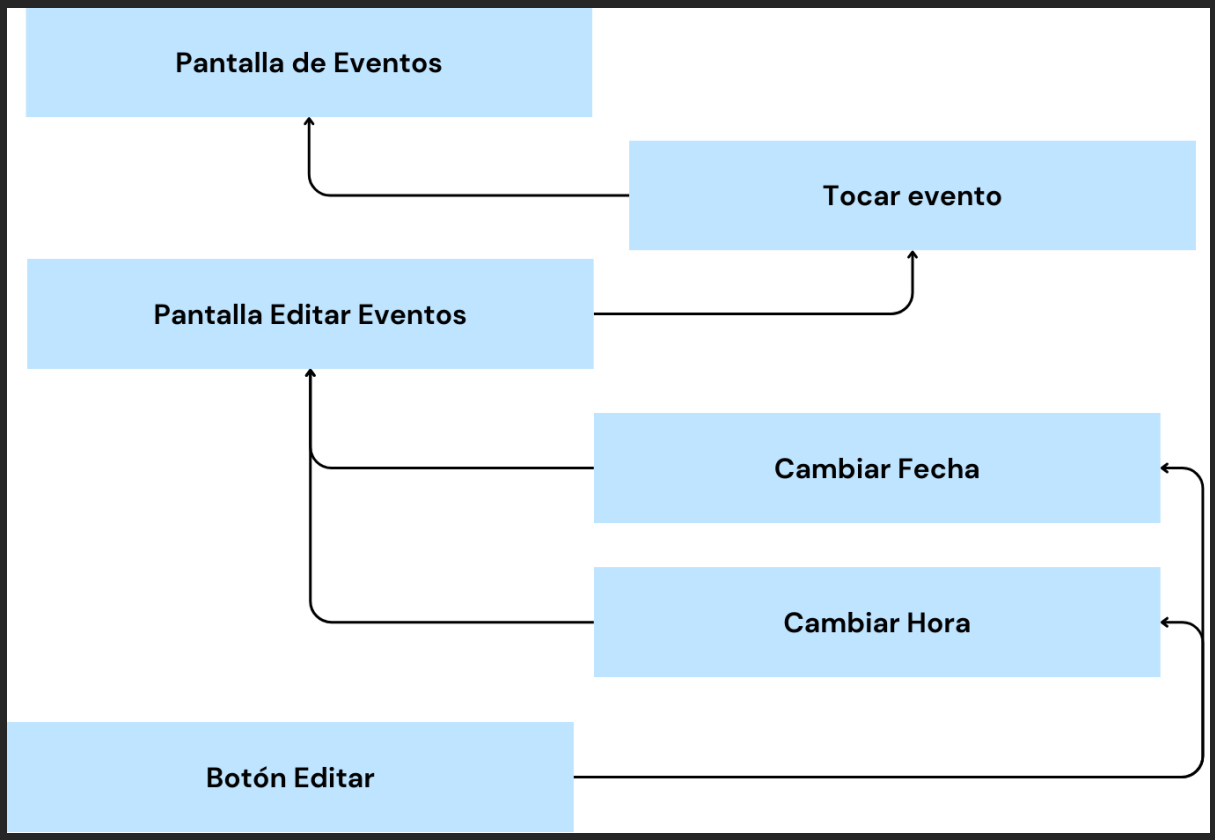
## Crear un Evento

Inicias sesión deberás pulsar el botón abajo a la derecha con el símbolo “+” al hacerlo se abrirá una ventana que tienes que rellenar con una fecha una hora y un nombre después de esto tocas el botón crear.



