# CONTEXTO DE LA APLICACIÓN

Eventify es una agenda multiplataforma desarrollada con Flutter y Dart, está diseñada para proporcionar a los usuarios una herramienta intuitiva potente y adaptable para la gestión de su tiempo y eventos. Eventify está disponible para Android, iOS, macOS, GNU/Linux y Windows, ofreciendo una experiencia unificada, fluida y asistida por un agente de inteligencia artificial integrado, independientemente del dispositivo que utilice el usuario.

La aplicación de Eventify destaca por su interfaz interactiva de uso sencillo, lo que minimiza la curva de aprendizaje al máximo, es una aplicación que toda persona situada en cualquier rango de edad puede usar. Un aspecto clave es la personalización visual, permitiendo a los usuarios modificar la apariencia de la aplicación mediante la selección de colores dinámicos según sus preferencias.

Eventify también incorpora funcionalidades avanzadas de gestión de eventos, como la posibilidad de añadir, modificar y eliminar eventos en cualquier fecha del calendario. Cuenta además con un sistema de búsqueda avanzada por palabras clave, niel de prioridad, tipo de evento y demás filtros según el contexto, como puede ser filtrar por localidad en conferencias.

Eventify cuenta con la integración de su propio agente inteligente, disponible en toda instancia para asistir al usuario con tareas como la sugerencia de recordatorios optimizados, la organización del calendario o la localización rápida de eventos relevantes.

Para evitar que se pasen por alto eventos importantes, Eventify ofrece un sistema de notificaciones que envían recordatorios con antelación según las preferencias del usuario. La persistencia de datos está asegurada mediante el almacenamiento en la nube, lo que permitirá la sincronización entre dispositivos y abrirá la puerta a futuras funcionalidades más inteligentes basadas en el análisis de patrones de uso.

En resumen, Eventify se presenta como una solución de agenda-calendario multiplataforma completa que redefine la organización personal al integrar un agente inteligente proactivo, una experiencia personalizable y un diseño centrado en la eficiencia, sincronización, escalabilidad y seguridad.

## Comparativa de eventify con otras aplicaciones en el mercado

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

### 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.
* Alertas basadas en ubicación.
* Integración con Siri para establecer recordatorios con comandos de voz.

Sincronización y compatibilidad

* Sincronización con iCloud.
* Compatibilidad con calendarios de terceros (Google Calendar, Microsoft Outlook, etc.)
* 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.

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

Gestión de eventos y tareas:

* Creación 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 y 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.

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 | Google Calendar | Eventify |
| Interactividad y facilidad de uso | Interfaz limpia y sencilla | Intuitivo y bien integrado | Totalmente personalizable según las necesidades del usuario |
| Personalización de colores | Solo modo claro/oscuro | Temas limitados (modo claro/oscuro) | Opción para elegir cualquier color para la interfaz |
| Gestión de eventos | Añadir, modificar y eliminar | Añadir, modificar y eliminar | Con funciones avanzadas y filtros personalizados |
| Búsqueda avanzada | Búsqueda básica | Búsqueda limitada a títulos y descripciones | Filtrado por palabras clave, prioridad, tipo, ubicación, etc. |
| Persistencia de datos | iCloud (requiere internet) | Google Drive (requiere internet) | Almacenamiento en la nube (requiere internet) |
| Notificaciones | Alerta estándar | Alertas configurables | Configuración personalizada para eventos según su prioridad |
| Compatibilidad | Solo en dispositivos Apple | Solo en dispositivos con Google | Soporte para Android, iOS, escritorio y más plataformas gracias a flutter |
| Escalabilidad | Limitado a funciones predeterminadas | Google puede actualizar, pero con restricciones | Capacidad de agregar nuevas funciones en el futuro |
| Seguridad | Datos encriptados en iCloud | Protección con Google Account | Seguridad garantizada mediante autenticación y almacenamiento en la nube |
| Rendimiento | Optimizado para Apple | Integrado con Google, pero puede tener lag en algunos dispositivos | Fluido y optimizado para dispositivos Android, iOS y escritorio |

### 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 Eventify ofrece esta opción de personalización.
* Búsqueda avanzada: Eventify permite filtrar eventos por múltiples criterios como palabras clave, prioridad, tipo, ubicación, etc. A diferencia de Apple Calendar y Google Calendar, que ofrecen opciones más limitadas.
* Escalabilidad: Eventify tiene la capacidad de agregar nuevas funciones en el futuro, permitiendo una mayor personalización y expansión de sus características.

# ANÁLISIS

## Necesidades del usuario

La aplicación debe ser interactiva y fácil de usar, con una interfaz intuitiva y sin una curva de aprendizaje elevada, debe poder adecuarse a todos los públicos, sin restringir ningún rango de edad.

El usuario deberá poder personalizar ciertos aspectos de la aplicación según sus gustos, eligiendo los colores de diversas partes de la aplicación, lo que favorecerá a la experiencia dentro de esta.

El sistema debe contestar a toda solicitud de forma rápida y eficiente, no debe retrasarse al interactuar con el usuario.

La aplicación deberá ser accesible desde dispositivos móviles Android y iOS, así como en escritorio desde ordenadores portátiles y sobremesa. Su diseño multiplataforma deberá garantizar una experiencia consistente y optimizada en los diversos entornos.

La gestión de notificaciones debe ser flexible, permitiendo configurar qué avisos recibir y en qué momentos, así como silenciar notificaciones temporal o permanentemente.

## Requisitos funcionales

Gestión de eventos. El usuario podrá añadir, modificar o eliminar cualquier evento en cualquier fecha del calendario.

Búsqueda avanzada. El usuario podrá buscar eventos a través de múltiples filtros.

Notificaciones. El usuario podrá configurar el evento para que se le mande notificaciones sobre el mismo, ya sea a través de su prioridad o del propio evento.

Modo offline. El usuario deberá poder interactuar con la aplicación de forma más limitada sin conexión a internet.

Soporte multiplataforma. La aplicación funcionará correctamente en Android, iOS, Windows, GNU/Linux y macOS.

