



Memoria del Proyecto Final de Grado

Autores:

Yosef Guillermo Karam Müller

Vladyslav Rosiyan

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

1. Introducción:	7
1.1 Introducción en español	7
1.2 Introducción en inglés	8
2. Objetivos de la aplicación	9
3. Análisis del entorno:	11
3.1 Identificación de factores externos	11
3.2 Investigación de la competencia	12
3.3 Recopilación de información	12
3.4 Análisis DAFO	13
4. Solución propuesta:	15
4.1 Tecnologías usadas en el desarrollo	15
4.2 Recursos necesarios	18
5. Planificación temporal del desarrollo del proyecto	21
6. Plan de mantenimiento de la aplicación	29
7. Documentación del diseño e implementación de la solución:	33
7.1 Diseño de la base de datos	33
7.2 Descripción modular del software	36
7.3 Diseño lógico de la aplicación	43
7.4 Diseño de interfaces	48
7.5 Estudio de la seguridad de la aplicación	63
8. Código fuente documentado	65
9. Manual de configuración y funcionamiento de la aplicación	67
10. Manual de usuario	91
11. Objetivos por implementar	117
12. Conclusiones	119
13. Bibliografía y fuentes de información	121

Figuras y tablas

- Tabla _Análisis_Dafo	13
- Tabla 2_Asignación_y_temporización_de_tareas	24
- Tabla 3_Frecuencia_de_mantenimiento	31
- Figura 1_Esquema_visual_JSON_User	35
- Figura 2_Esquema_visual_JSON_Event	35
- Tabla 4_Esquema_de_módulos	41
- Figura 3_Diagrama_deRepresentación	42
- Figura 4_Diagrama_de_contexto	47
- Figura 5_Actividades_y_funciones_principales_Eventia	68

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

1 Introducción

1.1 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 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.

1.2 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.

2 Objetivos

Eventia tiene el objetivo de facilitar y agilizar la gestión de eventos para grupos reducidos, como amigos o familiares. Pasamos a desglosar y enumerar aquellos puntos en los que nos enfocamos para cumplir para conseguir este fin:

1. Simplificar la gestión de eventos

Proporcionar una plataforma que centralice y simplifique todos los aspectos de la organización de eventos, desde la creación hasta la gestión de inscripciones y la comunicación con los asistentes.

2. Mejorar la coordinación logística

Facilitar la coordinación de los diferentes aspectos logísticos de los eventos, asegurando que todo se gestione de manera coherente y eficiente.

3. Optimizar la comunicación

Permitir una comunicación efectiva y fluida entre los organizadores y los asistentes, eliminando la necesidad de utilizar múltiples herramientas de comunicación.

4. Reducir tiempo y recursos

Minimizar el tiempo y los recursos necesarios para organizar eventos, permitiendo a los organizadores concentrarse en la calidad de la experiencia para los asistentes.

5. Facilitar la experiencia del usuario

Ofrecer una interfaz intuitiva y accesible que haga que tanto los organizadores como los asistentes encuentren fácil y agradable interactuar con la aplicación.

6. Adaptabilidad

Proveer una solución versátil que pueda adaptarse a diversos tipos de eventos y necesidades específicas de los organizadores.

7. Mejora continua

Emplear una metodología ágil que permita la evolución constante de la plataforma, incorporando feedback de usuarios y adaptándose a las nuevas tendencias y necesidades.

8. Seguridad y confianza

Garantizar la seguridad de los datos y la confianza en la plataforma, utilizando tecnologías robustas para la autenticación y almacenamiento de información.

3 Análisis del entorno

3.1 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.

3.2 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.

3.3 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.

3.4 Análisis DAFO

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

Tabla 1_Análisis_Dafo

Debilidades	Amenazas
<ul style="list-style-type: none">- Falta de reconocimiento de la marca- Recursos limitados- Dependencia de los servicios de Google	<ul style="list-style-type: none">- Alta competencia- Evolución rápida del mercado- Ataques ciberneticos- Cambios continuos de las tecnologías
Fortalezas	Oportunidades
<ul style="list-style-type: none">- Interfaz intuitiva- Funcionalidades integradas- Flexibilidad en diferentes tipos de eventos- Accesibilidad en dispositivos móviles	<ul style="list-style-type: none">- Crecimiento del mercado de eventos virtuales- Tendencias de personalización en Eventos- Expansión geográfica

4 Solución propuesta

4.1 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.

PlantUML

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.

4.2 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:
 - o 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.
 - o 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.
 - o Consultores y personal de pruebas:
 - Hemos consultado a personas con experiencia en desarrollo de aplicaciones Java, además, hemos tenido la ayuda de personas ajena al desarrollo para que prueben la aplicación y nos den su feedback.
- Recursos Hardware
 - o 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.

5 Planificación temporal del desarrollo del proyecto

Semana 1 (24 septiembre – 29 septiembre)

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

Semana 2 (30 septiembre – 2 de octubre)

- Diseño inicial:
 - o Diseño de la arquitectura de la aplicación
 - o Prototipos de la interfaz de usuario
- Desarrollo inicial:
 - o Configuración del entorno de desarrollo:
 - Android Studio
 - Git
 - Github
 - o Desarrollo de la estructura básica de la aplicación:
 - o Implementación del diseño de interfaz de la aplicación
 - o 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:
 - o 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 aplicación
- 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 aplicación y toda su estructura en Firebase Database
- Añadir función login recuérdame para 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 aplicación
- 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 aplicación
- Implementación de un primer sistema de notificaciones

Semana 10 (25 noviembre – 1 de diciembre)

- Realización de la memoria con los datos ya recogidos e indicados durante el desarrollo de la aplicación
- Desarrollo del sistema de notificaciones

Semana 11 (2 diciembre – 8 diciembre)

- Implementación de la pantalla de hoy, su lógica y su visualización tras salirse un integrante del grupo.
- Refinamiento y finalización del sistema de notificaciones
- Solución de errores
- Realización de la memoria con los datos ya recogidos e indicados durante el desarrollo de la aplicación

Semana 12 (9 diciembre – 15 diciembre)

- Finalización de la memoria
- Presentación y entrega al tutor
- Instalación de la aplicación en sistema de reproducción para defensa del TFG

Tabla 2_Asignación_y_temporización_de_tareas

Tarea	Estatus	Tam.	Asign.	Inicio	Fin
Crear las bases de la app y todo lo necesario para poder navegarla	Realizado	L	Diroween	28/09/2024	02/10/2024
Creación de una pantalla principal con botones en el inferior	Realizado	M	Diroween	02/10/2024	03/10/2024
Creación de una SplashScreen	Realizado	M	Diroween	03/10/2024	06/10/2024
Los botones deben cambiar su color al seleccionarlos	Realizado	S	Diroween	03/10/2024	06/10/2024
Crear base de datos Firebase	Realizado	M	Diroween	17/10/2024	19/10/2024
Creación de las notas y edición de estas	Realizado	M	Diroween	09/10/2024	20/10/2024
Creación de un login linkeado a bdd	Realizado	L	Diroween	17/10/2024	20/10/2024
Creación de signup linkeado a bdd	Realizado	L	Diroween	17/10/2024	20/10/2024
Los amigos deben aparecer en una lista del FriendsFragment	Realizado	M	Diroween	21/10/2024	25/10/2024
Los amigos tienen que poder mandar peticiones, aceptarlas y rechazarlas	Realizado	L	Diroween	21/10/2024	27/10/2024
Los amigos tienen que poder buscarse en la app	Realizado	L	Diroween	24/10/2024	27/10/2024
Creación del sistema de amigos	Realizado	XL	Diroween	21/10/2024	29/10/2024
Las notas deben poder eliminarse con un swipe	Realizado	S	Diroween	29/10/2024	29/10/2024
Los usuarios deben tener una foto de perfil y tienen que poder subirla	Realizado	M	Diroween	30/10/2024	30/10/2024

Crear y linkear servicio en la nube para subir ficheros	Realizado	M	Diroween	30/10/2024	30/10/2024
Unificar el estilo visual de todas las activities	Realizado	L	Diroween	16/10/2024	31/10/2024
Añadir la appbar a todas las activities que lo necesiten	Realizado	M	Diroween	30/10/2024	31/10/2024
Poder ver las imágenes de usuario en la pestaña de amigos y en la ventana de búsqueda	Realizado	M	Diroween	31/10/2024	31/10/2024
En una activity se tienen que mostrar los datos de cada amigo al pulsar en él	Realizado	L	Diroween	31/10/2024	03/11/2024
Hacer que en el signup se vuelva a la pantalla de login	Realizado	XS	Diroween	03/11/2024	03/11/2024
Poner al usuario en la navbar y que pueda ver sus datos y hacer logout	Realizado	L	Diroween	31/10/2024	04/11/2024
Dentro de la activity de mostrar los datos de los amigos hay que añadir una forma de eliminarlos	Realizado	M	Diroween	04/11/2024	04/11/2024
Hacer que en cada ventana de amigo se muestren los eventos del amigo que coincidan con los del usuario	Realizado	L	Diroween	04/11/2024	05/11/2024
Cada evento tiene que tener asignada una imagen que los usuarios puedan subir	Realizado	M	Diroween	06/11/2024	07/11/2024
Añadir al login un recuérdame	Realizado	S	Diroween	09/11/2024	09/11/2024

para poder logearse una única vez					
Crear una clase eventos y la forma de interactuar con ellos como usuario	Realizado	XL	Diroween	03/11/2024	13/11/2024
Dentro de cada evento creado, poder invitar a un amigo al evento	Realizado	L	Diroween	13/11/2024	14/11/2024
Que el usuario pueda salirse de un evento y si es la última persona registrada eliminarlo de la bdd	Realizado	S	Diroween	15/11/2024	15/11/2024
Hacer que se actualicen los eventos en el main cuando se crea o elimina un evento	Realizado	S	Diroween	16/11/2024	16/11/2024
Escribir un resumen de la app en inglés y español	Realizado	M	Diroween	18/11/2024	18/11/2024
Mejora - Pedir confirmación activa de un usuario para salir del evento	Realizado	XS	Diroween	20/11/2024	20/11/2024
Realizar fichero bajo especificaciones de la memoria e indicar los puntos a detallar	Realizado	S	Diroween	20/11/2024	20/11/2024
Improvement -> Los próximos eventos se ordenan además de por fecha por hora	Realizado	XS	Diroween	27/11/2024	27/11/2024
Hacer que una vez se termine un evento, quede registrado en un histórico	Realizado	M	reinnuS	22/11/2024	30/11/2024
Colores específicos para	Realizado	M	reinnuS	22/11/2024	30/11/2024

los eventos pasados y futuros en el calendario					
Pinchar en un día del calendario y muestre los eventos de ese día. También en esa activity que lleve un + para añadir otro evento	Realizado	L	Diroween, reinnuS	22/11/2024	30/11/2024
Un usuario con rol admin debe poder eliminar a usuarios del evento desde el menú contextual	Realizado	S	Diroween	30/11/2024	01/12/2024
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.	Realizado	L	Diroween	30/11/2024	01/12/2024
Cambiar el background de los TextViews en EventViewer a esquinas redondeadas en lugar de cuadradas	Realizado	XS	Diroween	30/11/2024	01/12/2024
Al salir de un evento, el cargo de admin se tiene que pasar automáticamente a otro usuario del evento.	Realizado	S	Diroween	30/11/2024	01/12/2024
Hacer que en la pestaña con el icono del sol, que representa el día actual, se muestren los eventos que tiene	Realizado	L	Diroween, reinnuS	30/10/2024	02/12/2024

para ese día el usuario					
Editar datos del evento.	Realizado	M	Diroween, reinnuS	16/11/2024	02/12/2024
Generar aspecto visual de la app	Realizado	L	Diroween, reinnuS	01/10/2024	04/12/2024
Realizar sistema de notificaciones con WorkManager	Realizado	L	reinnuS	24/10/2024	07/12/2024
Eliminar notificaciones de eventos cuando se sale de uno	Realizado	M	reinnuS	30/11/2024	07/12/2024
Incluir layouts de la app a la memoria	Realizado	L	Diroween	01/12/2024	08/12/2024
Manual de usuario	Realizado	L	reinnuS	07/12/2024	09/12/2024
Realizar la memoria de la aplicación	Realizado	XL	Diroween, reinnuS	16/11/2024	12/12/2024
Aplicar a la memoria todas las especificaciones del proyecto según instrucciones del TFG	Realizado	L	Diroween	18/11/2024	12/12/2024
Realizar los diagramas necesarios de la aplicación	Realizado	XL	Diroween, reinnuS	21/11/2024	12/12/2024
Pruebas y correcciones basadas en los resultados	Realizado	L	Diroween, reinnuS	04/12/2024	12/12/2024
Manual de configuración de la aplicación	Realizado	L	reinnuS	09/12/2024	12/12/2024
Preparar entorno y sistema para prueba en vivo	Realizado	L	Diroween, reinnuS	30/11/2024	15/12/2024

6 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:

Tabla 3_Frecuencia_de_mantenimiento

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.

7 Documentación del diseño e implementación de la solución

7.1 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:
 - o from: identificador del usuario que envió la solicitud.
 - o name: nombre del usuario que envió la solicitud.
 - o status: estado de la solicitud.
- friends: Lista de amigos del usuario:
 - o id: identificador del amigo.
 - o name: nombre del amigo.
- eventsRequests: Solicitudes de participación en eventos:
 - o id: identificador del evento.
 - o 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. Puede tener dos valores cada usuario:
 - o admin: representando el status de administrador del evento
 - o guest: representando ser un participante invitado
- user_id: Identificador del usuario.

- Relaciones:

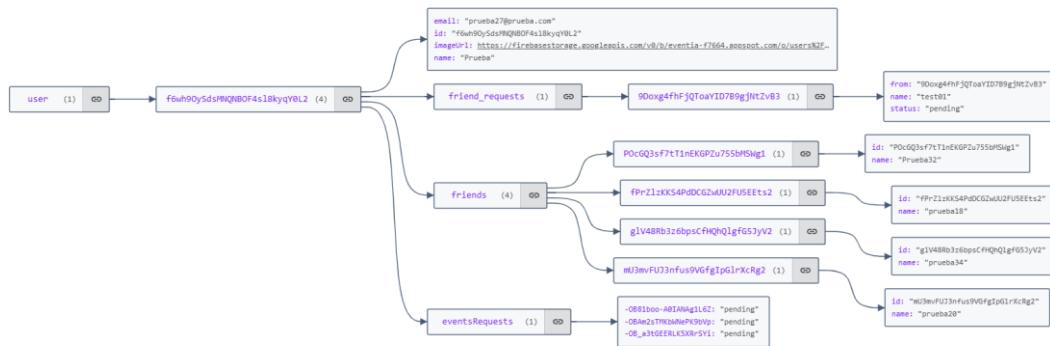
- Un evento puede tener múltiples usuarios registrados.
- Cada usuario registrado tiene un identificador y un estado booleano indicando si está registrado.

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:

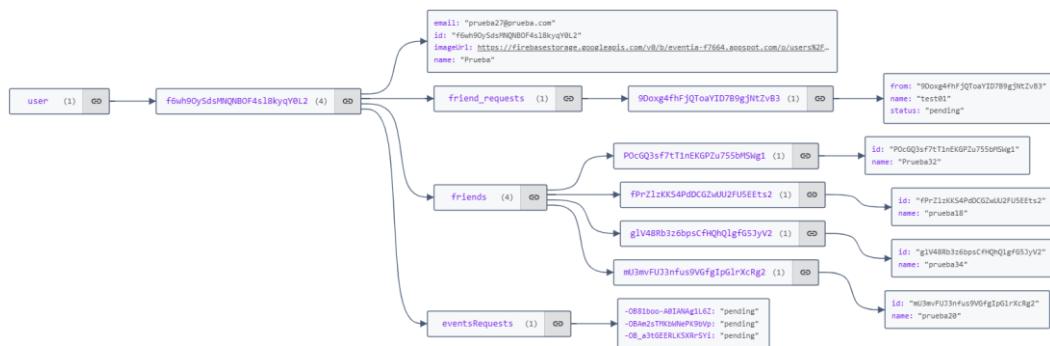
- User

Figura 1_Esquema_visual_JSON_User



- Event

Figura 2_Esquema_visual_JSON_Event



7.2 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.

2. 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:

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

3. 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.
- EventDateComparator: Comparador para las fechas de los 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.
- EventOnCurrentDayActivity: Actividad para mostrar los eventos de un día seleccionado.
- 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.
- PreviousEventsActivity: Actividad para mostrar los eventos pasados.

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.

4. 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.

5. 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.

6. 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.
- TodayListAdapter: Adaptador para la lista de actividades del día.
- TodayListViewHolder: ViewHolder para los elementos de la lista 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.

7. Módulo de Notificaciones (notification)

Gestiona las notificaciones de los eventos y recordatorios. Lo conforman los ficheros:

- NotificationHelper: Clase que maneja la lógica de las notificaciones.
- NotificationSettingsActivity: Actividad para configurar las notificaciones.
- ReminderWorker: Clase que utiliza WorkManager para programar y mostrar recordatorios.