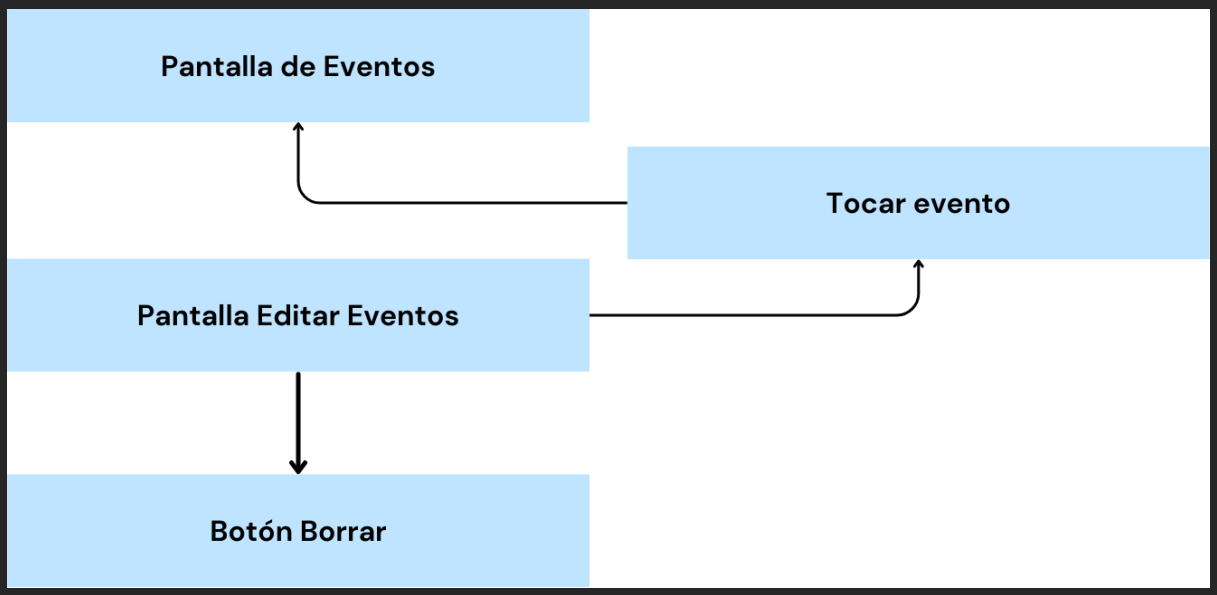
## Editar Eventos

Inicias sesión (Debes tener ya un evento creado) toca un evento de tu lista se te abrirá una ventana en esta cambia los datos de fecha y hora y toca el botón editar



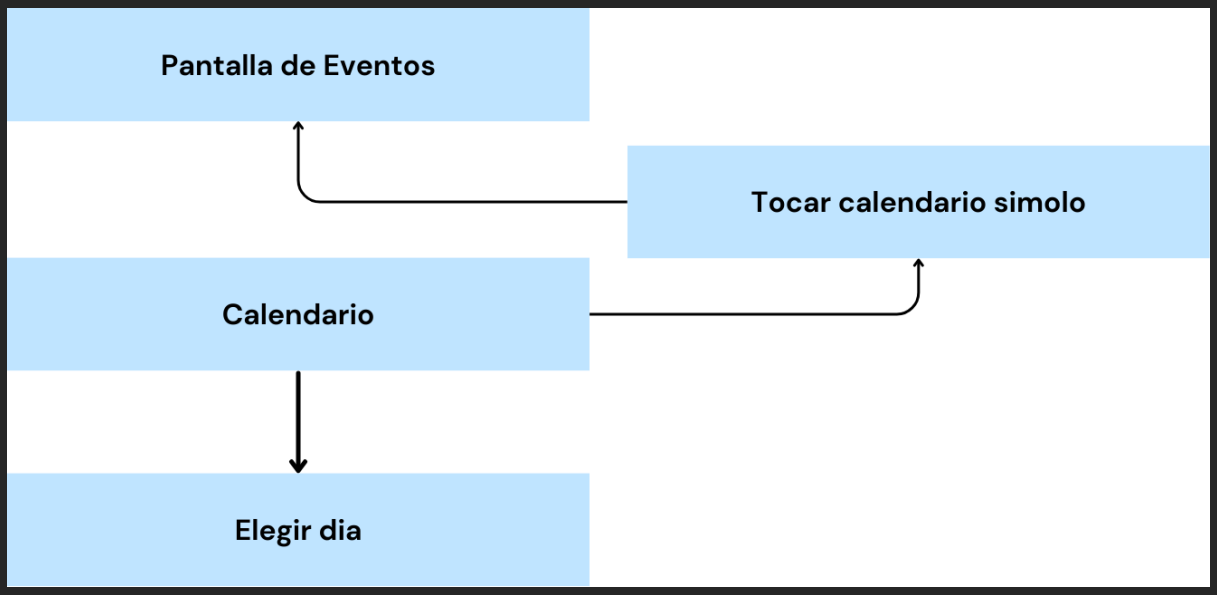
## Borrar Eventos

Inicias sesión (Debes tener ya un evento creado) toca un evento de tu lista se te abrirá una ventana en esta toca el botón borrar