Chat y mensajería. El usuario podrá enviar y recibir mensajes de la IA en tiempo real.

Internacionalización. La aplicación debe estar disponible en múltiples idiomas.

Seguridad. La aplicación deberá proteger los datos personales de los usuarios y requerir autentificación para acceder a las funciones principales.

Integración con servicios externos. El sistema deberá permitir la autenticación mediante proveedores externos.

Agente artificial especializado. El sistema deberá contar con un agente artificial especializado capaz de interpretar instrucciones del usuario para añadir, eliminar o modificar eventos en el calendario. El usuario podrá interactuar con el agente mediante un lenguaje neutral.

## Requisitos no funcionales

Características de los eventos. Los eventos se podrán diferenciar por el título, la descripción, la prioridad, la fecha y la hora, y el tipo de evento. Dependiendo del tipo de evento, se le podrán añadir unos atributos u otros.

Campo de notificación en los eventos. A la hora de añadir un evento el usuario deberá poder elegir si quiere que la aplicación le notifique o no antes de la fecha de realización de dicho evento.

Adaptabilidad de formularios El sistema deberá ser capaz de mostrar u ocultar campos en los formularios de eventos de manera fluida y sin recargar pantallas.

Mantenibilidad. El código debe ser legible y seguir buenas prácticas de desarrollo para facilitar futuras modificaciones.

Usabilidad. La interfaz debe ser intuitiva y fácil de usar para usuarios sin experiencia técnica, además todo texto debe estar correctamente traducido al idioma que corresponda al usuario.

Escalabilidad. El sistema debe soportar un crecimiento en el número de usuarios y eventos sin degradar el rendimiento.

Disponibilidad. La aplicación debe estar disponible en todo momento.

Seguridad. Los datos de usuario deben transmitirse cifrados mediante HTTPS, además las contraseñas deberán almacenarse de forma segura y nunca en texto plano.

Portabilidad. La aplicación deberá poder desplegarse en diferentes plataformas y entornos sin cambios significativos en el código.

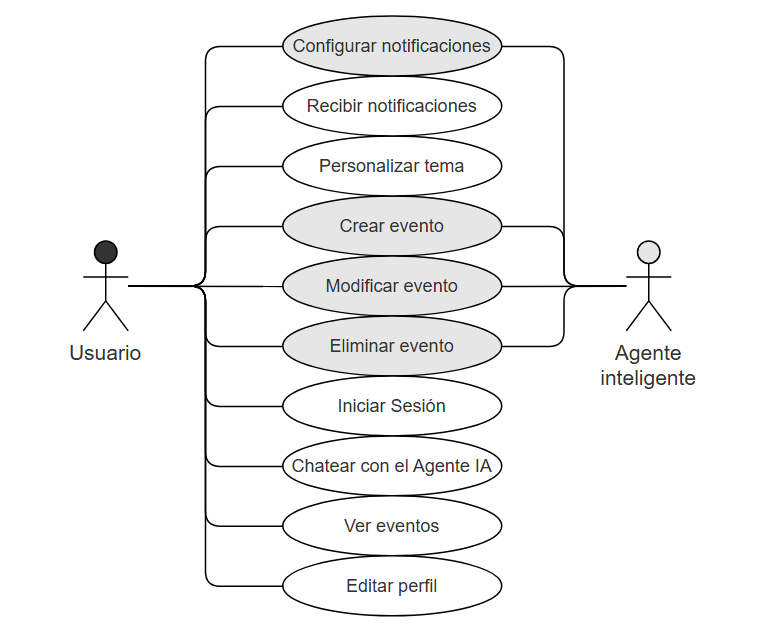
Consistencia visual. Los elementos de la interfaz deben respetar la paleta de colores marcada por el tema activo. Los iconos y textos mantendrán suficiente contraste para diferenciarse del fondo y garantizar la legibilidad en todos los temas.

Persistencia de preferencias. La preferencia de tema seleccionada por el usuario deberá guardarse y restaurarse automáticamente al reiniciar la aplicación.

Garantía de entrega de notificaciones. El sistema deberá garantizar que las notificaciones de eventos lleguen al usuario con suficiente antelación para que pueda actuar antes de la fecha y hora del evento. El tiempo de entrega de la notificación debe ser fiable y no superar el margen de antelación configurado por el usuario.

Interacción con el agente inteligente. El agente deberá responder a las instrucciones del usuario en menos de 5 segundos (siendo 5 segundos el caso en que la solicitud sea altamente compleja). Este mismo deberá responder coherentemente y con seguridad, sin perder el hilo de la conversación, evitando acciones ambiguas.

## Casos de uso



Un usuario puede registrarse en la aplicación utilizando su cuenta de Google que usa en su día a día. Una vez autenticado, el usuario accede a una interfaz intuitiva donde puede crear nuevos eventos, seleccionando el tipo de evento y completando los campos específicos requeridos para cada tipo. El usuario puede personalizar la apariencia de la aplicación eligiendo colores y temas, adaptando la experiencia a sus gustos.

La aplicación permite al usuario comunicarse en tiempo real a través de un chat integrado, facilitando el intercambio de mensaje relacionados con los eventos. El usuario puede buscar y filtrar eventos por diferentes criterios, como su título, descripción o fecha y configurar recordatorios personalizados para recibir notificaciones antes de la fecha de cumplimiento de cada evento.

El usuario puede acceder a la aplicación desde dispositivos móviles u ordenadores, disfrutando siempre de una experiencia consistente y optimizada. Además, puede gestiona su perfil, cambiar el idioma de la interfaz, configurar las notificaciones que desee recibir y consultar la ayuda integrada para resolver cualquier incidencia.

El usuario tiene la tranquilidad de que sus datos están a salvo, pudiendo sincronizarlos entre diferentes dispositivos y eliminarlos definitivamente si así lo desea, en cumplimiento de los principios de privacidad y control de la información personal.

