



CONSEJERÍA DE EDUCACIÓN  
**Comunidad de Madrid**

## **IES ENRIQUE TIERNO GALVAN** Parla

### **CFGS DESARROLLO DE APLICACIONES MULTIPLATAFORMA**

**Curso 2024/2025**

---

#### **Proyecto DAM**

**TITULO: Aplicación en flutter, EVENTIFY**

**Alumno: David Montero Cuesta**

**Tutor: Luis Alfonso González Marcos**

Abril de 2025

## Tabla de contenido

<b>DESCRIPCIÓN GENERAL DEL PROYECTO .....</b>	3
<b>Comparativa de Eventify con otras aplicaciones en el mercado .....</b>	4
Competencias en el mercado.....	4
Funcionalidades del Calendario en iOS (Apple Calendar).....	4
Funcionalidades del Calendario en Android (Google Calendar) .....	5
<b>Comparación: Apple Calendar vs. Google Calendar vs. Eventify .....</b>	7
<b>Análisis Comparativo.....</b>	9
<b>ANÁLISIS Y DISEÑO DE LA BASE DE DATOS .....</b>	10
<b>Requisitos .....</b>	10
<b>FLUTTER VS KOTLIN MULTIPLATFORM.....</b>	12
<b>FLUTTER .....</b>	12
<b>KOTLIN MULTIPLATFORM.....</b>	13
<b>CONCLUSIÓN .....</b>	14
<b>Elección de base de datos para Eventify .....</b>	16
<b>Objetivo .....</b>	16
<b>Comparativa de bases de datos en la nube gratuitas .....</b>	16
<b>Decisión: Firebase Firestore .....</b>	17
<b>DISEÑO DE LA BASE DE DATOS.....</b>	18
<b>ASPECTOS FUNCIONALES Y DISEÑO DE LA APLICACIÓN .....</b>	26
<b>PAQUETIZACIÓN .....</b>	27
<b>NAVEGACIÓN .....</b>	29
<b>IMPLEMENTACIÓN DE LA APLICACIÓN .....</b>	30
<b>INTERFAZ DEL USUARIO .....</b>	30
<b>DISEÑO DE LAS INTERFACES .....</b>	30
<b>ASPECTOS DESTACABLES DEL DESARROLLO .....</b>	32
<b>LÍNEAS DE TRABAJO FUTURO .....</b>	32
<b>CONCLUSIONES.....</b>	33

<b>BIBLIOGRAFÍA .....</b>	34
<b>ANEXOS .....</b>	35
<b><i>ENLACE AL GITHUB DEL PROYECTO .....</i></b>	35

## DESCRIPCIÓN GENERAL DEL PROYECTO

Eventify es una agenda móvil multiplataforma implementada con el framework Flutter y el lenguaje Dart. Eventify tiene como objetivo proporcionar a los usuarios una herramienta intuitiva y potente para la gestión de su tiempo y eventos tanto en dispositivos Android como iOS, ofreciendo una experiencia unificada, eficiente y asistida por un agente de inteligencia artificial integrado.

La aplicación se caracteriza por su interactividad y facilidad de uso, presentando una interfaz intuitiva que minimiza la curva de aprendizaje. Un aspecto central es la personalización visual, permitiendo a los usuarios seleccionar colores que dinámicamente transformarán la apariencia de la aplicación según sus preferencias. La eficiencia y la rapidez en la respuesta son fundamentales para garantizar una experiencia de usuario fluida y sin interrupciones.

Eventify incorpora funcionalidades avanzadas de gestión de eventos, permitiendo a los usuarios añadir, modificar y eliminar eventos en cualquier fecha del calendario. Destaca su sistema de búsqueda avanzada, que facilita la localización de eventos por palabras clave, nivel de prioridad y tipo. Además, la aplicación ofrece filtros específicos adaptados a diferentes tipos de eventos, como la localidad para conferencias y la asignatura para exámenes, optimizando la búsqueda de información relevante. Un elemento diferenciador clave es la integración de un agente de inteligencia artificial, que estará disponible en todo momento para asistir al usuario con diversas tareas, como la sugerencia de recordatorios optimizados o la ayuda en la búsqueda de eventos.

Para asegurar que los usuarios no olviden eventos importantes, Eventify incluye un sistema de notificaciones configurables que enviarán recordatorios previos a la fecha programada. La persistencia de los datos está garantizada mediante el almacenamiento de los eventos en una base de datos en la nube, lo que además sienta las bases para futuras funcionalidades de sincronización y la posible interacción inteligente del agente de IA con los datos del usuario.

En resumen, Eventify se presenta como una solución de calendario móvil multiplataforma que redefine la organización personal al integrar un agente de inteligencia artificial para ofrecer asistencia proactiva y mejorar la experiencia del usuario. Además de las potentes funciones de búsqueda y gestión de eventos, la aplicación se distingue por su diseño interactivo, personalizable y eficiente, respaldado por un enfoque en la seguridad, el rendimiento y la escalabilidad.

## **Comparativa de Eventify con otras aplicaciones en el mercado**

### Competencias en el mercado

Los calendarios de **iOS (Apple Calendar)** y **Android (Google Calendar)** ofrecen una variedad de funcionalidades diseñadas para la gestión de eventos, recordatorios y planificación.

### **Funcionalidades del Calendario en iOS (Apple Calendar)**

Gestión de eventos y recordatorios:

- Creación, edición y eliminación de eventos.
- Posibilidad de establecer eventos repetitivos (diarios, semanales, mensuales, anuales o personalizados).
- Agregar una ubicación al evento con integración de Apple Maps.
- Añadir notas y descripciones dentro del evento.
- Adjuntar archivos y enlaces a eventos.

Notificaciones y alertas:

- Recordatorios antes del evento con opciones personalizadas (minutos, horas o días antes).
- Alertas basadas en ubicación (por ejemplo, recibir un aviso al llegar a un lugar).
- Integración con Siri para establecer recordatorios con comandos de voz.

Sincronización y compatibilidad:

- Sincronización con iCloud, lo que permite acceder a eventos desde todos los dispositivos Apple.
- Compatibilidad con calendarios de terceros: Google Calendar, Microsoft Outlook, Exchange, Yahoo Calendar, CalDAV.
- Integración con aplicaciones de terceros para importar eventos.

Vista y organización:

- Diferentes modos de vista: día, semana, mes y año.
- Posibilidad de crear varios calendarios y asignarles colores diferentes para diferenciarlos.
- Opción de compartir calendarios con otros usuarios de iCloud.

Otras características:

- Modo oscuro compatible con iOS.
- Posibilidad de agregar eventos a través de Siri.
- Integración con Apple Watch para ver eventos en la muñeca.
- Posibilidad de responder a invitaciones de eventos desde la app.

### **Funcionalidades del Calendario en Android (Google Calendar)**

Gestión de eventos y tareas:

- Creación y edición de eventos con opciones de personalización.
- Eventos recurrentes con patrones flexibles de repetición.
- Posibilidad de adjuntar archivos desde Google Drive.
- Agregar descripciones y enlaces dentro del evento.
- Creación de tareas (integrado con Google Tasks).