Sus funcionalidades son:

- Gestión y envío de notificaciones.
- Configuración de ajustes de notificaciones.
- Programación de recordatorios para eventos.

La tecnología utilizada en este módulo es WorkManager, para programar tareas en segundo plano y enviar notificaciones.

8 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 y recuperar la información del usuario
- Glide: para mostrar las imágenes de los usuarios.

9 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

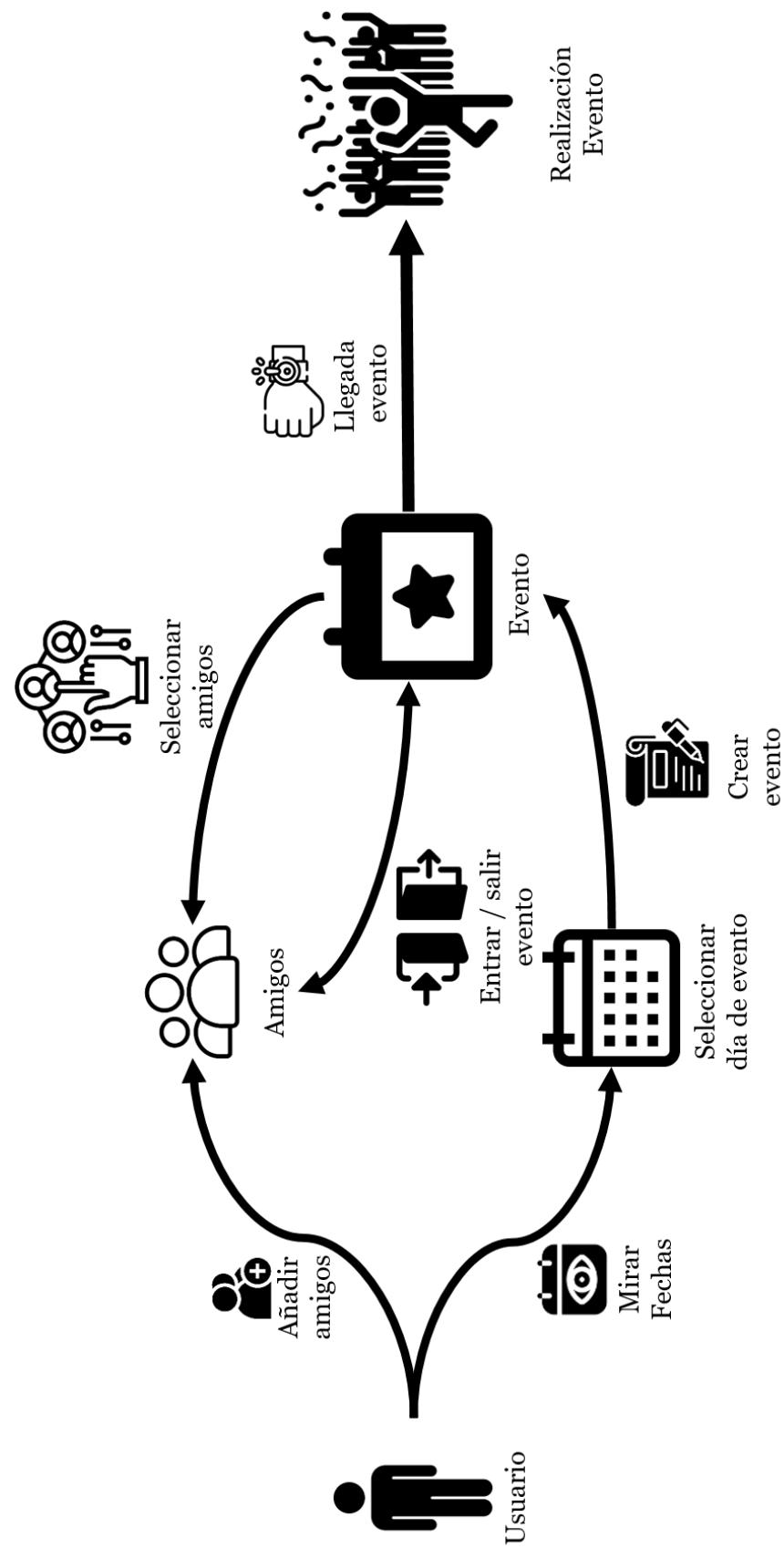
La tecnología usada en este módulo es Firebase Authentication, para poder realizar el inicio de sesión automático.

Incluimos diagramas para facilitar la comprensión del diseño modular de la aplicación. No se pueden incluir en la memoria debido a su tamaño y complejidad, sin embargo, pueden consultarse en la carpeta DOCUMENTOS, del repositorio de GitHub de la aplicación.

Tabla 4_Esquema_de_módulos

Nombre del Módulo	Archivos que lo Conforman
Módulo de Autenticación (authentication)	LoginActivity, SignupActivity
Módulo de Calendario (calendar)	CalendarFragment, CalendarFragmentAdapter, CalendarFragmentViewHolder
Módulo de Eventos (event)	Event, EventCreationActivity, EventDateComparator, EventEditorActivity, EventInviterActivity, EventListAdapter, EventListViewHolder, EventOnCurrentDayActivity, EventRequest, EventRequestAdapter, EventRequestsActivity, EventRequestViewHolder, EventViewerActivity, PreviousEventsActivity
Módulo de Amigos (friend)	FriendEventListFragment, FriendInfoPageFragment, FriendInviterAdapter, FriendRequest, FriendRequestActivity, FriendRequestAdapter, FriendsAdapter, FriendSearcherActivity, FriendSearcherAdapter, FriendSearcherViewHolder, FriendsFragment, FriendsRequestViewHolder, FriendsViewerPageAdapter, FriendsViewHolder, FriendViewerActivity
Módulo de Notas (note)	Note, NotesActivity, NotesAdapter, NotesFragment, NotesViewHolder
Módulo de Notificaciones (notification)	NotificationHelper, NotificationSettingsActivity, ReminderWorker
Módulo de Hoy (today)	TodayFragment, TodayListAdapter, TodayListViewHolder
Módulo de Gestión de Usuarios (user)	User, UserSettings
Módulo Principal y de Inicialización	MainActivity, Splash

Figura 3_Diagrama_deRepresentación



7.3 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.

2. Componentes Clave

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

- Vista de Hoy:
 - o TodayFragment: Proporciona un resumen de los eventos y actividades del día.
 - o TodayListAdapter y TodayListViewHolder: adaptan y muestran los eventos del día actual, dándoles la importancia que tienen

- Notificaciones:
 - o NotificationHelper: Clase para ayudar con la gestión de notificaciones.
 - o NotificationSettingsActivity: Actividad para la configuración de notificaciones.
 - o ReminderWorker: Clase para gestionar recordatorios de eventos.

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.

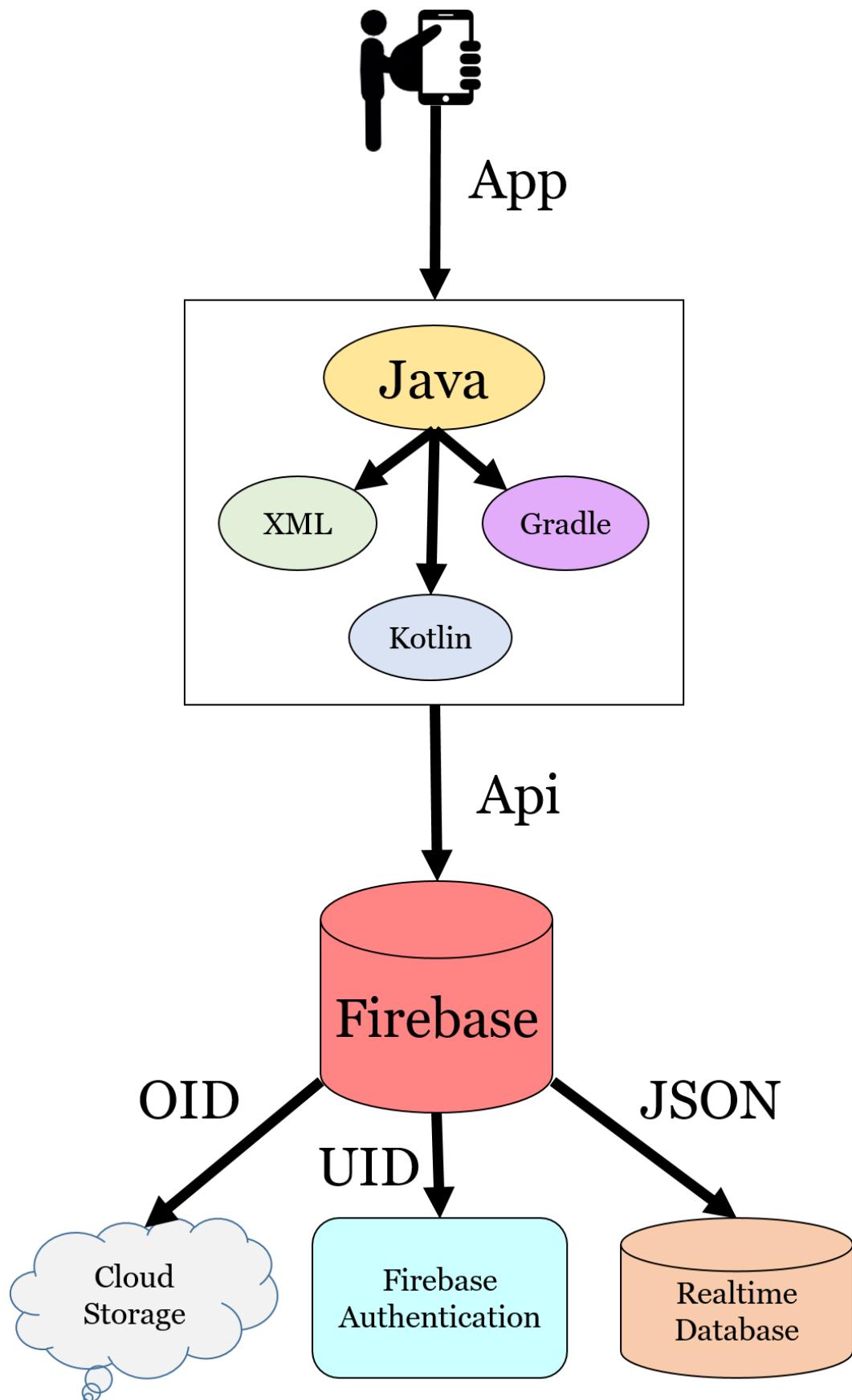
Gestión de Notas

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.

Figura 4_Diagrama_de_contexto



7.4 Diseño de interfaces

El diseño de interfaces de usuario en nuestra aplicación busca proporcionar una experiencia de usuario intuitiva, accesible y visualmente atractiva para facilitar la interacción con las distintas funcionalidades.

Para crear las interfaces primero se realizaron bocetos a mano en papel. Este método permitió explorar rápidamente diferentes ideas de diseño y proporcionar una base visual desde la cual trabajar. Los bocetos a papel fueron digitalizados y recreados en PowerPoint, facilitando prototipos más estructurados. Finalmente, los diseños fueron implementados y refinados en Android Studio.

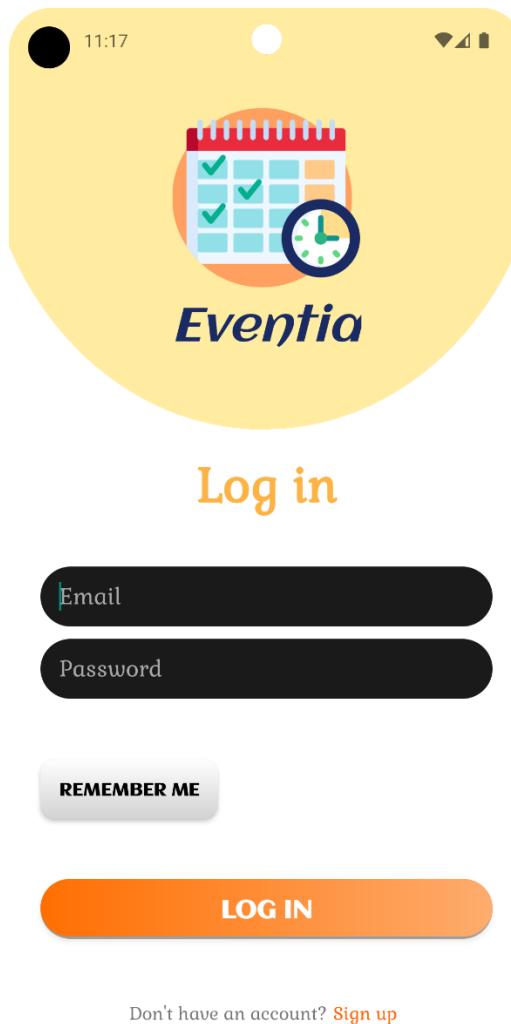
Para seguir un estilo durante toda la aplicación se utilizó la guía de estilo de Material Design de Google para asegurar un diseño consistente, intuitivo y accesible. Los principios clave aplicados han sido:

- Jerarquía visual: utilizamos tamaños de fuente, espaciados y colores para crear una jerarquía clara y comprensible.
- Componentes de interfaz: usamos componentes predefinidos como AppBar, RecyclerView, FloatingActionButton, siguiendo los estándares de Material Design.
- Colores y tipografía: aplicamos una paleta de colores y tipografías recomendadas por Material Design para asegurar una apariencia moderna y coherente.

Con esta metodología, logramos un diseño de interfaz cohesivo y funcional, visualmente atractivo, fácil de usar y accesible para todos los usuarios.

Todas las interfaces están pensadas y diseñadas para que se muestren con los textos en castellano o en inglés dependiendo del idioma del dispositivo. De esta manera, mostraremos capturas en un único idioma, ya que solo cambia el idioma de los textos, no su distribución.

- Interfaz de inicio de sesión

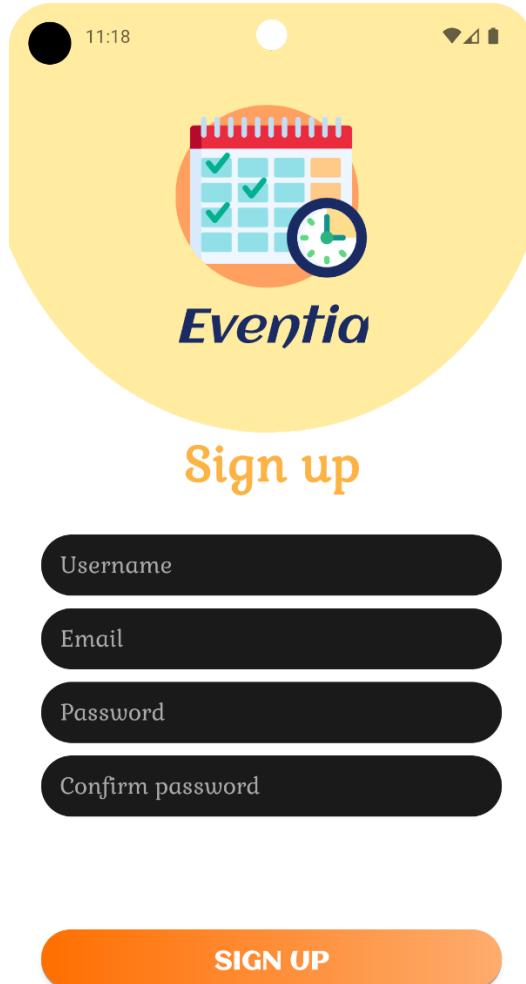


La pantalla de inicio de sesión permite a los usuarios iniciar sesión en la aplicación. Tiene un fondo blanco limpio y está diseñada para ser clara y fácil de usar. Los elementos principales de esta interfaz, mencionado en orden descendente según su posición en pantalla, son:

- Logo y nombre de la aplicación: Un ImageView muestra el logo junto con un TextView que contiene el nombre.
- Indicador de interfaz: Es un TextView que indica al usuario que está en la pantalla de inicio de sesión.
- Campos para introducir email y contraseña: Son dos EditText, uno para el email y otro para la contraseña, con una pista cada uno de lo que se debe introducir en cada campo.

- Botón “Recuérdame”: un ToggleButton que permite al usuario seleccionar si desea que la aplicación recuerde sus datos de inicio de sesión e inicie automáticamente la próxima vez que se abra la aplicación.
- Botón de inicio de sesión: un Button que, al ser presionado, permite al usuario iniciar sesión en la aplicación.
- Botón de texto para registrarse: son dos TextView. el primero indica al usuario que puede registrarse si no tiene una cuenta. El segundo, contiene la palabra "registrarse" está en color naranja y es un enlace clicable que redirige a la interfaz de registro.

