

DESARROLLO AVANZADO DE SOFTWARE

4º curso, grupo 46

Segundo cuatrimestre

PROYECTO GRUPAL: FESTUP

Aingeru Bellido, Nagore Gómez, Sergio Martín, Maitane Urruela

Bilbao, 19 de mayo de 2024

ÍNDICE

Índice

1. Descripción de la aplicación	3
1.1. Requisitos	3
2. Arquitectura del proyecto y sus clases	4
3. Características	6
3.1. Servidor	6
3.1.1. Documentación interactiva de <i>FastAPI</i>	7
3.2. Comunicación cliente-servidor	9
3.3. Base de datos	9
3.4. Uso de imágenes	10
3.5. <i>Intents</i> implícitos	11
3.6. Integración de códigos QR	12
3.7. Uso de <i>Google Maps</i> y geolocalización	13
3.8. Uso de preferencias	13
3.9. Uso de Content Provider	13
3.10. FCM	14
3.11. Diferencias entre orientación vertical y horizontal	14
3.12. Login automático	14
3.13. <i>Widget</i>	16
3.14. Trabajo en segundo plano	17
4. Manual de usuario	18
4.1. Inicio de sesión	18
4.2. Feed del usuario	19
4.3. Pantalla de búsqueda	20
4.4. Mapa y listado de eventos	21
4.5. Perfil de usuario	22
4.6. Perfil de cuadrilla	24
4.7. Pantalla del evento	25
4.8. Ajustes	27
4.9. <i>Widget</i>	28

ÍNDICE DE FIGURAS

Índice de figuras

1.	Arquitectura de datos.	4
2.	Diseño de los servicios presentes en el servidor.	7
3.	Documentación interactiva de <i>FastAPI</i>	7
4.	OAuth de <i>FastAPI</i>	8
5.	Respuesta de token de acceso caducado	8
6.	Diseño de ambas bases de datos. Los campos marcados en azul solo están disponibles en la base de datos remota.	9
7.	Diálogo con botones para compartir código de acceso a una cuadrilla por WhatsApp o Telegram.	11
8.	Pantalla de compartir (figura 8a) y pantalla de escanear (figura 8b) código QR para añadirse a una cuadrilla.	12
9.	La aplicación abrirá en la pantalla de la figura 9a y navegará al la pantalla de la figura 9c si el usuario ha iniciado sesión previamente o a la pantalla de la figura 9b en caso contrario.	15
10.	Diferentes estados del widget.	16
11.	Pantalla de inicio de sesión y registro de la aplicación	19
12.	Feed de los eventos de usuario (figura 12a) y los eventos recomendados (figura 12b).	20
13.	Pantallas de búsqueda.	21
14.	Mapa de eventos (figura 14) y lista de eventos (figura 14b).	22
15.	Perfiles de usuario.	23
16.	Otras opciones del perfil del usuario identificado.	24
17.	Perfiles de cuadrilla.	25
18.	Pantallas de evento.	26
19.	Pantalla de preferencias.	27
20.	Diferentes estados del <i>widget</i>	28

1. Descripción de la aplicación

FestUp es una aplicación que permite a los usuarios y a sus cuadrillas explorar los eventos y fiestas a su alrededor, obteniendo información detallada sobre los mismos así como de las personas y grupos de amigos que planean asistir a cada uno.

Una de las características principales de **FestUp** es su capacidad para personalizar la experiencia de cada usuario, pudiendo guardar sus cuadrillas y los eventos a los que asistirá tanto de forma individual como acompañado. Además, los usuarios pueden ver a qué eventos están planeando asistir sus amigos, lo que facilita la coordinación y la planificación de actividades en grupo.

La aplicación de **FestUp** se encuentra en el siguiente *enlace*:

<https://github.com/Maits27/FestUp.git>

Y la API correspondiente en el siguiente *enlace*:

<https://github.com/AingeruBeOr/FestUp-server.git>

Al servidor con la base de datos remota se accede mediante el siguiente comando:

```
ssh -i /path/a/clave/privada <usuario>@34.71.128.243
```

1.1. Requisitos

- **Versión de Android:** la versión para la cual se ha desarrollado **FestUp** es API 34 (Android 14) aunque como requisito mínimo se requiere tener API 26 (Android Oreo 8.0).
- **Conexión a Internet:** la aplicación usa una base de datos remota para la que será indispensable tener conexión a Internet en todo momento.
- **WhatsApp o Telegram instalado:** la aplicación utiliza un *Intent* (implícito) para mandar el código de acceso a la cuadrilla por WhatsApp o por Telegram. Para probar esta funcionalidad, será necesario tener instalada una de las aplicaciones (o ambas).
- **Aplicación de calendario en el dispositivo:** la aplicación nos permite añadir los eventos en el calendario de nuestro dispositivo para lo que es necesaria una aplicación de calendario.
- **Varios contactos en el teléfono:** la aplicación permite buscar amigos que usen **FestUp** en la libreta de contactos del teléfono. Para ello, hace falta tener contactos en el teléfono y que alguno de ellos tenga cuenta en **FestUp**. Para probar esta funcionalidad, se puede registrar un usuario con el número de teléfono de uno de los contactos de la libreta de contactos.
- **Permitir los permisos:** es imprescindible aceptar los permisos que se solicitan al iniciar la aplicación para poder probar todas las funcionalidades de la misma.

2. Arquitectura del proyecto y sus clases

La arquitectura de los datos está dividida en las capas recomendadas por Android en [10]. En la figura 1, se muestra el flujo de datos ordenado en las capas de:

- **Capa de IU (Interfaz de Usuario):** en esta capa se encuentran todos los elementos Componibles de *JetPack Compose* y los *ViewModel* que gestionan los eventos en la IU.
- **Capa de datos:** en esta capa, los repositorios se encargan de obtener datos de diferentes fuentes, como la base de datos local, el *DataStore* o servidores en la nube mediante clientes HTTP.