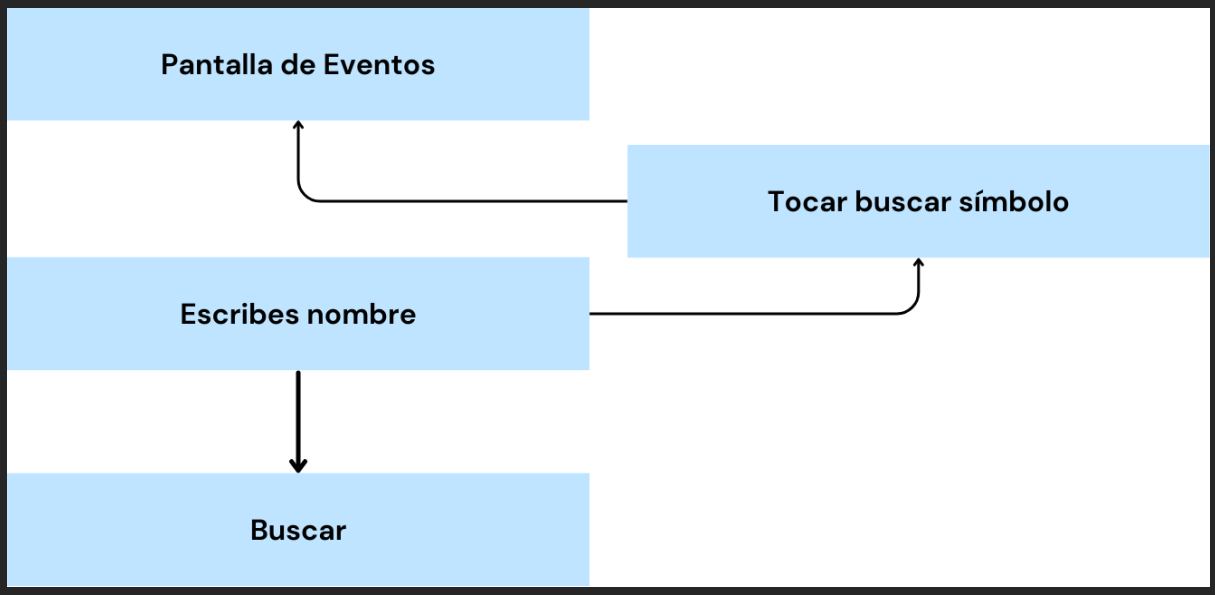

## Filtrar Eventos

Inicias sesión (Debes tener ya un evento creado) toca arriba en la barra de menú el símbolo de calendario te saldrá un calendario toca el día que quieres filtrar y darle ok.