- Interfaz de registro

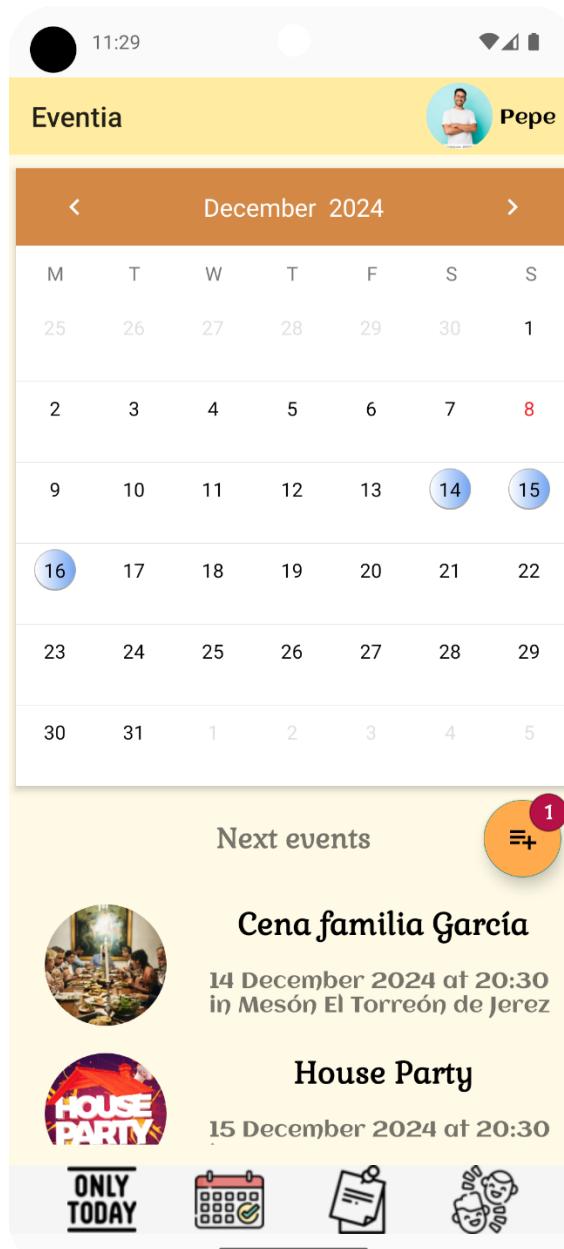


La pantalla de registro permite a los usuarios crear una nueva cuenta en la aplicación. Tiene un fondo blanco limpio y está diseñada para ser clara y fácil de usar, muy similar a la pantalla de inicio de sesión.

Los elementos principales de esta interfaz, mencionado en orden descendente según su posición en pantalla, son:

- Logo y nombre de la aplicación: Un ImageView muestra el logo junto con un TextView que contiene el nombre.
- Indicador de interfaz: Es un TextView que indica al usuario que está en la pantalla de inicio de sesión.
- Campos de texto para introducir los datos de usuario: Son cuatro EditText, uno para el nombre de usuario deseado, otro para el email, otro para la contraseña y el último para la confirmación de la contraseña. Cada uno de ellos lleva una pista de lo que se debe introducir en cada campo.
- Botón de registro: es un Button que, al ser presionado, permite al usuario registrar una nueva cuenta en la aplicación.

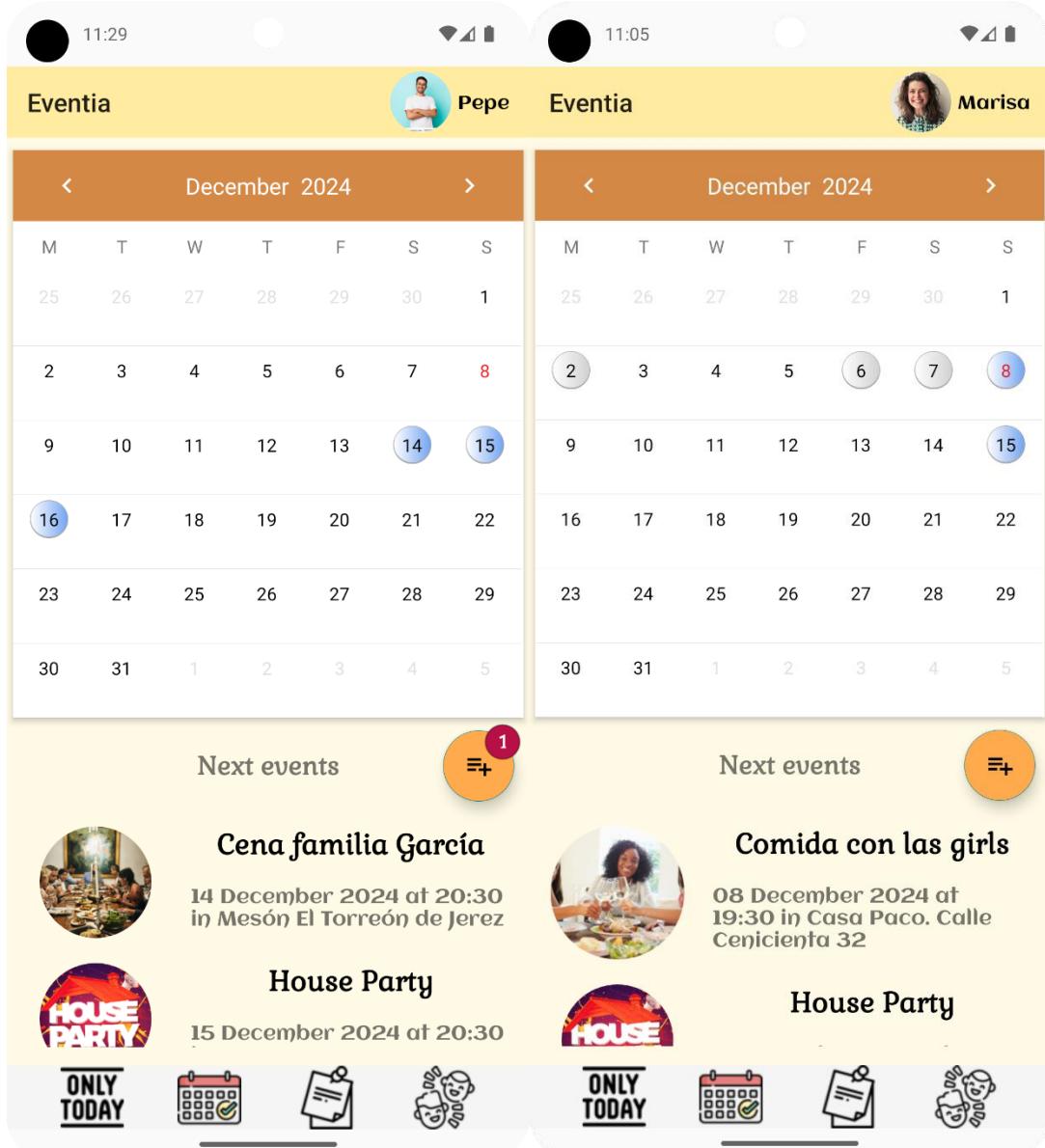
- Interfaz principal de la aplicación



La interfaz principal de la aplicación muestra la agenda de eventos del usuario. Contiene las distintas pantallas principales, además de acceso directo a la modificación del perfil del usuario y una barra de navegación inferior permite cambiar entre las pantallas principales, pulsando en cada uno de los iconos, simulando una BottomNavigationView.

Ahora procederemos a detallar cada una de las pantallas principales:

Pantalla Calendario de eventos



Contiene un calendario de eventos, en la que los eventos pasados se representan en gris, los eventos futuros en azul y el día actual con el número en rojo. Contiene además, una lista de los próximos eventos, en un RecyclerView, y un FloatingActionButton para poder añadir eventos a los que el usuario ha sido invitado., el cual puede tener un indicador TextView en caso de que haya eventos pendientes de aceptar.

Pantalla Eventos de hoy



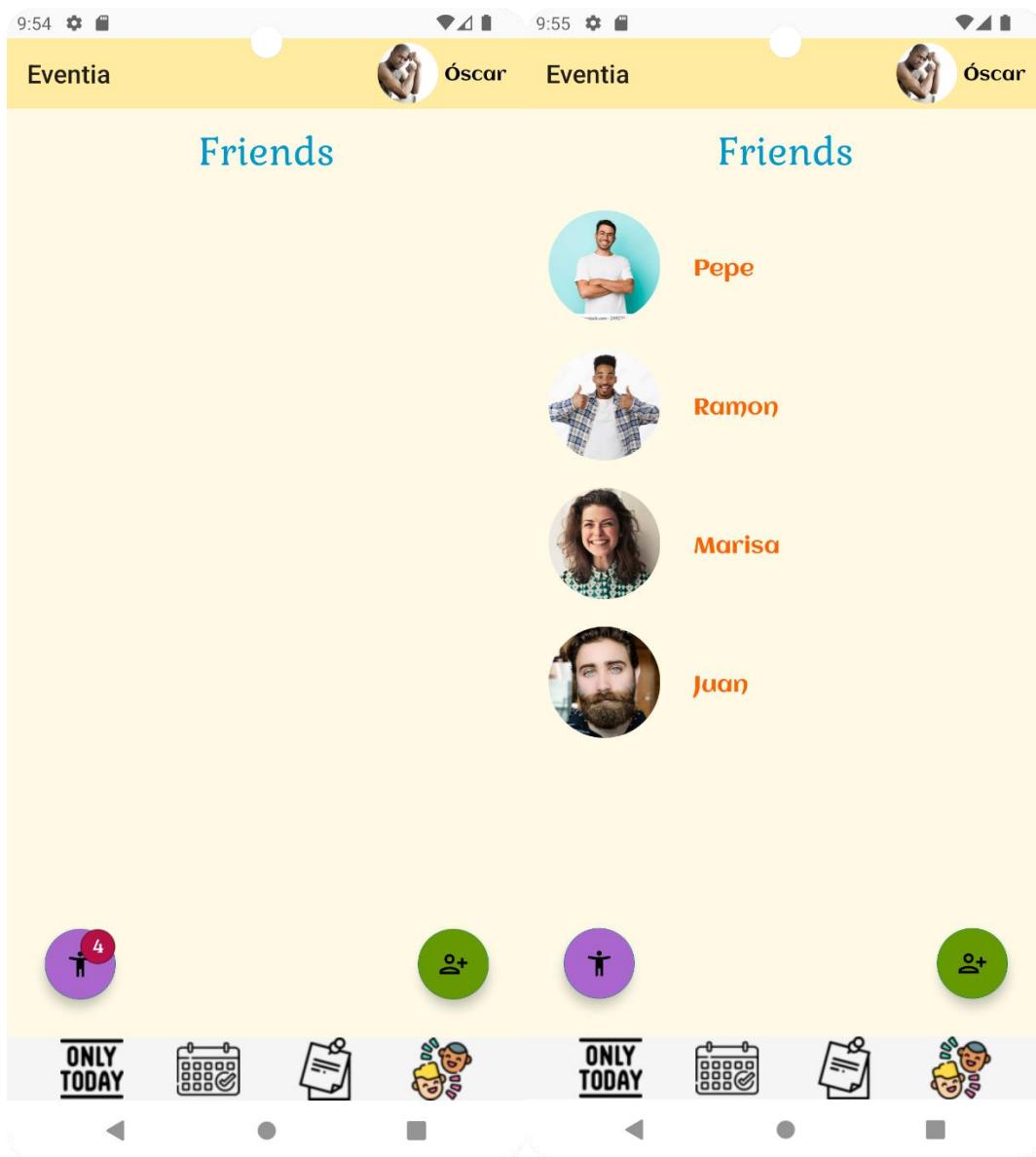
La pantalla del día de hoy presenta los detalles del evento o eventos programados para el día actual. Incluye la foto del evento en un ImageView, los datos del evento con TextViews y una lista de los participantes, un RecyclerView. Al pulsar en un evento se puede ir a la vista detalle de esta. En esta pantalla se le da importancia a cada evento, ocupando la mayor parte de la pantalla, ya que no es habitual tener más de un evento al día, centrando la atención del usuario en los más importante, el evento del día actual.

Pantalla de notas



La pantalla de notas permite a los usuarios ver y gestionar sus notas. Tiene un fondo con un color de post-it para identificarlo visualmente con las notas. Cada nota está contenida en una caja que separa el nombre de la nota de su cuerpo, y todas las notas están contenidas en un RecyclerView. También, hay un FloatingActionButton en la esquina inferior derecha para crear una nueva nota.

Pantalla de amigos



La pantalla de amigos permite a los usuarios ver y gestionar sus amistades dentro de la aplicación. Un RecyclerView muestra los amigos con su nombre y foto de perfil, en un TextView y un ImageView correlativamente. La pantalla tiene dos FloatingButtons en la parte inferior, uno para aceptar o rechazar peticiones de amistad, con un contador TextView de la cantidad de peticiones pendientes, y otro para buscar y enviar peticiones de amistad. Pulsando en un amigo podemos ver sus datos y los eventos en los que se coincide con él.

Otra interfaz de alta importancia es la pantalla de vista detalle de un evento, en ella se puede gestionar por completo un evento. No se detallarán los nombres de los elementos ya que se reutilizan en el resto de interfaces ya mencionadas y simplemente se vuelven a utilizar, por no ser redundantes, se muestra solo el diseño de la interfaz.

- Vista detalle de un evento



En esta interfaz se muestran los datos del evento, su foto y su título. En la parte baja del evento se muestran los participantes, diferenciando a los administradores de los participantes con un color malva y posicionándolos en la parte superior de la lista. Tres botones permiten realizar distintas acciones. Un botón para eliminar el evento para el usuario, otro para modificarlo y otro para enviar peticiones de evento a los amigos del usuario. Pulsando en cada uno de los participantes, que no sea el propio usuario, se podrá visualizar los datos del amigo.

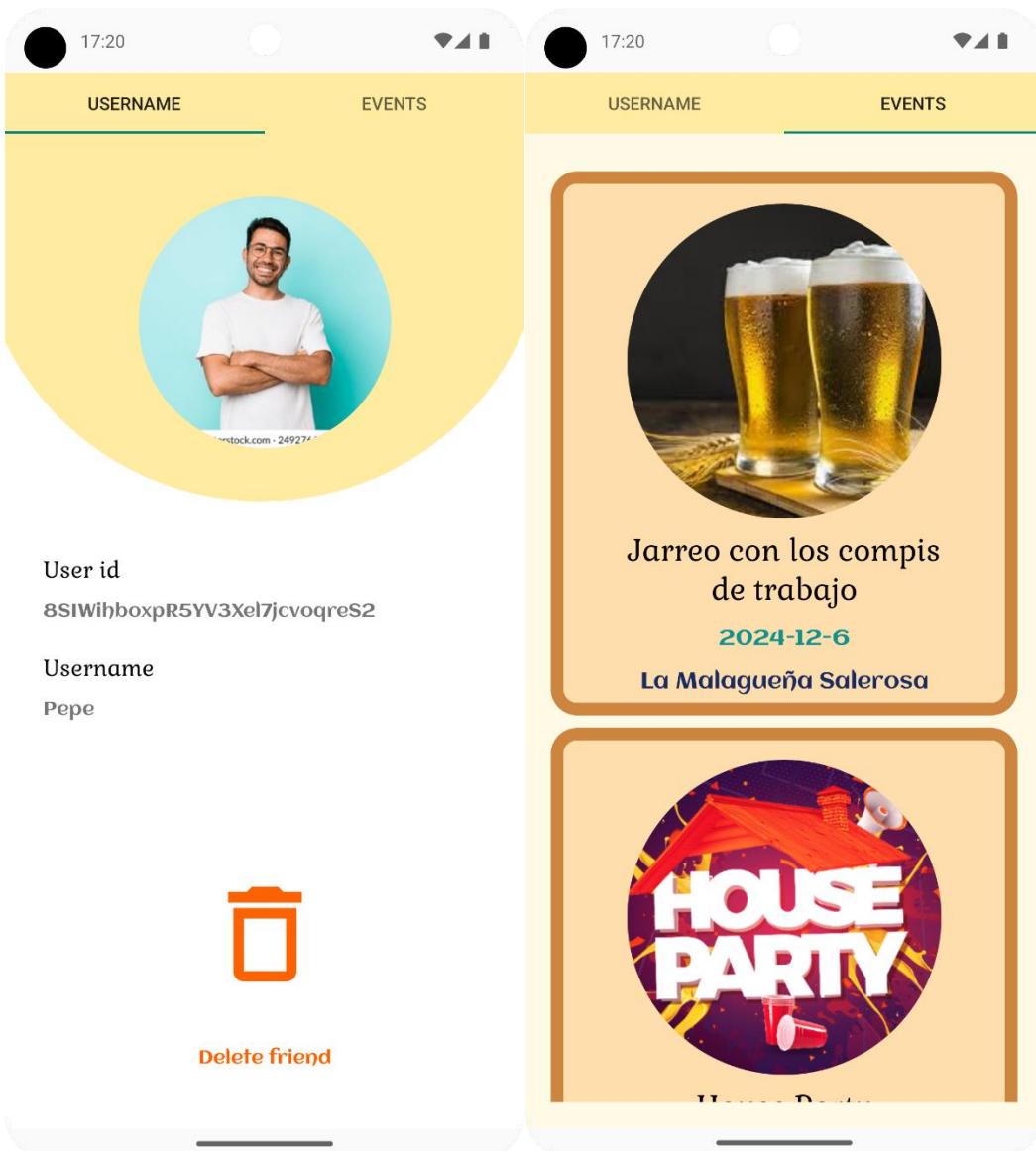
El resto de interfaces siguen siendo de importancia, se mostrarán, pero dada su sencillez y fácil reconocimiento de elementos e interactividad no se explicarán en detalle su diseño, solo se harán pequeñas descripciones para mejorar la lectura del documento.

