C:\Users\alumno\Desktop\cropped-logo_ies-1-1.png

Memoria del Proyecto Final de Grado

# Autores:

# Yosef Guillermo Karam Müller

# Vladyslav Rosiyan

# Antonio Águila

# Tutor del proyecto:

# Alejandro de Acuña Jiménez

Eventia: Donde cada evento, importa

## Grado Superior de Desarrollo de Aplicaciones Multiplataforma

## Centro educativo: IES Alonso de Avellaneda

## Año académico: 2024 – 2025

# Índice

## Introducción: 7

* 1. Introducción en español 7
  2. Introducción en inglés 8

## Análisis del entorno: 9

* 1. Identificación de factores externos 9
  2. Investigación de la competencia 10
  3. Recopilación de información 10
  4. Análisis DAFO 11

## Solución propuesta: 13

* 1. Tecnologías usadas en el desarrollo 13
  2. Recursos necesarios 16

## Planificación temporal del desarrollo del proyecto 19

## Plan de mantenimiento de la aplicación 29

## Documentación del diseño e implementación de la solución: 33

* 1. Diseño de la base de datos 33
  2. Descripción modular del software 36
  3. Diseño lógico de la aplicación 43
  4. Diseño de interfaces 48
  5. Estudio de la seguridad de la aplicación 50

## Código fuente documentado 51

## Manual de configuración y funcionamiento de la aplicación 53

## Manual de usuario

## Bibliografía y fuentes de información

# Figuras y tablas

* Tabla 1\_Análisis\_Dafo 11
* Tabla 2\_Asignación\_y\_temporización\_de\_tareas 23
* Figura 1\_Esquema\_de\_módulos 41
* Figura 2\_Diagrama\_de\_representación 42

# Notaciones y convenciones

• Cuerpo del texto Georgia 11 puntos

• Interlineado 1,5 puntos

• Margen izquierdo 3,5 cm

• Margen derecho 3 cm

• Márgenes superior e inferior 2,5 cm

• Paginación en el pie de página

• Rótulos de capítulos, apartados y sub apartados Georgia con un máximo de 16 puntos

• Nuevos capítulos comienzan por página impar

# Introducción

## Introducción en español

## Eventia: Donde cada evento, importa

Nuestro proyecto, Eventia, aborda la creciente necesidad de una herramienta eficiente y versátil para la gestión de eventos de eventos entre amigos. La necesidad de un sistema que facilite y optimice cada aspecto del proceso es más relevante que nunca. Eventia, busca ofrecer una solución integral y accesible para todos los organizadores de eventos.

La gestión de eventos tradicionales es un proceso complejo y laborioso. Los organizadores enfrentan desafíos como la coordinación de múltiples aspectos logísticos, la comunicación efectiva con los asistentes y la gestión de inscripciones. Muchos de estos procesos se manejan utilizando diversas herramientas que no siempre se integran de forma fluida. Esta situación plantea la necesidad de una herramienta unificada que centralice y simplifique la organización de eventos, asegurando una experiencia satisfactoria.

Para desarrollar Eventia, se ha adoptado una metodología ágil que permite iteraciones que coinciden con las tutorías de seguimiento del proyecto y feedback continuo. El proceso de desarrollo se ha dividido en cada una de esas iteraciones, enfocándose en la implementación de funcionalidades específicas. Esta metodología no solo facilita la detección y corrección temprana de errores, sino que también asegura que la plataforma evolucione de acuerdo a las necesidades específicas de la aplicación.

Eventia promete facilitar la forma en que se gestionan los eventos. Se espera que la aplicación reduzca el tiempo y los recursos necesarios para organizar un evento, mejorando al mismo tiempo la calidad de la experiencia para los asistentes.

En resumen, Eventia se posiciona como una herramienta esencial para cualquier persona que quiera agilizar la organización de eventos, con su enfoque integral y su capacidad de adaptarse a diferentes tipos de eventos. A medida que la tecnología continúa avanzando, herramientas como Eventia, facilitan la vida de los organizadores, mejorando su calidad de vida y el tiempo necesario para la creación de eventos.

## Introducción en inglés

## Eventia: Where every event matters

Our project, Eventia, addresses the growing need for an efficient and versatile tool for managing events among friends. In a world where event organization ranges from small gatherings to large international conferences, the need for a system that eases and optimizes every aspect of the process is needed more than ever. Eventia seeks to fill that gap by offering a comprehensive and accessible solution for all event organizers.

Traditional event management is often a complex and laborious process. Organizers face challenges such as coordinating multiple logistical aspects, effective communication with attendees and managing registrations. Many of these processes are handled using various tools that do not always integrate seamlessly, resulting in inefficiencies and errors. This situation creates a need for a unified tool that centralizes and simplifies event organization, ensuring a more coherent and professional experience.

For the development of Eventia, an agile methodology has been adopted, allowing iterations that match with project follow-up tutorials and a continuous feedback. The development process has been divided into these iterations, focusing on the implementation of specific functionalities. This methodology not only facilitates early detection and correction of errors but also ensures that the platform evolves according to the specific needs of the application.

Eventia promises to make easier the way events are managed. The application goal is to reduce the time and resources needed to organize an event while improving the quality of the experience for attendees.

In summary, Eventia positions itself as an essential tool for anyone looking to streamline event organization, with its comprehensive approach and ability to adapt to different types of events. As technology continues to advance, tools like Eventia not only make the lives of organizers easier, but also improve the quality and efficiency of event creation.

# Análisis del entorno

## Identificación de factores externos

La identificación de factores externos es fundamental para poder ubicar nuestra aplicación en el mercado. Procedemos a detallar estos factores de manera desglosada.

• Económicos: El crecimiento económico y el poder adquisitivo de los usuarios pueden influir en el uso de Eventia. En tiempos de bonanza económica, es más probable que las personas participen en eventos y busquen herramientas para gestionarlos eficientemente. La competencia de aplicaciones similares también puede afectar al uso Eventia como solución integral.

• Sociales: Las tendencias sociales, como el aumento de la socialización virtual y la organización de eventos entre amigos, son clave para Eventia. La creciente preferencia por soluciones digitales para la gestión de eventos destaca la relevancia de nuestra aplicación en el mercado actual.

• Tecnológicos: Las nuevas tecnologías que se desarrollan en el mercado actual por las empresas más grandes pueden hacer que el diseño o implementaciones realizadas en Eventia queden desfasadas por técnicas modernas de reciente desarrollo, haciendo más importante fijarse en estas nuevas mejoras y actualizaciones para incluirlas en la aplicación lo más pronto posible.

• Ecológicos: Aunque el impacto ecológico de Eventia es relativamente bajo, es importante considerar prácticas sostenibles en el desarrollo y mantenimiento de la aplicación. La promoción de eventos ecológicos también puede ser una característica atractiva para los usuarios conscientes del medio ambiente.

• Legales: Aunque los datos que Eventia recoge de los usuarios no son personales, es esencial cumplir con las leyes de protección de datos y privacidad, como el Reglamento General de Protección de Datos (GDPR). Esto garantiza que la información de los usuarios esté segura y que la aplicación cumpla con las regulaciones vigentes, evitando posibles sanciones.

## Investigación de la competencia

En el competitivo mercado de las aplicaciones móviles para gestión de eventos, Eventia se enfrenta varios rivales que ofrecen un sistema muy similar, destacando algunos competidores como:

* TimeTree: Una aplicación multiplataforma de calendario compartido que permite a los usuarios crear y gestionar eventos en grupo, con funcionalidades de recordatorios y sincronización de agendas.
* Eventbrite: Una aplicación multiplataforma que web y móvil que facilita la creación, promoción y gestión de eventos. Incluye funciones para venta de entradas, gestión de asistentes y seguimiento en tiempo real de la asistencia.
* ColorNote: Una aplicación móvil de notas y planificación que permite a los usuarios organizar sus tareas y eventos de manera eficiente. Aunque no es exclusivamente para eventos, su flexibilidad y facilidad de uso la hacen una herramienta útil para la gestión de eventos pequeños y personales.