## Comparativa de tecnologías multiplataforma: Flutter vs. KTM

Flutter y Kotlin Multiplatform son dos soluciones potentes para el desarrollo de aplicaciones multiplataforma. Mientras Flutter ofrece una única base de código con una interfaz coherente en todas las plataformas, KTM destaca por su interoperabilidad nativa y reutilización de lógica de negocio. La elección entre ambas opciones depende del enfoque deseado en diseño, desarrollo y escalabilidad.

|  |  |  |
| --- | --- | --- |
| Características | Flutter | KTM |
| Lenguaje | Dart | Kotlin |
| Base de código compartida | 100% compartida para Android, iOS, Web y escritorio | Lógica compartida, pero UI separada |
| UI | Propia, consistente y personalizable | Usa componentes nativos |
| Hot Reload | Sí | No |
| Tamaño de la app | Mayor, debido al motor de renderizado incluido | Más ligero al usar componentes nativos |
| Rendimiento | Alto, gracias al motor Skía | Nativo, sin capa adicional |
| Curva de aprendizaje | Requiere aprender Dar y estructura Flutter | Baja si ya conoces Kotlin, pero la integración con iOS requiere aprendizaje |
| Web y escritorio | Buen soporte e integración | Soporte más limitado, especialmente en la UI |
| Ecosistema y comunidad | Amplia comunidad y muchos paquetes disponibles | Comunidad más pequeña, menos recursos disponibles |
| Escalabilidad en proyectos grandes | Buena | Excelente |
| Reutilización de código | Completa | Parcial (principalmente en la lógica de negocio) |
| Popularidad del lenguaje | Dart es menos adoptado | Kotlin es más popular, especialmente en Android |
| Desarrollo ágil | Ideal, gracias al Hot Reload y un solo código | Menos ágil por la necesidad de compilar múltiples targets |
| Conclusión | Ideal para apps con UI consistente en múltiples plataformas | Ideal si se necesita rendimiento nativo con personalización de UI por plataforma |

### Decisión: Flutter vs. KTM

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 así la necesidad de desarrollar interfaces separadas para Android, iOS y escritorio. En contraste, KTM requiere escribir la lógica compartida, pero las interfaces deben desarrollarse por separado en cada plataforma.
2. Interfaz personalizada y flexible. Flutter ofrece un control total sobre la apriencia 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 del usuario y la usabilidad son aspectos fundamentales.
3. Desarrollo ágil con Hot Reaload. El Hot Reaload de Flutter permite visualizar los cambios en tiempo real sin necesidad de recompilar el código por completo, esto agiliza en proceso de desarrollo comparándolo con KTM, 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, KTM 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 extensa comunidad y un ecosistema de paquetes bien desarrollado, lo que facilia la integración de funcionalidad clavo como bases de datos, notificaciones, sincronización en la nube y autenticación. Esto reduce el tiempo de desarrollo en comparación con KTM, donde algunos recursos se muestran más 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.

## Comparativa de bases de datos

Para el desarrollo de Eventify, se necesita una base de datos que cumpla una serie de requisitos imprescindibles:

* Debe permitir guardar eventos personalizados (fecha, título, descripción, etc.).
* Debe ser accesible la máxima diversidad de dispositivos posible.
* Debe tener soporte para múltiples usuarios y gestión de sus eventos.
* Debe sincronizar los cambios en tiempo real y de forma eficiente.
* Debe albergar una opción gratuita suficiente para el uso inicial de la app.

Basado en estos requisitos, se ha decidido utilizar una base de datos en la nube, ya que así los datos se almacenarán de forma online sin perderse, incluso si el usuario cambia de dispositivo o reinstala la app.

|  |  |  |  |
| --- | --- | --- | --- |
| Características | Firebase Firestore | Firebase Realtime | Supabase |
| Tipo | NoSQL | NoSQL | SQL |
| Plan gratuito | 50K lecturas / día  20K escrituras / día  1 GB de almacenamiento | 1 GB de almacenamiento  100K conexiones simultáneas | 500MB de BBDD  2 GB de almacenamiento  10K usuarios autenticados |
| Pros | Tiempo real  Filtros y orden por campos  Excelente soporte en Flutter  Autenticación integrada | Sincronización ultra rápida  Muy fácil de usar | Consultas SQL reales  Código abierto  Autenticación integrada |
| Contras | Puede escalar rápido si se sobrepasan ciertos límites  Estructura por documentos | Menos flexible para estructuras complejas  Consultas limitadas | En desarrollo  Menor comunidad que Firebase |

### Decisión: Firebase Firestore

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

* El plan gratuito de Firebase Firestore 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.
* Soporte autenticación, por lo que cada usuario tendrá su propia agenda
* Se sincroniza en tiempo real entre dispositivos.
* Está perfectamente integrada con Flutter y tiene documentación abundante.

# DISEÑO

## UX – Experiencia del usuario en la aplicación

Para lograr la mejor experiencia posible para el usuario, la aplicación se ha ideado para cumplir con el objetivo de que cualquier usuario, independientemente de su edad o experiencia tecnológica, pueda utilizar la aplicación de forma intuitiva y sin barreras.

La interfaz ha sido diseñada para ser clara, atractiva y personalizable, permitiendo a cada usuario adaptar colores y estilos a su gusto. Los contenidos y las interacciones se han optimizado para que las tareas mas frecuentes (crear, modificar o eliminar eventos, chatear, recibir notificaciones, etc.) sean rápidas y sencillas, minimizando el número de pasos y evitando la sobrecarga de información.

Se ha puesto especial atención en la accesibilidad, garantizando que la aplicación sea usable por personas con diferentes capacidades, y en la coherencia visual y funcional entre las versiones móvil y escritorio, asegurando una experiencia consistente y fluida en cualquier plataforma. Además, la aplicación responde de forma ágil a las acciones del usuario, evitando esperas innecesarias y proporcionando feedback inmediato ante cualquier solicitud. El usuario puede personalizar su experiencia, gestionar sus notificaciones, y contar con ayuda accesible en todo momento.