Notificaciones y alertas:

- Recordatorios programables con distintas opciones de tiempo.
- Notificaciones basadas en ubicación.
- Posibilidad de recibir notificaciones por correo electrónico.

Sincronización y compatibilidad:

- Sincronización con Google Drive, Gmail y Google Tasks.
- Compatibilidad con calendarios de Microsoft Outlook, Exchange, iCloud, CalDAV.
- Integración con asistentes de voz como Google Assistant para programar eventos por voz.

Vista y organización:

- Diferentes modos de vista: día, semana, mes y agenda.
- Uso de colores personalizables para organizar eventos por categoría.
- Posibilidad de superponer varios calendarios.
- Sincronización con eventos de Gmail (reservas de vuelos, reuniones, envíos, etc.).

Otras características:

- Modos de sugerencias automáticas, que completan eventos basados en patrones previos.
- Posibilidad de crear reuniones con enlaces directos de Google Meet.
- Creación de encuestas de disponibilidad con Google Workspace.
- Accesibilidad desde la web en cualquier dispositivo.
- Integración con Wear OS para ver eventos desde un smartwatch.

## Comparación: Apple Calendar vs. Google Calendar vs. Eventify

FUNCIONALIDAD	APPLE CALENDAR (IOS)	GOOGLE CALENDAR (ANDROID)	EVENTIFY (APLICACIÓN PROPIA)
INTERACTIVIDAD Y FACILIDAD DE USO	<input checked="" type="checkbox"/> Interfaz limpia y sencilla	<input checked="" type="checkbox"/> Intuitivo y bien integrado con Google	<input checked="" type="checkbox"/> Totalmente personalizable
PERSONALIZACIÓN DE COLORES	<input checked="" type="checkbox"/> Solo modo claro/oscuro	<input checked="" type="checkbox"/> Temas limitados (modo claro/oscuro)	<input checked="" type="checkbox"/> Opción para elegir cualquier color de la interfaz
GESTIÓN DE EVENTOS	<input checked="" type="checkbox"/> Añadir, modificar y eliminar	<input checked="" type="checkbox"/> Añadir, modificar y eliminar	<input checked="" type="checkbox"/> Con funciones avanzadas y filtros personalizados
BÚSQUEDA AVANZADA	<input checked="" type="checkbox"/> Búsqueda básica	<input checked="" type="checkbox"/> Búsqueda limitada a títulos y descripciones	<input checked="" type="checkbox"/> Filtrado por palabras clave, prioridad, tipo, ubicación, etc.
PERSISTENCIA DE DATOS	<input checked="" type="checkbox"/> iCloud (requiere internet)	<input checked="" type="checkbox"/> Google Drive (requiere internet)	<input checked="" type="checkbox"/> Almacenamiento en la nube, sin necesidad de base de datos local
NOTIFICACIONES	<input checked="" type="checkbox"/> Alertas estándar	<input checked="" type="checkbox"/> Alertas configurables	<input checked="" type="checkbox"/> Configuración personalizada para eventos críticos
MODO OFFLINE	<input checked="" type="checkbox"/> No disponible sin internet	<input checked="" type="checkbox"/> No disponible sin internet	<input checked="" type="checkbox"/> Algunas funciones sin conexión

<b>COMPATIBILIDAD</b>	<input checked="" type="checkbox"/> Solo en dispositivos Apple	<input checked="" type="checkbox"/> Solo en dispositivos con Google	<input checked="" type="checkbox"/> Soporte para Android, iOS, escritorio
<b>ESCALABILIDAD</b>	<input checked="" type="checkbox"/> Limitado a funciones predeterminadas	<input type="triangle-down"/> Google puede actualizar, pero con restricciones	<input checked="" type="checkbox"/> Capacidad de agregar nuevas funciones en el futuro
<b>SEGURIDAD</b>	<input checked="" type="checkbox"/> Datos encriptados en iCloud	<input checked="" type="checkbox"/> Protección con Google Account	<input checked="" type="checkbox"/> Seguridad garantizada mediante autenticación y almacenamiento en la nube
<b>RENDIMIENTO</b>	<input checked="" type="checkbox"/> Optimizado para Apple	<input checked="" type="checkbox"/> Integrado con Google, pero puede tener lag en algunos dispositivos	<input checked="" type="checkbox"/> Fluido y optimizado para dispositivos Android, iOS y escritorio (gracias a Flutter)
<b>INTEGRACIÓN CON OTROS SERVICIOS</b>	<input checked="" type="checkbox"/> Apple Maps, Siri, Apple Watch	<input checked="" type="checkbox"/> Google Maps, Google Meet, Gmail	<input checked="" type="checkbox"/> No requiere integración con servicios externos, pero puede agregarse en el futuro

## Análisis Comparativo

Ventajas de la aplicación frente a Apple Calendar y Google Calendar

**Mayor personalización:** Apple Calendar y Google Calendar no permiten cambios en los colores de la interfaz, mientras que la aplicación de agenda ofrece esta opción de personalización.

**Búsqueda avanzada:** La aplicación permite filtrar eventos por múltiples criterios como palabras clave, prioridad, tipo, ubicación, etc., a diferencia de Apple y Google Calendar, que tienen opciones más limitadas.

**Escalabilidad:** La aplicación tiene la capacidad de agregar nuevas funciones en el futuro, permitiendo una mayor personalización y expansión de sus características.

## Conclusión

La aplicación de agenda, desarrollada con Flutter, ofrece una solución flexible y altamente personalizable, con ventajas como la personalización de colores y la búsqueda avanzada. Sin embargo, debido a la decisión de no utilizar una base de datos local, la persistencia de datos estará a cargo del almacenamiento en la nube, lo que puede ser una limitación si el usuario no dispone de conexión a internet. A pesar de ello, la aplicación sigue siendo una opción viable y atractiva para aquellos que busquen una experiencia de usuario adaptada a sus necesidades, con la posibilidad de agregar nuevas funcionalidades en el futuro.

# ANÁLISIS Y DISEÑO DE LA BASE DE DATOS

## Requisitos

### Necesidades del Usuario

- La aplicación debe ser **interactiva y fácil de usar**, con una interfaz intuitiva y sin curva de aprendizaje elevada.
- El usuario podrá **personalizar** el aspecto de la app, eligiendo colores que cambiarán dinámicamente la apariencia.
- El sistema debe ser **eficiente y rápido**, sin retrasos al interactuar.

### Requisitos Funcionales

#### 1. Gestión de eventos

- El usuario podrá **añadir, modificar y eliminar** eventos en cualquier fecha del calendario.

#### 2. Búsqueda avanzada

- Se podrán buscar eventos por:
  - Palabras clave
  - Nivel de prioridad
  - Tipo de evento
  - Otros filtros específicos, como:
    - Localidad (para conferencias)
    - Asignatura (para exámenes)

#### 3. Notificaciones

- Los eventos importantes podrán configurarse para **enviar recordatorios** antes de su fecha.