Estos competidores ofrecen funcionalidades robustas y una experiencia de usuario destacada, lo que subraya la importancia de diferenciar a Eventia a través de características únicas, ofreciendo un servicio rápido y eficaz.

## Recopilación de información

También se ha realizado una recopilación de información continua para poder saber en todo momento el rumbo que debía tomar nuestra aplicación. Esta recopilación se ha realizado de la siguiente manera:

• Consultar a Usuarios de otras aplicaciones similares: Hemos consultado a amigos, familiares, compañeros de trabajo, compañeros de prácticas y conocidos para saber qué requerimientos introducir en la aplicación durante todo el desarrollo de la aplicación.

• Estudio de competidores: realizado para identificar oportunidades y áreas de mejora

• Pruebas de Usabilidad: Realiza pruebas periódicas para asegurar que la interfaz de usuario sea intuitiva.

Con esta recopilación esperamos que Eventia pueda adaptarse mejor a las necesidades del mercado y pueda ofrecer una mejor experiencia a los usuarios.

## Análisis DAFO

Con los datos recogidos y analizados realizamos un análisis DAFO aplicado a nuestra solución:

Tabla 2\_Análisis\_Dafo

|  |  |
| --- | --- |
| Debilidades | Amenazas |
| * Falta de reconocimiento de la marca * Recursos limitados * Dependencia de los servicios de Google | * Alta competencia * Evolución rápida del mercado * Ataques cibernéticos * Cambios continuos de las tecnologías |
| Fortalezas | Oportunidades |
| * Interfaz intuitiva * Funcionalidades integradas * Flexibilidad en diferentes tipos de eventos * Accesibilidad en dispositivos móviles | * Crecimiento del mercado de eventos virtuales * Tendencias de personalización en Eventos * Expansión geográfica |

# Solución propuesta

## Tecnologías utilizadas en el desarrollo

## Android Studio

La plataforma de desarrollo integrada (IDE) oficial para Android.

Utilizada para el desarrollo, depuración y prueba de Eventia. Proporciona un entorno completo para programar en Android y facilita la integración de bibliotecas y servicios.

## Git

El sistema de control de versiones distribuido.

Permite el seguimiento de cambios en el código fuente y la colaboración entre desarrolladores. Facilita la gestión de versiones y la implementación de nuevas funcionalidades de manera ordenada.

## GitHub

Una plataforma basada en la web que utiliza Git para el control de versiones.

Utilizado para alojar el código fuente de Eventia, facilitar la colaboración entre el equipo de desarrollo y gestionar las distintas ramas de desarrollo. Además, permite la integración continua y la revisión de código.

## Java

El lenguaje principal utilizado para la implementación de la lógica de la aplicación en Android. Java permite escribir código eficiente y robusto.

## Firebase

La plataforma de desarrollo de aplicaciones móviles proporcionada por Google. Firebase facilita la gestión de usuarios y datos, mejorando la eficiencia y la seguridad. Implementada para varios servicios:

+ Autenticación con Authentication: Para gestionar el acceso de los usuarios de manera segura.

+ Base de datos en tiempo real con Realtime database: Para almacenar y sincronizar datos entre usuarios en tiempo real.

+ Almacenamiento en la nube: Para guardar archivos y documentos relacionados con los eventos y los usuarios.

## Gradle

Utilizado para la gestión de dependencias y la configuración del proyecto en Android Studio. Gradle simplifica el proceso de construcción y despliegue de la aplicación, garantizando la integración de todas las librerías y recursos necesarios.

## XML

Lenguaje de marcado utilizado para definir layouts y recursos gráficos en aplicaciones Android. Utilizado para crear las interfaces de usuario, layouts, y varios elementos gráficos, drawables, de la aplicación. Permite una separación clara entre la lógica de la aplicación y la presentación visual.

## Glide

Una biblioteca de terceros para la carga y gestión de imágenes. Facilita la carga, almacenamiento en caché y presentación de imágenes dentro de la aplicación.

JUnit y Espresso

Herramientas de pruebas para Java y Android. JUnit se ha utilizado para pruebas unitarias, mientras que Espresso se ha empleado para pruebas de la interfaz de usuario, asegurando que la aplicación funcione correctamente en diversas situaciones.

## Material Design

El lenguaje de diseño de Google. Utilizado para crear interfaces de usuario consistentes y atractivas que sigan las mejores prácticas de diseño de Android

## Applandeo: Calendar View

Una biblioteca de calendario de terceros. Utilizada para implementar calendarios interactivos y funcionales en la aplicación. La biblioteca Material-Calendar-View facilita la integración de componentes de calendario en Eventia.

## WorkManager

Una biblioteca de Android para la programación de tareas diferidas. Utilizada para gestionar las notificaciones de eventos, garantizando que se entreguen en el momento adecuado sin afectar el rendimiento de la aplicación.

## LucidChart

Una herramienta web para la creación de esquemas y diagramas. Utilizada para la creación de los diagramas y esquemas de distinto tipo, mostrando visualmente los elementos más importantes sobre el funcionamiento de la aplicación.

## Recursos necesarios

Para el desarrollo de Eventia, se han requerido varios recursos esenciales que abarcan desde hardware y software hasta recursos humanos e informativos para poder guiarnos y nutrir nuestro proyecto de las mejores ideas. Detallamos a continuación los recursos necesarios que hemos podido emplear durante el desarrollo:

* Recursos humanos:
  + Estudiantes desarrolladores con conocimientos Android:
    - Tras conversar por un post creado por el tutor de las prácticas nos juntamos un grupo tres personas para el desarrollo de una aplicación móvil.
  + Mentores y tutores:
    - Profesores y tutores que guíen el proceso de desarrollo y ofrezcan asesoramiento técnico y académico. En este caso, tanto el tutor del proyecto como el tutor de prácticas designado.
  + Consultores y personal de pruebas:
    - Hemos consultado a personas con experiencia en desarrollo de aplicaciones Java, además, hemos tenido la ayuda de personas ajenas al desarrollo para que prueben la aplicación y nos den su feedback.
* Recursos Hardware
  + Computadoras para realizar el desarrollo:
    - Computadoras con especificaciones adecuadas para poder Android Studio, así como sus emuladores. En este caso se han usado computadoras personales, así como equipos proporcionados para la realización de la Formación en Centros de Trabajo, FCT.
  + Dispositivos de prueba:
    - Smartphones y tablets con sistema operativo Android para probar la app en diferentes dispositivos y versiones del sistema operativo. Se han utilizado tanto dispositivos virtualizados en Android Studio como dispositivos físicos personales, móviles antiguos y actuales, así como una tableta con Android puro.
* Recursos de Software
  + Licencias de Software:
    - Android Studio: el programa de JetBrains basado en IntelliJ tiene licencia gratuita
    - Firebase: Nos adscribimos a las opciones gratuitas de Firebase para poder desarrollar nuestro proyecto
  + Herramientas de colaboración:
    - Git y GitHub: para poder llevar el control de versiones y colaborar entre los distintos miembros del equipo. Además, en GitHub se ha creado un proyecto en el que los miembros pueden ir registrando y asignando las tareas que van realizando o tienen por realizar.
* Otros recursos
  + Documentación y formación:
    - Acceso a recursos en línea, tutoriales y documentación que hemos usado durante la realización del grado superior y las FCT, así como mucho material novedoso que se ha investigado para poder usarlo en la aplicación.
  + Infraestructura física:
    - Espacios de trabajo físicos tanto de carácter personales, como los proporcionados por el centro educativo y por el centro de las FCT.