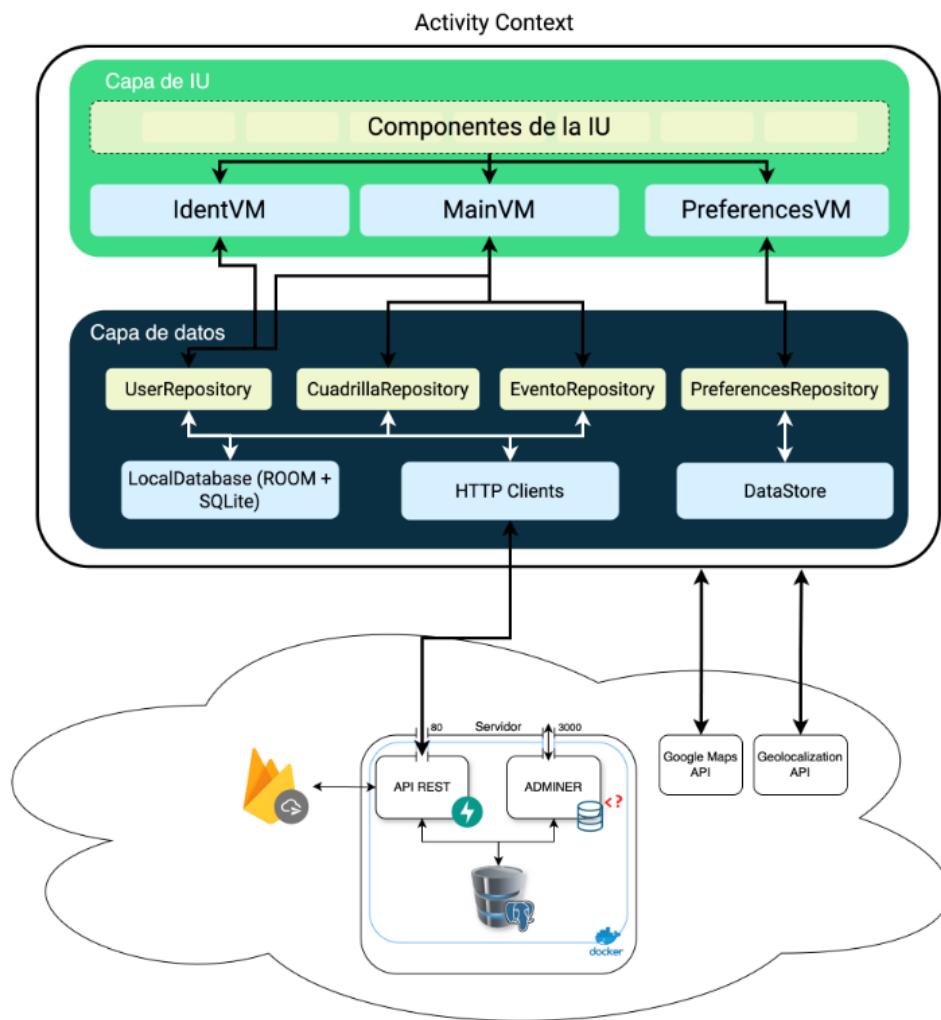


Figura 1: Arquitectura de datos.

Sin embargo, se ha prescindido de la capa de dominio [5] debido a que el flujo desde los repositorios a la capa de IU es directo.

Por otra parte, el código fuente de la aplicación está organizado acorde con la arquitectura de datos en su mayoría. La estructura de este está organizada en paquetes que siguen la siguiente distribución, donde se describen los paquetes y archivos más relevantes:

- **com.gomu.festup** : Paquete principal.
 - **FestUpApp.kt** : Archivo principal de la aplicación *FestUp*.
 - **MainActivity.kt** : Actividad principal.
 - **data** : Capa de datos de la aplicación. Archivos relacionados con la obtención de datos de diferentes fuentes.
 - **http** : Paquete con archivos para la conexión al servidor mediante HTTP.
 - ◊ **httpClients.kt** : Clientes “AuthClient” y “HttpClient”.
 - ◊ **dataModels.kt** : Modelos de transferencia de datos mediante HTTP.
 - **localDatabase** : Paquete de archivos para la gestión de la base de datos local.
 - ◊ **DAO** : DAOs (Data Access Object) que proporcionan acceso mediante *Room* a la base de datos.
 - ◊ **entities** : Entidades de la base de datos.
 - ◊ **database.kt** : Archivo principal de la base de datos.
 - **repositories** : Repositorios para la conexión a diferentes fuentes de datos.
 - **com.gomu.festup.ui** : Componentes gráficos.
 - **elements** : Componentes de la actividad principal.
 - ◊ **components** : Componentes comunes reutilizables de la aplicación.
 - ◊ **screens** : Pantallas de la aplicación.
 - ◊ **theme** : Archivos de configuración del tema de la aplicación, así como los colores y la tipografía.
 - ◊ **widget** : Componentes del *widget* de la aplicación.
 - **vm** : *ViewModels* de la aplicación. Se encargan de transmitir los datos y a los elementos de IU y de responder a los eventos de estos.
 - **com.gomu.festup.di** : Archivos de configuración de la inyección de dependencias mediante Hilt.
 - **com.gomu.festup.utils** : Archivos de herramientas y utilidades reusables desde diferentes partes del código.
 - **com.gomu.festup.alarmMng** : Archivos de gestión de alarmas.

3 CARACTERÍSTICAS

3. Características

3.1. Servidor

Para el servidor de la aplicación se ha empleado el servicio de *Cloud Computing* (IaaS) de GCP (*Google Cloud Platform*), en el cual se ha instanciado una máquina virtual a modo de servidor.

Los componentes principales del servidor son la API REST y la base de datos (más adelante en la sección 3.3). La API REST se ha implementado usando el *framework FastAPI* [14] en Python. Con este, se definen todos los *endpoints* que requiere la aplicación para acceder a la base de datos mediante HTTP. Además, genera automáticamente documentación interactiva para la API, lo que simplifica el desarrollo de esta.

La API REST hace uso de librerías de *Python* para almacenar las contraseñas de los usuarios *hash*-eadas y *SQLAlchemy* [19] para proporcionar un ORM para la base de datos. También, como se tratará más adelante en la sección 3.10, desde este servicio de la API REST se usa el servicio de mensajería de *Firebase FCM* [15].

Además, se ha usado el protocolo *OAuth* (*Open Authorization*) que es un estándar de autorización que permite restringir las acciones que la aplicación puede realizar en los recursos del servidor en nombre del usuario, sin compartir nunca las credenciales. Se ha utilizado este protocolo para gestionar la identificación de los usuarios, generando y validando códigos de acceso.

Cuando un usuario se identifica, se genera un código de acceso y otro de actualización. El token de acceso actúa como una credencial temporal que se incluye en cada petición HTTP al servidor para demostrar que el usuario está autorizado a realizar dicha acción. El token de actualización se utiliza para obtener un nuevo token de acceso una vez que el actual ha expirado. Esto permite que el cliente mantenga una sesión activa sin requerir que el usuario vuelva a autenticarse con sus credenciales.

Por otra parte, se ha usado la herramienta *Adminer* [1], una interfaz gráfica mediante la cual es posible acceder a la base de datos y hacer gestiones sobre ella a través de una interfaz gráfica.

Para el despliegue de estos servicios, se ha usado Docker [13] en un entorno de *Docker Compose*. El diseño de la conexión de estos servicios entre sí y el exterior se resume en la figura 2.

Además de todo esto, el servidor implementa una tarea de *crontab* para eliminar todos los eventos con una fecha pasada. Esto sirve para no acumular datos que no se van a usar en el servidor, especialmente las imágenes, que son lo que más espacio ocupan.