#### 4. Persistencia de datos

- Los eventos se almacenarán de forma **persistente** en una base de datos en la nube.

## Requisitos No Funcionales

### 1. Rendimiento

- La aplicación debe **responder rápidamente** y mantener una experiencia fluida.

### 2. Seguridad

- Los datos introducidos por el usuario deberán almacenarse de forma **segura y protegida**.

### 3. Compatibilidad

- La app será compatible con **múltiples dispositivos Android**.

### 4. Escalabilidad

- El sistema debe permitir añadir nuevas funcionalidades en el futuro, como:
  - Sincronización con cuenta
  - Exportación a otros formatos
  - Widgets de escritorio

### 5. Personalización

- El usuario podrá **cambiar el tema de la app** mediante una paleta amplia de colores.

## FLUTTER VS KOTLIN MULTPLATFORM

### FLUTTER

Flutter, desarrollado por Google, es un framework que permite crear aplicaciones multiplataforma con un solo código base, utilizando el lenguaje *Dart*.

Pros:

- Código 100% compartido. Puedes escribir una única base de código para Android, iOS, web y escritorio.
- UI consistente y personalizable. Usa su propio motor de renderizado para generar la interfaz, lo que garantiza que la apariencia sea la misma en todas las plataformas.
- Hot Reload. Permite ver cambios en tiempo real sin recomilar, lo que acelera el desarrollo.
- Rápido y eficiente. Su motor gráfico Skia le permite ofrecer un rendimiento fluido en animaciones e interfaces complejas.
- Gran comunidad y soporte. Hay muchos paquetes, documentación y soporte en foros como Stack Overflow.
- Web y escritorio mejor integrados. Puede compilar para navegadores y aplicaciones de escritorio más fácilmente que KMP.

Contras:

- Mayor tamaño de la app. Las aplicaciones de Flutter suelen ser más pesadas que las nativas debido al motor de renderizado incorporado.
- No usa componentes nativos. Aunque Flutter imita el aspecto de los widgets nativos, no los usa realmente, lo que puede causar diferencias en comportamiento.
- Dart es menos popular. Comparado con Kotlin, el ecosistema de Dart es más pequeño y menos adoptado fuera de Flutter.
- Integración con plataformas nativas. Aunque es posible acceder a APIs nativas con canales de plataforma, no es tan directo como en KMP.
- Curva de aprendizaje. Requiere aprender Dart y la estructura de Flutter, lo que puede ser un obstáculo si vienes de Kotlin o Java.

## KOTLIN MULTIPLATFORM

Kotlin Multiplatform permite compartir lógica entre diferentes plataformas (Android, iOS, web, escritorio), pero mantiene la posibilidad de escribir código específico para cada una cuando sea necesario.

Pros:

- Reutilización de código. Puedes compartir gran parte de la lógica de negocio (por ejemplo, controladores, modelos de datos, etc.) entre Android, iOS y otras plataformas.
- Código nativo. Utiliza Kotlin para Android y se puede integrar fácilmente con código Swift en iOS, lo que permite una experiencia de rendimiento casi nativo.
- Interoperabilidad total. En Android funciona como código Kotlin nativo y en iOS puede integrarse con Swift y Objective-C sin problemas.
- Menor curva de aprendizaje. Si ya conoces Kotlin, la transición es bastante sencilla.
- Uso de UI nativa. No impone un framework de UI, sino que permite utilizar las herramientas nativas de cada plataforma (Jetpack Compose en Android, SwiftUI o UIKit en iOS).
- Escalabilidad. Ideal para proyectos grandes que requieren compartir lógica sin sacrificar la personalización de la UI en cada plataforma.

Contras:

- No incluye UI multiplataforma. Debes escribir las interfaces de usuario por separado en Android e iOS, lo que aumenta el trabajo en comparación con Flutter.
- Menos maduro que Flutter. Aunque Kotlin es un lenguaje sólido, KMP aún está en fase experimental y puede tener cambios inesperados.
- Menor comunidad y recursos. Hay menos documentación y paquetes listos para usar en comparación con Flutter.

Curva de aprendizaje en iOS. Aunque Kotlin es fácil para desarrolladores de Android, integrarlo con Swift en iOS requiere aprendizaje extra.

Soporte web limitado. No está tan desarrollado como Flutter para aplicaciones web.

## CONCLUSIÓN

Dado que el objetivo es desarrollar una aplicación de agenda disponible en Android, iOS y escritorio, Flutter representa la mejor opción por las siguientes razones:

### 1. Un solo código para todas las plataformas

Flutter permite escribir una única base de código, evitando la necesidad de desarrollar interfaces separadas para Android, iOS y escritorio. En contraste, Kotlin Multiplatform requiere escribir la lógica compartida, pero las interfaces deben desarrollarse por separado en cada plataforma (Jetpack Compose para Android, SwiftUI para iOS, etc.).

### 2. Interfaz personalizada y flexible

Flutter ofrece un control total sobre la apariencia de la aplicación, permitiendo diseñar una UI atractiva y homogénea en todas las plataformas sin depender de los componentes nativos. Esto resulta ideal para una aplicación de agenda, donde la experiencia de usuario y la usabilidad son aspectos fundamentales.

### 3. Desarrollo rápido con Hot Reload

El Hot Reload de Flutter permite visualizar los cambios en tiempo real sin necesidad de recompilar la aplicación por completo. Esto agiliza el proceso de desarrollo en comparación con Kotlin Multiplatform, donde las pruebas pueden ser más lentas debido a la necesidad de compilar código nativo para cada plataforma.

### 4. Soporte para escritorio y web

Flutter ofrece un soporte más avanzado para aplicaciones de escritorio y web, facilitando la expansión de la aplicación de agenda a otras plataformas en el futuro. Por otro lado, Kotlin Multiplatform todavía presenta limitaciones en este aspecto, especialmente en la parte de la UI.

### 5. Amplia comunidad y paquetes listos para usar

Flutter cuenta con una comunidad extensa y un ecosistema de paquetes bien desarrollado, lo que facilita la integración de funcionalidades clave como bases de datos, notificaciones, sincronización en la nube y autenticación. Esto reduce el tiempo de desarrollo en comparación con Kotlin Multiplatform, donde algunos recursos aún son limitados.

Para desarrollar una aplicación multiplataforma eficiente, con una interfaz moderna y un desarrollo ágil, Flutter se presenta como la mejor opción. Gracias a su capacidad para compartir código entre todas las plataformas y a su amplio ecosistema de herramientas, permite optimizar el tiempo de desarrollo y garantizar una experiencia de usuario consistente.

## Elección de base de datos para Eventify

### Objetivo

Para el desarrollo de Eventify, se necesita una base de datos que:

- Permita guardar eventos personalizados (fecha, título, descripción, etc.).
- Sea accesible desde varios dispositivos.
- Soporte usuarios distintos y gestión de sus eventos.
- Sincronice cambios en tiempo real o al menos de forma eficiente.
- Tenga una opción gratuita suficiente para el uso inicial de la app.