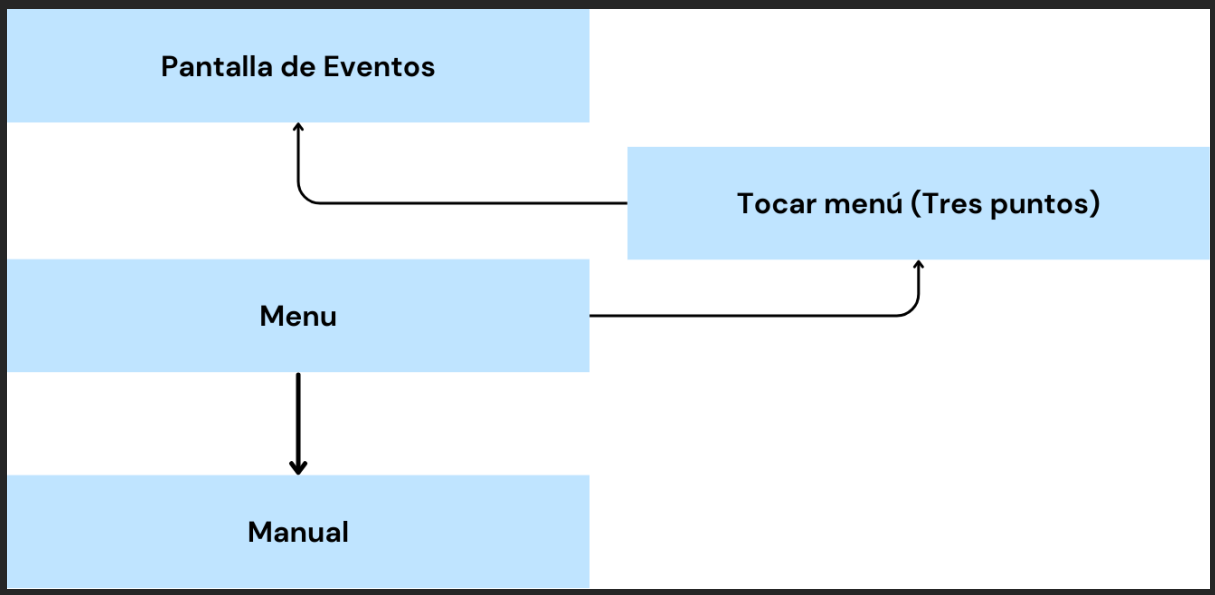
## Buscar Eventos

Inicias sesión (Debes tener ya un evento creado) toca arriba en la barra de menú el símbolo de buscar y escribe el nombre de evento que quieres buscar.



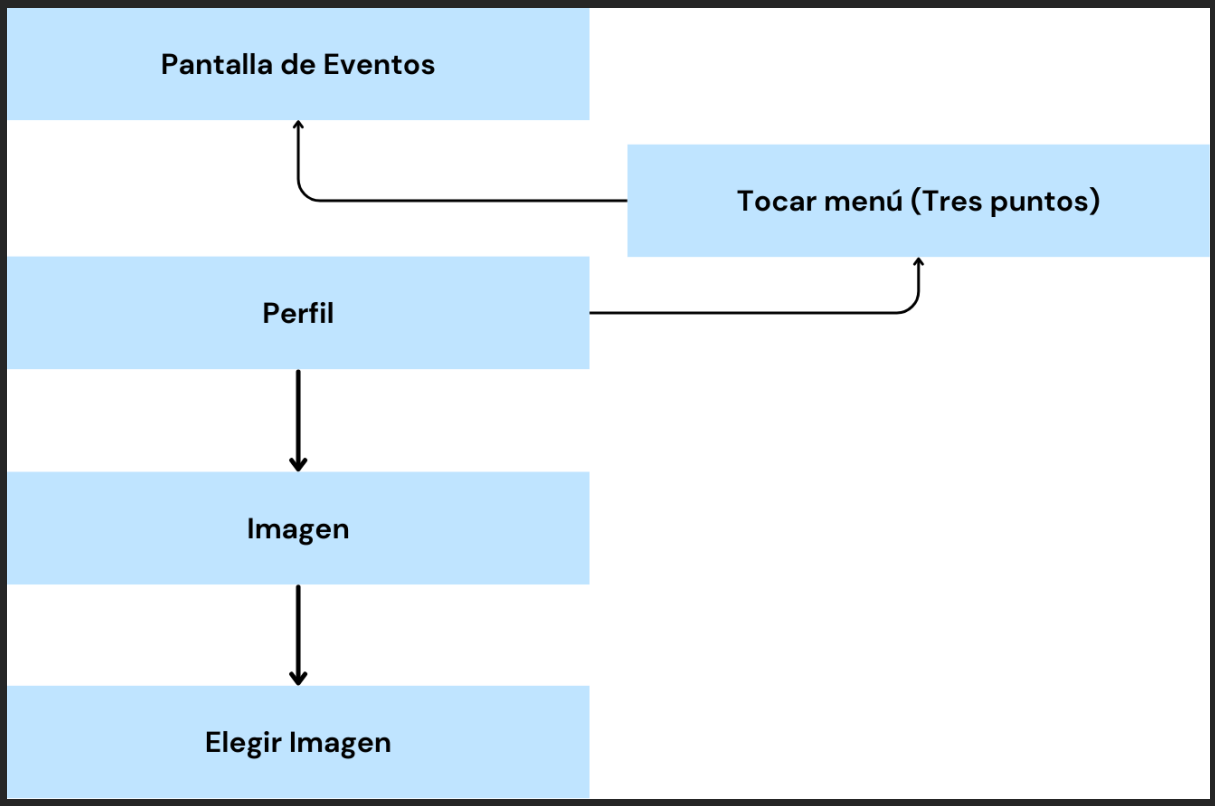

## Ver Manual

Inicias sesión deberás pulsar el Menú (tres botones) de arriba a la derecha se abrirá el menú hay toca Manual usuario.