3 CARACTERÍSTICAS

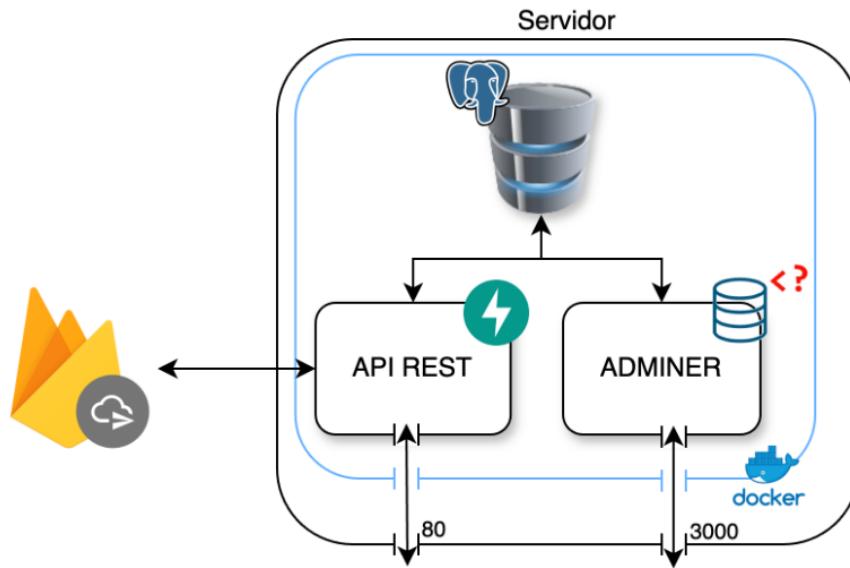
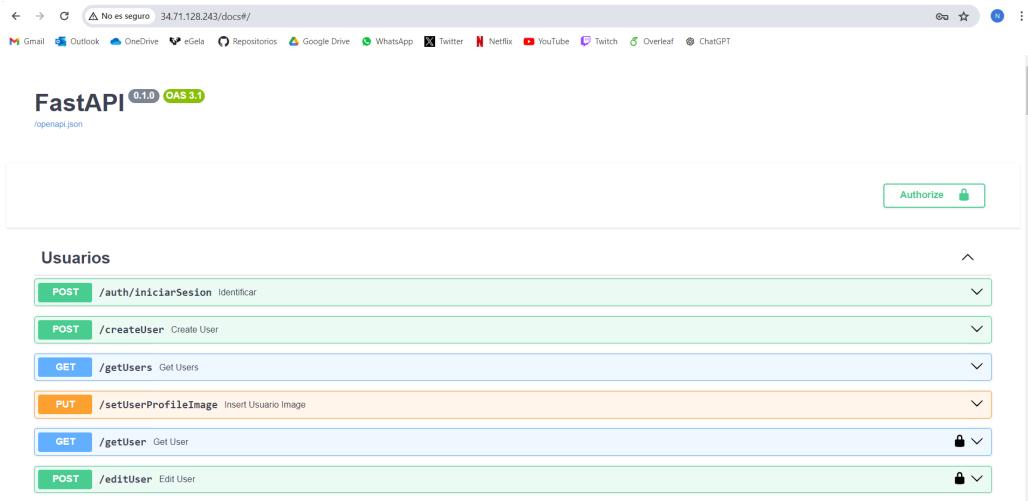


Figura 2: Diseño de los servicios presentes en el servidor.

3.1.1. Documentación interactiva de *FastAPI*

Como se ha mencionado, *FastAPI* proporciona una documentación interactiva que se encuentra en la dirección IP del servidor (34.71.128.243) para probar todas las peticiones HTTP creadas, como se muestra en la figura 3. Al acceder a la dirección IP será redirigido automáticamente a `/docs`.



La captura de pantalla muestra la interfaz web de la documentación de *FastAPI*. En la parte superior, se indica la versión 0.1.0 y CAS 3.1. La barra de navegación incluye enlaces a Gmail, Outlook, OneDrive, eGela, Repositorios, Google Drive, WhatsApp, Twitter, Netflix, YouTube, Twitch, Overleaf y ChatGPT. La sección principal titulada 'Usuarios' enumera las siguientes rutas:

- POST /auth/iniciarSesion**: Identificar
- POST /createUser**: Create User
- GET /getUsers**: Get Users
- PUT /setUserProfileImage**: Insert Usuario Image
- GET /getUser**: Get User
- POST /editUser**: Edit User

En la parte superior derecha, hay un botón 'Authorize' con un candado.

Figura 3: Documentación interactiva de *FastAPI*

3 CARACTERÍSTICAS

Para poder probar las peticiones marcadas con un candado abierto es necesario identificarse, al haber utilizado el protocolo OAuth mencionado. Para ello, es necesario pulsar el botón "Authorize" (seguido de un candado) que se muestra en la parte superior derecha e ingresar un nombre de usuario y una contraseña válidos (si no se dispone de una cuenta, se puede crear en la petición *createUser*), posteriormente se debe pulsar el botón "Authorize", como se muestra en la figura 4. De esta manera, los candados de las peticiones se cerrarán y podrán ser probadas.

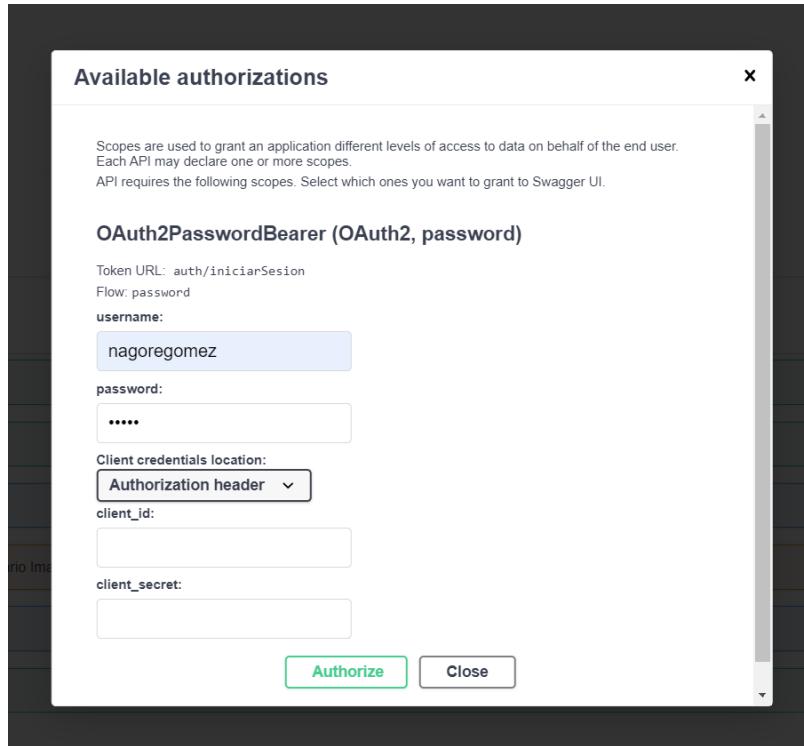


Figura 4: OAuth de *FastAPI*

Es importante destacar que 5 minutos después de identificarse, el token de acceso caducará y las peticiones devolverán la respuesta mostrada en la figura 5. Será necesario volver a identificarse en el protocolo OAuth para poder realizar las peticiones de nuevo.