Por ello, se ha decidido utilizar una base de datos en la nube, ya que así los datos se almacenan online y no se pierden, incluso si el usuario cambia de dispositivo o reinstala la app.

### Comparativa de bases de datos en la nube gratuitas

BASE DE DATOS	TIPO	PLAN GRATUITO	PROS	CONTRAS
FIREBASE FIRESTORE	NoSQL	50K lecturas/día 20K escrituras/día 1 GB de almacenamiento	- Tiempo real - Filtros y orden por campos - Excelente soporte en Flutter	- Puede escalar rápido si se sobrepasan los límites  - Estructura por documentos
FIREBASE REALTIME DB	NoSQL (JSON)	1 GB almacenamiento 100K conexiones simultáneas	- Sincronización ultra rápida  - Muy fácil de usar	- Menos flexible para estructuras complejas  - Consultas limitadas
SUPABASE	SQL (PostgreSQL)	500 MB base de datos 2 GB almacenamiento 10K usuarios autenticados	- Consultas SQL reales  - Código abierto  - Autenticación integrada	- En desarrollo  - Menor comunidad que Firebase

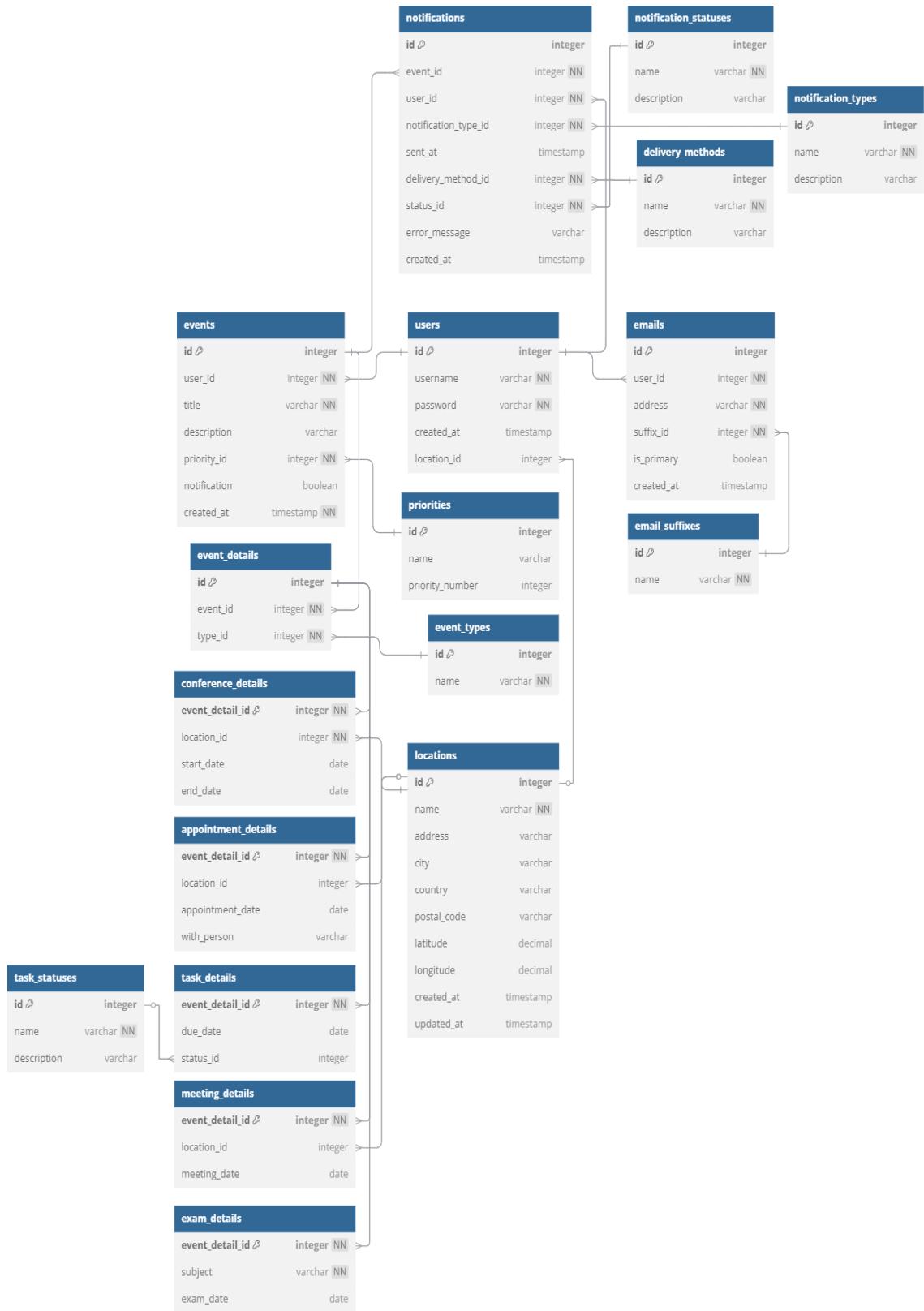
### **Decisión: Firebase Firestore**

Se ha decidido usar Firebase Firestore como base de datos para Eventify por las siguientes razones:

- Su plan gratuito es suficiente para empezar, con 50.000 lecturas y 20.000 escrituras al día.
- Permite almacenar los eventos por usuario y ordenarlos por fecha fácilmente.
- Soporta autenticación, por lo que cada usuario tendrá su propia agenda.
- Se sincroniza en tiempo real entre dispositivos.
- Está muy bien integrada con Flutter y tiene documentación abundante.

## DISEÑO DE LA BASE DE DATOS





**Tabla: *notifications***

Almacena información sobre las notificaciones enviadas a los usuarios.

- ***id*** (Integer, PK): Identificador único de la notificación (Clave Primaria).
- ***event\_id*** (Integer, FK): Identificador del evento asociado a la notificación (Clave Foránea, referencia a events.id). Puede ser nulo si la notificación no está directamente ligada a un evento.
- ***user\_id*** (Integer, FK): Identificador del usuario al que se envió la notificación (Clave Foránea, referencia a users.id).
- ***notification\_type\_id*** (Integer, FK): Identificador del tipo de notificación (Clave Foránea, referencia a notification\_types.id).
- ***sent\_at*** (Timestamp): Fecha y hora en que se envió la notificación.
- ***delivery\_method\_id*** (Integer, FK): Identificador del método de entrega de la notificación (ej: email, push) (Clave Foránea, referencia a delivery\_methods.id).
- ***status\_id*** (Integer, FK): Identificador del estado de la notificación (ej: enviada, entregada, fallida) (Clave Foránea, referencia a notification\_statuses.id).
- ***error\_message*** (Varchar): Mensaje de error si la entrega de la notificación falló.
- ***created\_at*** (Timestamp): Fecha y hora en que se creó el registro de la notificación.

**Tabla: *events***

Contiene información sobre los diferentes eventos.