- Interfaces para la creación y edición de un evento



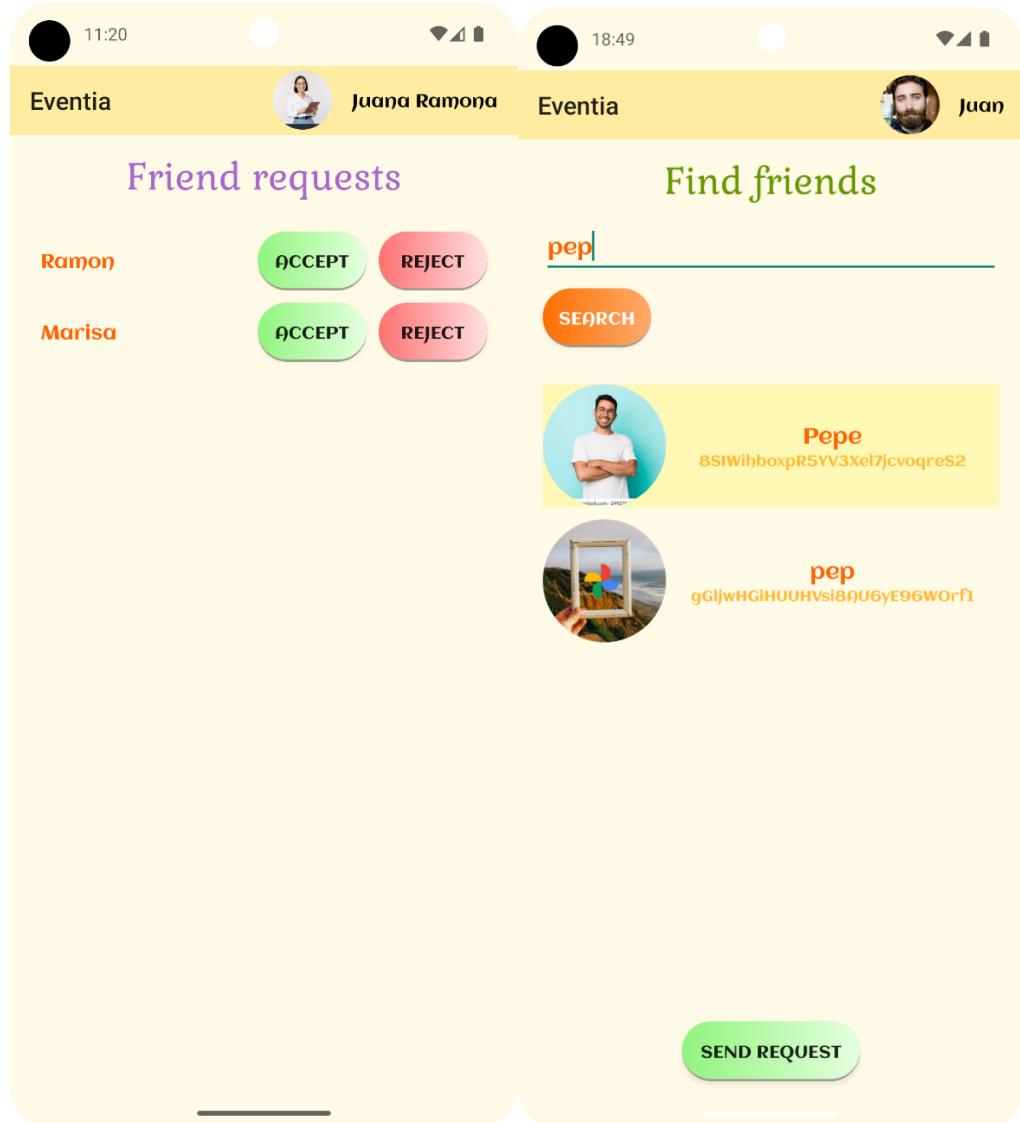
Formularios sencillos e intuitivos con un botón para crear o salvar cambios

- Interfaces para la visualización de los datos de un amigo



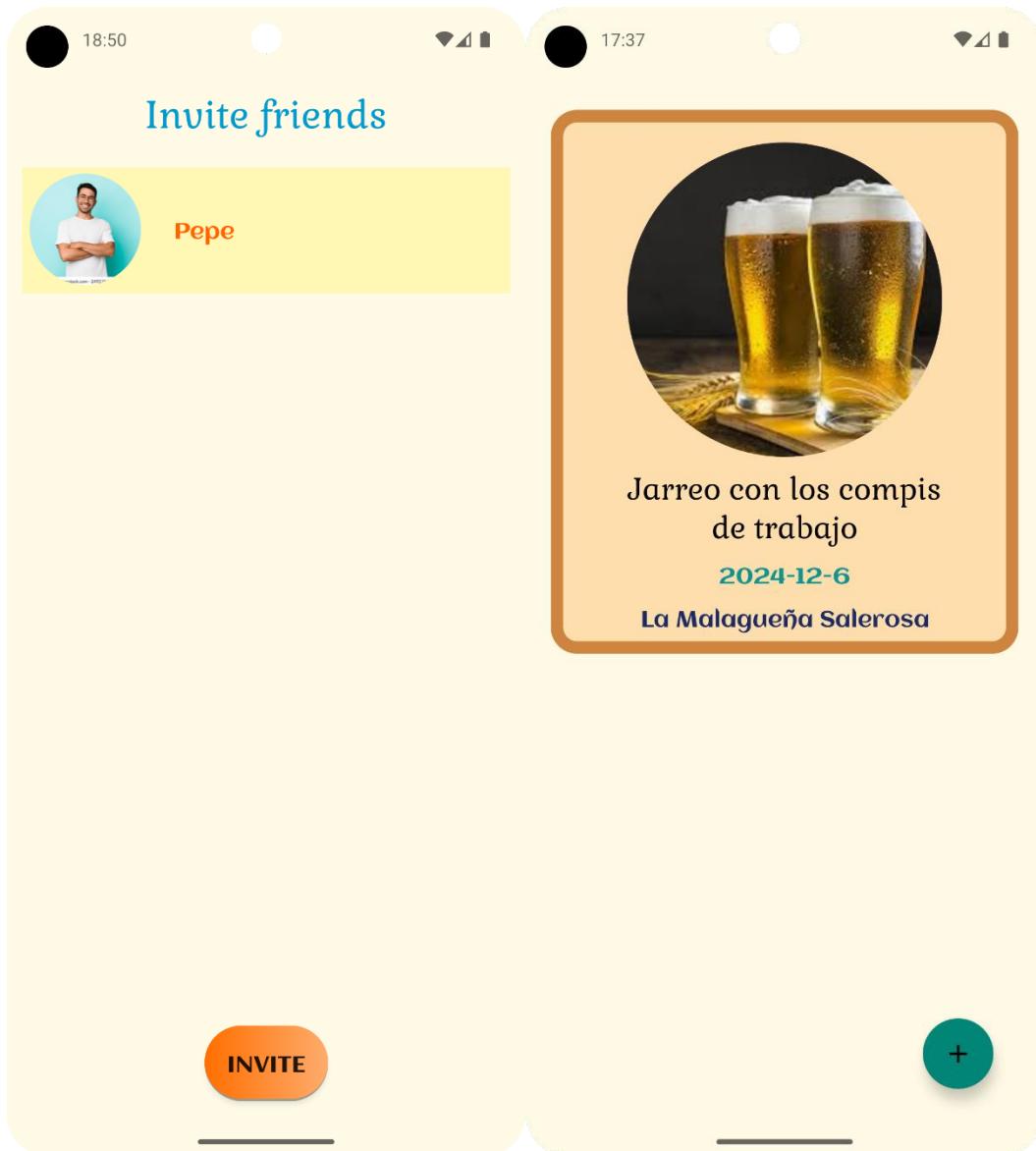
Vistas para la visualización de los datos de un amigo y los eventos conjuntos en los que se están inscritos.

- Interfaces para aceptar, buscar amigos



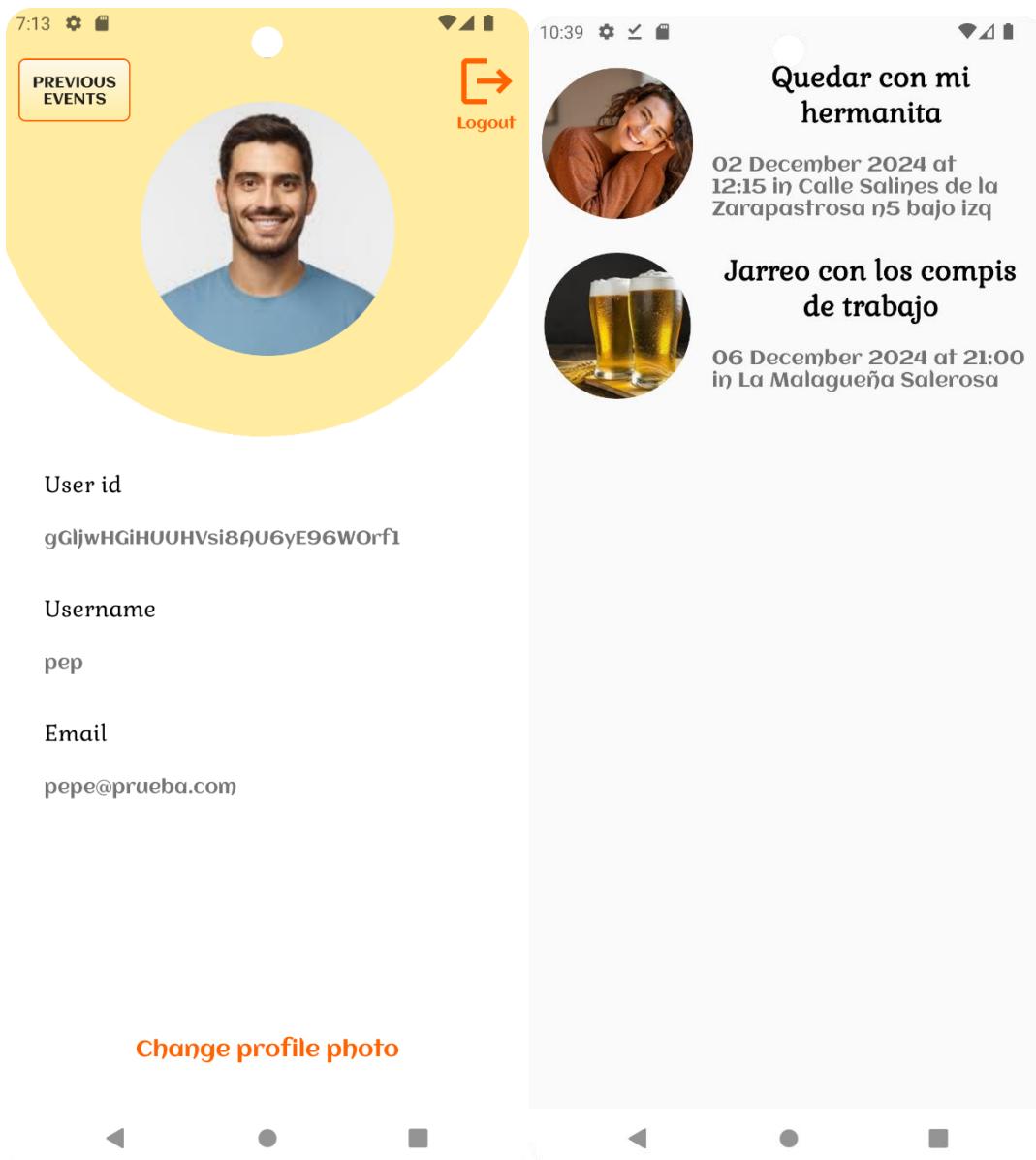
Botones simples y concisos. En la búsqueda de amigos se añade el id del usuario, como en otras aplicaciones, como Discord, para poder identificar a usuarios que tuviesen el mismo nombre o imagen.

- Interfaces para invitar a un amigo y para ver los eventos del día actual



Interfaces sencillas, la primera una lista de amigos y un botón, la segunda con el evento y un botón para poder crear uno nuevo.

- Interfaz de usuario y eventos previos



Una interfaz clara, que muestra la imagen del usuario y sus datos. Permite con un botón ir a los eventos previos, una lista sencilla que permite clicar cada evento para ir a su vista detalle. Con otro botón se puede cerrar sesión y con el último, un TextView clicable, se puede cambiar la foto de perfil del usuario.

7.5 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.

8. Código fuente documentado

Incluimos el código fuente documentado a través del enlace a nuestro repositorio público de GitHub, para que esté disponible y se pueda consultar en todo momento.

<https://github.com/Diroween/Eventia>

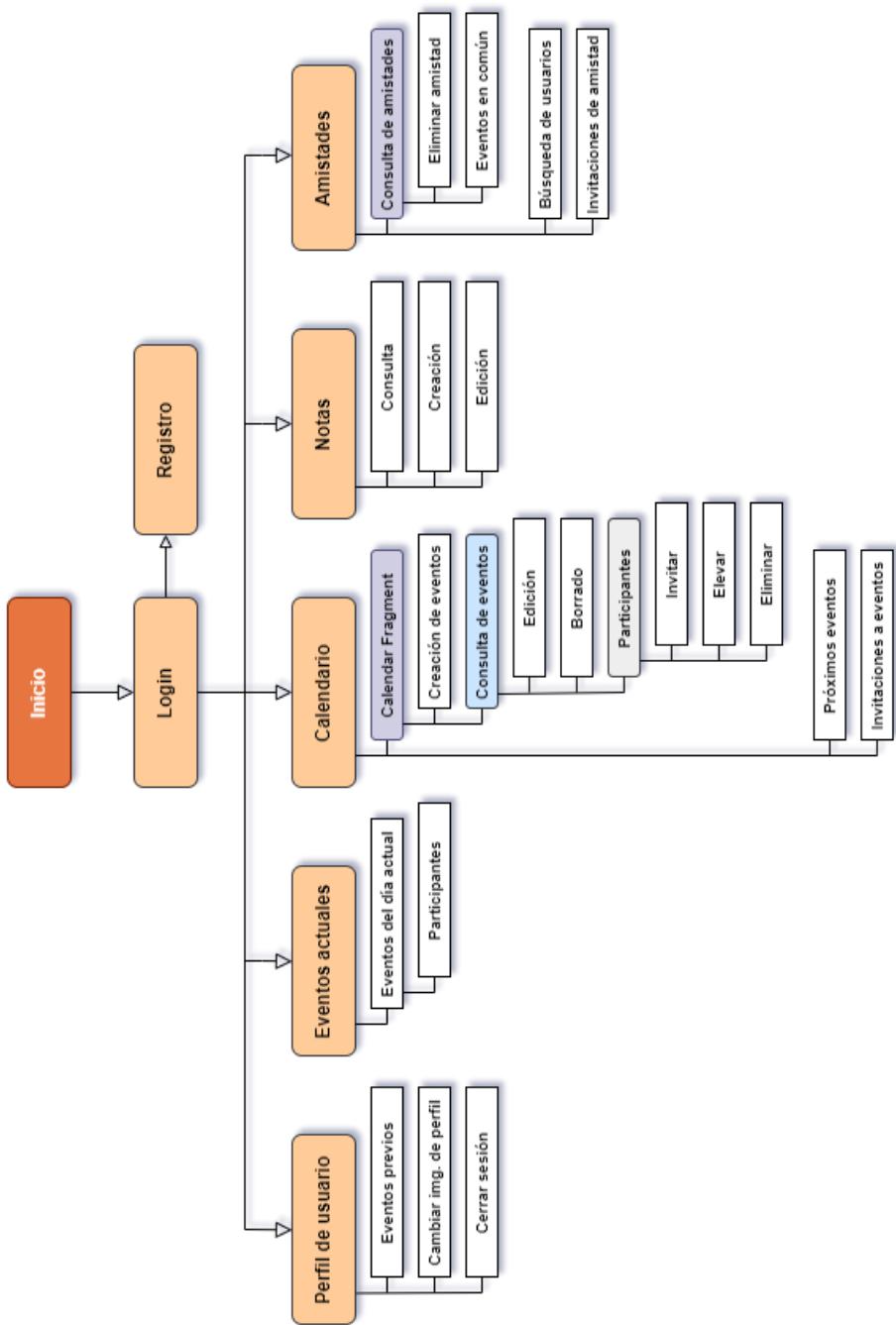
En el propio GitHub se ha dejado creado un README.md para darle un aspecto visual en GitHub al proyecto. Se puede consultar también en el enlace proporcionado.

9. Manual de configuración y funcionamiento de la aplicación

Contenidos

1. Configuración inicial.
2. Inicio.
 - 2.1. SignupActivity.
 - 2.2. LoginActivity.
3. Clase User.
4. MainActivity.
5. CalendarFragment.
 - 5.1. NotificationHelper.
6. Clase Event.
 - 6.1. EventCreationActivity.
 - 6.2. EventOnCurrentDayActivity.
 - 6.3. EventViewerActivity.
 - 6.4. EventEditorActivity.
7. TodayFragment.
8. NotesFragment.
9. FriendsFragment.
 - 9.1. FriendSearcherActivity.
 - 9.2. FriendRequestActivity.
 - 9.3. FriendViewerActivity.
10. UserSettings.
 - 10.1. PreviousEventsActivity.