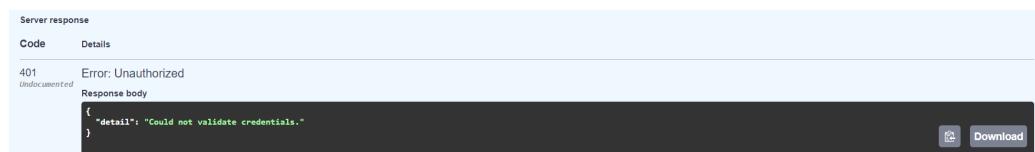


Figura 5: Respuesta de token de acceso caducado

3 CARACTERÍSTICAS

3.2. Comunicación cliente-servidor

En esta aplicación se ha empleado *Ktor* [16] para gestionar las comunicaciones entre la aplicación y el servidor remoto mediante peticiones HTTP. Para ello, se utiliza la clase "HttpClient" de *Ktor*, creando instancias de clientes HTTP que permiten enviar solicitudes a los *endpoints* de la API de *FastAPI*. Estos clientes se configuran con opciones específicas, como el tiempo de espera y los encabezados de solicitud, según el tipo de petición que se requiera hacer.

Una vez que se envían las solicitudes, *Ktor* maneja la comunicación con la API de manera eficiente y transparente, permitiendo recibir las respuestas correspondientes. Estas respuestas incluyen datos en formato JSON, entre otros, que luego pueden procesarse en la aplicación *Kotlin*.

3.3. Base de datos

El diseño de la base de datos que se ha usado para persistir los datos de la aplicación se puede ver en la figura 6. Ambas bases de datos son muy similares, aunque por razones de optimización de flujo de datos y tiempo, algunos datos solo se almacenan en la base de datos remota.

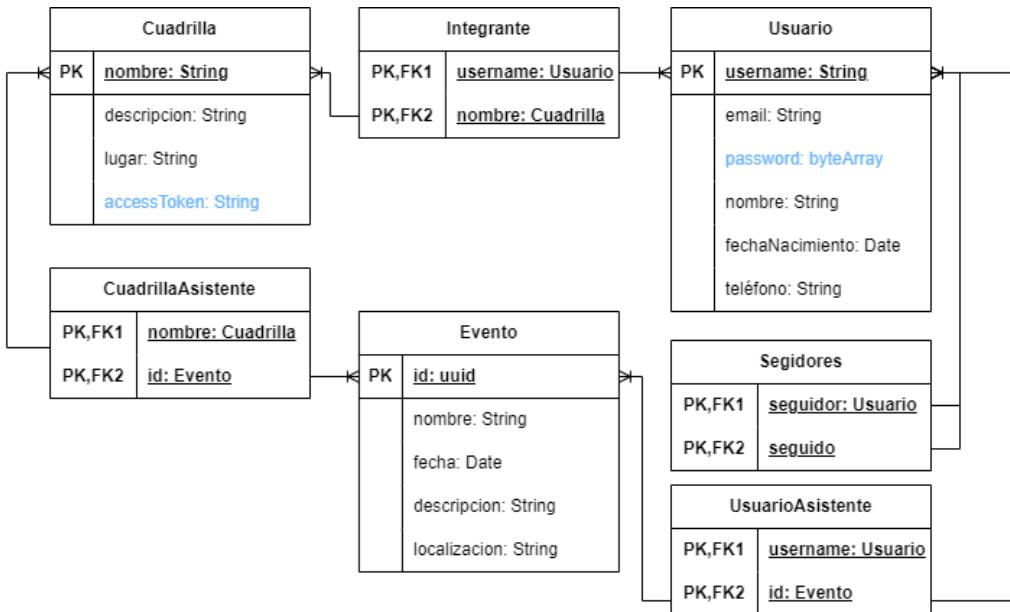


Figura 6: Diseño de ambas bases de datos. Los campos marcados en azul solo están disponibles en la base de datos remota.

La base de datos remota se ha implementado sobre el SGDB (Sistema Gestor de Bases de Datos) relacional PostgreSQL [18].

Para la base de datos local en el entorno Android, se ha usado *Room* [3], un complemento de *Jetpack Compose* que ofrece una capa de abstracción sobre el SGDB relacional *SQLite* [20], sistema de base de datos que se utiliza en Android.

Además de aislar las interacciones de la aplicación con la base de datos, ofrece diferentes herra-

3 CARACTERÍSTICAS

mientas que permiten actualizar las vistas de forma programada cuando se realizan cambios en los datos.

3.4. Uso de imágenes

Debido a los casos de uso de la aplicación, se ha decidido hacer uso de imágenes para los siguientes casos:

- **Foto de perfil del usuario.** Esta foto la podrá elegir por primera vez en la pantalla de registro y podrá editarla más adelante desde su perfil dentro de la aplicación.
- **Foto de perfil de la cuadrilla.** Al igual que con la foto de perfil de un usuario, se podrá modificar a la hora de crear la cuadrilla y modificarla.
- **Foto del evento.** Esta imagen se establecerá a la hora de crear el evento.

Estas imágenes se han guardado en el servidor en tres directorios específicos. Para poder tenerlas asignadas a un usuario, cuadrilla o evento, el nombre de la imagen contendrá el identificador único de esa instancia, por lo que no hará falta guardar el directorio de la imagen en la base de datos ni ningún otro tipo de relación.

Para cargar las fotos, se utilizada el *PhotoPicker* [12] implementado por Android. Este ofrece la posibilidad de cargar una imagen de manera sencilla desde el álbum de fotos de nuestro dispositivo.

Por otra parte, las fotos se cargan desde el servidor usando *Coil* [2] (*Coroutine Image Loader*). Esta librería externa a Android, permite recuperar fotos de Internet de una manera ágil, ya que se encarga ella misma de gestionar todo el proceso en segundo plano, como por ejemplo, la memoria caché, las animaciones de carga y el estado de la solicitud. Además, es la recomendada por Android [11].

Gracias a *Coil*, se ha podido establecer una política correcta de caché para los casos de uso de la aplicación. La aplicación hará la solicitud de la imagen al servidor, siempre y cuando sea una imagen de foto de perfil, tanto de cuadrilla como de usuario.

Por el contrario, la imagen del evento no se volverá a solicitar, al no poder ser modificada desde la aplicación. De esta manera, las imágenes de los eventos cargan al momento.

3 CARACTERÍSTICAS

3.5. *Intents* implícitos

En la aplicación se hace uso de *Intents* implícitos para enviar el código de acceso a una cuadrilla mediante WhatsApp o Telegram. De esta manera, se puede compartir este código fácilmente y permitir que nuevos usuarios formen parte de una cuadrilla ya existente.