- ***id*** (Integer, PK): Identificador único del evento (Clave Primaria).
- ***user\_id*** (Integer, FK): Identificador del usuario que creó el evento (Clave Foránea, referencia a users.id).
- ***title*** (Varchar, NN): Título del evento (No Nulo).
- ***description*** (Varchar): Descripción del evento.
- ***priority\_id*** (Integer, FK): Identificador de la prioridad del evento (Clave Foránea, referencia a priorities.id).
- ***notification*** (Boolean): Indica si se debe enviar una notificación para este evento.
- ***created\_at*** (Timestamp): Fecha y hora en que se creó el evento.

**Tabla: *users***

Almacena información sobre los usuarios del sistema.

- **id** (Integer, PK): Identificador único del usuario (Clave Primaria).
- **username** (Varchar, NN, Unique): Nombre de usuario único (No Nulo, Único).
- **password** (Varchar, NN): Contraseña del usuario (No Nulo).
- **created\_at** (Timestamp): Fecha y hora en que se creó la cuenta del usuario.
- **location\_id** (Integer, FK): Identificador de la ubicación del usuario (Clave Foránea, referencia a locations.id).
- **name** (Varchar): Nombre completo del usuario.
- **priority\_number** (Integer): Número de prioridad asociado al usuario (podría usarse para algún tipo de clasificación).

**Tabla: *notification\_types***

Define los diferentes tipos de notificaciones (ej: recordatorio de evento, actualización de estado).

- **id** (Integer, PK): Identificador único del tipo de notificación (Clave Primaria).
- **name** (Varchar, NN): Nombre del tipo de notificación (No Nulo).
- **description** (Varchar): Descripción del tipo de notificación.

**Tabla: *delivery\_methods***

Define las formas en que se pueden enviar las notificaciones (ej: email, SMS, push).

- **id** (Integer, PK): Identificador único del método de entrega (Clave Primaria).
- **name** (Varchar, NN): Nombre del método de entrega (No Nulo).
- **description** (Varchar): Descripción del método de entrega.

**Tabla: *notification\_statuses***

Define los posibles estados de una notificación (ej: pendiente, enviada, entregada, fallida).

- **id** (Integer, PK): Identificador único del estado de la notificación (Clave Primaria).
- **name** (Varchar, NN): Nombre del estado de la notificación (No Nulo).
- **description** (Varchar): Descripción del estado.

**Tabla: *priorities***

Define los diferentes niveles de prioridad para los eventos (ej: alta, media, baja).

- **id** (Integer, PK): Identificador único de la prioridad (Clave Primaria).
- **name** (Varchar, NN): Nombre de la prioridad (No Nulo).

**Tabla: *locations***

Almacena información sobre las ubicaciones.

- **id** (Integer, PK): Identificador único de la ubicación (Clave Primaria).
- **name** (Varchar): Nombre de la ubicación.
- **address** (Varchar): Dirección de la ubicación.
- **city** (Varchar): Ciudad de la ubicación.
- **country** (Varchar): País de la ubicación.
- **postal\_code** (Decimal): Código postal de la ubicación.
- **latitude** (Decimal): Latitud de la ubicación.
- **longitude** (Decimal): Longitud de la ubicación.
- **created\_at** (Timestamp): Fecha y hora en que se creó el registro de la ubicación.
- **updated\_at** (Timestamp): Fecha y hora en que se actualizó el registro de la ubicación.

**Tabla:** *event\_details*

Actúa como una tabla padre para los detalles específicos de los eventos.

- **id** (Integer, PK): Identificador único del detalle del evento (Clave Primaria).
- **event\_id** (Integer, FK, Unique): Identificador del evento asociado (Clave Foránea, referencia a events.id, Único - indica una relación de uno a uno o uno a cero con events).
- **type\_id** (Integer, FK): Identificador del tipo de detalle del evento (Clave Foránea, referencia a event\_types.id).

**Tabla:** *event\_types*

Define los diferentes tipos de eventos que pueden existir (ej: reunión, cita, tarea).

- **id** (Integer, PK): Identificador único del tipo de evento (Clave Primaria).
- **name** (Varchar, NN): Nombre del tipo de evento (No Nulo).

**Tabla:** *conference\_details*

Detalles específicos para eventos de tipo conferencia.

- **event\_detail\_id** (Integer, FK, PK, Unique): Identificador del detalle del evento asociado (Clave Foránea, referencia a **event\_details.id**, Clave Primaria, Único - relación de uno a uno con event\_details).
- **location\_id** (Integer, FK): Identificador de la ubicación de la conferencia (Clave Foránea, referencia a locations.id).
- **start\_date** (Date): Fecha de inicio de la conferencia.
- **end\_date** (Date): Fecha de fin de la conferencia.

**Tabla: *appointment\_details***

Detalles específicos para eventos de tipo cita.

- **event\_detail\_id** (Integer, FK, PK, Unique): Identificador del detalle del evento asociado (Clave Foránea, referencia a event\_details.id, Clave Primaria, Único - relación de uno a uno con event\_details).
- **location\_id** (Integer, FK): Identificador de la ubicación de la cita (Clave Foránea, referencia a locations.id).
- **appointment\_date** (Date): Fecha de la cita.
- **with\_person** (Varchar): Nombre de la persona con la que es la cita.

**Tabla: *task\_details***

Detalles específicos para eventos de tipo tarea.

- **event\_detail\_id** (Integer, FK, PK, Unique): Identificador del detalle del evento asociado (Clave Foránea, referencia a event\_details.id, Clave Primaria, Único - relación de uno a uno con event\_details).
- **due\_date** (Date): Fecha de vencimiento de la tarea.
- **status\_id** (Integer, FK): Identificador del estado de la tarea (Clave Foránea, referencia a task\_statuses.id).

**Tabla: *task\_statuses***

Define los posibles estados de una tarea (ej: pendiente, en progreso, completada).

- **id** (Integer, PK): Identificador único del estado de la tarea (Clave Primaria).
- **name** (Varchar, NN): Nombre del estado de la tarea (No Nulo).
- **description** (Varchar): Descripción del estado.

**Tabla: *meeting\_details***

Detalles específicos para eventos de tipo reunión.

- **event\_detail\_id** (Integer, FK, PK, Unique): Identificador del detalle del evento asociado (Clave Foránea, referencia a event\_details.id, Clave Primaria, Único - relación de uno a uno con event\_details).
- **location\_id** (Integer, FK): Identificador de la ubicación de la reunión (Clave Foránea, referencia a locations.id).
- **meeting\_date** (Date): Fecha de la reunión.

**Tabla: *exam\_details***

Detalles específicos para eventos de tipo examen.

- **event\_detail\_id** (Integer, FK, PK, Unique): Identificador del detalle del evento asociado (Clave Foránea, referencia a event\_details.id, Clave Primaria, Único - relación de uno a uno con event\_details).
- **subject** (Varchar, NN): **Asignatura del examen (No Nulo)**.
- **exam\_date** (Date): Fecha del examen.

**Tabla: *emails***

Almacena información sobre los correos electrónicos enviados.