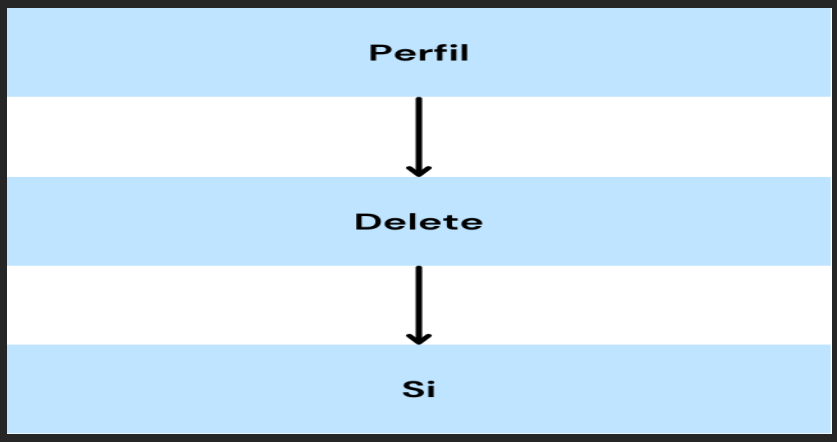
## Poner imagen

Inicias sesión deberás pulsar el Menú (tres botones) de arriba a la derecha se abrirá el menú hay toca perfil hay se te abre el menú de perfil toca el símbolo de perfil y elige la imagen.



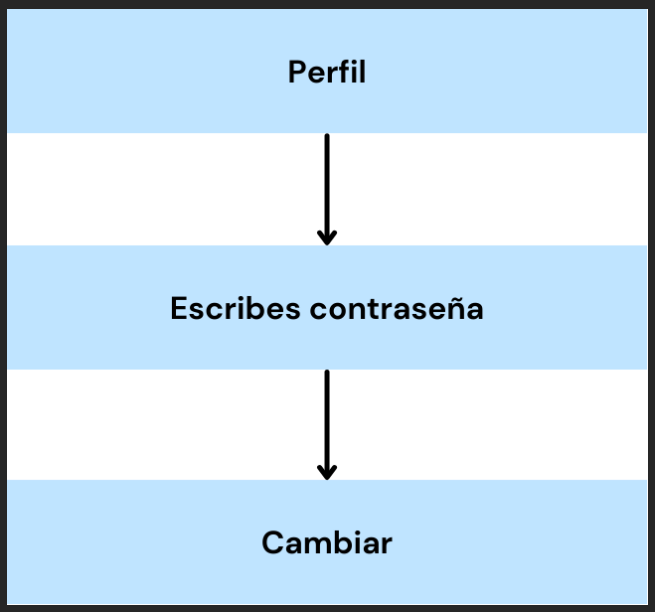

## Borrar cuenta

Dentro de perfil toca en delete te saldrá un mensaje de quieres borrar la cuenta le das a que si.