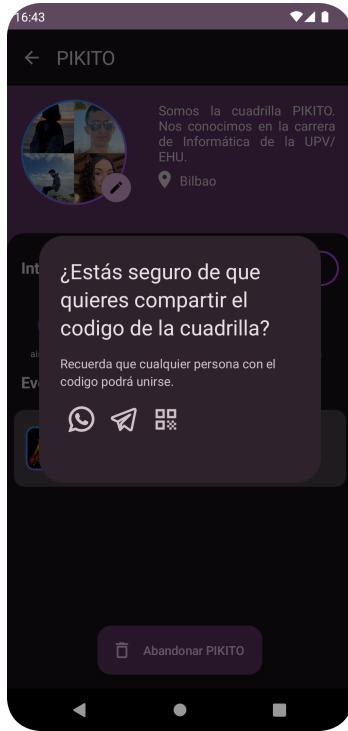


Figura 7: Diálogo con botones para compartir código de acceso a una cuadrilla por WhatsApp o Telegram.

3 CARACTERÍSTICAS

3.6. Integración de códigos QR

En *FestUp* se ha integrado una librería llamada *ZXing* [21] para permitir compartir las cuadrillas a través de códigos QR. Si bien esta funcionalidad ya estaba disponible mediante códigos numéricos que se compartían por WhatsApp o Telegram, se considera más rápida y cómoda la opción de unirse a otras cuadrillas tan solo escaneando un código con la cámara del teléfono.

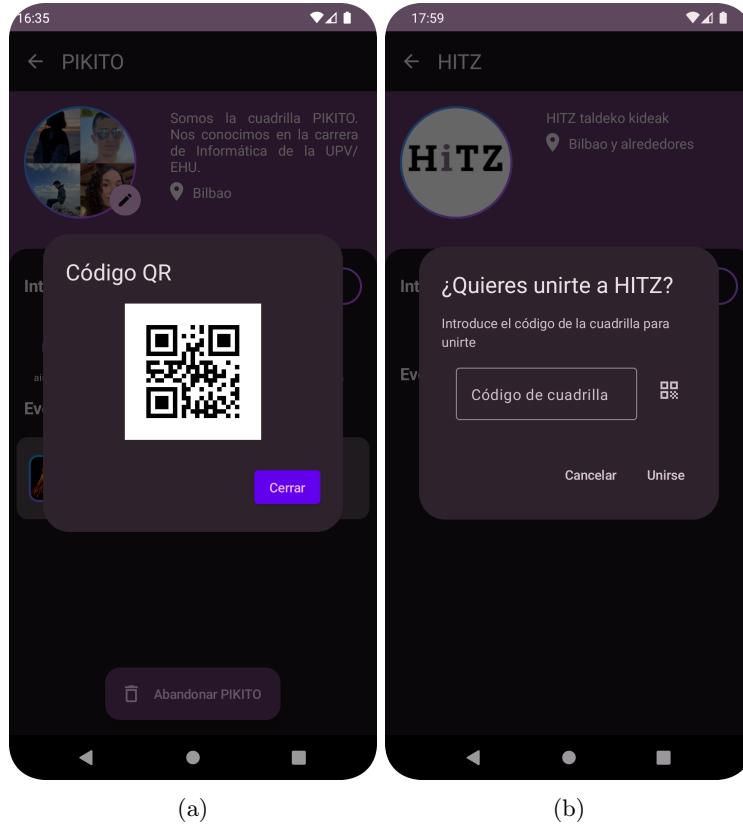


Figura 8: Pantalla de compartir (figura 8a) y pantalla de escanear (figura 8b) código QR para añadirse a una cuadrilla.

Zxing (*Zebra Crossing*) es un paquete disponible para el desarrollo en Android que permite la lectura y generación de diferentes tipos de códigos, tales como códigos de barra, códigos QR y muchos otros formatos.

El funcionamiento de *ZXing* se basa en la captura de imágenes a través de la cámara del dispositivo, seguido del procesamiento de esas imágenes para detectar y decodificar los códigos presentes.

La librería utiliza algoritmos de análisis de imagen para identificar los patrones característicos de los códigos de barras y QR, y luego aplica técnicas de decodificación para extraer la información contenida.

ZXing también ofrece integración sencilla con las aplicaciones a través de APIs, permitiendo a

3 CARACTERÍSTICAS

los desarrolladores implementar funcionalidades de escaneo con un mínimo de esfuerzo. Además, proporciona herramientas para la generación de códigos, facilitando así la creación de aplicaciones completas que puedan tanto leer como producir códigos de barras y QR.

3.7. Uso de Google Maps y geolocalización

Para la visualización de los eventos en la pantalla de explorar se ha incluido una vista de mapa. Este mapa se despliega en pantalla gracias al SDK de *Google Maps* para Android [17].

El mapa contiene marcadores en los puntos donde tendrán lugar los eventos, ofreciéndole al usuario un despliegue gráfico de estos a su alrededor. Al mismo tiempo, con el objetivo de centrar la vista en los alrededores de la ubicación del usuario, se ha hecho uso de los servicios de geolocalización. De este modo, en el mapa, la cámara se coloca sobre un pequeño círculo azul que indica la posición del usuario en tiempo real.

Para situar los marcadores en el mapa, se precisan las coordenadas de latitud y longitud. Teniendo en cuenta que las ubicaciones de los eventos son cadenas de texto introducidas por los usuarios, se emplea un objeto *Geocoder* [8] para obtener dichas coordenadas a partir del texto introducido por el usuario.

Finalmente, con la intención de evitar ubicaciones erróneas para los eventos, se ha incluido en la pantalla de adición de los mismos un pequeño mapa que indica el punto que le corresponde a la ubicación introducida al pulsar el botón de lupa.

3.8. Uso de preferencias

Con el fin de ofrecer un servicio y una experiencia más personalizada, se permite a los usuarios guardar sus propias preferencias en los dispositivos donde se use la aplicación. Estos ajustes son personalizados para cada uno de los usuarios y se almacenan en el *DataStore*, de modo que son datos persistentes a nivel dispositivo.

Las preferencias se distribuyen en dos secciones: ajustes del sistema y ajustes de visualización. Las preferencias de visualización incluyen la selección del idioma (castellano y euskera disponibles), así como la posibilidad de seleccionar diferentes temas (claro y oscuro disponibles).

Por otro lado, en el apartado de ajustes del sistema se encuentra la opción para habilitar o deshabilitar las notificaciones de la aplicación. Es importante recordar que para que esta configuración cobre sentido, la aplicación debe tener permisos para emitir notificaciones.

3.9. Uso de Content Provider

FestUp es una aplicación que, además de ofrecer funcionalidades limitadas al entorno de la aplicación, también incluye interacciones con algunas de las aplicaciones que vienen por defecto en los dispositivos Android. Anteriormente, se han mencionado los *intents* implícitos; no obstante, no son la única herramienta que contribuye a ello.