- **id** (Integer, PK): Identificador único del correo electrónico (Clave Primaria).
- **user\_id** (Integer, FK): Identificador del usuario al que se envió el correo electrónico (Clave Foránea, referencia a users.id).
- **address** (Varchar, NN): Dirección de correo electrónico del destinatario (No Nulo).
- **suffix\_id** (Integer, FK): Identificador del sufijo del correo electrónico (Clave Foránea, referencia a email\_suffixes.id).
- **is\_primary** (Boolean): Indica si esta es la dirección de correo electrónico principal del usuario.
- **created\_at** (Timestamp): Fecha y hora en que se creó el registro del correo electrónico.

**Tabla: *email\_suffixes***

Define los diferentes sufijos de correo electrónico (ej: @gmail.com, @example.com).

- **id** (Integer, PK): Identificador único del sufijo de correo electrónico (Clave Primaria).
- **name** (Varchar, NN): Nombre del sufijo (No Nulo).

## ASPECTOS FUNCIONALES Y DISEÑO DE LA APLICACIÓN

En Eventify, para la pantalla de login, se ha utilizado la arquitectura Clean combinada con el patrón Model-View-ViewModel (MVVM).

**Arquitectura Clean:** Se ha comenzado a implementar para separar la lógica de autenticación (la parte importante de la funcionalidad del login) de los detalles de cómo se muestra la interfaz de login (los campos de texto y el botón) y de cómo se acceden a los datos de autenticación (por ejemplo, a través de una llamada a un servidor). Esto hace que la lógica de login sea más independiente, fácil de probar y reutilizable en el futuro si se cambia la interfaz o la forma de autenticar. Aunque actualmente solo se ha aplicado en la pantalla de login, el objetivo es extender esta arquitectura a toda la aplicación para lograr una mayor separación de responsabilidades y mejorar la mantenibilidad general.

**Model-View-ViewModel (MVVM):** Se ha aplicado específicamente en la pantalla de login porque simplifica la gestión de la interfaz de usuario y su estado. El ViewModel actúa como un intermediario entre la interfaz (View) y la lógica de autenticación (Model en este contexto). La interfaz de login se limita a mostrar la información que le proporciona el ViewModel y a notificarle las acciones del usuario (como escribir en un campo o pulsar un botón). El ViewModel contiene toda la lógica de presentación, como validar los datos de entrada, iniciar la autenticación y manejar los estados de carga o error. Esto resulta en un código de interfaz más limpio y una lógica de presentación más fácil de probar. Al igual que con la Arquitectura Clean, la intención es adoptar el patrón MVVM en otras pantallas y funcionalidades de la aplicación para una gestión de UI consistente y eficiente.

## PAQUETIZACIÓN

La estructura de código de la aplicación Eventify, ubicada en la carpeta lib, se organiza en varios paquetes y ficheros clave. Para la gestión de la autenticación de usuarios, incluyendo las funcionalidades de inicio y registro de sesión, se ha creado el paquete auth. El paquete common alberga código reutilizable y utilidades compartidas por diferentes partes de la aplicación. La gestión de dependencias e inyección se realiza a través del paquete di. Por último, el paquete l10n está dedicado a la internacionalización y localización, conteniendo los recursos para las traducciones de la aplicación.

Además de estos paquetes, se encuentran dos ficheros principales en la raíz de lib. firebase\_options.dart contiene la configuración necesaria para establecer la conexión con la base de datos Firebase. El fichero main.dart sirve como punto de entrada de la aplicación, encargándose de su inicialización y el arranque de la interfaz de usuario.

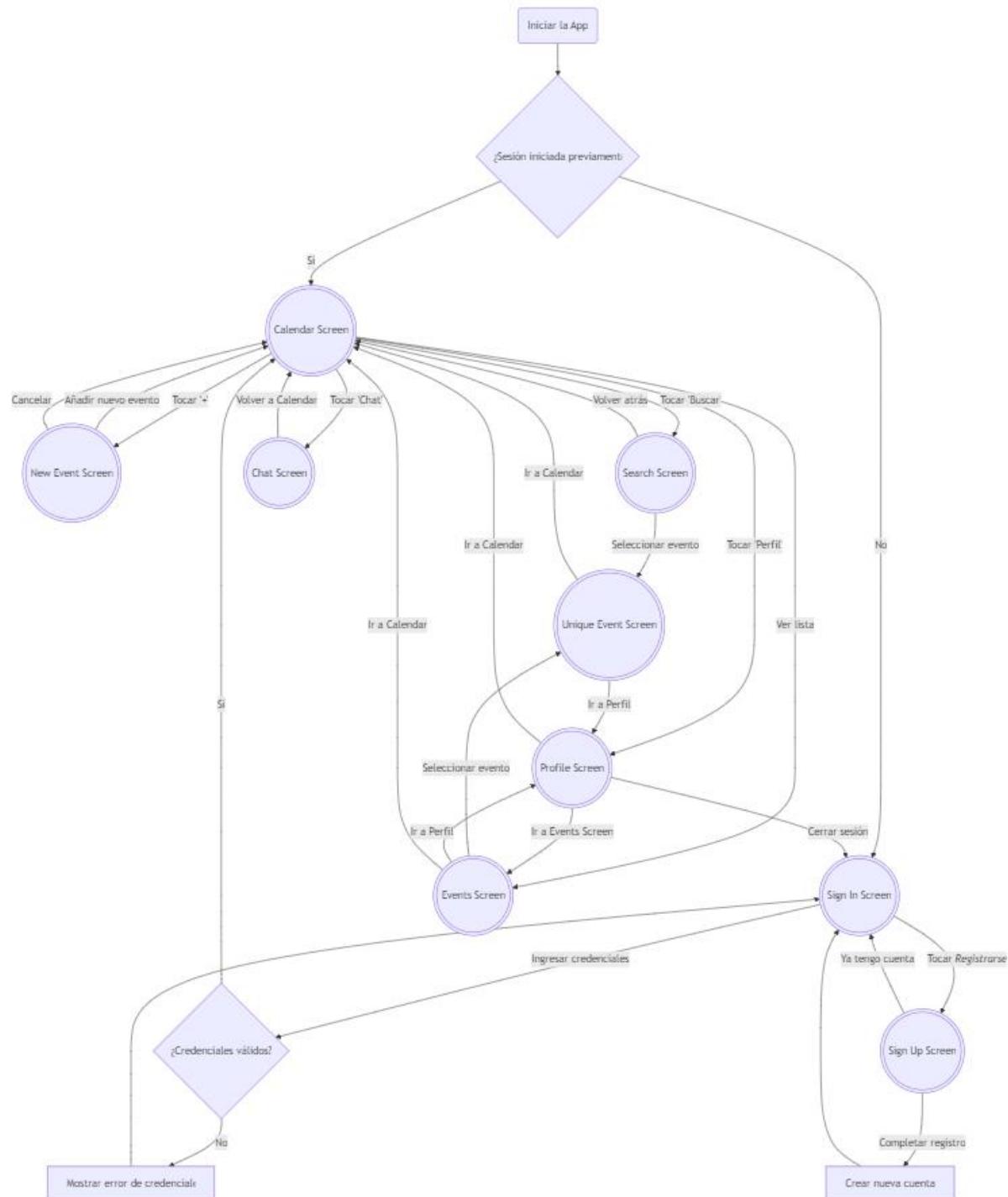
Contenido de auth (implementada arquitectura clean y MVVM):

- data
  - data\_sources
  - models
  - repositories
- domain
  - entities
  - presentation
    - screen
    - view\_model
  - repositories
  - use\_cases

Contenido de common:

- theme
  - colors
- utils
  - auth
  - dates
  - errors
- widgets (en este momento dentro de la carpeta se encuentran widgets que no son comunes en varias pantallas)
  - auth
    - animations
    - widgets
  - calendar
    - animations
    - widgets
  - chat
    - ...
  - Profile
    - ...