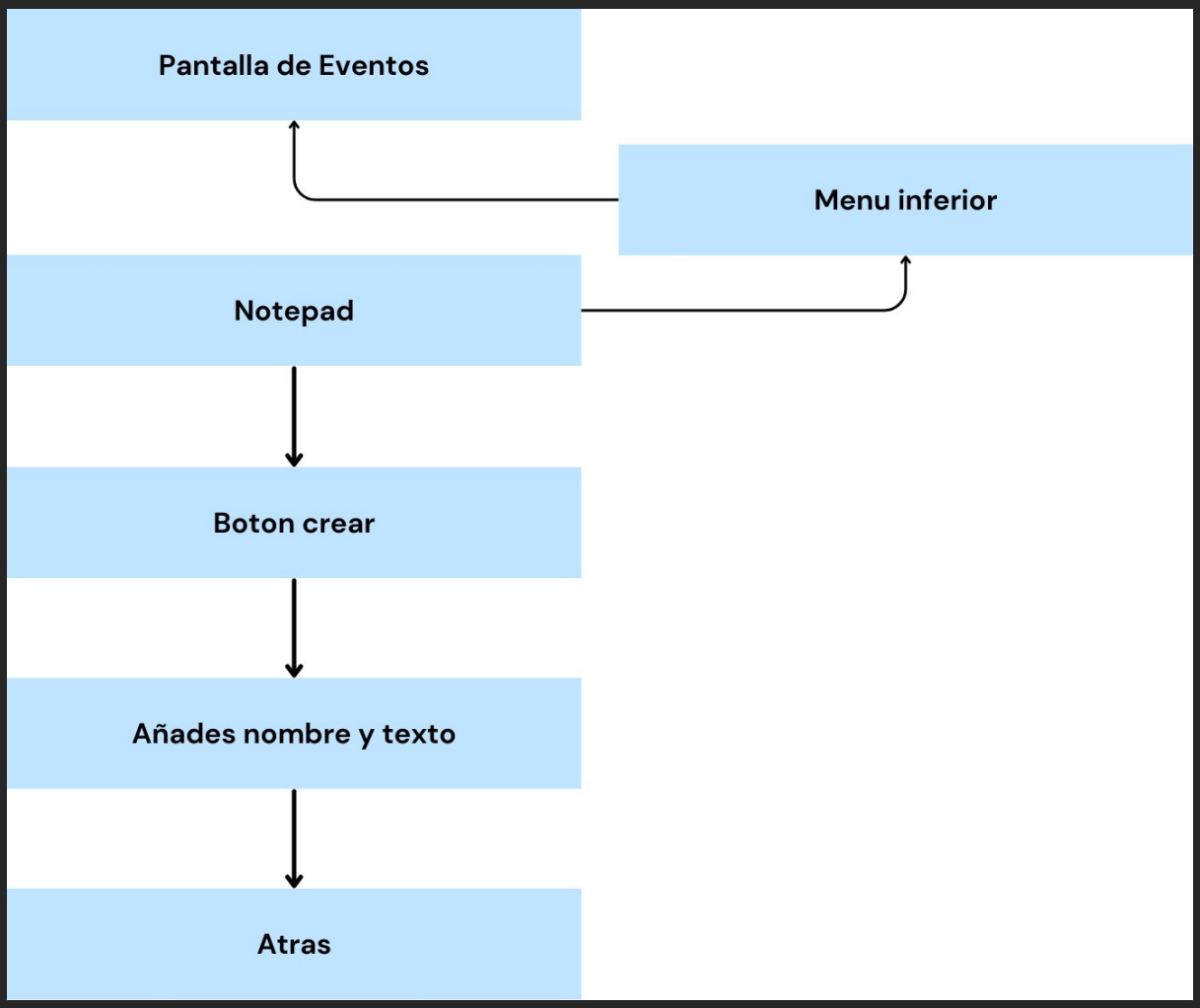
## Cambiar contraseña

En perfil en el campo contraseña escribes otra contraseña y le das a cambiar.