## UI – Interfaz del usuario de la aplicación

Eventify ofrece una presentación clara, atractiva y coherente de los contenidos, facilitando la interacción y el acceso a todas las funcionalidades. El diseño de la interfaz abarca la disposición de los menús, botones, formularios, iconos, tipografías y colores, asegurando que cada elemento sea fácilmente identificable y accesible para el usuario. Se ha puesto especial atención en la consistencia visual, de modo que la navegación y el aspecto general sean coherentes en todas las pantallas y plataformas, evitando confusiones y mejorando la percepción de calidad.

Además del aspecto gráfico, la interfaz también contempla la organización de los textos y contenidos, priorizando la legibilidad y la jerarquía visual. Los mensajes, títulos y descripciones están diseñados para ser claros y directos, ayudando al usuario a comprender rápidamente cada sección y acción disponible.

La aplicación permite la personalización de ciertos aspectos visuales, como la selección de temas de color, para que cada usuario pueda adaptar la interfaz a sus preferencias. Así, de igual manera, se han utilizado recursos gráficos modernos y adaptativos, que garantizan una correcta visualización en diferentes tamaños y resoluciones de pantalla, desde móviles hasta escritorios.

## IxD – Diseño de la Interacción de la aplicación

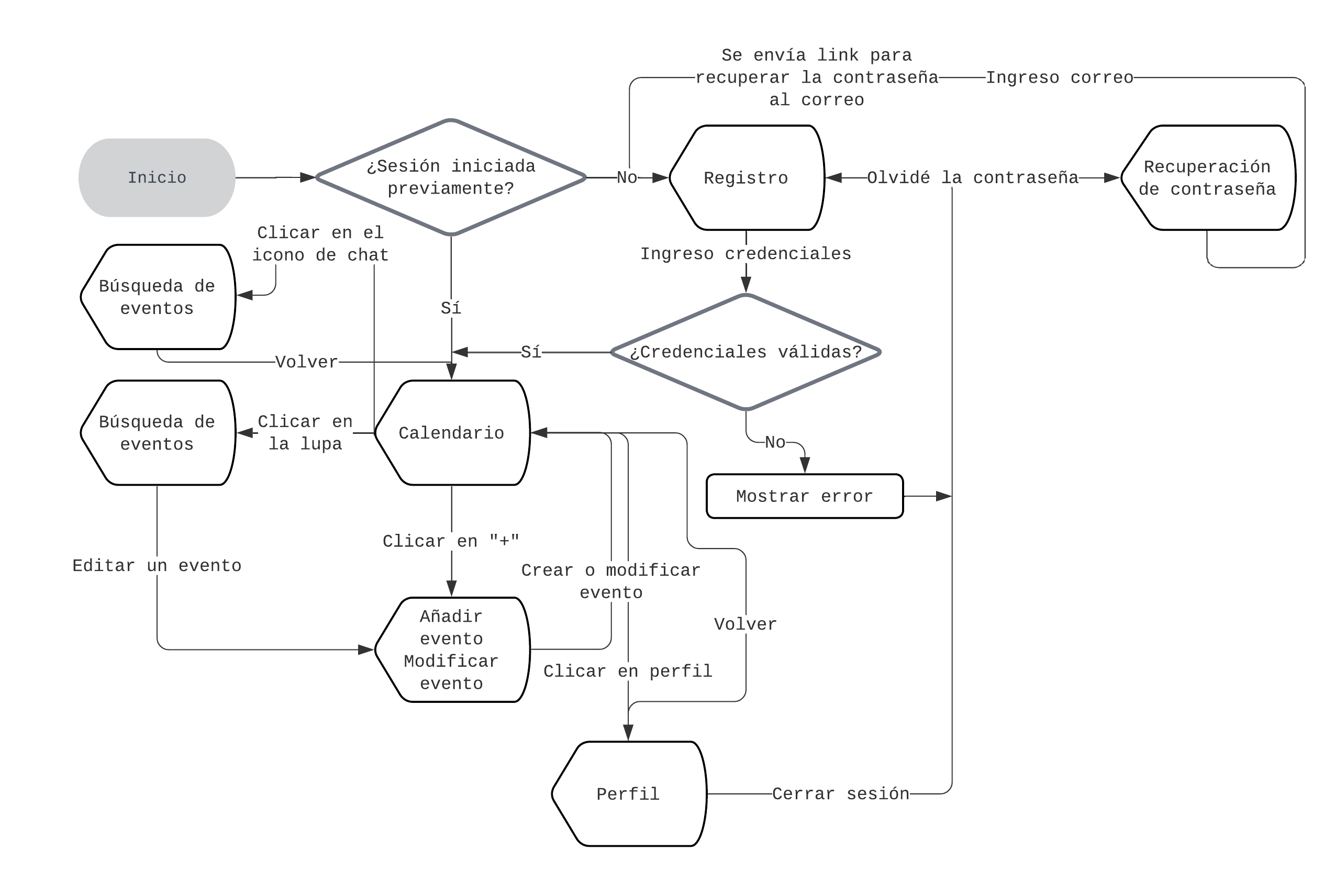
En Eventify se han diseñado flujos de navegación claros e intuitivos, donde el usuario puede acceder a las principales funciones mediante gestos naturales y acciones sencillas. Por ejemplo, un toque común permite acceder a la pantalla de búsqueda de eventos, mientras que con un deslizamiento puedes navegar en el historial de mensajes con el chat en su pantalla. Los formularios de creación y edición de eventos responden de manera inmediata a la introducción de datos, mostrando u ocultando campos dinámicamente según el tipo de evento seleccionado.

El sistema proporciona un feedback visual ante una acción relevante, como la creación, modificación o eliminación de eventos, el envío de mensajes o la recepción de notificaciones. Este feedback ayuda al usuario a comprender el resultado de sus acciones y a mantener el control durante su estancia en la aplicación.

Por otro lado, se han implementado atajos y accesos directos para agilizar tareas frecuentes, como la búsqueda de eventos similares a uno clicando sobre el mismo o la filtración según la fecha clicando sobre un día del calendario. La aplicación también adapta sus interacciones a las características del dispositivo, permitiendo, por ejemplo, el uso de gestos multitáctiles en dispositivos móviles o atajos de teclado en escritorio.