## NAVEGACIÓN

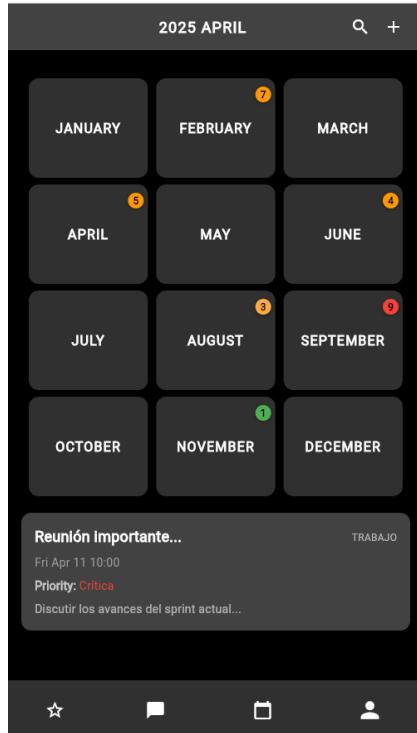


# IMPLEMENTACIÓN DE LA APLICACIÓN

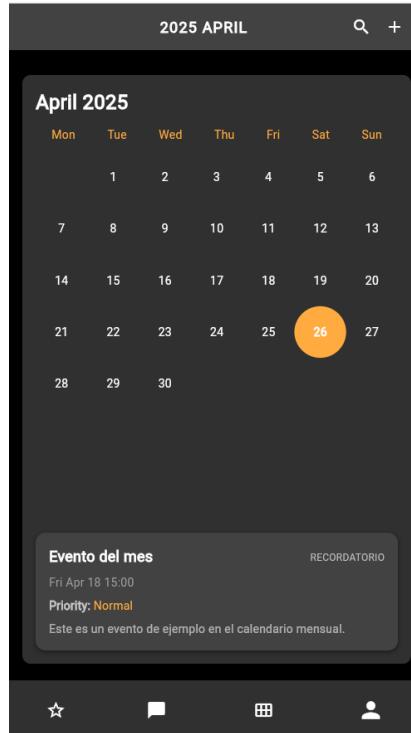
## INTERFAZ DEL USUARIO

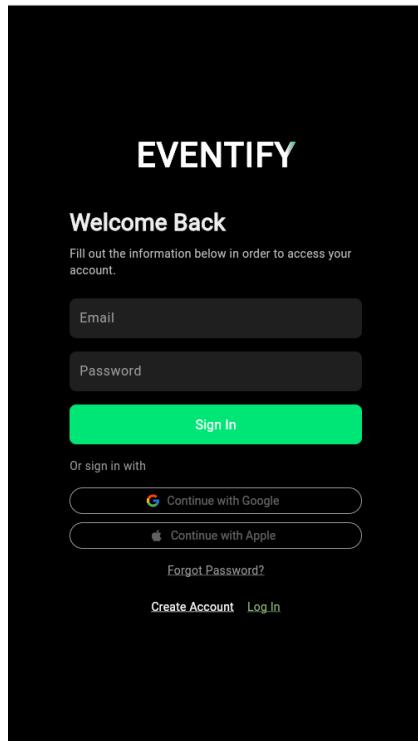
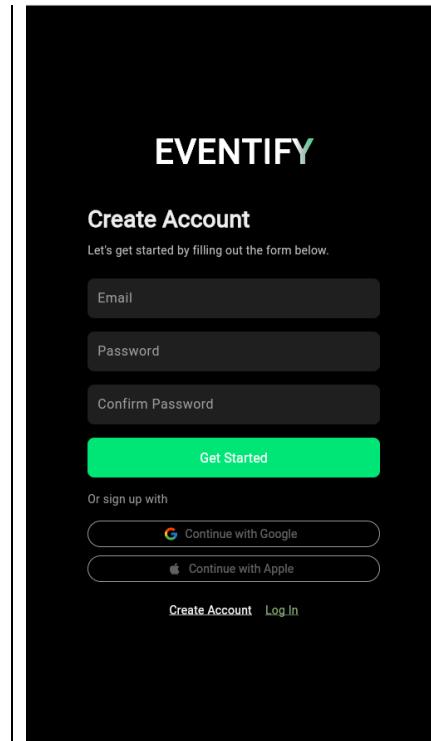
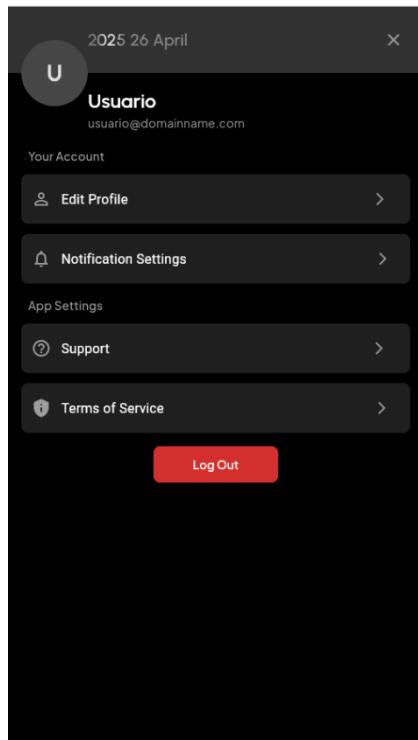
### DISEÑO DE LAS INTERFACES

CALENDARIO ANUAL



CALENDARIO MENSUAL



**INICIO DE SESIÓN****REGISTRO****PERFIL****CHAT CON EVENTIFY**

## ASPECTOS DESTACABLES DEL DESARROLLO

**Investigación sobre Mistral (Fase Inicial):** La exploración de la IA con Mistral presentó complicaciones iniciales debido a la novedad de la tecnología y la necesidad de comprender sus fundamentos.

**Adaptación a Firebase (Fase de Backend):** La transición a Firebase implicó una complejidad significativa al requerir un cambio de mentalidad respecto a la gestión y estructura de los datos.

**Problemas de Traducción (Fase de Localización):** La implementación de la localización con l10n en Flutter generó diversas complicaciones relacionadas con la configuración, la gestión de los archivos de traducción y la correcta visualización en la aplicación.