Estos recursos han sido fundamentales para asegurar el desarrollo de Eventia, garantizando que la aplicación funcione de manera eficiente y ofrezca una experiencia óptima a los usuarios, pudiendo cumplir en el proceso con los objetivos académicos del proyecto fin de grado.

# Planificación temporal del desarrollo del proyecto

Semana 1 (24 septiembre – 29 septiembre)

* Investigación y planificación:
  + Análisis de necesidades
  + Definición de los requisitos y especificaciones del proyecto
  + Creación del plan de proyecto y asignación de tareas

Semana 2 (30 septiembre – 2 de octubre)

* Diseño inicial:
  + Diseño de la arquitectura de la aplicación
  + Prototipos de la interfaz de usuario
* Desarrollo inicial:
  + Configuración del entorno de desarrollo:
    - Android Studio
    - Git
    - Github
  + Desarrollo de la estructura básica de la aplicación:
  + Implementación del diseño de interfaz de la aplicación
  + Creación de una memoria provisional para hacer un seguimiento de las tareas y las tecnologías usadas durante el proyecto.

Semana 3 (7 octubre – 13 octubre)

* Migrar proyecto a otro repositorio debido a que un compañero debe abandonar el proyecto
* Desarrollo avanzado de la aplicación:
  + Creación de los distintos layouts principales de la aplicación
  + Creación del sistema de navegación entre las distintas pantallas
  + Definir un sistema de amigos para la aplicación y su lógica
  + Adaptar aplicación a las especificaciones indicadas por tutor de prácticas

Semana 4 (14 octubre – 20 octubre)

* Crear ficheros para internacionalización de la aplicación y traducción de todas las cadenas de texto.
* Creación de Firebase y suscripción al servicio Firebase Authenticator
* Creación del Login y el Sign up de la aplicación y enlazarla con Firebase

Semana 5 (21 octubre – 27 octubre)

* Realización del sistema de amigos linkeado con base de datos
* Creación de base de datos en Realtime Database
* Adaptación de los elementos al estilo visual de la app
* Realización del sistema de notas

Semana 6 (28 octubre – 3 noviembre)

* Incluir app bar a las actividades que lo necesiten
* Completa implementación del sistema de notas
* Creación de la visualización y personalización de los datos del usuario
* Creación y suscripción a Firebase CloudStorage
* Añadir Glide como sistema para cargar imágenes en línea

Semana 7 (4 noviembre – 10 noviembre)

* Finalización del sistema de amigos viendo sus datos y los eventos en los que coincide con el usuario
* Creación de los eventos, la forma de crearlos y visualizarlos en la app y toda su estructura en Firebase Database
* Añadir función login recuérdame parar inicio de sesión automático

Semana 8 (11 noviembre – 17 noviembre)

* Realización del sistema para invitar amigos a los eventos, salirse de los eventos y eliminarlos.
* Pequeños cambios visuales para adaptar las nuevas funcionalidades al aspecto visual de la app
* Ordenación de todas las clases en paquetes específicos para cada una de las vistas principales

Semana 9 (18 noviembre – 24 noviembre)

* Desarrollo de la memoria con los datos ya recogidos e indicados durante el desarrollo de la app

Tabla 2\_Asignación\_y\_temporización\_de\_tareas

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tarea | Asignación | Estatus | Prioridad | Tam. | Inicio | Fin |
| Crear las bases de la app y todo lo necesario para poder navegarla | Diroween | Realizado | Alta | L | 28-oct. | 2-oct. |
| Creación de una pantalla principal con botones en el inferior | Diroween | Realizado | Alta | M | 2-oct. | 3-oct. |
| Creación de una SplashScreen | Diroween | Realizado | Media | M | 3-oct. | 6-oct. |
| Los botones deben cambiar su color al seleccionarlos | Diroween | Realizado | Baja | S | 3-oct. | 6-oct. |
| Crear base de datos Firebase | Diroween | Realizado | Muy alta | M | 17-oct. | 19-oct. |
| Creación de las notas y edición de estas | Diroween | Realizado | Media | M | 9-oct. | 20-oct. |
| Creación de signup linkeado a bdd | Diroween | Realizado | Alta | L | 17-oct. | 20-oct. |
| Los amigos deben aparecer en una lista del FriendsFragment | Diroween | Realizado | Media | M | 21-oct. | 25-oct. |
| Los amigos tienen que poder mandar peticiones, aceptarlas y rechazarlas | Diroween | Realizado | Media | L | 21-oct. | 27-oct. |
| Los amigos tienen que poder buscarse en la app | Diroween | Realizado | Media | L | 24-oct. | 27-oct. |
| Las notas deben poder eliminarse con un swipe | Diroween | Realizado | Baja | S | 29-oct. | 29-oct. |
| Creación del sistema de amigos | Diroween | Realizado | Muy alta | XL | 21-oct. | 29-oct. |
| Crear y linkear servicio en la nube para subir ficheros | Diroween | Realizado | Muy alta | M | 30-oct. | 30-oct. |
| Los usuarios deben tener una foto de perfil y tienen que poder subirla | Diroween | Realizado | Media | M | 30-oct. | 30-oct. |
| Unificar el estilo visual de todas las activities | Diroween | Realizado | Baja | L | 16-oct. | 31-oct. |
| Añadir la appbar a todas las activities que lo necesiten | Diroween | Realizado | Baja | M | 30-oct. | 31-oct. |
| Poder ver las imágenes de usuario en la pestaña de amigos y en la ventana de búsqueda | Diroween | Realizado | Media | M | 31-oct. | 31-oct. |
| En una activity se tienen que mostrar los datos de cada amigo al pulsar en él | Diroween | Realizado | Media | L | 31-oct. | 3-nov. |
| Hacer que en el signup se vuelva a la pantalla de login | Diroween | Realizado | Baja | XS | 3-nov. | 3-nov. |
| Poner al usuario en la appbar y que pueda ver sus datos y hacer logout | Diroween | Realizado | Media | L | 31-oct. | 4-nov. |
| Dentro de la activity de mostrar los datos de los amigos hay que añadir una forma de eliminarlos | Diroween | Realizado | Baja | M | 4-nov. | 4-nov. |
| Hacer que en cada ventana de amigo se muestren los eventos del amigo que coincidan con los del usuario | Diroween | Realizado | Alta | L | 4-nov. | 5-nov. |
| Cada evento tiene que tener asignada una imagen que los usuarios puedan subir | Diroween | Realizado | Baja | M | 6-nov. | 7-nov. |
| Añadir al login un recuérdame para poder logearse una única vez | Diroween | Realizado | Muy baja | S | 9-nov. | 9-nov. |
| Crear una clase eventos y la forma de interactuar con ellos como usuario | Diroween | Realizado | Muy alta | XL | 3-nov. | 13-nov. |
| Dentro de cada evento creado, poder invitar a un amigo al evento | Diroween | Realizado | Alta | L | 13-nov. | 14-nov. |
| Que el usuario pueda salirse de un evento y si es la última persona registrada eliminarlo de la bdd | Diroween | Realizado | Baja | S | 15-nov. | 15-nov. |
| Hacer que se actualicen los eventos en el main cuando se crea o elimina un evento | Diroween | Realizado | Baja | S | 16-nov. | 16-nov. |
| Escribir un resumen de la app en inglés y español | Diroween | Realizado | Alta | M | 18-nov. | 18-nov. |
| Creación de un login linkeado a bdd | Diroween | Realizado | Muy alta | L | 17-nov. | 19-nov. |
| Mejora - Pedir confirmación activa de un usuario para salir del evento | Diroween | Realizado | Muy baja | XS | 20-nov. | 20-nov. |
| Realizar fichero bajo especificaciones de la memoria e indicar los puntos a detallar | Diroween | Realizado | Alta | S | 20-nov. | 20-nov. |
| Hacer que en la pestaña con el icono del sol, que representa el día actual, se muestren los eventos que tiene para ese día el usuario | GitAguila | En progreso | Muy alta | L | 30-oct. |  |
| Realizar sistema de notificaciones con WorkManager | reinnuS | En progreso | Media | L | 24-oct. |  |
| Realizar la memoria de la aplicación | Diroween | En progreso | Muy alta | XL | 16-nov. |  |
| Aplicar a la memoria todas las especificaciones del proyecto | Diroween | En progreso | Muy alta | L | 18-nov. |  |
| Cambiar el background de los TextViews en EventViewer a esquinas redondeadas en lugar de cuadradas | GitAguila | Asignadas | Muy baja | XS |  |  |
| Editar datos del evento. | GitAguila | Asignadas | Baja | M |  |  |
| Hacer que una vez se termine un evento, quede registrado en un histórico | reinnuS | Asignadas | Muy alta | M |  |  |
| Los usuarios deben tener un rol asignado para los eventos. El creador debe tener rol de admin y tiene que poder asignar ese rol a otros usuarios mediante un menú contextual. | GitAguila | Asignadas | Media | M |  |  |
| Poder consultar el histórico de eventos | reinnuS | Asignadas | Alta | M |  |  |
| Si se almacena en la BDD el formato LocalDate o LocalZoneDate para las fechas en lugar del formato yyyy-MM-dd, hay que modificar todas las clases que recogen la fecha para que funcionen según este nuevo formato |  | Propuestas | Alta | L |  |  |
| Un usuario con rol admin debe poder eliminar a usuarios del evento desde el menú contextual | GitAguila | Asignadas | Baja | S |  |  |
| Pruebas de integración |  | Para hacer | Media | M |  |  |
| Pruebas de rendimiento |  | Para hacer | Media | M |  |  |
| Pruebas funcionales |  | Para hacer | Media | M |  |  |
| Pruebas unitarias |  | Para hacer | Media | M |  |  |
| Realizar los diagramas necesarios de la aplicación | Diroween | Asignadas | Muy alta | XL |  |  |
| Generar aspecto visual de la app | Diroween, GitAguila, reinnuS | En progreso | Media | L | 1-oct. |  |
| Crear una lista de tareas dentro de cada evento y que se muestren en la pestaña de hoy |  | Propuestas | Alta | L |  |  |
| Iconos de calendario personalizados |  | Propuestas | Media | M |  |  |
| Probar sistema de notificaciones con AlarmManager |  | Propuestas | Media | L |  |  |