Siguiendo esta serie de principios, el diseño de la interacción en la aplicación hará que el usuario se sienta cómodo con la propia aplicación.

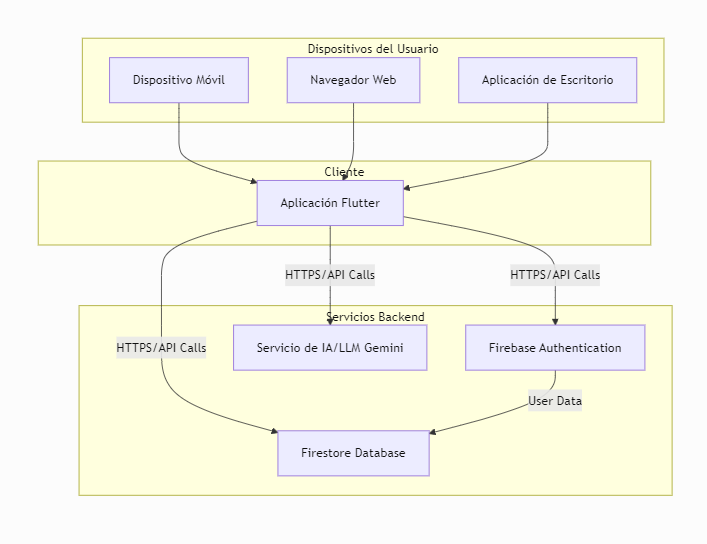
## Diagrama de despliegue



## Arquitectura

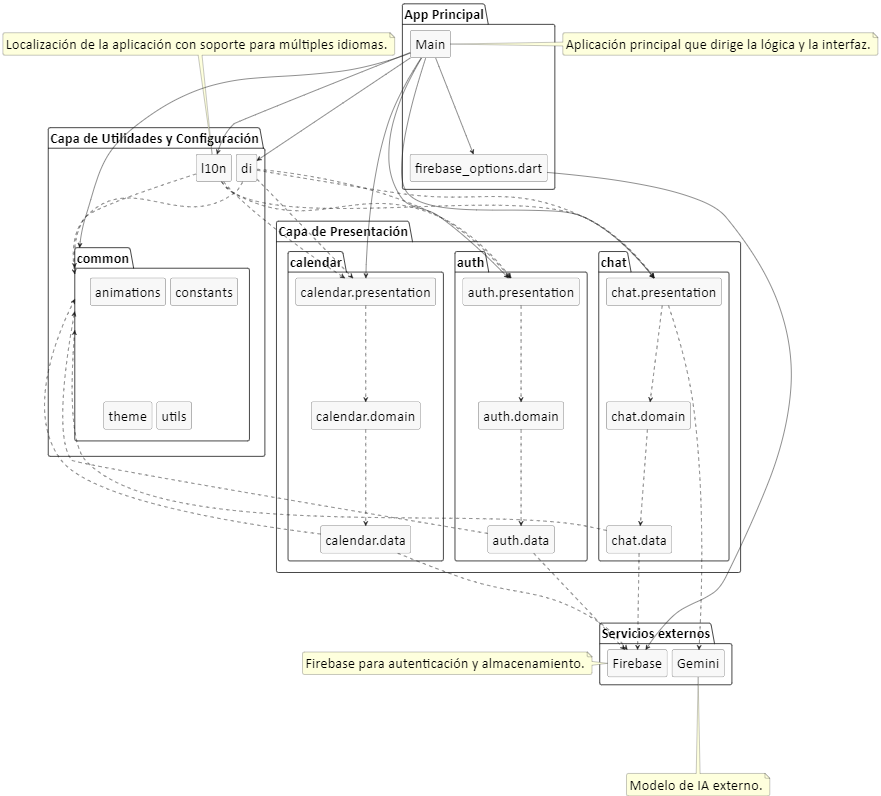
### Diagrama de despliegue

La arquitectura de Eventify ha sido diseñada para ser robusta, escalable y accesible desde múltiples plataformas.



El diagrama muestra como interactúan los diferentes componentes y servicios, ilustrando la separación de preocupaciones y el uso de tecnologías modernas de nube para ofrecer una experiencia de usuario fluida y variada en funcionalidades. Se centra en tres partes: los puntos de interacción del usuario, la lógica de la aplicación del cliente y los servicios de backend.

### Diagrama de componentes



## Diseño de la base de datos

El diseño de la base de datos de la aplicación se ha realizado pensando en la escalabilidad, la eficiencia y la facilidad de acceso a la información relevante para cada usuario. La estructura está orientada a soportar funcionalidades como la gestión de eventos, la creación y personalización de eventos y la gestión de notificaciones.

La estructura de los registros en la base de datos de Eventify en Firebase Firestore es el siguiente:

*UserId*

*Email*

*Username*

*Events*

*EventId*

*dateTime (Timestamp)*

*createdAt (Timestamp)*

*description (Text)*

*hasNotification (Boolean)*

*priority (Text) (low/médium/high/critical)*

*title (Text)*

*type (Text) (task, appointment, exam, meet, conference)*

*userId*

*subject (si es task)*

*withPersonYesNo (Boolean) (si es appointment)*

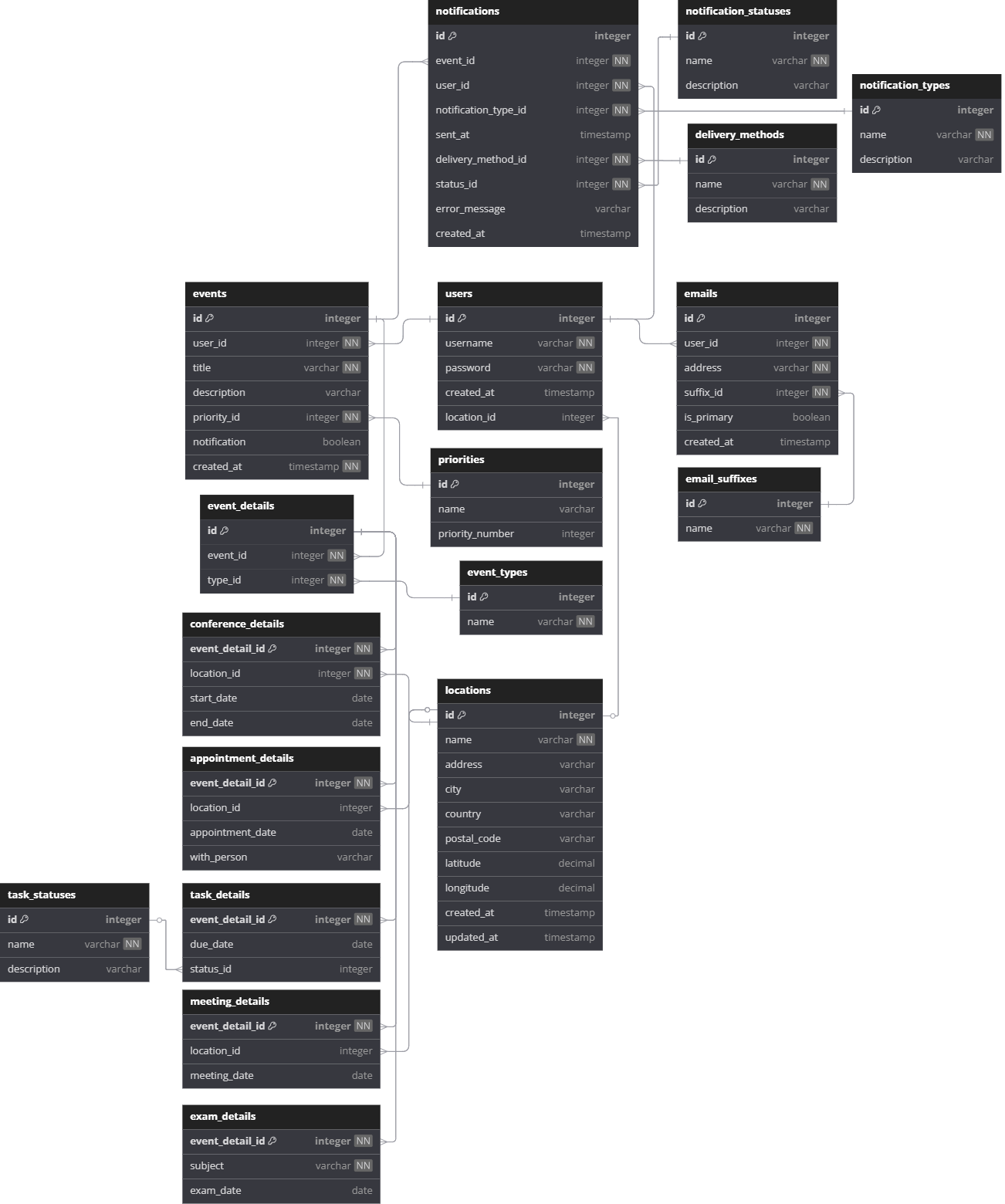
*withPerson (Text) (si es appointment)*

*location (Text) (si es conference, appointment o meeting)*

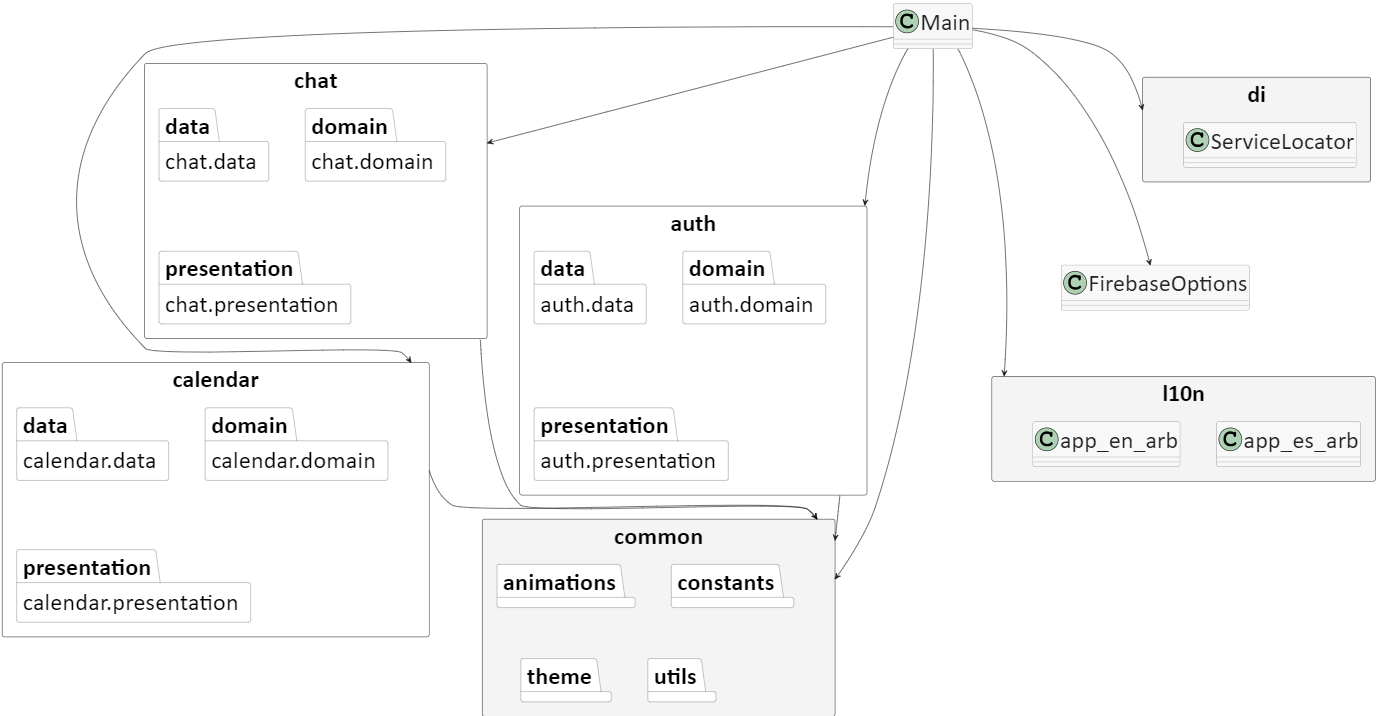
*LocationId*

*Name (Text)*

*createdAt (Timestamp)*



## Paquetización y organización



La organización del proyecto sigue una estructura modular y clara, facilitando el mantenimiento, la escalabilidad y la colaboración entre desarrolladores. Además, implementa el patrón MVVM (Model-View-ViewModel) combinado con principios Cleab Architecture, lo que permite una separación estricta de responsabilidades, facilita la testabilidad, y promueve la reutilización de componentes en distintas capas de la aplicación.