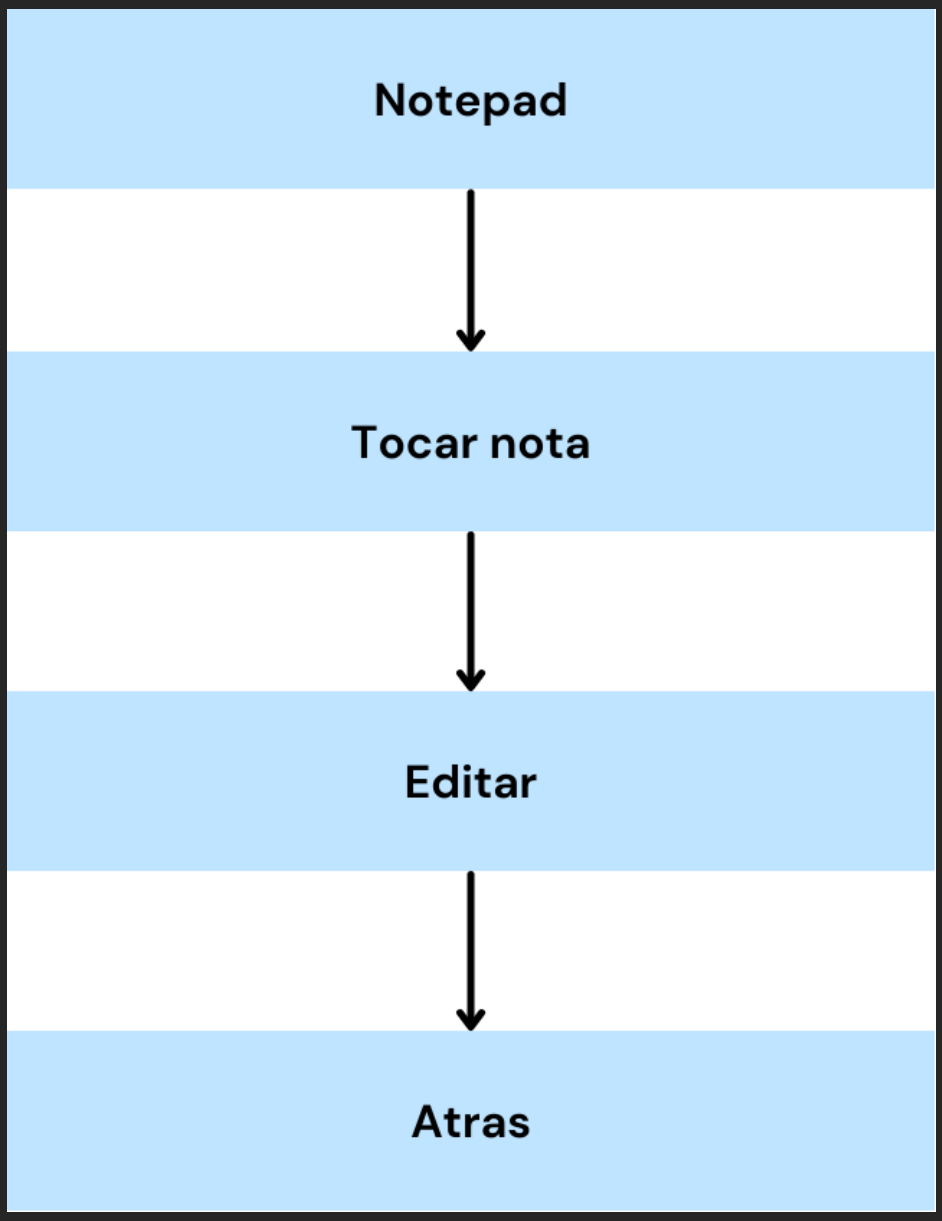
## Crear nota

En la vista eventos tienes abajo un menú para cambiar a la vista notepad hay le das al botón y crear tu nota poniendo y un título y texto y dándole para atrás ya se guarda.



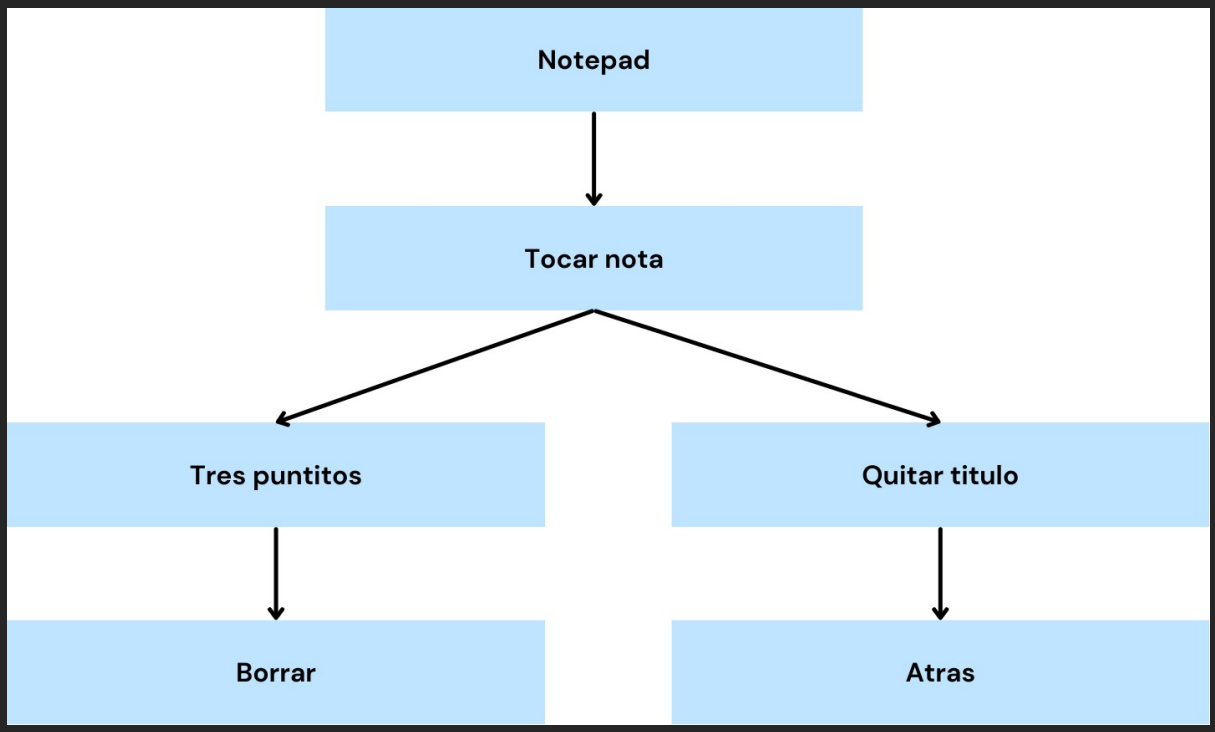
## Editar nota

Una vez hayas creado una nota esta te saldrá en la vista notepad toca la nota y ya desde dentro edita el titulo o el texto.