# Plan de mantenimiento de la aplicación

Un plan de mantenimiento es fundamental para asegurar el correcto funcionamiento de la aplicación y la buena evolución a lo largo del tiempo.

Nuestro plan de mantenimiento para la aplicación consta de varios tipos de mantenimiento, con una temporización periódica asignada. Procedemos a detallarlos:

## • Mantenimiento correctivo

Enfocado en la solución de errores que se presenten en la aplicación. Sus actividades son:

* Monitoreo continuo de la aplicación para identificar y registrar errores.
* Priorizar y corregir errores críticos que afecten la funcionalidad básica de la aplicación.
* Lanzamiento de actualizaciones para solucionar errores menores.

## • Mantenimiento preventivo

Enfocado en la prevención de problemas futuros de la actividad realizando un mantenimiento proactivo de la aplicación. Sus actividades son:

* Revisión periódica del código para identificar posibles puntos de fallo y áreas de mejora.
* Realización de pruebas de seguridad para proteger los datos de los usuarios.
* Creación y actualización de documentación técnica para facilitar el mantenimiento futuro.

## • Mantenimiento adaptativo

Enfocado en la adaptación de la aplicación a cambios en el entorno de software y hardware. Sus actividades son:

* Actualización de la aplicación para ser compatible con nuevas versiones del sistema operativo Android.
* Adaptación a nuevas tecnologías y dispositivos emergentes.
* Ajustes en la integración con servicios de terceros, como Firebase, para asegurar su compatibilidad y funcionamiento.

## • Mantenimiento perfectivo

Enfocado en la mejora de la aplicación para optimizar su rendimiento y añadir nuevas funcionalidades. Sus actividades son:

* Optimización del código para mejorar la velocidad y la eficiencia de la aplicación.
* Implementación de nuevas características y mejoras basadas en el feedback de los usuarios.
* Realización de pruebas de rendimiento y ajustes para asegurar una experiencia de usuario fluida.

Además de estos mantenimientos fundamentales, también se van a realizar tareas necesarias, en las que son imprescindibles la participación del usuario final, que no son un mantenimiento como tal, pero son necesarias para poder llevar a cabo el plan de mantenimiento:

## • Soporte y actualización continua

Esta tarea provee de soporte técnico y lanzamiento de actualizaciones regulares. Consta de:

* Establecimiento de canales de comunicación para que los usuarios puedan reportar problemas y sugerencias.
* Programación de actualizaciones regulares que incluyan mejoras, correcciones y nuevas funciones.
* Monitoreo del rendimiento de la aplicación y recolección de estadísticas de uso para informar futuras mejoras.

## • Gestión del feedback de los usuarios

Esta tarea se basa en la recopilación y análisis del feedback de los usuarios para guiar las futuras mejoras a implementar. Consta de:

* Implementación de mecanismos dentro de la aplicación para que los usuarios puedan proporcionar retroalimentación.
* Análisis de las reseñas y comentarios de los usuarios en plataformas como Google Play Store.
* Realización de encuestas y entrevistas periódicas para obtener la perspectiva personal de los usuarios.

La temporización del plan de mantenimiento será la siguiente:

|  |  |
| --- | --- |
| Tipo de Mantenimiento | Frecuencia |
| Correctivo | Continuo, según se identifiquen errores |
| Adaptativo | Cada nueva versión del sistema operativo o tecnología |
| Perfectivo | Trimestral, basado en feedback y análisis de rendimiento |
| Preventivo | Semestral, para revisiones y pruebas de seguridad |
| Soporte y Actualización | Mensual, con actualizaciones menores y mejoras continuas |
| Gestión del Feedback | Continuo, con revisiones mensuales del feedback |

Este plan de mantenimiento asegura que Eventia se mantenga actualizada, funcional y relevante para los usuarios, garantizando una experiencia de alta calidad y una aplicación robusta y segura.

# Documentación del diseño e implementación de la solución

## Diseño de la base de datos

Para Eventia, el diseño de la base de datos es fundamental para asegurar una gestión eficiente de la información relacionada con los eventos y sus usuarios. La base de datos se ha estructurado utilizando Firebase Realtime Database, lo que permite sincronizar datos en tiempo real y mantenerlos actualizados en todos los dispositivos. A continuación, detallamos las entidades principales y sus relaciones:

## **User**

### **• Atributos**:

### id: Identificador único del usuario.

### email: Correo electrónico del usuario.

### friend\_requests: Solicitudes de amistad pendientes:

* + from: identificador del usuario que envió la solicitud.
  + name: nombre del usuario que envió la solicitud.
  + status: estado de la solicitud.
* friends: Lista de amigos del usuario:
  + id: identificador del amigo.
  + name: nombre del amigo.
* eventsRequests: Solicitudes de participación en eventos:
  + id: identificador del evento.
  + status: estado de la solicitud.