En la carpeta principal lib/ se encuentran los paquetes y archivos fundamentales que conforman la lógica y presentación de la aplicación.

El paquete auth contiene toda la lógica y las pantallas relacionadas con la autenticación de usuarios, incluyendo el inicio de sesión, registro, recuperación de contraseña y la gestión de sesiones. Este paquete centraliza los flujos de acceso y seguridad, permitiendo una gestión independiente y reutilizable de la autenticación.

El paquete calendar/ está dedicado a la gestión y visualización de eventos en formato de calendario. Aquí se implementan las funcionalidades para crear, modificar, eliminar y mostrar eventos, así como la lógica de navegación entre eventos y la integración con recordatorios.

El paquete chat/ agrupa los componentes y la lógica necesarios para la mensajería en tiempo real con el agente inteligente de Eventify. Incluye el envío y recepción de mensajes junto a la visualización de conversaciones.

El paquete common/ reúne utilidades, constantes, temas de diseño, animaciones y funciones compartidas por toda la aplicación. Este paquete promueve la reutilización del código y la coherencia visual y funcional en todas las pantallas.

El paquete di/ (dependency injection) contiene la configuración e implementación del service locator y la inyección de dependencias, facilitando la gestión de instancias y la escalabilidad del proyecto.

El paquete l10n/ almacena los archivos de localización (traducciones) en diferentes idiomas, permitiendo que la aplicación sea accesible y usable por usuarios de distintas regiones y lenguas.

El archivo firebase\_options.dart configuración específica de Firebase para la aplicación, necesaria para la integración con los servicios de autenticación, base de datos y notificaciones.

Por último, el archivo main.dart es el unto de entrada de la aplicación, donde se inicializan los servicios principales, se configura la localización, los temas y se define la estructura básica de la navegación.

### ¿Por qué se ha decidido utilizar MVVM con Clean Architecture?

Se ha optado por el uso de MVVM con Clean Architecture debido a que proporciona una separación más estricta de responsabilidades, favoreciendo tanto a la escalabilidad, como a la testabilidad y el mantenimiento de proyectos complejos. Esta combinación permite estructurar la lógica de negocio, la interfaz del usuario y la infraestructura de forma independiente, lo que resulta ideal para aplicaciones modernos como lo es Eeventify, donde hay múltiples funcionalidades y necesidad de una arquitectura robusta.

|  |  |  |  |
| --- | --- | --- | --- |
| Características | MVC | MVP | MVVM |
| Facilidad de uso | Simple | Moderado | Complejo |
| Testabilidad | Limitada | Buena | Excelente |
| Organización de código | Básica | Muy buena | Excelente |
| Rendimiento | Bueno | Muy bueno | Excelente |
| Destaca en | Aplicaciones pequeñas | Proyectos medianos | Aplicaciones complejas y proyectos grandes |

## Plan de pruebas

El plan de pruebas del proyecto tiene como objetivo asegurar que todas las funcionalidades principales y secundarias se comportan correctamente, tanto desde el punto de vista técnico como de la experiencia del usuario. Para ello se han predefinido diferentes tipos de pruebas que abarcan desde la verificación de requisitos funcionales hasta la evaluación de aspectos no funcionales como el rendimiento, la seguridad y la accesibilidad.

En primer lugar, se realizarán pruebas funcionales para comprobar procesos clave, como el registro y autenticación de usuarios, la gestión de los eventos, la mensajería en tiempo real y la personalización de la aplicación, funcionan según lo esperado. Estas pruebas incluirán la creación, edición y eliminación de eventos, la gestión de asistentes, la configuración de recordatorios y notificaciones, así como la interacción con el agente artificial para la gestión inteligente de eventos. También se verificará la correcta integración con servicios externos como Firebase y la sincronización de datos entre dispositivos.

Además, se llevarán a cabo pruebas de integración para garantizar que los distintos módulos de la aplicación colaboran de forma adecuada, especialmente en lo relativo a la comunicación con la base de datos, el almacenamiento de archivos y la gestión de notificaciones. Se comprobará que los datos fluyen correctamente entre las diferentes capas y que no existen pérdidas ni inconsistencias.

Las pruebas de usabilidad y accesibilidad serán fundamentales para asegurar que la aplicación es fácil de usar para cualquier persona, independientemente de su edad o experiencia tecnológica. Se evaluará la claridad de la interfaz, la facilidad de navegación, la comprensión de los mensajes y la adaptación a diferentes dispositivos y tamaños de pantalla. También e comprobará la compatibilidad con lectores de pantalla y otras ayudas técnicas.

Por último, se realizarán pruebas no funcionales para medir el rendimiento de la aplicación, el tiempo de respuesta entre acciones del usuario, el consumo de recursos y la robustez ante posibles fallos o desconexiones. Se pondrá especial atención en la seguridad de los datos personales y en la protección de la privacidad del usuario.

El éxito del plan de pruebas se medirá en función de que todas las funcionalidades principales estén libres de errores críticos, la experiencia del usuario sea satisfactoria y la aplicación mantenga un rendimiento y una seguridad adecuados en todos los dispositivos y plataformas soportadas.

### Ejemplos de casos de prueba

Creación de evento. El usuario accede al formulario de creación de nuevo evento, introduce los datos requeridos y guarda. Se comprueba que el evento aparece correctamente en el calendario y en la base de datos.