3 CARACTERÍSTICAS

La aplicación emplea *Content Providers* [7] para permitir que el usuario pueda guardar los eventos en alguno de los calendarios locales que incluya su dispositivo. Sin embargo, no todos los dispositivos incluyen calendarios locales y, por lo tanto, en dichos casos se trata de añadir el evento en *Google Calendar* asociado a la cuenta de Google que haya registrada en el dispositivo. De ser así, el evento se añade al calendario y se sube a la nube, permitiendo la visualización del evento en todos los dispositivos que tengan acceso a ese mismo calendario.

Por otra parte, también mediante el uso de *Content Providers* se recogen los contactos que se almacenan en el dispositivo del usuario (figura 16b) y se contrastan con los usuarios que se han registrado previamente en la aplicación. La finalidad de esto último es poder mostrar una pantalla donde un usuario pueda consultar si alguno de sus contactos tiene cuenta en la aplicación para poder seguirlo.

3.10. FCM

En una aplicación como *FestUp*, es importante la interacción entre los usuarios. Por ello, mediante el servicio de *Firebase Cloud Messaging* (FCM) se han implementado notificaciones personalizadas para cada usuario en función de las personas a las que sigue en la aplicación.

Para hacer esto, se han creado diferentes canales de difusión a los que están suscritos los usuarios en función de las personas a las que siguen. Básicamente, cuando un usuario comienza a seguir a otro, automáticamente se le suscribe al canal de este último, y cuando el usuario seguido se apunta a algún evento, se les notifica tal acción a todas las personas que lo siguen.

3.11. Diferencias entre orientación vertical y horizontal

Es evidente que la distribución de los elementos de la IU debe ser adaptativa en función de la orientación que tenga el dispositivo al momento de operar sobre la aplicación. Por ello, *FestUp* es compatible tanto con la orientación vertical como horizontal. Tanto en el formato *landscape* como en el formato *portrait*, las funcionalidades de la aplicación son completamente accesibles, lo que permite realizar las mismas acciones independientemente de la orientación.

En *AndroidView* se requiere de algo conocido como *fragments* para modular este tipo de comportamientos; sin embargo, en *JetPack Compose*, cada uno de los elementos composable funcionan como un *fragment*.

3.12. Login automático

Como se ha explicado en anteriores secciones, es imprescindible tener un usuario para poder acceder a la aplicación, de tal forma que sería necesario introducir valores para estos campos cada vez que se desea usar la aplicación.

Con la intención de hacer este proceso más rápido y cómodo para el usuario, se ha implementado un servicio automático de inicio de sesión. De esta forma, solo será necesario iniciar sesión la primera

3 CARACTERÍSTICAS

vez que se acceda a **FestUp**; en las siguientes ocasiones, se cargará directamente la pantalla principal porque el proceso de inicio de sesión se realiza en segundo plano.

Para el proceso en segundo plano, se utiliza una pantalla de carga que comprueba si el usuario ya ha iniciado sesión anteriormente para decidir si abrir la aplicación o redirigir al inicio de sesión. Estas pantallas se pueden ver en la figura 9.

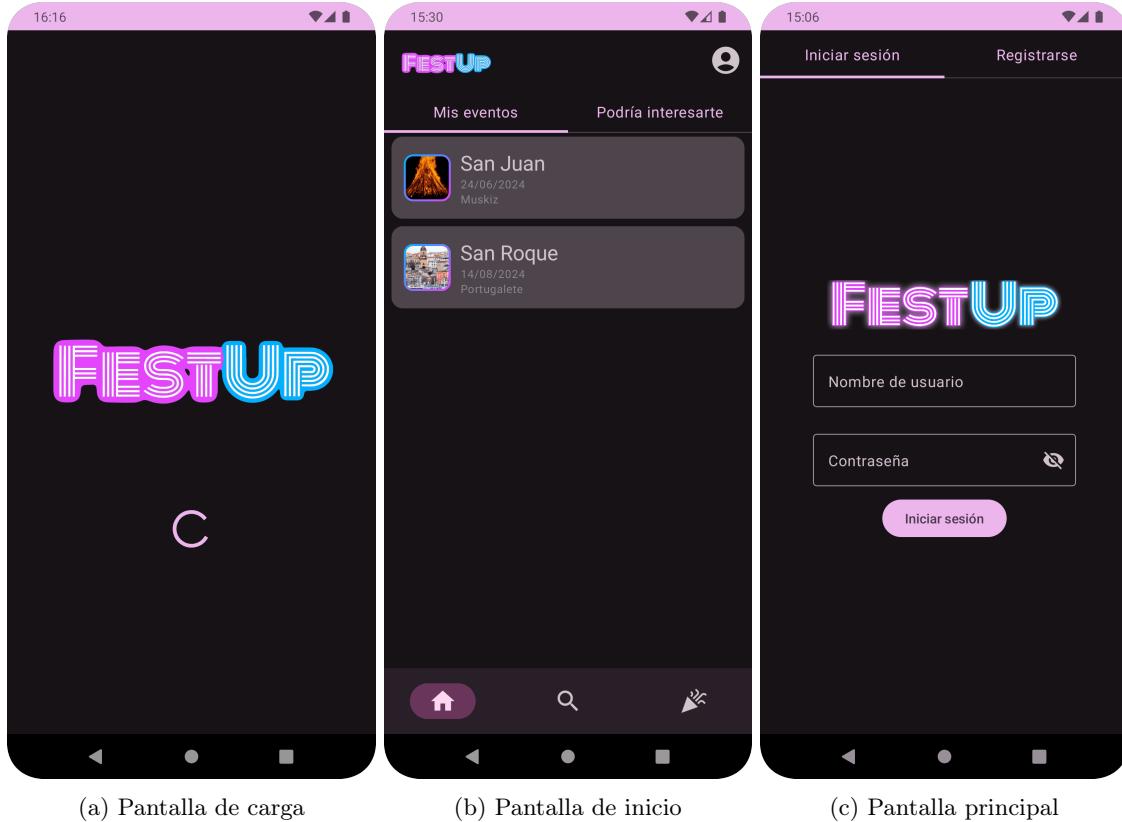


Figura 9: La aplicación abrirá en la pantalla de la figura 9a y navegará al la pantalla de la figura 9c si el usuario ha iniciado sesión previamente o a la pantalla de la figura 9b en caso contrario.

Por último, el servicio se cancela cuando un usuario desde dentro de la aplicación cierra sesión. En este caso, la próxima vez que ese usuario quiera acceder a **FestUp** deberá introducir su usuario y contraseña en el formulario de login.

3 CARACTERÍSTICAS

3.13. Widget

El *widget* de la aplicación, sirve para mostrar los próximos eventos a los que el usuario o una de sus cuadrillas está apuntado. El *widget* tiene 3 estados posibles que se ven representados en la figura 20.