### imageUrl: dirección de la imagen de perfil del usuario.

### name: Nombre del usuario.

### • Relaciones:

* Un usuario puede tener múltiples amigos y solicitudes de amistad
* Un usuario puede tener múltiples solicitudes de participación en eventos.

## Event

### • Atributos:

* id: Identificador único del evento.
* date: Fecha del evento.
* hour: Hora del evento.
* image: URL de la imagen del evento.
* name: Nombre del evento.
* place: Lugar del evento.
* registeredUsers: Lista de usuarios registrados.
* user\_id: Identificador del usuario.

### • Relaciones:

* Un evento puede tener múltiples usuarios registrados.
* Cada usuario registrado tiene un identificador y un estado booleano indicando si está registrado.

Este diseño permite una gestión eficiente y escalable de los datos, sincronizando cambios en tiempo real en todos los dispositivos conectados. Además, garantiza la seguridad, haciendo que las contraseñas de los usuarios se guarden exclusivamente en Firebase Authenticator, estando cifradas y no teniendo registro de estas en la base de datos. Por último, el diseño modular de la base de datos permite añadir nuevas funcionalidades y expandir las capacidades de la aplicación sin afectar la integridad de los datos existentes.

Añadimos a continuación la estructura simplificada de los elementos antes detallados en formato JSON, que es la forma en la que se guardan los datos en Firebase Database:

### User

{

"user": {

"f6wh9OySdsMNQNBOF4sl8kyqY0L2": {

"email": "prueba27@prueba.com",

"friend\_requests": {

"9Doxg4fhFjQToaYID7B9gjNtZvB3": {

"from": "9Doxg4fhFjQToaYID7B9gjNtZvB3",

"name": "test01",

"status": "pending"

}

},

"friends": {

"POcGQ3sf7tT1nEKGPZu755bMSWg1": {

"id": "POcGQ3sf7tT1nEKGPZu755bMSWg1",

"name": "Prueba32"

},

"fPrZlzKKS4PdDCGZwUU2FU5EEts2": {

"id": "fPrZlzKKS4PdDCGZwUU2FU5EEts2",

"name": "prueba18"

},

"glV48Rb3z6bpsCfHQhQlgfG5JyV2": {

"id": "glV48Rb3z6bpsCfHQhQlgfG5JyV2",

"name": "prueba34"

},

"mU3mvFUJ3nfus9VGfgIpGlrXcRg2": {

"id": "mU3mvFUJ3nfus9VGfgIpGlrXcRg2",

"name": "prueba20"

}

},

"eventsRequests": {

"-OB81boo-A0IANAg1L6Z": "pending",

"-OBAm2sTMKbWNePK9bVp": "pending",

"-OB\_a3tGEERLK5XRrSYi": "pending"

},

"id": "f6wh9OySdsMNQNBOF4sl8kyqY0L2",

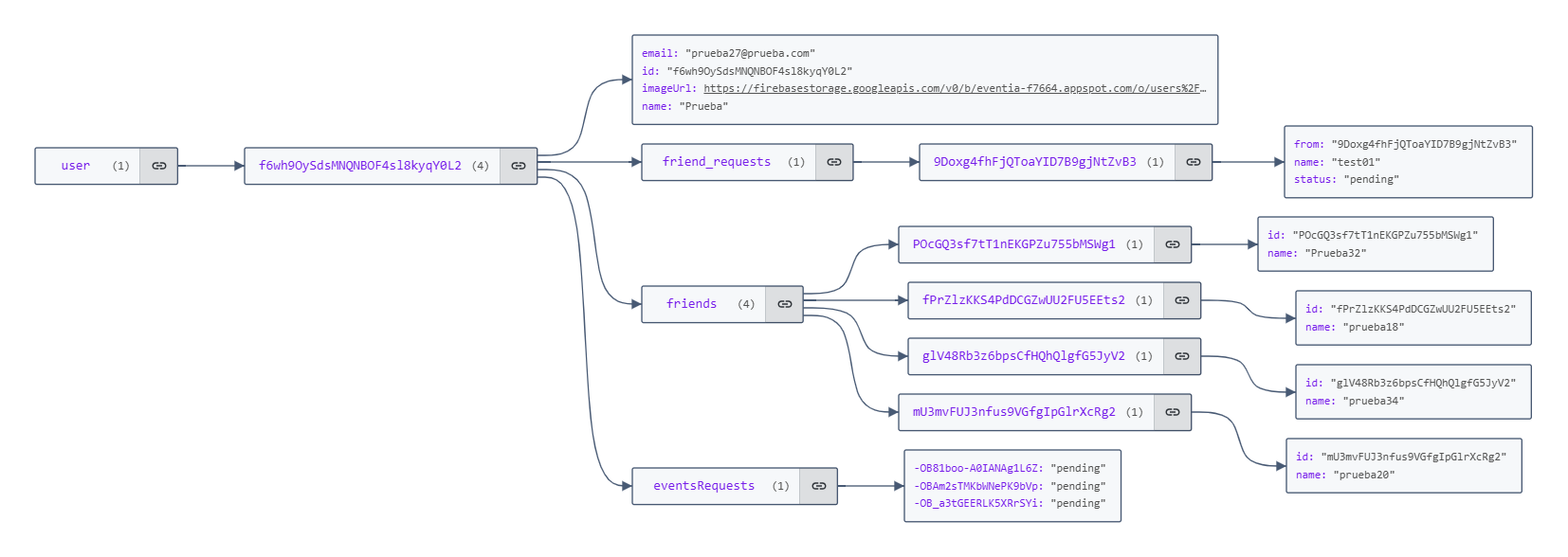
"imageUrl": "https://firebasestorage.googleapis.com/v0/b/eventia-f7664.appspot.com/o/users%2Ff6wh9OySdsMNQNBOF4sl8kyqY0L2%2Fprofile.jpg?alt=media&token=49f0889f-d71b-4e38-981a-02067ccd2e4a",

"name": "Prueba"

}

}

}



### Event

{

"event": {

"-OBC6zEztUFcy0hXpIkA": {

"date": "2024-11-15",

"hour": "14:15",

"id": "-OBC6zEztUFcy0hXpIkA",

"image": "https://firebasestorage.googleapis.com/v0/b/eventia-f7664.appspot.com/o/events%2F-OBC6zEztUFcy0hXpIkA%2FeventImage.jpg?alt=media&token=94583dac-ea95-4ea0-8b1d-7a39e5508a91",