## LÍNEAS DE TRABAJO FUTURO

**Conexión Completa con la Base de Datos:** Se planea implementar la totalidad de las funcionalidades que requieran interacción con la base de datos Firebase para la gestión integral de los datos de la aplicación.

**Implementación de Arquitectura Clean y MVVM:** Se extenderá la adopción de la arquitectura Clean y el patrón MVVM a todas las capas y módulos del proyecto para mejorar la organización, testabilidad y mantenibilidad del código.

**Pantalla de Carga Inicial:** Se desarrollará e integrará una pantalla de carga que se mostrará al iniciar la aplicación, proporcionando una experiencia de usuario más fluida durante el proceso de inicialización.

**Implementación de Pantallas Clave:** Se crearán las interfaces de usuario y la lógica de negocio para las pantallas de búsqueda de eventos, creación de nuevos eventos y visualización de la lista de eventos.

**Integración del Agente de IA:** Se implementarán las funcionalidades del agente de inteligencia artificial para ofrecer asistencia inteligente y proactiva al usuario dentro de la aplicación.

## CONCLUSIONES

El proyecto Eventify es una iniciativa en curso para crear una agenda móvil multiplataforma intuitiva y potente, desarrollada con Flutter y Dart para su despliegue en Android e iOS. La base de datos en la nube de Firebase asegura la persistencia de los datos y sienta las bases para la futura escalabilidad de la aplicación.

Si bien la aplicación ofrece funcionalidades fundamentales de gestión y búsqueda de eventos, características distintivas como la asistencia inteligente (similar a la de un agente de IA) y la amplia gama de opciones de personalización de colores aún se encuentran en fase de implementación. La visión del proyecto es ofrecer una experiencia de usuario enriquecida con capacidades proactivas y una interfaz altamente adaptable a las preferencias individuales.

La estructura de la aplicación, con una adopción inicial de principios de arquitectura limpia y MVVM, busca facilitar la futura expansión y el mantenimiento del código. En resumen, Eventify, actualmente en desarrollo, se perfila como una solución de agenda móvil con un potencial significativo, combinando la eficiencia de Flutter con la promesa de funcionalidades avanzadas de asistencia y personalización que la diferenciarán en el mercado.

El proyecto Eventify representa una emocionante aventura en el desarrollo de una agenda móvil multiplataforma, construida con Flutter y Dart para alcanzar usuarios en Android e iOS. La elección de Firebase como backend proporciona una base sólida para la persistencia de datos y la escalabilidad futura.

El desarrollo de Eventify, aunque actualmente enfocado en funcionalidades esenciales de gestión y búsqueda de eventos, es un proceso que encuentro profundamente gratificante y que siento que está impulsando significativamente mi crecimiento como desarrollador. **La perspectiva de implementar características distintivas como la asistencia inteligente y una amplia paleta de personalización visual, actualmente en desarrollo, me entusiasma enormemente.**

Anticipo que la culminación de este proyecto, con su combinación de una interfaz de usuario intuitiva creada con Flutter y la promesa de funcionalidades avanzadas, me brindará una satisfacción increíble al ver cómo una herramienta tan

completa y personalizada llega a los usuarios. Eventify no es solo una aplicación; es un reflejo de mi progreso y dedicación como desarrollador.

## BIBLIOGRAFÍA

OpenAI. ChatGPT. Recuperado en abril de 2025. Utilizado como asistente virtual para generar contenido, resolver dudas técnicas y aclarar conceptos de programación.

Google. Gemini de Google. Recuperado en abril de 2025. Empleado para contrastar información y obtener diferentes perspectivas sobre temas de desarrollo.

YouTube. Desarrollo de Aplicaciones móviles en 2024. Recuperado en abril de 2025. Este video proporciona una guía completa para aspirantes a desarrolladores de aplicaciones móviles en 2024, cubriendo el conocimiento esencial para comenzar en este campo. Explica las dos principales metodologías de desarrollo: nativo y multiplataforma. Para el desarrollo nativo, se mencionan Objective-C y Swift para iOS, y Java y Kotlin para Android. En cuanto al desarrollo multiplataforma, se discuten frameworks como React Native, Flutter, Kotlin Multiplatform y .NET MAUI. El video también ofrece una hoja de ruta para el aprendizaje, incluyendo el aprendizaje de un lenguaje de programación base, un framework/SDK, Git y GitHub, y servicios backend.

YouTube. Desarrollo de Aplicaciones móviles en 2024. Recuperado en abril de 2025. Utilizado como referencia para analizar diferentes variantes de estructuras de proyecto y tecnologías como Flutter y Kotlin Multiplatform, con el objetivo de tomar decisiones informadas durante el desarrollo de la aplicación.

YouTube. Flutter vs Kotlin Multiplatform ¿Por qué Google invierte en dos tecnologías enfrentadas?. Recuperado en abril de 2025. Se ha tenido en cuenta la opinión del creador del video y también las reflexiones aportadas por los usuarios en la sección de comentarios, como parte del análisis sobre la viabilidad y enfoque del uso de Flutter en el desarrollo del proyecto.

YouTube. Flutter vs Kotlin Multiplatform: Google se posiciona. Recuperado en abril de 2025. Se analiza la estrategia de Google al invertir en dos tecnologías aparentemente competidoras para el desarrollo de aplicaciones móviles: Flutter y Kotlin Multiplatform. El video explora los posibles enfoques y el posicionamiento de Google frente a estas dos

opciones, considerando sus fortalezas y debilidades en el ecosistema del desarrollo móvil.

FlutterFlow. *FlutterFlow - UI Builder para Flutter*. Recuperado en abril de 2025. Se utilizará como herramienta principal para el diseño visual de la interfaz de usuario del proyecto. Gracias a su editor gráfico tipo *drag and drop*, permite construir pantallas de manera intuitiva, similar al sistema de diseño de vistas de Android Studio. Esta plataforma acelera el proceso de prototipado y desarrollo frontend, facilitando también la exportación del código Flutter resultante para integrarlo con la lógica personalizada del proyecto.

DartPackages. *Firebase Core - Instalación de Firebase en Flutter*. Recuperado en abril de 2025. Se ha utilizado este recurso para instalar y configurar **Firebase Core**, la biblioteca base necesaria para la integración de Firebase en Flutter. La documentación oficial ha servido de guía para la correcta implementación de los servicios de Firebase dentro del proyecto, asegurando compatibilidad y optimización en el entorno de desarrollo.

YouTube. *Tutorial Crear Agentes AI Mistral - Crea tus propios agentes de IA con Mistral*. Recuperado en abril de 2025. Utilizado como referencia para la creación y comprensión del funcionamiento del agente de IA basado en Mistral.

YouTube. *Traducciones y localización en Flutter con l10n | Internacionaliza tu app*. Recuperado en abril de 2025. Utilizado como referencia para comprender el proceso de internacionalización y localización de aplicaciones desarrolladas con Flutter, específicamente mediante el uso de la herramienta l10n.

## ANEXOS

[ENLACE AL GITHUB DEL PROYECTO](#)