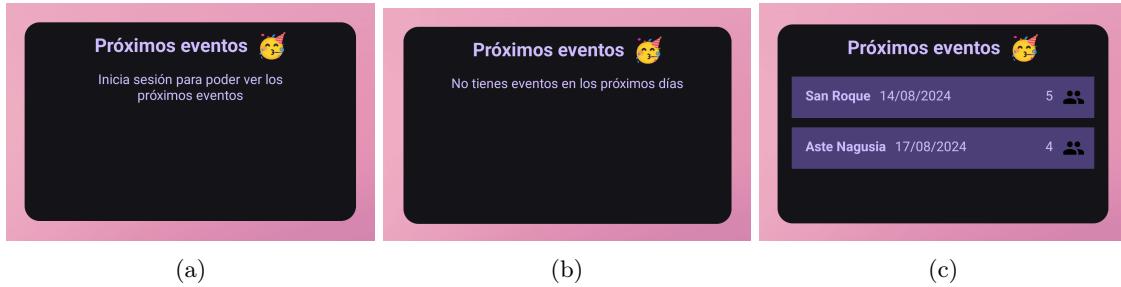


Figura 10: Diferentes estados del widget.

El *widget* no mostrará ninguna información hasta que el usuario haya iniciado sesión en la aplicación. Esto se comprobará en el *DataStore* que la aplicación utiliza como preferencias. Aquí se guardará el nombre del usuario que tiene la sesión iniciada; si no existe, se mostrará el mensaje de la figura 20a.

Una vez dentro de la aplicación, el *widget* mostrará los eventos del usuario (figura 20c) o un mensaje en caso de que el usuario no esté apuntado a ningún evento (figura 20b).

La actualización del *widget* se llevará a cabo en los siguientes casos:

- El usuario se apunta o se desapunta de un evento tanto como usuario o como cuadrilla.
- Se inicia o se cierra sesión.
- Se crea un evento nuevo al que implícitamente se añade al usuario como asistente.
- Cada una hora como se especifica en el archivo de configuración XML del widget.

Por defecto, el *widget* tiene un tamaño de 2 celdas de alto y 4 de ancho, aunque este se puede reescalar tanto vertical como horizontalmente. Con este tamaño, el *widget* se podrá visualizar en la mayoría de dispositivos móviles modernos. Sin embargo, se recomienda ampliar el ancho de este al máximo de la pantalla para poder visualizar mejor la información.

Para el desarrollo de este, se ha utilizado *JetPack Glance*, una librería moderna para desarrollar *widgets* nativos de Android en conjunto con *JetPack Compose* [9].

3 CARACTERÍSTICAS

3.14. Trabajo en segundo plano

La aplicación realiza diferentes tareas en segundo plano, como comunicaciones con la base de datos remota, obtención de la ubicación del usuario, entre otras, para ofrecer una experiencia de usuario más limpia y fluida. Para ello, en la mayoría de los casos, se utilizan las corrutinas de *Kotlin*. Estas corrutinas son una funcionalidad del lenguaje que facilita la ejecución de código en diferentes hilos de ejecución [6].

Además, se ha hecho uso de *Alarm Manager* para notificar a los usuarios que están apuntados a un evento el día previo a que este tenga lugar. Se trata de una funcionalidad que viene por defecto en la aplicación. Cuando un usuario se apunta a un evento, automáticamente se programa una alarma en segundo plano que el usuario recibirá a través del canal de notificaciones de la aplicación [4].

Asimismo, en las preferencias del usuario se permite gestionar si las notificaciones están activadas o no, afectando a las notificaciones tanto de FCM como del *Alarm Manager*. En otras palabras, si el usuario desactiva las notificaciones, se borrarán las alarmas que tenga programadas por estar apuntado a ciertos eventos. Por el contrario, si tenía las notificaciones desactivadas y las habilita, se añadirán alarmas para los eventos en los que esté suscrito.

4. Manual de usuario

Se puede acceder a la aplicación mediante registro o usando el siguiente usuario creado previamente:

- Usuario: ikersobron
- Contraseña: 12345

4.1. Inicio de sesión

Nada más abrir **FestUp**, aparecerá una pantalla de carga (figura 11a) que nos llevará directamente a la pantalla de inicio de sesión, de forma que los usuarios puedan registrarse (figura 11c) o iniciar sesión (figura 11b) en caso de disponer ya de una cuenta. En el registro se pedirán los siguientes campos:

- **Nombre de usuario:** Este es el nombre que se utiliza para identificar al usuario en el inicio de sesión y en el resto de la aplicación, por lo que deberá ser único, en minúsculas y sin espacios.
- **Email:** Correo electrónico del usuario que se utiliza para añadir los eventos al calendario de Google en caso de solicitarlo.
- **Nombre:** El número de teléfono del usuario se utiliza para conectarle con sus contactos que también están registrados en la aplicación.
- **Nombre:** El nombre del usuario. Puede ser el nombre real de la persona, un mote, etc. Aparecerá en el perfil de forma que el resto de usuarios sepan de quién se trata. No hay restricciones en lo que poner.
- **Fecha de nacimiento:** La fecha se utiliza para dar la edad del usuario y calcular la edad media de los eventos disponibles en la aplicación, en base a sus asistentes.
- **Contraseña:** La contraseña deberá ser la misma en los dos campos (de forma que se verifique que el usuario la está añadiendo correctamente), además de tener cinco o más caracteres.
- **Foto de perfil:** Se da la opción de añadir una foto de perfil en el registro, pero se podrá añadir más adelante en caso de no querer hacerlo al momento.

Una vez iniciada la sesión, esta se mantendrá abierta en el dispositivo hasta que se pulse el botón "cerrar sesión" del perfil del usuario. Si este es el caso, al abrir la aplicación se pasará de la pantalla de carga directamente al contenido de la app.

En caso de que no se pueda conectar al servidor por un problema de red o similar, aparecerá un ícono avisando de ello y se deshabilitará el botón para acceder a la aplicación. Para solucionarlo, será necesario cerrar y volver a abrir la aplicación después de activar la red.