"name": "chantaje vehiculo",

"place": "la fabrica",

"registeredUsers": {

"POcGQ3sf7tT1nEKGPZu755bMSWg1": true,

"RSDebFFtJdZ4POELvsOrTYJEp7H3": true,

"f6wh9OySdsMNQNBOF4sl8kyqY0L2": true,

"fPrZlzKKS4PdDCGZwUU2FU5EEts2": true,

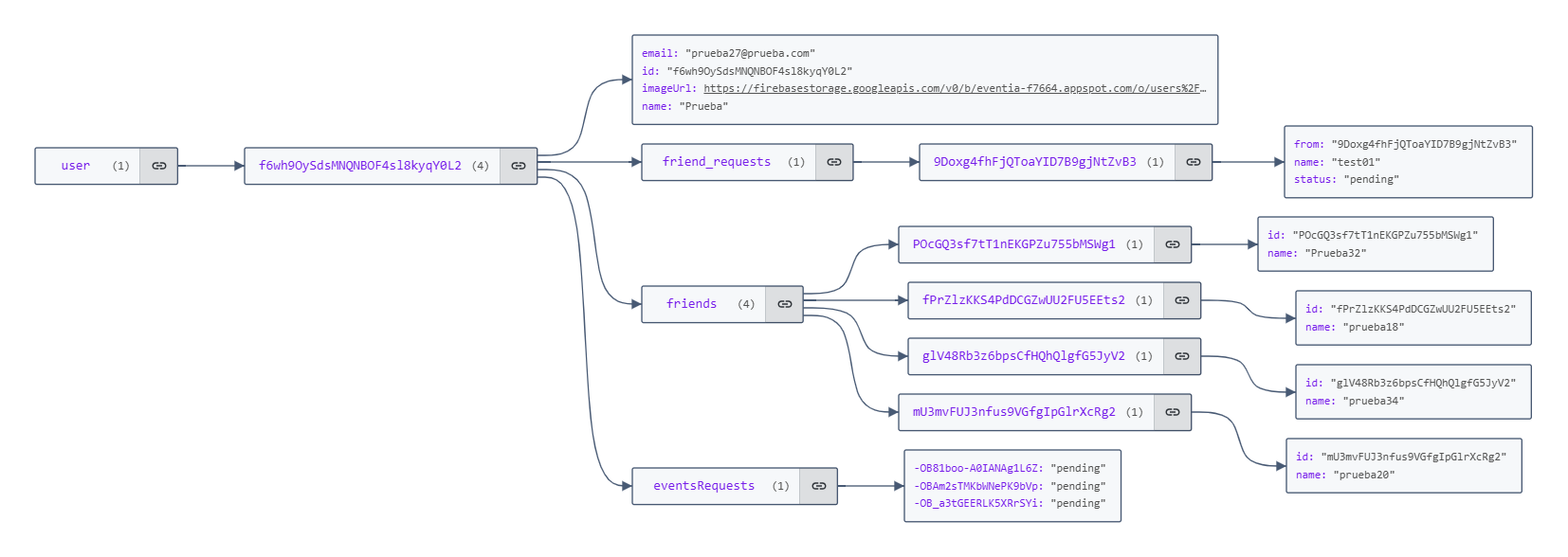
"mU3mvFUJ3nfus9VGfgIpGlrXcRg2": true

}

}

}

}



## Descripción modular del software

Eventia se ha desarrollado utilizando un enfoque modular para facilitar su gestión, mantenimiento y escalabilidad. A continuación, se describimos los principales módulos, representados por las carpetas de código, y sus funcionalidades:

### 1. Módulo de Autenticación (authentication)

Gestiona el registro y autenticación de usuarios. Lo conforman los ficheros:

+ LoginActivity: Actividad para el inicio de sesión de usuarios.

+ SignupActivity: Actividad para el registro de nuevos usuarios.

Sus funcionalidades son:

* Registro de nuevos usuarios.
* Inicio de sesión de usuarios existentes.

La tecnología usada para este módulo ha sido Firebase Authentication, que asegura la seguridad de los datos de inicio de los usuarios.

### Módulo de Calendario (calendar)

Administra la visualización y gestión de eventos en formato de calendario. Lo conforman los ficheros:

* CalendarFragment: Fragmento para mostrar el calendario de eventos.
* CalendarFragmentAdapter: Adaptador para gestionar la visualización de eventos en el calendario.
* CalendarFragmentViewHolder: ViewHolder para los elementos del calendario.

Sus funcionalidades son:

* Visualización de eventos en una vista de calendario.
* Gestión de eventos desde el calendario.

La tecnología utilizada para este módulo ha sido:

* Applandeo Material Calendar View, que ofrece un calendario personalizable con iconos
* Glide para mostrar las imágenes de los eventos

### Módulo de Eventos (event)

Maneja la creación, edición y visualización de eventos. Lo conforman los ficheros:

* Event: Clase modelo para eventos.
* EventCreationActivity: Actividad para la creación de nuevos eventos.
* EventEditorActivity: Actividad para la edición de eventos existentes.
* EventInviterActivity: Actividad para invitar amigos a eventos.
* EventListAdapter: Adaptador para la lista de eventos.
* EventListViewHolder: ViewHolder para los elementos de la lista de eventos.
* EventRequest: Clase modelo para solicitudes de eventos.
* EventRequestAdapter: Adaptador para solicitudes de eventos.
* EventRequestsActivity: Actividad para gestionar las solicitudes de eventos.
* EventRequestViewHolder: ViewHolder para los elementos de las solicitudes de eventos.
* EventViewerActivity: Actividad para ver los detalles de un evento.

Sus funcionalidades son:

* Creación de nuevos eventos.
* Edición y eliminación de eventos existentes.
* Visualización de eventos próximos y pasados.
* Gestión de solicitudes de eventos.

Las tecnologías utilizadas para este módulo han sido:

* Firebase Realtime Database, para almacenar los datos de los eventos y los usuarios registrados en ellos
* Firebase Cloud Storage: para almacenar imágenes de eventos.
* Glide: para mostrar las imágenes de los eventos.

### Módulo de Amigos (friend)

Gestiona la lista de amigos y las solicitudes de amistad de los usuarios. Lo conforman los ficheros:

* FriendEventListFragment: Fragmento para mostrar los eventos de los amigos.
* FriendInfoPageFragment: Fragmento para mostrar la información del amigo.
* FriendInviterAdapter: Adaptador para invitar amigos a eventos.
* FriendRequest: Clase modelo para solicitudes de amistad.
* FriendRequestActivity: Actividad para gestionar solicitudes de amistad.
* FriendRequestAdapter: Adaptador para las solicitudes de amistad.
* FriendsAdapter: Adaptador para la lista de amigos.
* FriendSearcherActivity: Actividad para buscar amigos.
* FriendSearcherAdapter: Adaptador para la búsqueda de amigos.
* FriendSearcherViewHolder: ViewHolder para los resultados de la búsqueda de amigos.
* FriendsFragment: Fragmento para la lista de amigos.
* FriendsRequestViewHolder: ViewHolder para las solicitudes de amistad.
* FriendsViewerPageAdapter: Adaptador para la visualización de amigos.
* FriendsViewHolder: ViewHolder para los elementos de la lista de amigos.
* FriendViewerActivity: Actividad para ver la información de un amigo.

Sus funcionalidades son:

* Envío y aceptación de solicitudes de amistad.
* Visualización de la lista de amigos.
* Gestión de eventos de amigos.

Las tecnologías utilizadas para este módulo son:

* Firebase Realtime Database: para guardar los datos de los usuarios y sus relaciones.
* Glide: para mostrar las imágenes de los usuarios.

### Módulo de Notas (note)

Permite a los usuarios crear y gestionar notas rápidas personales. Lo conforman los ficheros:

* Note: Clase modelo para las notas.
* NotesActivity: Actividad para gestionar las notas.
* NotesAdapter: Adaptador para la lista de notas.
* NotesFragment: Fragmento para mostrar las notas.
* NotesViewHolder: ViewHolder para los elementos de la lista de notas.

Sus funcionalidades son:

* Creación de notas.
* Edición y eliminación de notas.

La tecnología usada en este módulo es Shared Preferences del propio Android, que permite guardar las notas por cada usuario y recuperarlas para mostrarlas.

### Módulo de Hoy (today)

Proporciona una vista de los eventos y actividades del día actual. Lo conforman los ficheros:

* TodayFragment: Fragmento para mostrar los eventos y actividades del día.

Sus funcionalidades son:

* Visualización de eventos del día.
* Resumen de actividades y notificaciones relevantes para el día.

La tecnología utilizada en este módulo es Firebase Realtime Database, para poder recuperar la información de los eventos del día actual.

### Módulo de Gestión de Usuarios (user)

Administra la información de los usuarios y sus configuraciones. Lo conforman los ficheros:

* User: Clase modelo para el usuario.
* UserSettings: Configuraciones del usuario.

Sus funcionalidades son:

* Gestión de perfiles de usuario (nombre, email, imagen de perfil).
* Configuraciones del usuario.