Figura 5_Actividades_y_funciones_principales_Eventia



Actividades y funciones principales Eventia

1. Configuración inicial

La aplicación está configurada para emplear un SDK mínimo de nivel 26 y nivel 31 como API objetivo.

A lo largo de este manual se hace referencia a varias de las clases proporcionadas en la librerías:

- androidx.work
- androidx.recyclerview
- androidx.swiperefreshlayout
- com.google.firebaseio
- com.github.bumptech.glide
- com.applandeo:material-calendar-view

Los permisos requeridos para el uso óptimo de Eventia son:

- android.permission.INTERNET
- android.permission.READ_MEDIA_IMAGES
- android.permission.WRITE_MEDIA_IMAGES
- android.permission.READ_MEDIA_VISUAL_USER_SELECTED
- android.permission.READ_EXTERNAL_STORAGE
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.POST_NOTIFICATIONS

2. Inicio

La aplicación arranca mostrando un splash screen del logo de Eventia. Para ello, se emplea la clase Splash, en la que se hace uso de un objeto AnimationDrawable para realizar la animación.

En dicha clase también se establece la conexión con el proyecto en Firebase con el método:

```
1     FirebaseAuth mAuth = FirebaseAuth.getInstance();
```

Para simplificar y de cara al resto del documento, se indica a continuación las instancias que se realizan para poder emplear las funciones y conexiones proporcionadas por Firebase:

```
1     FirebaseAuth mAuth = FirebaseAuth.getInstance();  
2     FirebaseDatabase database =  
         FirebaseDatabase.getInstance().getReference();
```

Continuando con la ejecución, se verifica si el usuario ha optado por recordar su inicio de sesión mediante el método:

```
1     private void rememberLogIn(String email, String password)
```

obteniendo las credenciales almacenadas con SharedPreferences en el dispositivo y dando paso a la ejecución de MainActivity en caso de que el valor "rememberme" sea "true".

En caso contrario, se da paso a la ejecución de LoginActivity.

2.1. SignupActivity

Si no se dispone de una cuenta es posible realizar el registro al pulsar sobre "Registrarse", lo cual ejecuta SignupActivity.

En dicha clase disponemos del siguiente método para generar el usuario en FirebaseAuth:

```
1     private void createAccount(String email, String password, String username)
```

y para almacenar la referencia de dicho usuario en la base de datos se emplea:

```
1     private void writeNewUser (String userId, String name, String email)
```

2.2. LoginActivity

LoginActivity verifica mediante "Patterns.EMAIL_ADDRESS" que el formato del campo "Correo" sean correctos, ejecutando en tal caso:

```
1     private void logIn(String email, String password)
```

que pasa a iniciar sesión y almacenar las credenciales (si aplica) con:

```
1     firebaseAuth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>()
```

si la tarea se ha completado con éxito, se inicia MainActivity.

3. Clase User

La clase User cuenta con unos atributos, constructores y métodos getters y setters básicos:

```
1  private String email;  
2  private String name;  
3  private String id;  
4  private String imageUrl;
```

```
1  public User(String name, String email, String id, String imageUrl)  
2  public User(String name, String email, String id)  
3  public User(String name, String id)
```

4. MainActivity

La ejecución de MainActivity comprueba si la aplicación cuenta con los permisos de notificación habilitados con el método:

```
1     private void requestNotificationPermission()
```

Dependiendo de la versión de Android y del permiso establecido en el dispositivo genera el cuadro de dialogo de permisos o lanza NotificationSettingsActivity, indicando al usuario la importancia de las mismas y facilitando un acceso a los ajustes de la aplicación en el sistema.

Una vez comprobados, carga por defecto CalendarFragment y establece sobrescribiendo el método onClick que emplearan las distintas imágenes/secciones de la aplicación:

- TodayFragment.
- CalendarFragment.
- NotesFragment.
- FriendsFragment.
- UserSettings.

5. CalendarFragment

CalendarFragment se compone principalmente de:

- CalendarView.
- RecyclerView.
- FloatingActionButton.
- Un adaptador personalizado (CalendarFragmentAdapter).
- Una serie de ArrayLists para manejar los datos.

Al crear la vista de CalendarFragment se cargan los eventos del usuario mediante el método:

```
1     public void loadUserEvents()
```

En él, se escucha activamente cualquier cambio en la base de datos con el listener:

```
1     databaseReference.child("events").orderByChild("date")  
2         .addValueEventListener(new ValueEventListener() { ... })
```

y se criba por fecha en los distintos ArrayLists los eventos del usuario, asignando un fondo determinado en el calendario en función del tipo de evento (pasado o futuro).

En RecyclerView se cargan todos los eventos futuros a partir de la fecha y hora actual, almacenados en el ArrayList empleado por CalendarFragmentAdapter.

Finalmente, se realiza una llamada al método:

```
1     private void checkPendingEventRequests()
```

el cual el emplea el mismo tipo de listener que hemos mencionado previamente, pero comprobando en esta ocasión si existen invitaciones a eventos pendientes, mostrando un contador con la cifra en tal caso.

5. CalendarFragment

Al pulsar en FloatingActionButton accederíamos a EventRequestsActivity en el que nos es posible aceptar o rechazar dichas invitaciones mediante los métodos:

```
1  private void acceptEventRequest(String eventId)  
2  private void declineEventRequest(String eventId)
```

De ser el primer caso, se registra un nuevo par clave-valor para dicho evento en la base de datos (JSON), siendo éste el userId y rol, que por defecto es “invitado”.

Por último y como comentábamos antes, mostramos dentro de un RecyclerView los eventos que todavía no han llegado a término adaptando la forma en la que se visualizan mediante CalendarFragmentAdapter.

Hemos hecho uso de la librería com.bumptech.glide para manejar la obtención y formato de las imágenes que se muestran en cada elemento. Al pulsar sobre cualquiera de ellos accederíamos a la vista proporcionada por EventViewerActivity para dicho evento.

5.1. NotificationHelper

Los métodos estáticos de la clase [NotificationHelper](#) sirven para programar las notificaciones correspondientes según los segundos (delay) que queden hasta dicho momento:

```
1  public static void enqueueNotifications(Context context, Event event, long delay)
2  public static long getSecondsUntilEvent(Event event)
```

Las notificaciones (hasta un máximo de 4 por evento) se generan empleando [WorkManager](#) y son del tipo [OneTimeWorkRequest](#) y [Unique Work](#). Con ello nos es posible lanzar un único trabajo por cada notificación pendiente y tener controlados los mismos mediante el nombre (o mejor dicho ID) del evento.

Incluimos ciertos datos mediante los setters de OneTimeWorkRequest para que a la hora de crear la notificación podamos mostrar la información del evento, así como disponer de su ID para que al pulsar sobre el dialogo de notificación podamos mostrar el evento desde [EventViewerActivity](#).

Ésta última funcionalidad se logra mediante un objeto [TaskStackBuilder](#) en el cual creamos una pila de actividades estableciendo [MainActivity](#) como principal y agregando [EventViewerActivity](#) encima.

Todo ello ocurre en el método:

```
1  public void createNotification(String title, String message, String eventId)
```

6. Clase Event

La clase Event cuenta con los siguientes atributos, constructores y métodos getters y setters acordes:

```
1  private String id;  
2  private String name;  
3  private String date;  
4  private String place;  
5  private String image;  
6  private String hour;  
7  private Map<String, String> registeredUsers;
```

```
1  public Event(String id, String name, String date, String place, String image, String  
hour, Map<String, String> registeredUsers)  
2  public Event(String id, String name, String date, String place, String image)
```

6.1. EventCreationActivity

Esta clase toma la fecha seleccionada en el calendario y facilita la edición de:

- La imagen del evento (opcional).
- Título.
- Lugar.
- Hora.

Al ejecutar EventCreationActivity se genera un nuevo objeto de tipo Event y se establece un id aleatorio mediante:

```
1     databaseReference.push().getKey();
```

Para realizar la carga de imágenes se ejecuta:

```
1     private void requestPermissions()
```

se hace una diferencia entre versiones Android (superior/inferior a Tiramisú) y se solicitan los permisos de lectura de imágenes del dispositivo con:

```
1     ContextCompat.checkSelfPermission(this,  
        Manifest.permission.READ_MEDIA_IMAGES)
```

Si se cuenta con ellos, se lanza un Intent de tipo “image/*”. Con todo ello se genera un objeto:

```
1     private final ActivityResultLauncher<Intent> pickImageLauncher =  
        registerForActivityResult (new  
    ActivityResultContracts.StartActivityForResult(), result → { ... })
```

Subiendo la previsualización de la imagen en el ImageView de la actividad y ejecutando:

```
1     private void uploadImage()
```

Que mediante el uso de StorageReference carga la imagen en la base de datos, , generando un objeto URI con la referencia.

6.1. EventCreationActivity

La selección de la hora se realiza mostrando un TimePickerDialog (posicionado en la hora actual del dispositivo).

Finalmente, al pulsar en el botón de guardar se toma todos los valores de los campos verificando que no estén vacíos previamente con:

```
1 private boolean checkEmptyFields()
```

y se pasa a registrar el evento en la base de datos con:

```
1 private void saveEvent(String imageUri)
```

6.2. EventOnCurrentDayActivity

Volviendo a las funcionalidades de CalendarFragment, hemos implementado un listener para que a la hora de pulsar sobre una fecha del calendario que contenga al menos un evento se nos dirija a EventOnCurrentDayActivity, en la que se muestra un listado simple de los eventos que tienen o tuvieron lugar así como un botón para poder generar un nuevo evento.

Si pulsásemos sobre uno de los eventos veríamos la información en detalle del mismo mediante EventViewerActivity.

6.3. EventViewerActivity

En EventViewerActivity mostramos en detalle toda la información del evento entregado como extra por la actividad que la inició.

Se incluyen varias funcionalidades:

- Eliminar el evento.
- Editar dicho evento.
- Agregar participantes.
- Consultar y/o administrar participantes.

Para ello hacemos una llamada única a la base de datos en busca del ID del evento dentro del método:

```
1  private void loadData() {  
2      reference.child("events").child(eventId).addListenerForSingleValueEvent(new  
3          ValueEventListener() { ... }  
4      )  
5  }
```

Obtenemos los usuarios registrados para dicho evento, lo cual se mostrará mas adelante en el RecyclerView destinado para ello con:

```
1  private void loadRegisteredUsers(String eventId)  
2      reference.child("events").child(eventId).child("registeredUsers")  
3          .addValueEventListener(new ValueEventListener() { ... })  
4  }
```

y asignamos a cada uno el rol que tengan en el mismo:

```
1  private void loadRoles() {  
2      roles.clear();  
3      reference.child("events").child(eventId).addListenerForSingleValueEvent(  
4          new ValueEventListener() { ... })  
5  }
```

6.3. EventViewerActivity

Los permisos para editar un evento, invitar y administrar participantes solo está disponible para los usuarios que tengan el rol "admin". Esto se comprueba mediante el método:

```
1  private void checkAdmin(String eventId) {  
2      String userId = FirebaseAuth.getInstance().getUid();  
3      reference.child("events").child(eventId).child("registeredUsers").child(userId)  
4          .addListenerForSingleValueEvent(new ValueEventListener() { ... })
```

Desde esta actividad nos es posible administrar el rol de cada participante. Para ello mostramos mediante el siguiente método un menú que nos permite eliminar o elevar los permisos del participante seleccionado:

```
1  private void showContextMenu(View view, User user)
```

Manejamos las acciones según la opción seleccionada con:

```
1  private boolean handleMenuItemSelected(MenuItem item, User user)
```

Que ejecuta respectivamente:

```
1  private void removeUserAsAdmin(User user)  
2  private void makeUserAdmin(User user)
```

Ambos métodos realizan una llamada a la BBDD en la cual o bien se elimina el par con el userId en cuestión o se modifica el valor del mismo a "admin" dentro de "registeredUsers".

6.3. EventViewerActivity

Para finalizar la sección de participantes, nos es posible agregar nuevos al pulsar en el botón destinado para ello, ejecutando EventInviterActivity. En ella, primero se carga el listado de amistades del usuario:

```
1 private void loadFriends()
```

Y si invitamos a una de ellas se ejecuta:

```
1 private void inviteFriend();
```

que finalmente muestra con un Toast si la invitación se ha enviado con éxito al haber agregado un nuevo par con valor "pending" dentro del campo "eventsRequests" dentro del usuario objetivo.

A la hora de querer eliminar o abandonar un evento se hace uso de:

```
1 private void removeUserFromEvent(String eventId)
```

método por el cual se muestra un Snackbar solicitando la confirmación del usuario.

En caso afirmativo, se procede a ejecutar los siguientes métodos:

```
1 checkAndPromoteToAdmin(userId);  
2 noRegisteredUsersDelete(eventId);  
3 NotificationHelper.cancelAllWorkRequests(eventId);
```

1. Promociona al siguiente participante a administrador en caso de no existir ninguno.
2. Comprueba si quedan mas participantes en el evento, eliminando el evento por completo en caso contrario.
3. Cancela todas las notificaciones pendientes para dicho evento.

También se implementa un comparador personalizado para ordenar a los usuarios por roles:

```
1 public class UserRoleComparator implements Comparator<User>
```

6.4. EventEditorActivity

EventEditorActivity recibe mediante extras de la actividad anterior toda la información del evento para establecerla en los campos correspondientes.

Adicionalmente a EventCreationActivity, esta clase implementa un listener que abre un [DatePickerDialog](#) al pulsar sobre la fecha del evento.

Al guardar el evento se cancelan todos los trabajos pendientes para el evento en cuestión y se generan los necesarios con la nueva fecha mediante los métodos de NotificationHelper mencionados anteriormente.

7. TodayFragment

En esta actividad mostramos con ListView y el adaptador personalizado TodayListAdapter los eventos que tienen lugar en el día actual. La comparación se realiza mediante el uso de objetos [LocalDate](#).

También se implementa un [SwipeRefreshLayout](#) para poder refrescar la vista deslizando hacia abajo.

TodayListAdapter, a su vez, hace uso de RecyclerView para mostrar los participantes de cada evento sin tener que acudir a EventViewActivity, aunque sigue siendo posible si pulsamos sobre el evento.

8. NotesFragment

Se compone principalmente de un RecyclerView que implementa el adaptador personalizado NotesAdapter.

Es posible crear una nota pulsando el botón habilitado para ello, que lanza NotesActivity.

El almacenamiento de notas se gestiona mediante SharedPreferences con los siguientes métodos:

```
1     public ArrayList<Note> getNotes(SharedPreferences sharedNotes)  
2     public void deleteNote(SharedPreferences sp, String noteName)
```

Además, mediante un objeto ItemTouchHelper se permite eliminar notas deslizandolas hacia cualquiera de los lados.

9. Friends fragment

En esta sección se muestra mediante un RecyclerView y el adaptador personalizado FriendsAdapter la lista de amistades del usuario junto a dos botones, uno para buscar y agregar amistades mediante FriendSearcherActivity y otro para las solicitudes de amistad pendientes desde FriendRequestActivity. Éste último muestra un contador de las solicitudes pendientes en caso de existir.

9.1. FriendSearcherActivity

En esta actividad también empleamos un adaptador personalizado, FriendSearcherAdapter, y destacan principalmente los métodos:

```
1     private void searchUsers(String username)
```

que realiza una búsqueda de usuarios en la base de datos tomando como mínimo 3 caracteres (que deben ser los iniciales del nombre del usuario), normalizando el texto para quitar posibles acentos:

```
1     String databaseName = Normalizer.normalize(user.getName(),  
2                                         Normalizer.Form.NFD).replaceAll("\\p{M}", "");
```

y devolviendo los resultados que coincidan con la expresión regular:

```
1     String regex = "^" + introducedName + ".*";
```

Al pulsar en invitar, se ejecuta:

```
1     private void sendFriendRequest(String currentUserId, String targetUserId)
```

que comprueba si el usuario:

- No es una amistad ya agregada
- No somos nosotros mismos
- Si se ha enviado una petición previamente
- El estado de dicha petición

Tanto en el caso de ser null como al no existir, manda una petición estableciendo un par clave-valor en el campo "friend_requests" del usuario objetivo con el userId y el string "pending".

9.2. FriendRequestActivity

En ella mostramos de forma sencilla mediante un RecyclerView y el adaptador personalizado FriendRequestAdapter las peticiones de amistad pendientes.

Se toman todos los valores (si existen) del campo "friend_requests" del usuario y mediante los métodos:

```
1     private void acceptFriendRequest(String currentUserId, String targetUserId)  
2     private void rejectFriendRequest(String targetUserId)
```

agregamos o eliminamos los pares en cuestión con los métodos proporcionados por DatabaseReference.

9.3. FriendViewerActivity

En esta actividad se establece una vista mediante ViewPager2, TabLayout y el adaptador personalizado FriendsViewerPageAdapter, el cual muestra FriendInfoPageFragment o FriendEventListFragment según la pestaña en la que nos encontremos.