## Borrar nota

Toca una nota y puedes borrarla tanto en el menú superior en los tres puntos donde tendrás la opción borrar tanto si borras el titulo y le das atrás.



# VII. Conclusiones y Futuro

## Aplicaciones futuras

### 1. Mejora general en la interfaz:

- Refinar el diseño para una experiencia de usuario más intuitiva y atractiva.

- Optimizar la disposición de elementos para facilitar la navegación y el uso de la aplicación.

### 2. Optimización de la aplicación:

- Identificar y corregir posibles cuellos de botella en el rendimiento para mejorar la velocidad y la eficiencia.

- Reducir el consumo de recursos, como la memoria y la batería, para una experiencia más fluida y duradera.

### 3. Implementar diferentes formas de filtrado para los eventos:

- Agregar opciones de filtrado para que los usuarios puedan organizar sus eventos según criterios como fecha, categoría o prioridad.

- Permitir personalización en la visualización de los eventos para adaptarse a las preferencias individuales de los usuarios.

### 4. Implementar el login con Google:

- Facilitar el acceso y la sincronización de datos al permitir que los usuarios se autentiquen con sus cuentas de Google.

- Mejorar la seguridad y la privacidad al utilizar la autenticación de Google como opción de inicio de sesión.

## 5. Más opciones al crear un evento:

- Agregar la capacidad de configurar recordatorios para eventos, como la opción de recibir una notificación 5 minutos antes del inicio de un evento.

- Permitir la personalización de eventos con más detalles, como ubicación, invitados o repetición.

## 6. Funcionalidad de compartir eventos:

- Permitir a los usuarios compartir eventos con amigos, familiares o colegas a través de diferentes plataformas de mensajería o redes sociales.

- Facilitar la colaboración y la coordinación de actividades al permitir que múltiples usuarios vean y participen en eventos compartidos.

## 7. Modo de visualización personalizable:

- Agregar opciones de personalización en la vista de calendario, como la capacidad de cambiar entre vistas diaria, semanal o mensual.

- Permitir que los usuarios elijan qué información desean ver en la vista del calendario, como eventos, tareas pendientes o notas.

## 8. Sugerencias inteligentes de eventos:

- Utilizar algoritmos de aprendizaje automático para ofrecer sugerencias inteligentes de eventos en función de los hábitos y patrones de uso del usuario.

- Ayudar a los usuarios a descubrir y planificar eventos relevantes de manera más eficiente.

## 9. Integración con asistentes virtuales:

- Permitir la integración con asistentes virtuales como Google Assistant o Amazon Alexa para la creación y gestión de eventos mediante comandos de voz.

- Mejorar la accesibilidad y la usabilidad al ofrecer una forma alternativa de interactuar con la aplicación.

## 10. Funcionalidad de análisis de tiempo:

- Agregar herramientas de análisis de tiempo que proporcionen a los usuarios información sobre cómo están utilizando su tiempo.

- Ofrecer estadísticas y gráficos que muestren la distribución del tiempo entre diferentes actividades y categorías.

## 11. Modo offline mejorado:

- Mejorar la funcionalidad offline para permitir a los usuarios acceder y editar eventos incluso cuando no tienen conexión a internet.

## Conclusión

La aplicación TimeFlow, diseñada para gestionar el tiempo en eventos, bloc de notas y listas de hábitos para diferentes días, ha sido sometida a rigurosas pruebas de estrés para evaluar su robustez y estabilidad.

TimeFlow ha demostrado ser altamente eficiente en el manejo de múltiples tareas y funcionalidades sin experimentar fallos críticos ni comportamientos anómalos. La aplicación ha superado con éxito todas las pruebas, asegurando su capacidad para proporcionar una experiencia de usuario fluida y confiable incluso bajo condiciones de uso extremo. Estos resultados destacan la solidez del diseño y la implementación de TimeFlow, reafirmando su capacidad para ayudar a los usuarios a gestionar su tiempo de manera efectiva y sin inconvenientes.

## Bibliografía


- GitHub
- Material Design Guidelines
- Stack Overflow
- Android Developers
- YouTube - Android Developers
- Google Developers - Android