Las tecnologías utilizadas para este módulo han sido:

* Firebase Realtime Database.
* Firebase Cloud Storage: para almacenar imágenes de los usuarios.
* Firebase Authentication: para almacenar la y recuperar la información del usuario
* Glide: para mostrar las imágenes de los usuarios.

### Módulo Principal y de Inicialización

Conformado por un fichero cada uno, contenidos cada uno en la raíz del proyecto:

* MainActivity: Actividad principal de la aplicación.
* Splash: Actividad de pantalla de bienvenida, splash screen, desde la que se puede hacer Login automáticamente, si el usuario eligió ser recordado

Sus funcionalidades son:

* Contener los elementos y clases principales de la aplicación
* Dar un aspecto personalizado al inicio de sesión
* Hacer un login automático para usuarios previamente recordados

La tecnología usada en este módulo es Firebase Authentication, para poder realizar el inicio de sesión automático.

Figura 1\_Esquema\_de\_módulos

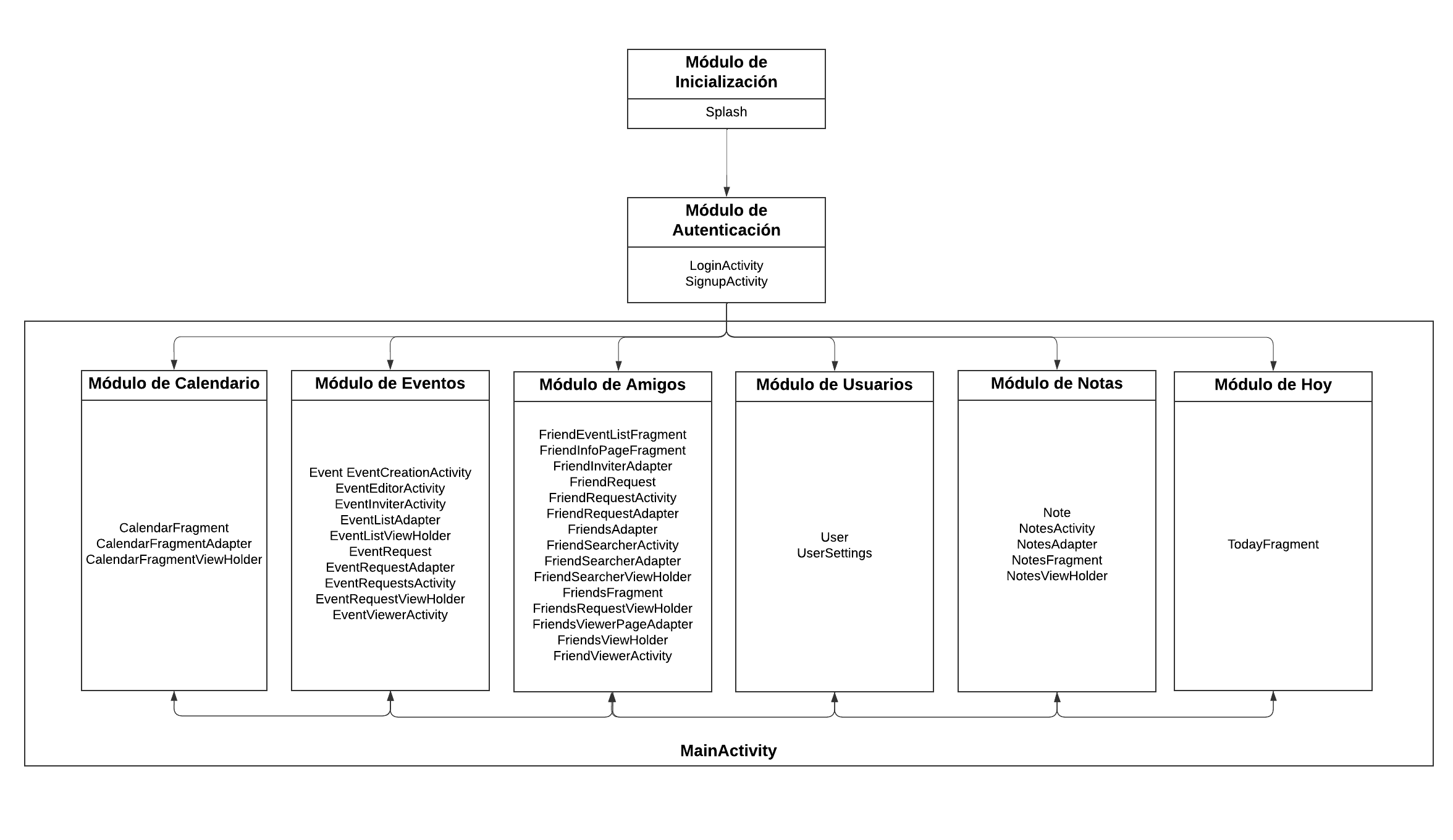
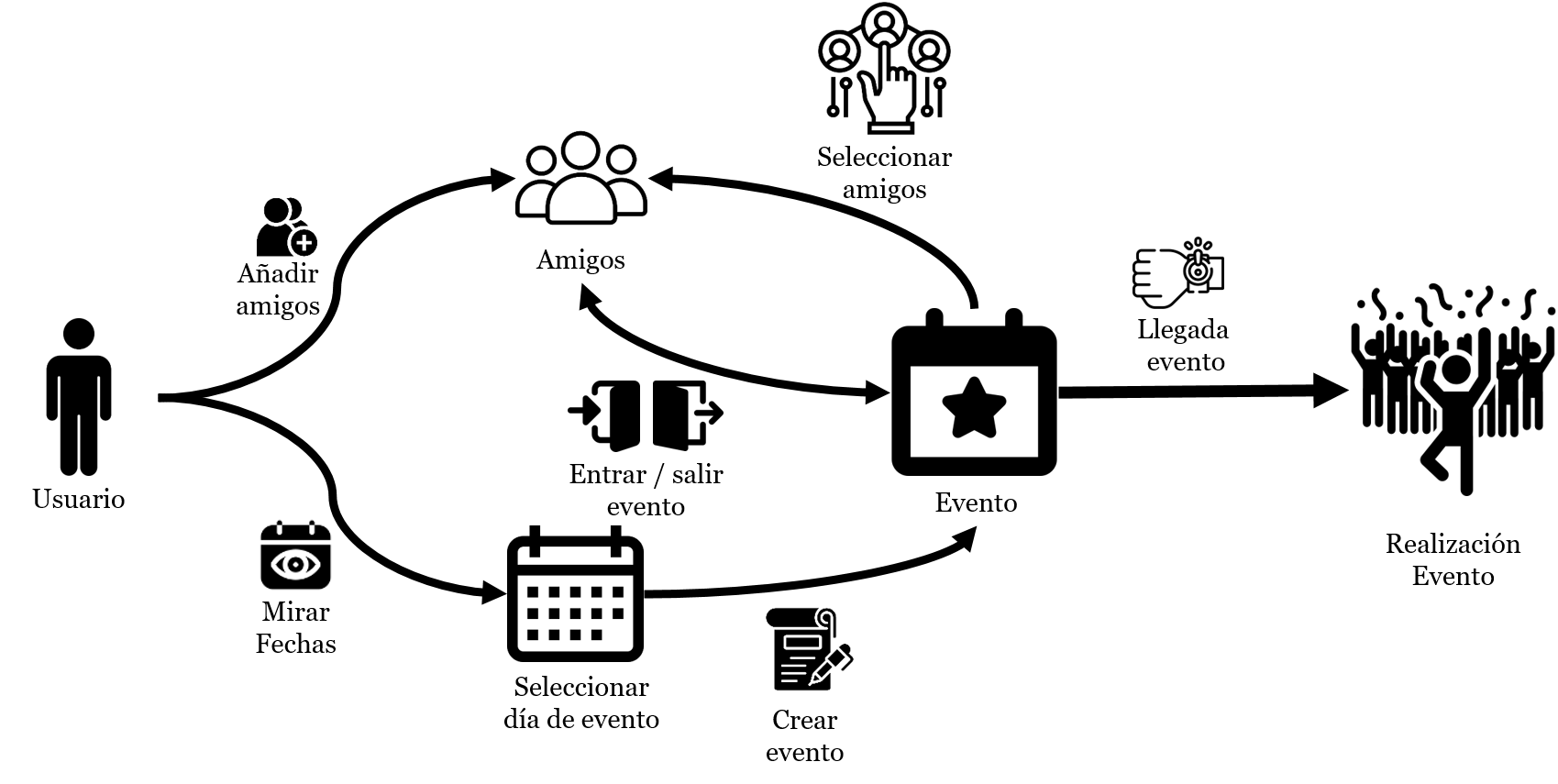