FriendInfoPageFragment nos habilita información detallada de la amistad junto a la posibilidad de eliminarlo como amistad, borrando el id de dicha amistad del campo "friends" del usuario. Se solicita confirmación previa mediante un Snackbar como ocurre con los eventos.

FriendEventListFragment muestra los eventos que tenemos en común con dicha amistad. Emplea la misma vista que EventOnCurrentDayActivity y nos permite acceder a los detalles con EventViewerActivity.

10. UserSettings

En esta sección tenemos disponible información detallada de la cuenta activa junto a un acceso a PreviousEventsActivity, un TextView para cerrar sesión y otro que nos permite cambiar nuestra imagen de perfil haciendo uso de los métodos:

```
1     private void uploadImage()
```

(prácticamente igual al que se emplea en los eventos salvo que la imagen se almacena en la carpeta "users/" en Firebase) y:

```
1     private void updateProfile(String imageUri)
```

que se encarga de cargar la imagen a través de:

```
1     UserProfileChangeRequest profileUpdate = new UserProfileChangeRequest.Builder()
2             .setPhotoUri(Uri.parse(imageUri)).build();
```

Para finalizar sesión, se ejecuta dentro del listener del correspondiente TextView:

```
1     FirebaseAuth.getInstance().signOut();
```

se borran las preferencias que almacenan las claves almacenadas por el botón "Remember me":

```
1     SharedPreferences sharedPreferences =
2             getSharedPreferences("login", MODE_PRIVATE);
3     sharedPreferences.edit().clear().apply();
```

se cancelan por completo todos los trabajos pendientes:

```
1     WorkManager.getInstance(getApplicationContext()).cancelAllWork();
```

y finalmente nos devuelve a LoginActivity.

10.1. PreviousEventsActivity

Muestra mediante un RecyclerView y el adaptador CalendarFragmentAdapter la lista de eventos previos a la fecha actual del usuario.

Pulsando en uno de ellos accederíamos a los detalles facilitados por EventViewerActivity.

10. Manual de usuario

Contenidos

- 11. Instalación y requisitos mínimos.**
- 12. Registro.**
- 13. Inicio de sesión.**
- 14. Configuración inicial.**
- 15. Secciones de la aplicación.**
- 16. Perfil de usuario.**
- 17. Eventos actuales.**
- 18. Calendario.**
- 19. Eventos.**
 - 19.1. Como crear un evento.**
 - 19.2. Detalles del evento.**
 - 19.3. Como eliminar un evento.**
 - 19.4. Como editar un evento.**
 - 19.5. Como agregar participantes.**
 - 19.6. Participantes y roles.**
 - 19.7. Notificaciones.**
- 20. Notas.**
- 21. Amistades.**
- 22. Problemas y preguntas frecuentes.**

1. Instalación y requisitos mínimos

Los requisitos mínimos para la instalación y uso de Eventia son:

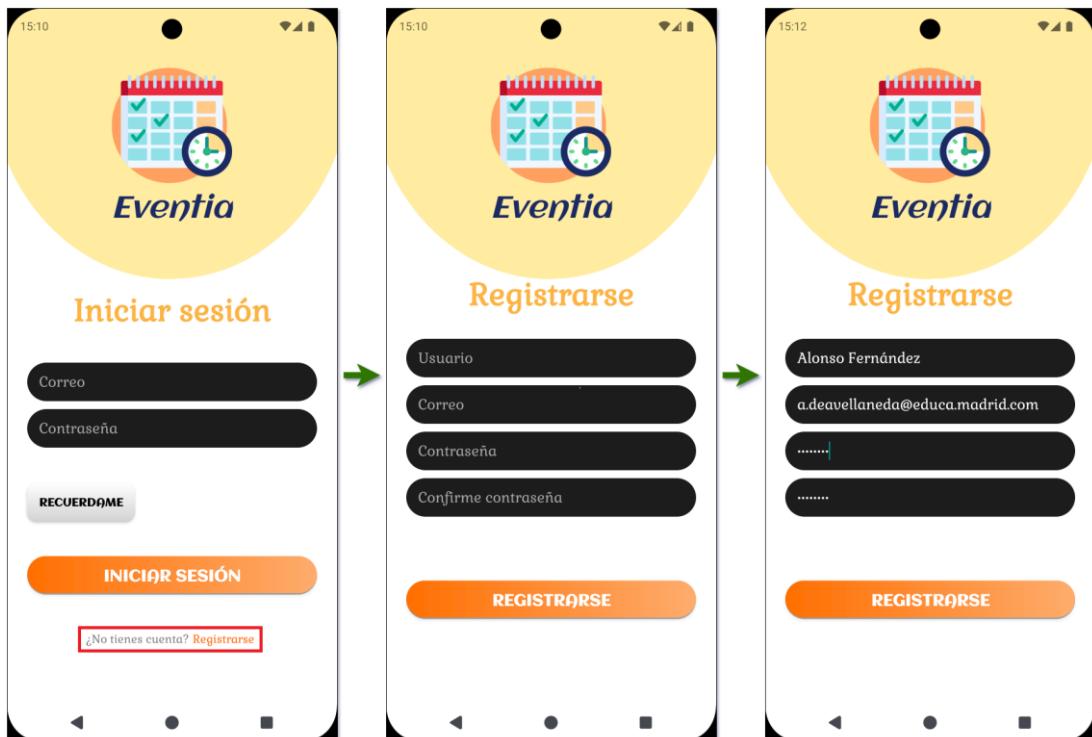
- Un dispositivo con Android Oreo o nivel de API 26.
- Conexión a internet.
- 50MB de memoria disponible.



2. Registro

Para poder emplear Eventia y todas sus funcionalidades debemos registrar una cuenta en la aplicación.

Al pulsar en “Registrarse” desde la pantalla inicial accederemos al formulario que debemos cumplimentar.



3. Inicio de sesión

A la hora de iniciar sesión debemos introducir los datos con los que nos dimos de alta.

Adicionalmente podemos dejar marcado el botón “Recuérdame” para que no sea necesario iniciar sesión de nuevo al finalizar la aplicación.

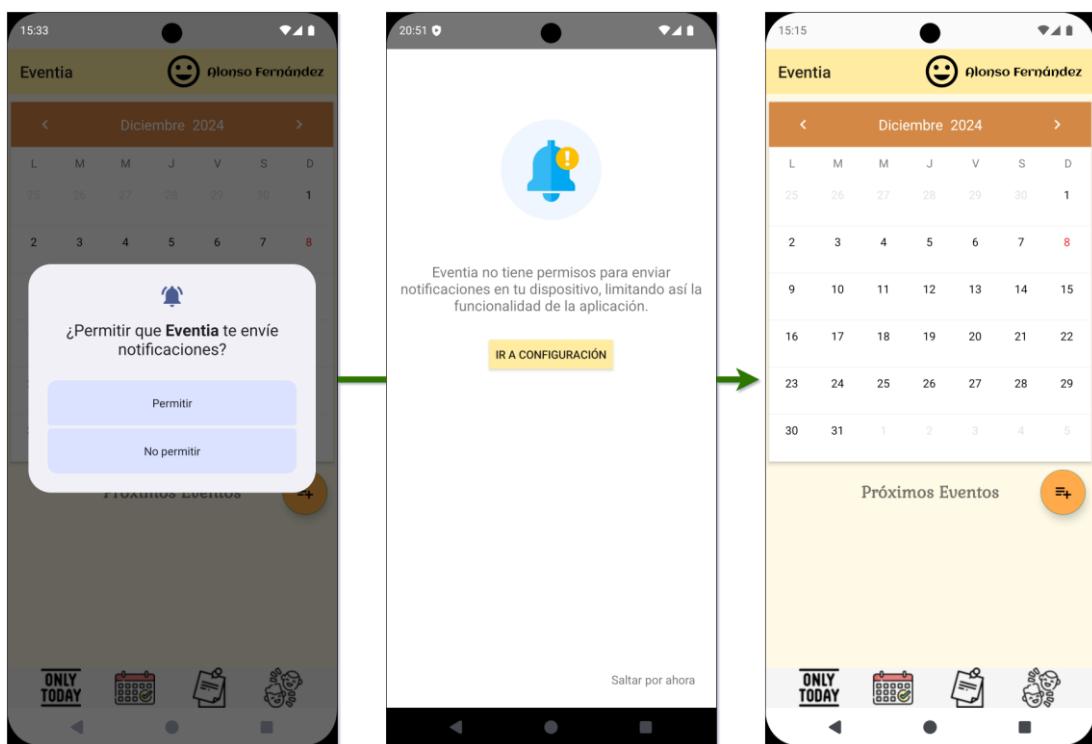


4. Configuración inicial

Al acceder por primera vez a Eventia se nos solicitará permisos para poder recibir notificaciones de los próximos eventos.

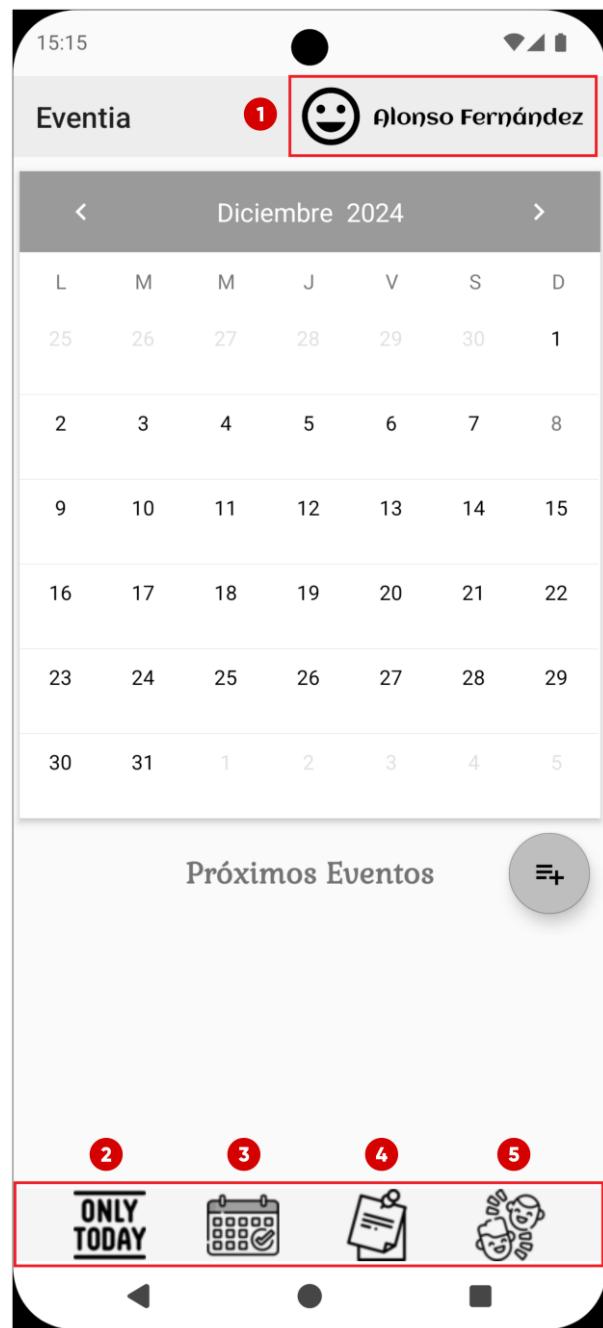
Dependiendo de la versión de Android que use el dispositivo (previo o posterior a Tiramisu), es posible que la pantalla inicial sea directamente la que aparece en la figura central.

En ambos casos terminaremos en la pantalla inicial de la aplicación.



5. Secciones de la aplicación

Eventia se compone principalmente de cinco secciones de navegación:



6. Perfil de usuario

Al acceder a los detalles de nuestro perfil distinguimos los siguientes elementos:

1. Eventos previos.
2. Cerrar sesión.
3. Datos del usuario.
4. Cambio de imagen de perfil.



6. Perfil del usuario

1. Eventos previos

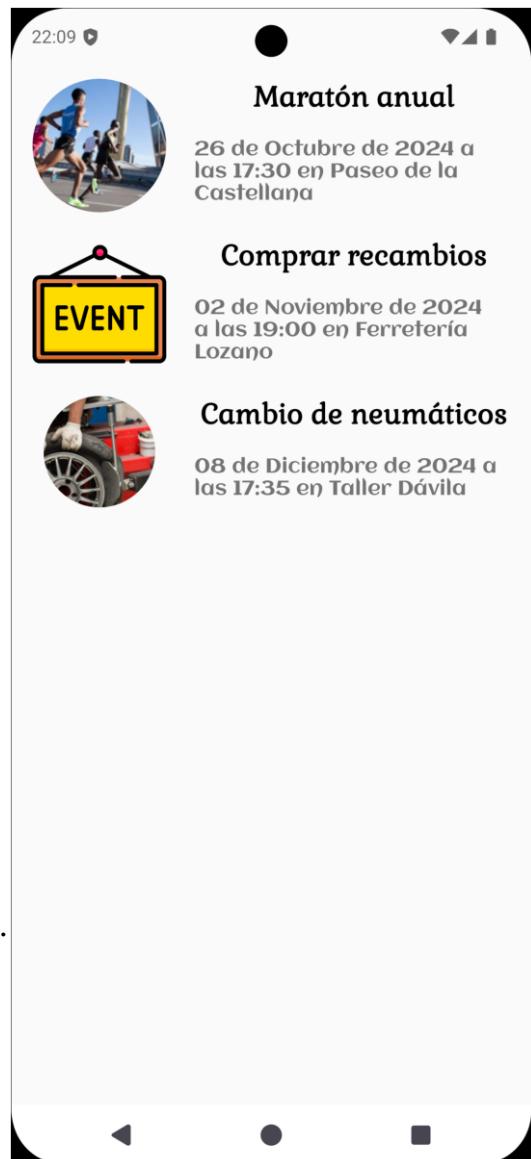
En esta sección se pueden consultar los eventos previos a la fecha actual.

2. Cerrar sesión

Al pulsar en “Cerrar sesión” finalizaremos la sesión de la cuenta activa, cancelando además todas las notificaciones pendientes.

3. Datos del usuario

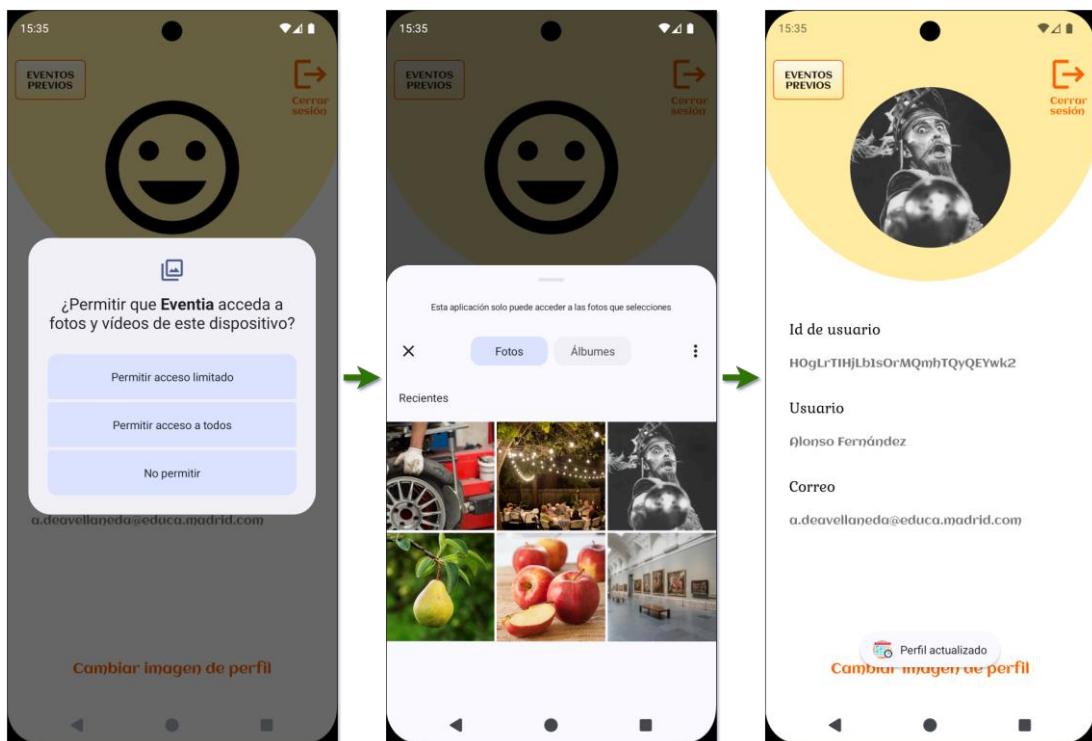
Contiene información relevante sobre el usuario.



6. Perfil del usuario

4. Cambio de imagen de perfil

Podemos cambiar la imagen de nuestro perfil al pulsar en el texto, se solicitarán permisos de acceso a los archivos del sistema previamente.



7. Eventos actuales

En esta sección se visualizan todos los eventos que han tenido o van a tener lugar en el día actual, así como sus participantes.