Edición de evento. El usuario selecciona un evento existente, modifica algún campo y guarda los cambios. Se verifica que la información se actualiza en todas las vistas y en la base de datos.

Eliminación de evento. El usuario elimina un evento y se comprueba que desaparece del calendario y de la base de datos.

Personalización de tema. El usuario accede a la configuración, cambia el color principal de la aplicación y verifica que el cambio se aplica instantáneamente en toda la interfaz.

Recepción de notificaciones. Se configura un recordatorio para un evento y se comprueba que la notificación llega antes de la fecha y hora de cumplimiento del evento, según la antelación establecida.

Accesibilidad. Se navega por la app utilizando un lector de pantalla y se comprueba que todos los elementos sean accesibles y comprensibles.

Interacción con el agente artificial. El usuario solicita mediante lenguaje natural la creación de un evento al agente artificial. Se verifica que el agente interpreta correctamente la orden y realiza la acción solicitada.

Pruebas multiplataforma. Se instala y utiliza la aplicación en Android, iOS y escritorio, comprobando que la experiencia es consistente y que no hay errores visuales ni funcionales.

# IMPLEMENTACIÓN DE LA APLICACIÓN

## Entorno de desarrollo

El desarrollo e la aplicación se ha realizado utilizando Flutter, un framework open source de Google que permite crear aplicaciones multiplataforma con una única base de código en Dart. El entorno de desarrollo principal ha sido Visual Studio Code, aprovechando su integración con herramientas de depuración, control de versiones y extensiones específicas para Flutter y Dart.

Para la gestión de dependencias y la compilación, se ha utlizado el sistema de paquetes de Flutter (pubsec.yaml), permitiendo la integración sencilla de librerías externas como Firebase, paquetes de internacionalización, gestión de estado y utilidades de UI. El proyecto está configurado para soportar desarrollo y pruebas en Android, iOS, web Y escritorio, facilitando la validación multiplataforma desde etapas tempranas.

Se ha empleado Firebase como backend principal, integrando servicios de autenticación, base de datos en tiempo real (Firestore), almacenamiento de archivos y notificaciones push. La configuración de Firebase se gestiona mediante el archivo firebase\_options.dart, generado automáticamente para cada plataforma.

El control de versiones se realiza con Git, permitiendo la colaboración y el seguimiento de cambios en el código fuente. Además, se han definido reglas de análisis estático y buenas prácticas mediante el archivo análisis\_options.yaml, asegurando la calidad y consistencia del código.

El entorno de desarrollo está preparado para facilitar la escalabilidad, la colaboración y la integración continua, permitiendo un ciclo de desarrollo ágil y eficiente.

### Requisitos previos en el entorno de desarrollo

1. Instalar Flutter y Dart. Descarga e instala el SDK de Flutter. Asegúrate de que Dart esté incluido y correctamente configurado en tu PATH.
2. Instala VS Code.
3. Instala las extensiones recomendadas en VS Code:

* Flutter (by Dart-Code)
* Dart (by Dart-Code)
* GitLens (opcional, para control de versiones)

1. Abre el proyecto en VS Code. Selecciona la carpeta /eventify/ como directorio de trabajo
2. Instala las dependencias del proyecto. Ejecuta en la terminal:

* flutter pub get

1. Configura un emulador o conecta un dispositivo físico.

* Para Android usa Android Studio.
* Para iOS usa Xcode en macOS.
* Para web asegúrate de tener Chrome o Edge instalado.
* Para escritorio verifica los requisitos de Flutter para Windows, macOS o GNU/Linux.

1. Ejecuta la aplicación. Ejecuta en la terminal

* flutter run

# ELEMENTOS DESTACABLES DEL DESARROLLO

## Retos y problemas afrontados

Uno de los principales retos durante el desarrollo de la aplicación ha sido el aprendizaje y la adaptación a nuevas tecnologías como Flutter y Dary. Al tratarse de un framework y lenguaje diferentes a los aprendidos en desarrollo móvil, fue necesario invertir tiempo en comprender su filosofía, la estructura de los proyectos, la gestión de estados y la integración de widgets personalizados. Este proceso implicó familiarizarse con conceptos como el hot reload, la composición de interfaces mediante widgets (similar al uso de fragmentos visto en la asignatura de Móviles) y la gestión eficiente de recursos en una aplicación multiplataforma.

Otro desafío importante fue la integración de Firebase como backend principal. No solo fue necesario aprender a con figurar los servicios de autenticación, almacenamiento y notificaciones push, sino que también supuso el primer contacto con una base de datos en forma de documento como Firestore. Acostumbrado a modelos relacional, trabajar con una base de datos NoSQL basada en documentos y colecciones requirió un cambio de mentalidad, especialmente a la hora de estructurar los datos, definir las relaciones y optimizar las consultas. Aprender a manejar la estructura de datos en Firestore, definir reglas de acceso y gestionar la sincronización en tiempo real supuso un avance significativo en el dominio de las tecnologías cloud modernas.

Además, surgieron complicaciones relacionadas con la adaptación de la interfaz a diferentes tamaños y resoluciones de pantalla. Muchos elementos y diseños que se visualizaban correctamente en dispositivos móviles presentaban problemas al ejecutar en escritorio y viceversa, como desalineaciones, escalados incorrectos o espacios vacíos poco estéticos. Esto obligó a revisar y ajustar los layouts, emplear widgets responsivos y realizar pruebas exhaustivas en distintas plataformas para garantizar la experiencia de usuario coherente y agradable en todos los dispositivos.

Otro reto también fue la gestión de las localizaciones y traducciones. Durante las primeras fases del desarrollo, Flutter generaba las dependencias de localización en ubicaciones incorrectas, lo que provocaba errores al intentar acceder a los recursos traducidos desde distintas partes de la aplicación. Fue necesario reorganizar la estructura de archivos de localización, ajustar la configuración del proyecto y revisar la integración de las dependencias para asegurar que la internacionalización funcionara correctamente en todos los idiomas soportados.