Figura 2\_Diagrama\_de\_representación



## Diseño Lógico de la Aplicación

El diseño lógico de Eventia define la estructura y flujo de datos dentro de la aplicación, asegurando que las diversas funcionalidades interactúen de manera coherente y eficiente. Aquí se describen los componentes lógicos principales y sus interacciones. Definimos a continuación los elementos que conforman el diseño

### 1. Arquitectura de la Aplicación

Eventia sigue una arquitectura MVC, Modelo Vista Controlador, que facilita la separación de las responsabilidades y mejora la mantenibilidad del código:

* Modelo:

Representa la lógica de los datos de la aplicación.

Sus componentes son las clases User, Event, Friend, Note.

* Vista:

Encargada de la presentación de los datos al usuario y la captura de las interacciones.

Sus componentes son las actividades, los fragmentos, los adaptadores y los ViewHolder, como, por ejemplo, LoginActivity, CalendarFragment, EventViewerActivity, etc.

* Controlador:

Maneja la lógica de la aplicación y las interacciones entre el modelo y la vista.

Su componente principal es la lógica implementada en las actividades y los fragmentos para gestionar eventos y actualizar vistas.

### 3. Componentes Clave

* Autenticación y Gestión de Usuarios:
  + LoginActivity y SignupActivity: Gestionan el inicio de sesión y registro de usuarios.
  + UserSettings: Gestiona las configuraciones del usuario.
* Gestión de Eventos:
  + EventCreationActivity y EventEditorActivity: Permiten la creación y edición de eventos.
  + EventViewerActivity: Muestra los detalles de los eventos.
* Interacción Social:
  + FriendsFragment y FriendRequestActivity: Gestionan la lista de amigos y solicitudes de amistad.
  + FriendViewerActivity: Muestra información detallada de los amigos.
* Visualización del Calendario:
  + CalendarFragment: Muestra los eventos en una vista de calendario.
  + CalendarFragmentAdapter y CalendarFragmentViewHolder: Adaptan y muestran los eventos en el calendario.
* Gestión de Notas
  + NotesActivity y NotesFragment: Permiten la creación y gestión de notas.
  + NotesAdapter y NotesViewHolder: Adaptan y muestran las notas.
* Vista de Hoy
  + TodayFragment: Proporciona un resumen de los eventos y actividades del día.

### 3. Flujos de Datos

Para poder detallar correctamente el flujo de los datos, vamos a numerar los pasos que se realizan en cada uno de los flujos:

### Autenticación y Registro

1. El usuario ingresa sus credenciales en LoginActivity, se registra en SignupActivity o tiene sus credenciales guardadas de un anterior inicio de sesión.

2. Las credenciales se validan con Firebase Authentication.

3. Una vez autenticado, los datos del usuario se cargan desde Firebase Realtime Database.

### Creación y Gestión de Eventos

1. El usuario crea un evento en EventCreationActivity.

2. Los detalles del evento se guardan en Firebase Realtime Database.

3. Los eventos se visualizan en EventViewerActivity y CalendarFragment.

### Interacción con Amigos

1. Los usuarios pueden enviar peticiones en FriendSearcherActivity

2. Los usuarios pueden recibir solicitudes de amistad y gestionarlas en FriendRequestActivity.

3. La lista de amigos se visualiza en FriendsFragment.

### Gestión de Notas

1. Los usuarios pueden crear y editar notas en NotesActivity.

2. Las notas se guardan en local gracias a las Shared Preferences y se visualizan en NotesFragment.

### Notificaciones

1. Al crear o editar un evento se registran las notificaciones del evento en el dispositivo

2. Se comprueba la fecha de un evento y se gestionan con las notificaciones sobre eventos próximos con WorkManager.

Este diseño lógico asegura que los componentes de la aplicación interactúen de manera eficiente, proporcionando una experiencia de usuario fluida y coherente.

AQUÍ IRÁN LOS DIAGRAMAS UML CUANDO ESTÉ TERMINADA LA APP

## Diseño de interfaces

## Estudio de la Seguridad de la Aplicación

La seguridad es una prioridad fundamental para Eventia, considerando la información personal y de eventos manejada por la aplicación. Este estudio aborda las principales áreas de seguridad que se debemos considerar para proteger los datos y asegurar la integridad de la aplicación.

Detallaremos una serie de apartados en los que desgranar cada uno de los pasos de este estudio.

### Seguridad en la autenticación

Una autenticación segura utilizando Firebase Authentication. Soporta métodos como correo electrónico y contraseña. Otorga protección contra ataques de fuerza bruta mediante la implementación de políticas de bloqueo después de múltiples intentos fallidos de inicio de sesión.

En adición, las contraseñas de los usuarios no se almacenan en texto plano. En su lugar, se utilizan algoritmos de hash seguros, como bcrypt, para cifrar las contraseñas antes de almacenarlas en Firebase, una función ofrecida por el propio servicio de Firebase.

### Protección de datos

Gracias a Firebase, utilizamos protocolos HTTPS y TLS, para cifrar todas las comunicaciones entre la aplicación y los servidores de Firebase, evitando la intercepción de datos sensibles durante el transporte.

Firebase proporciona cifrado automático para los datos almacenados en su base de datos en reposo, asegurando que los datos sensibles estén protegidos contra accesos no autorizados.

### Control de Acceso

Se han implementado reglas de seguridad de Firebase Realtime Database para controlar el acceso a los datos. Estas reglas determinan quién puede leer y escribir en la base de datos, asegurando que solo los usuarios autorizados tengan acceso a la información.

### Protección contra Amenazas Comunes

Firebase ofrece una validación de todos los datos de entrada para proteger contra ataques de inyección de SQL, XSS y otros tipos de inyección. Su configuración de reglas de seguridad valida la estructura y el contenido de los datos en Firebase, asegurando que los datos que se escriben en la base de datos cumplen con los requisitos esperados.

### Respuesta a Incidentes de Seguridad

Se usan mecanismos de monitoreo y detección para identificar actividades sospechosas y posibles brechas de seguridad gracias a las opciones de monitorización de Firebase Console. En un futuro sería interesante desarrollar planes de respuesta a incidentes que detallan los pasos a seguir en caso de una brecha de seguridad, incluyendo la notificación a los usuarios afectados y las medidas correctivas necesarias.

### Evaluaciones y Auditorías de Seguridad

En caso de ser comercializada, sería importante realizar auditorías de seguridad periódicas para identificar y corregir vulnerabilidades en la aplicación. También se podría plantear la contratación de expertos en seguridad para llevar a cabo pruebas y evalúen la resistencia de la aplicación frente a ataques externos.