8. Calendario

En la pestaña “Calendario” diferenciamos tres secciones principales:

1. Calendario.
2. Próximos eventos.
3. Invitaciones a eventos.



8. Calendario

1. Calendario

Dentro de las funcionalidades del calendario se incluyen:

- Seleccionar una fecha para generar un evento.
- Visualizar fechas que contengan eventos.
- Consultar eventos de una fecha dada.

Los días con eventos previos se indican mediante un fondo gris, mientras que los eventos futuros son señalados en color azul.

2. Próximos eventos

En este listado se muestran todos los eventos que tendrán lugar desde la fecha y hora actuales en adelante.

3. Invitaciones a eventos

Sección desde la cual se accede a las peticiones de eventos pendientes.

9. Eventos

A la hora de generar un nuevo evento distinguimos los siguientes atributos:

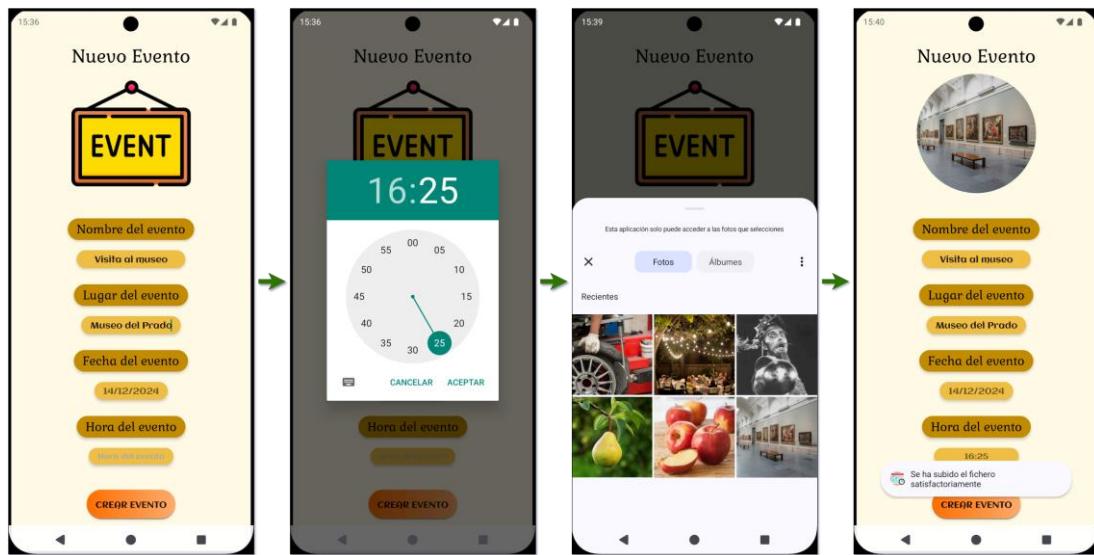
1. Una imagen (opcional).
2. Título.
3. Lugar.
4. Fecha y hora.



9.1. Como crear un evento

Salvo la imagen, el resto de campos son obligatorios.

Una vez rellenado los datos, podemos pulsar en “Crear evento” para registrarlos en el calendario.



Una vez creado, podemos consultar la información de un evento si pulsamos en:

- La fecha en el calendario.
- La lista de próximos eventos.
- Desde eventos actuales (de ser el caso).
- Desde eventos previos (de ser el caso).
- Eventos en común con nuestras amistades.

Esto nos llevará a la vista en detalle.

9.2. Detalles del evento

Dentro de la misma, disponemos de varias funcionalidades:

1. Eliminar el evento.
2. Editar el evento.
3. Agregar amistades al evento.
4. Lista de participantes y roles.



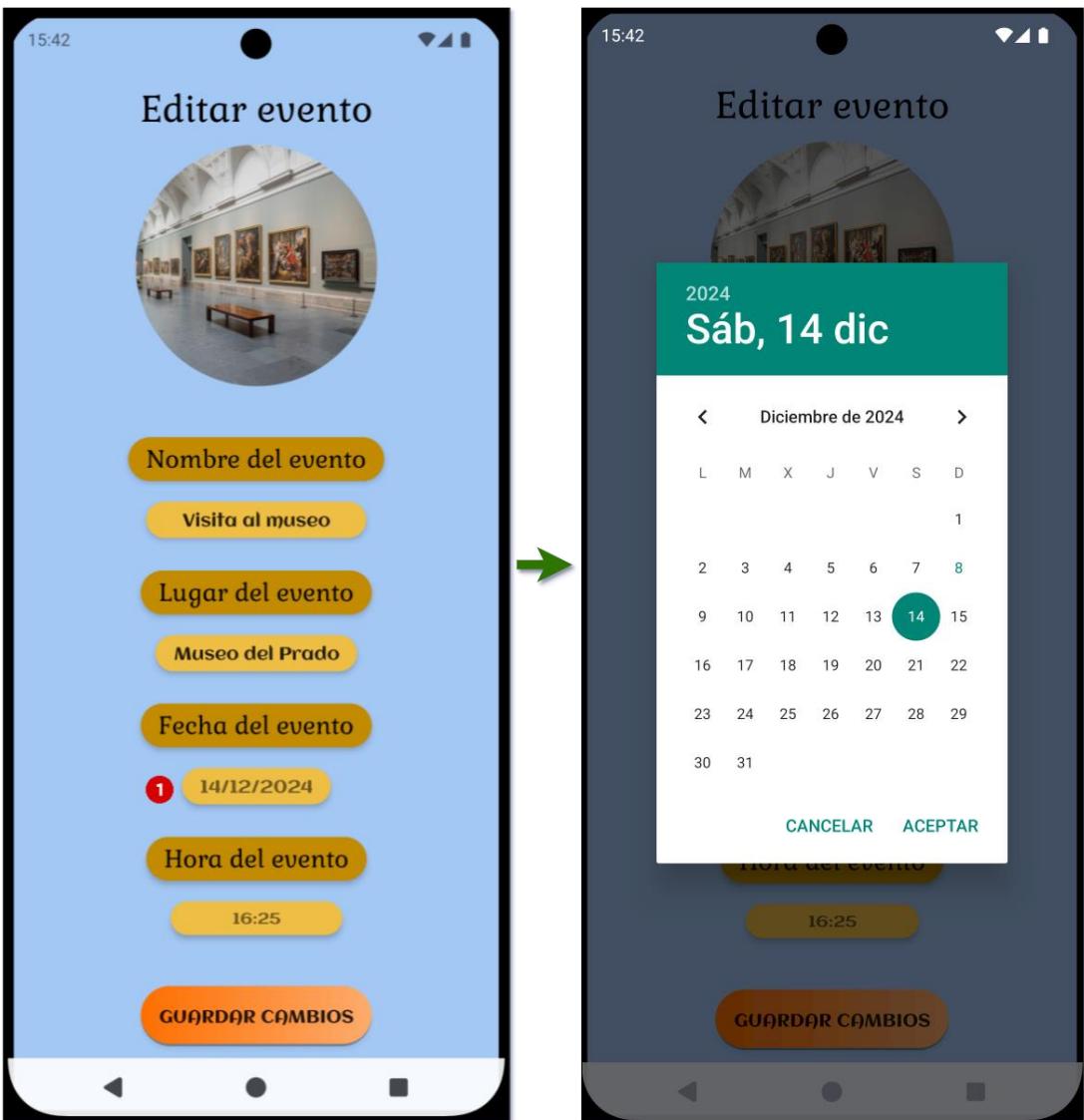
9.3. Como eliminar un evento

Al pulsar en eliminar un evento se nos solicitará una confirmación. Si la aceptamos, saldremos del evento.

Si además somos los últimos integrantes de dicho evento, éste se eliminara por completo.

9.4. Como editar un evento

Editar un evento emplea las mismas funcionalidades que la creación, incluyendo la posibilidad de cambiar la fecha del mismo.



9.5. Como agregar participantes

Podemos agregar amistades a cualquier evento en el que tengamos permisos de anfitrión pulsando sobre el botón de la esquina inferior derecha.

Una vez aceptada la invitación, se mostrará en el listado de participantes.



9.6. Participantes y roles

En ella se distingue mediante un marco a quienes tienen permisos de anfitrión, y por tanto, edición y control sobre el resto de participantes.

Si mantenemos seleccionado a uno de los participantes se mostrarán las opciones:

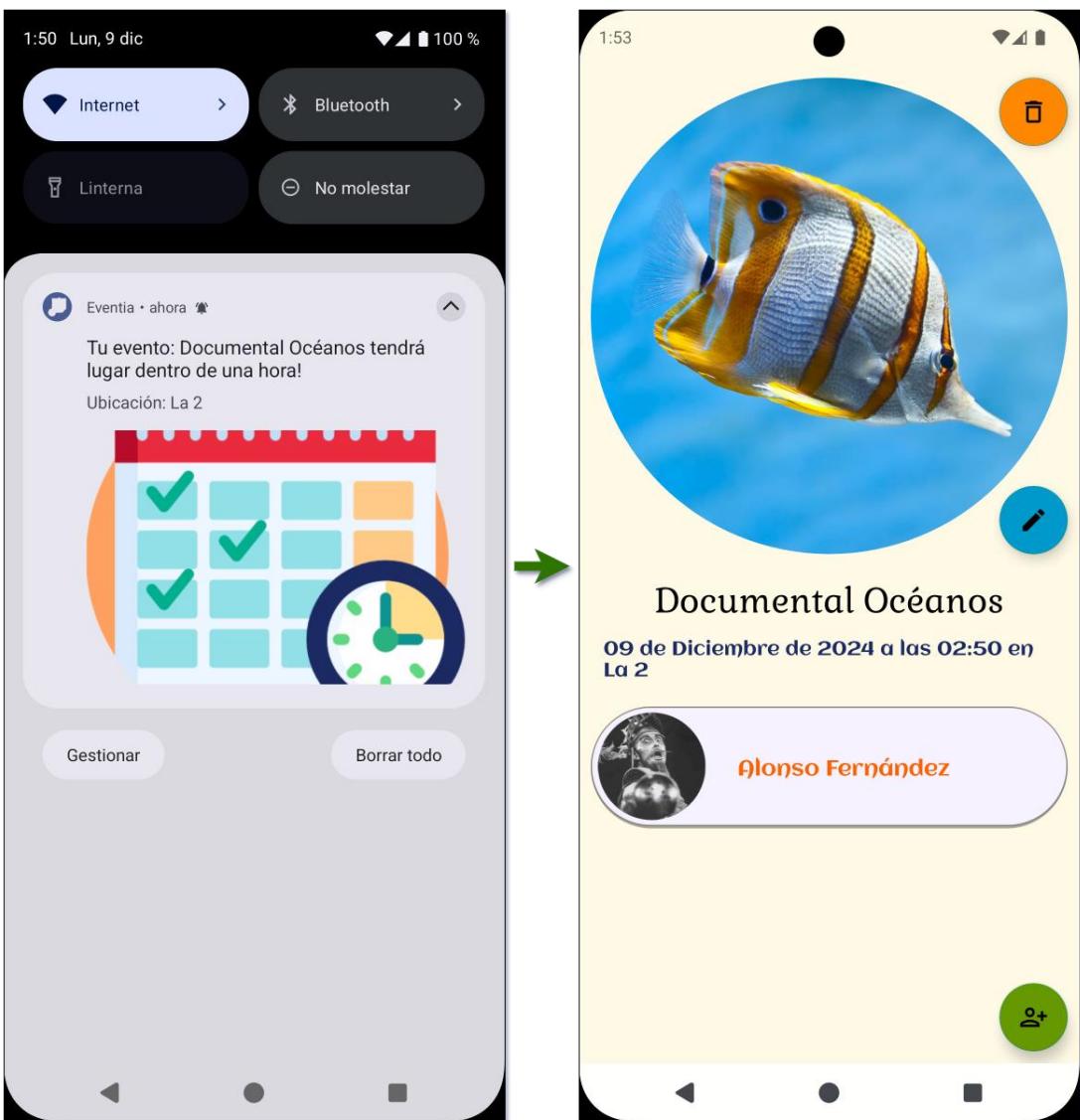


9.7. Notificaciones

Los eventos enviarán las siguientes notificaciones en caso de aplicar según el tiempo restante:

- 7 días antes.
- 1 día antes.
- 1 hora antes.
- Al cumplirse la hora del evento.

Al pulsar sobre la notificación nos llevará directamente a los detalles del evento.

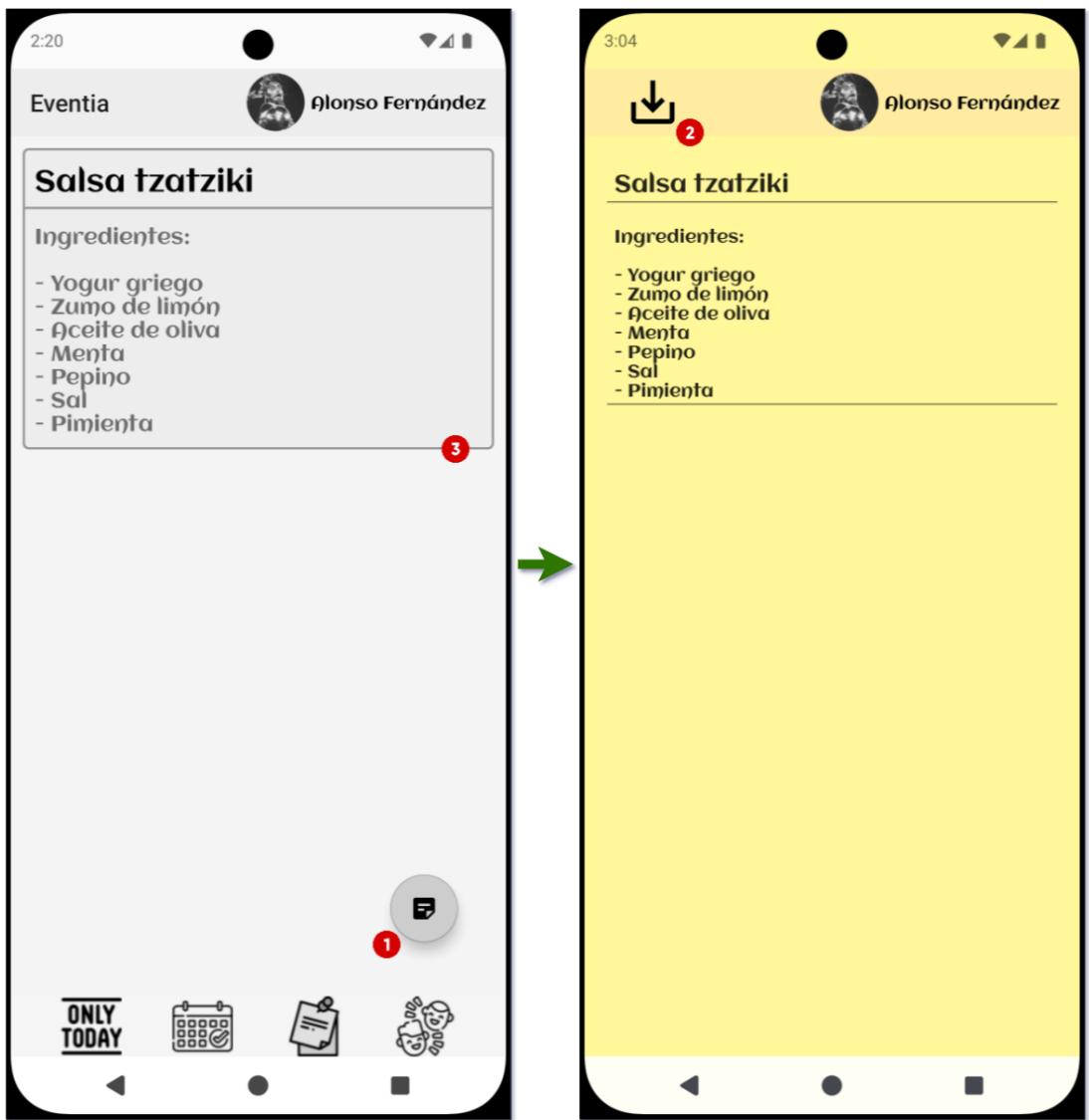


10. Notas

En esta sección puedes crear y gestionar anotaciones del día a día.

- Pulsando el botón (1) podrás generar nuevas anotaciones.
- Al pulsar sobre una de ellas (3) accedemos a la sección de edición.
- Una vez finalizada, podemos guardar pulsando el botón (2).

Para eliminarlas basta con deslizarlas a uno de los lados.



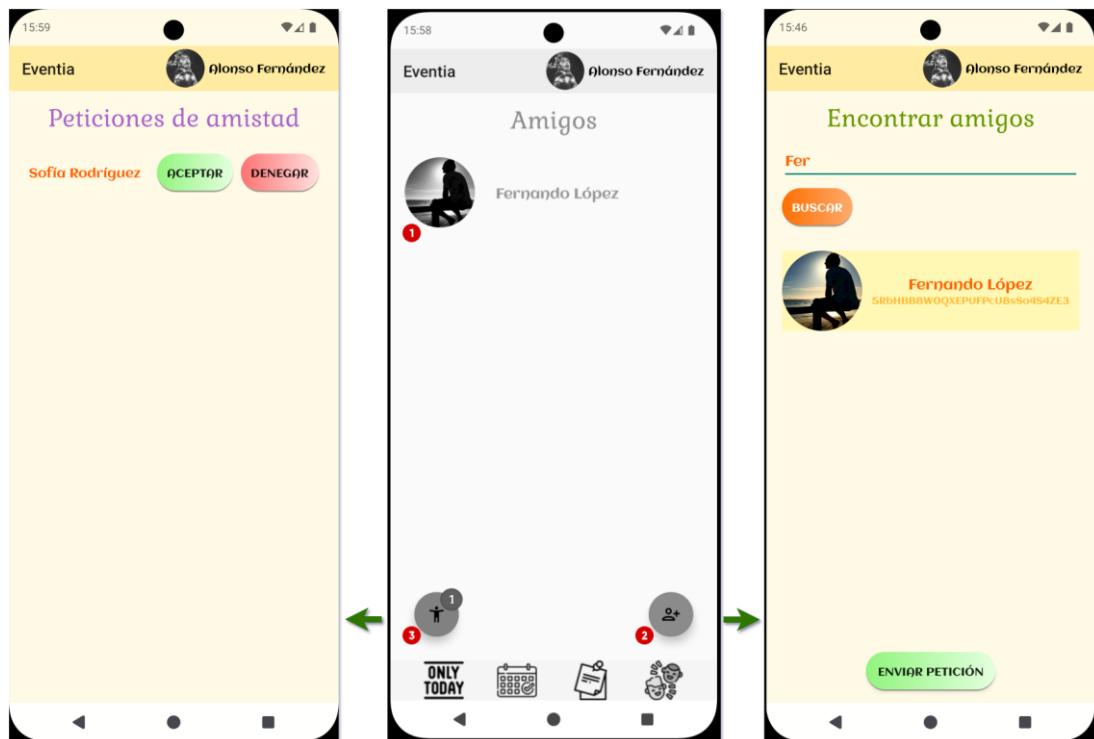
11. Amistades

En esta sección nos es posible añadir, consultar y gestionar nuestras amistades dentro de Eventia.

Se distinguen tres elementos principales:

1. Listado de amistades.
2. Buscador de usuarios.
3. Peticiones de amistad.

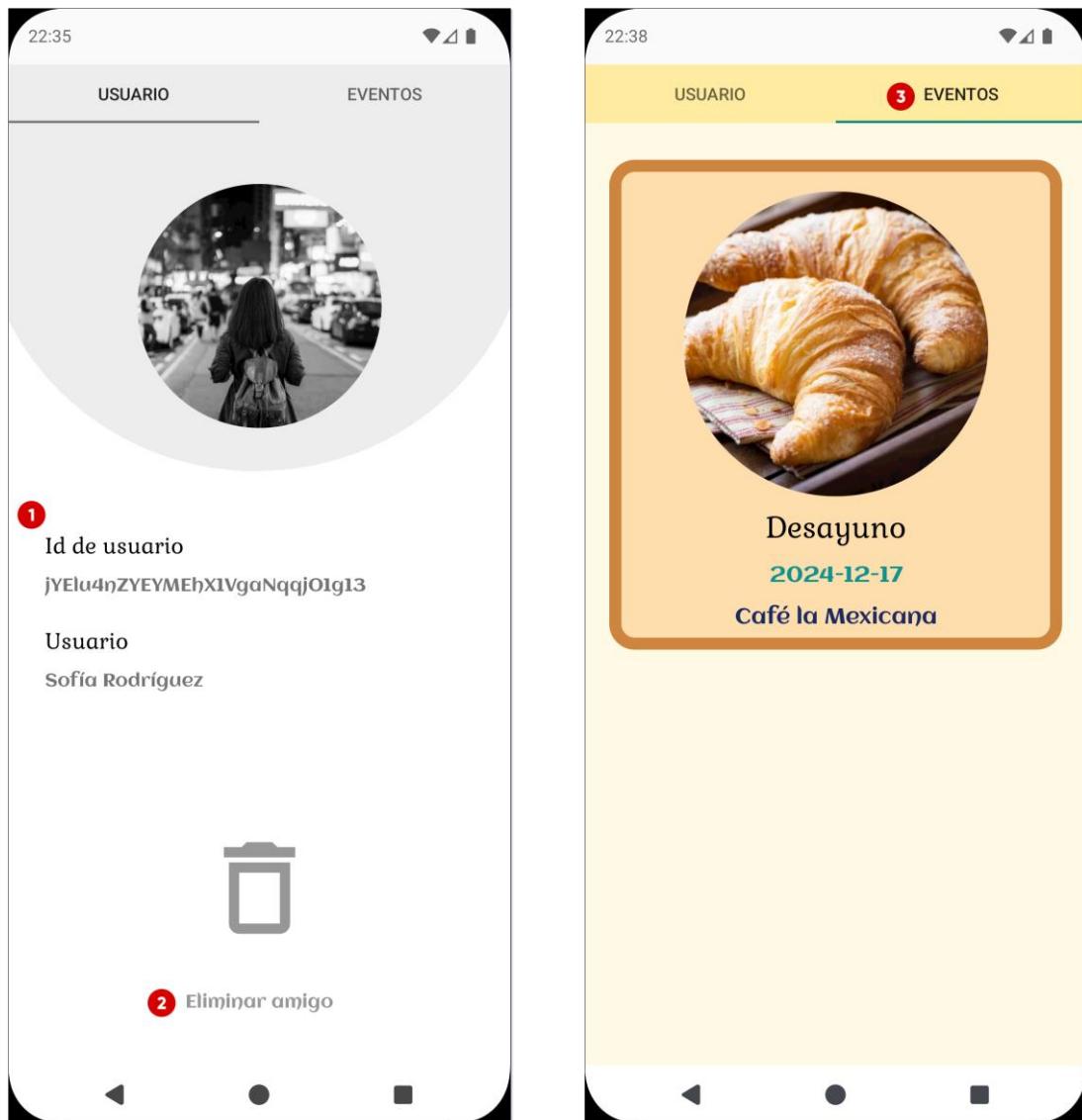
Hay que tener en cuenta que a la hora de realizar búsquedas se requieren como mínimo tres caracteres.



11. Amistades

Podemos consultar en detalle la información de nuestras amistades al pulsar sobre su perfil.

Adicionalmente, tenemos la opción de eliminar dicha amistad (2) y consultar los eventos que tengamos en común (3).



12. Problemas y preguntas frecuentes

No recibo notificaciones de mis eventos, ¿a que se debe?

- Es probable que tengas desactivadas las notificaciones que envía Eventia en tu dispositivo. Puedes consultar las mismas manteniendo pulsado el icono de Eventia y seleccionando “información”.

Al intentar subir una imagen aparece el mensaje “permiso denegado”.

- Igual que en el caso anterior, es posible que hayas denegado los permisos de forma permanente para el acceso que requiere Eventia a los archivos en tu dispositivo.

¿Si reinicio mi teléfono, seguiré recibiendo notificaciones?

- Todos los eventos futuros que se sincronizaron en tu último acceso a Eventia se notificarán, aunque las modificaciones de un evento por parte de un tercero no se sincronizarán hasta acceder a la aplicación de nuevo, incluyendo notificaciones.
- Además, señalar que los dispositivos que empleen algún sistema de ahorro de batería o limitación del sistema pueden tener impacto en el comportamiento de Eventia.

11. Objetivos por implementar

Eventia ya ofrece una amplia gama de funcionalidades para gestionar eventos, notas y amigos siempre hay espacio para mejorar y expandir la aplicación. A continuación, se presentamos algunos objetivos clave que podrían implementarse en futuras versiones para mejorar aún más la experiencia del usuario y la funcionalidad general de la aplicación:

- **Sincronización con calendarios externos**

Una primera propuesta de mejora es integrar la aplicación con servicios de calendario externos como Google Calendar u Outlook, para sincronizar eventos y evitar conflictos de programación. Consiguiendo con ello, la importación y exportación de eventos y una sincronización bidireccional de estos.

- **Creación de la página web de la aplicación**

Para complementar la experiencia de usuario y proporcionar una plataforma accesible desde cualquier dispositivo, se ha propuesto la creación de una página web para Eventia. Esta página web serviría como una extensión de la aplicación móvil, ofreciendo funcionalidades clave y proporcionando información relevante sobre la aplicación.

- **Mejoras en la gestión de amigos**

También sería buena opción, añadir funcionalidades adicionales para la gestión de amigos, como la capacidad de ver el estado en línea de los amigos, enviar mensajes directos, y crear grupos de amigos.

- **Funciones avanzadas de búsqueda y filtrado**

Sería recomendable en un futuro, implementar funciones avanzadas de búsqueda y filtrado para que los usuarios puedan encontrar rápidamente eventos. Se podrían usar, por ejemplo, palabras clave y filtrados por fechas o ubicaciones.

- Integración con redes sociales

Una buena integración sería permitir a los usuarios compartir eventos y notas directamente en sus redes sociales favoritas. Pudieran ser opciones como compartir en redes sociales como Facebook, Twitter o Instagram y la previsualización de publicaciones antes de compartir.

- Soporte Multilingüe Adicional

Si la aplicación tuviera éxito, sería recomendable expandir el soporte de la aplicación para incluir más idiomas además del inglés y español. Añadir un selector de idiomas en las configuraciones de la aplicación o soporte para idiomas RTL como el árabe y el hebreo.

- Integración con Asistentes Virtuales

Una función actual y avanzada, que supondría un largo desarrollo, sería permitir la integración con asistentes virtuales como Google Assistant y Amazon Alexa para gestionar eventos y notas mediante comandos de voz. Algunas de sus funciones podrían ser comandos de voz para añadir, editar y eliminar eventos y notas, consultar el calendario y notas a través de asistentes virtuales o tener notificaciones y recordatorios basados en comandos de voz.

- Mejoras en el Login

Una vez despegase la aplicación y se hiciera con ámbito mercantil, sería bueno implementar una funcionalidad que permita a los usuarios recuperar sus contraseñas olvidadas de manera sencilla y segura, principalmente cuando estuviese la página web, para poder usar todas las funcionalidades de Firebase Authenticator para cambiar contraseñas de forma segura.

Estos objetivos por implementar no solo buscan ampliar las funcionalidades de Eventia, sino también mejorar la experiencia del usuario y mantener la aplicación relevante y competitiva en el mercado. Con estas mejoras, Eventia podría ofrecer un servicio más completo y adaptado a las necesidades de sus usuarios.

12. Conclusiones

Los objetivos iniciales, planteados en el anteproyecto, se han superado, resultando en un proyecto de mayor escala de lo esperado. A pesar de que algunos compañeros abandonaron el proyecto por diversas circunstancias después de planificar una escalabilidad mayor, logramos avanzar hasta un punto satisfactorio de desarrollo.

La organización planificada de tareas temporizadas en el tiempo fue fundamental para alcanzar el objetivo. Utilizamos un fichero compartido donde registrábamos las herramientas utilizadas y las mejoras a implementar. Aunque finalmente no se recibió feedback y no se actualizó por parte de los integrantes, se consiguió llevar el proyecto a término exitosamente.

Durante el desarrollo, hemos ampliado y superado nuestros conocimientos, aplicando lo aprendido en el grado superior y enriqueciéndolo con el material disponible en internet. Empresas de software como Google, desarrolladores que contribuyen en FullStack y GitHub han sido fundamentales al compartir soluciones, aplicaciones y herramientas. Estos recursos no solo nos ayudaron a completar el proyecto, sino que también los hemos interiorizado para futuros desarrollos.

Un factor limitante significativo fue la falta de compromiso y el abandono del grupo por parte de algunos integrantes debido a diversas causas. Esto no solo ralentizó el progreso en múltiples ocasiones, sino que también generó desafíos adicionales tras haber acordado los objetivos y las tareas del proyecto. A pesar de estos obstáculos, el equipo que permaneció logró adaptarse y continuar adelante, demostrando resiliencia y capacidad de gestión en situaciones adversas.

Como recomendaciones para futuros proyectos, serían asegurar que la documentación del proyecto sea actualizada regularmente y realizada por todos los integrantes del equipo, para mantener toda la información accesible y organizada, así como, desarrollar planes de contingencia para abordar posibles problemas como la falta de compromiso de algunos miembros del equipo. Es crucial identificar soluciones alternativas y recursos adicionales que puedan ser utilizados en caso de necesidad.

Futuras líneas de investigación podrían ser investigar y aplicar técnicas de optimización para mejorar el rendimiento y la escalabilidad de la aplicación, también explorar la integración de tecnologías emergentes para mejorar la funcionalidad y usabilidad de la aplicación, así como ampliar la aplicación a otras plataformas y dispositivos para aumentar su accesibilidad y alcance.

A lo largo de este proyecto, hemos aprendido la importancia de la planificación, la colaboración y la adaptabilidad en el desarrollo de aplicaciones. Las habilidades adquiridas y las lecciones aprendidas durante este proceso serán valiosas para futuros proyectos y para nuestro desarrollo profesional continuo.

13. Bibliografía y fuentes de información

- Firebase Authentication: <https://firebase.google.com/docs/auth>
- Firebase Realtime Database: <https://firebase.google.com/docs/database>
- Firebase Cloud Storage: <https://firebase.google.com/docs/storage>
- Android Developer: <https://developer.android.com/>
- WorkManager: <https://developer.android.com/topic/libraries/architecture/workmanager>
- Persistent Work: <https://developer.android.com/develop/background-work/background-tasks/persistent>
- Task Stack Builder:
<https://developer.android.com/reference/android/app/TaskStackBuilder>
- NotificationCompat.Builder:
<https://developer.android.com/reference/androidx/core/app/NotificationCompat.Builder>
- LocalDate: <https://developer.android.com/reference/java/time/LocalDate>
- Doze and App Standby: <https://developer.android.com/training/monitoring-device-state/doze-standby>
- RecyclerView: <https://developer.android.com/develop/ui/views/layout/recyclerview>
- Ciclo de vida: <https://developer.android.com/guide/components/activities/activity-lifecycle>
- Fragment lifecycle: <https://developer.android.com/guide/fragments/lifecycle>
- GitHub: <https://github.com/>
- Git: <https://git-scm.com/>
- Gradle: https://docs.gradle.org/current/userguide/gradle_basics.html
- Entorno de desarrollo de Android Studio: <https://developer.android.com/studio>
- Material Design de Google: <https://m3.material.io/>
- Applandeo Material Calendar: <https://github.com/Applandeo/Material-Calendar-View>
- Glide: <https://github.com/bumptech/glide>
- PlantUML: <https://plantuml.com/>

Agradecimientos

Finalmente, nos gustaría dedicar un apartado especial a todas las personas que, con su esfuerzo y apoyo, han hecho posible la finalización de este proyecto:

Yosef Guillermo Karam Müller:

- La primera persona a la que quiero agradecerle todo lo que ha hecho por mí es a mi madre. Gracias a ella, combinar el trabajo, las prácticas y el proyecto ha podido ser una realidad. Su apoyo constante y motivación en los momentos más cruciales han sido fundamentales para poder dar lo mejor de mí en este proyecto.
- La segunda persona es el tutor de las prácticas José María. Gracias a él he conseguido aprender muchísimo y he podido aplicar mucho de lo aprendido al proyecto. Aunque no creo que él lea nunca estas palabras, no podía olvidar su gran aporte al proyecto y a mi vida personal gracias a sus enseñanzas.
- La tercera persona es mi amigo Miguel Ángel. Él ha sido siempre un referente para mí, como persona, como trabajador y como estudiante. Él fue quien me motivó a realizar estos estudios, dedicándome yo a un campo muy distinto, y gracias a él he conseguido terminarlos, gracias a su motivación incansable hasta en los momentos más complicados.
- La cuarta persona es mi amiga Alba María. Sin ella no habría sido posible nada de esto. Fue ella quien me ha motivado durante años a conseguir finalizar estos estudios y me ha apoyado en todo momento para que no flaquease ante las vicisitudes que se han presentado durante la realización del grado superior durante estos tres años y durante el proyecto.
- Por último, quiero destacar la labor de mi hermana. Para mí, es un ejemplo de constancia y tesón, una referente que me ha guiado siempre por el mejor de los caminos. Sin su ayuda y su soporte no podría haber terminado en pie lo que para mí ha sido un arduo camino. Muchas gracias.

Vladyslav Rosiyan:

- Me gustaría dedicar los esfuerzos de este trabajo a mi familia, en especial a mi madre y a mi abuela, quienes nunca han dejado de confiar en mi y me han prestado su apoyo incondicional en todo momento.
- También a todos los amigos que han sido partícipes desde el comienzo de esta experiencia, particularmente a Nuran, que ha visto gran parte de todo el proceso y siempre me brindó una perspectiva diferente en los momentos mas complicados.
- A Paloma, por haberme acogido como a uno mas y mostrarme el buen humor con el que hay que afrontar la vida.
- A todo el equipo docente y trabajadores implicados, que con tanto esfuerzo mantienen la enseñanza pública, gratuita y de calidad. Gracias por brindarme esta oportunidad.
- A mi compañero Yosef, por haber sido el hilo conductor de este proyecto desde el comienzo y no cesar en su labor a pesar de todas las complicaciones.