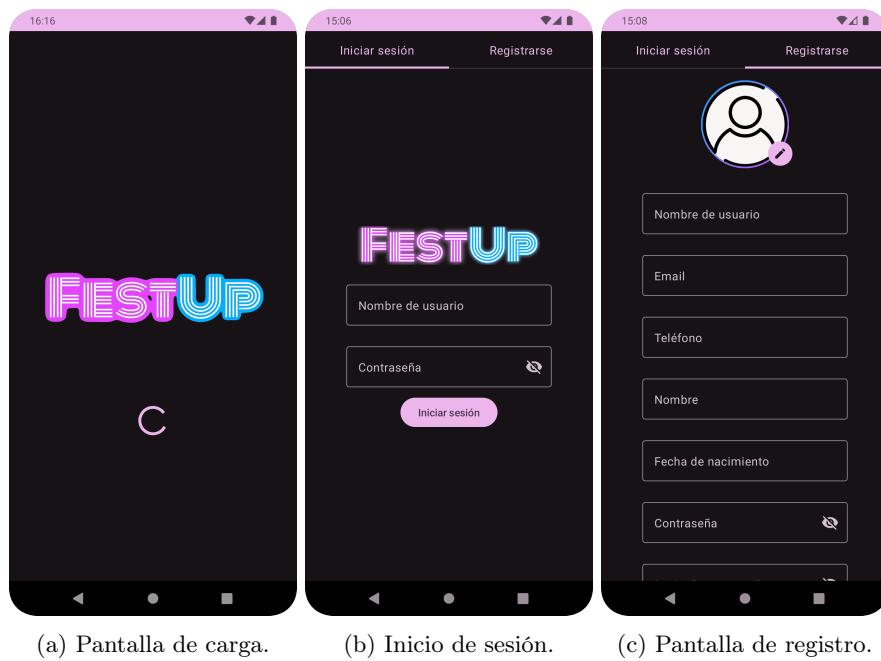


Figura 11: Pantalla de inicio de sesión y registro de la aplicación

4.2. Feed del usuario

Al iniciar sesión, se abrirá la pantalla principal con un listado ordenado por fecha de los eventos (figura 12a) a los que está apuntado el usuario, ya sea como individual o con alguna de sus cuadrillas. A medida que pasen los eventos, estos se eliminarán de la lista.

Mediante el *tab* superior, se podrá acceder a “Podría interesarte” (figura 12b), una lista similar a la anterior, pero referente a los eventos en los que participarán las personas a las que se sigue.

Por otro lado, en las tres pantallas principales, haciendo uso de la barra de navegación inferior se podrá navegar entre esta pantalla y las dos siguientes. Mientras que, haciendo uso de la barra de navegación superior, se podrá acceder al perfil del usuario que ha iniciado sesión pulsando sobre el icono de perfil.

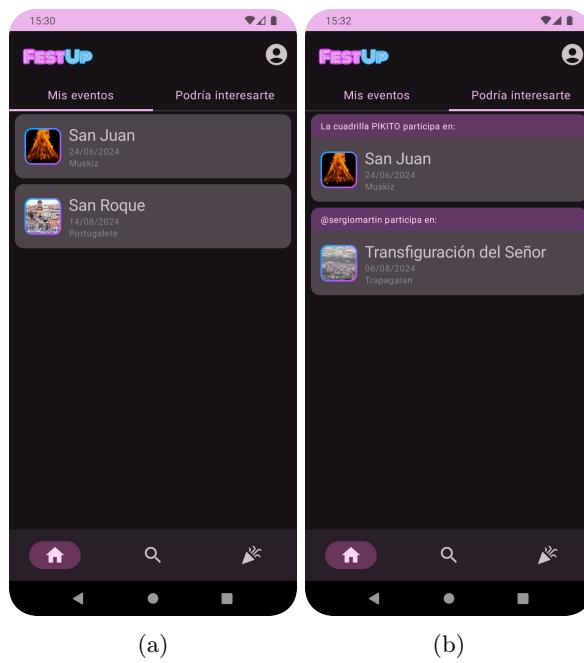


Figura 12: Feed de los eventos de usuario (figura 12a) y los eventos recomendados (figura 12b).

4.3. Pantalla de búsqueda

En esta pantalla se muestran tres paneles de búsqueda: uno para los usuarios de la aplicación (figura 13a), otro para las cuadrillas (figura 13b) y uno último para los eventos (figura 13c). Todos los elementos de ese tipo que se encuentren en la aplicación se mostrarán (en el caso de los usuarios, aparecerán todos excepto uno mismo), y a medida que se escriba un término de búsqueda, la lista mostrará los elementos que coincidan.

Al igual que en la pantalla anterior, al pulsar un elemento de la lista se accederá al mismo, ya sea un perfil de usuario, una cuadrilla o un evento.

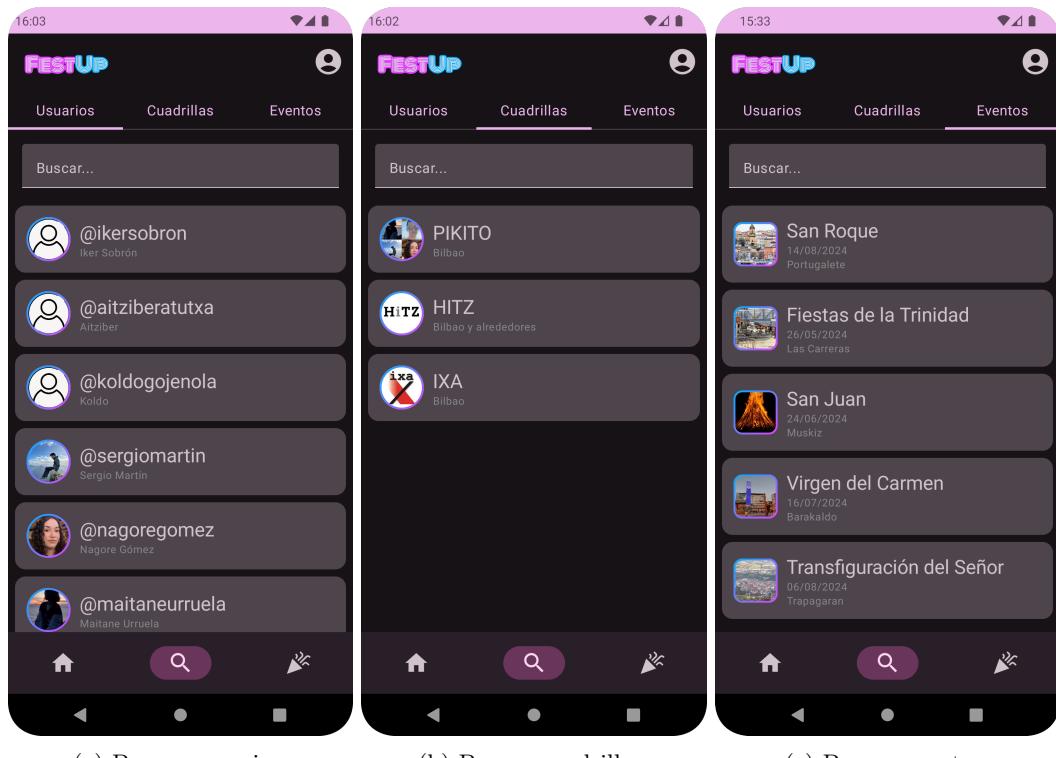


Figura 13: Pantallas de búsqueda.

4.4. Mapa y listado de eventos

En esta pantalla, aparecerán todos los eventos señalados por marcadores en un mapa. Al pulsar sobre alguno de estos marcadores se abrirá una pantalla con la información del mismo. Desde esta pantalla, se podrán crear nuevos eventos pulsando el botón de "Nuevo evento".

En caso de preferir un listado de eventos en lugar de un mapa, se podrá pulsar el botón de la esquina superior derecha para visualizar todos los eventos en formato de lista.

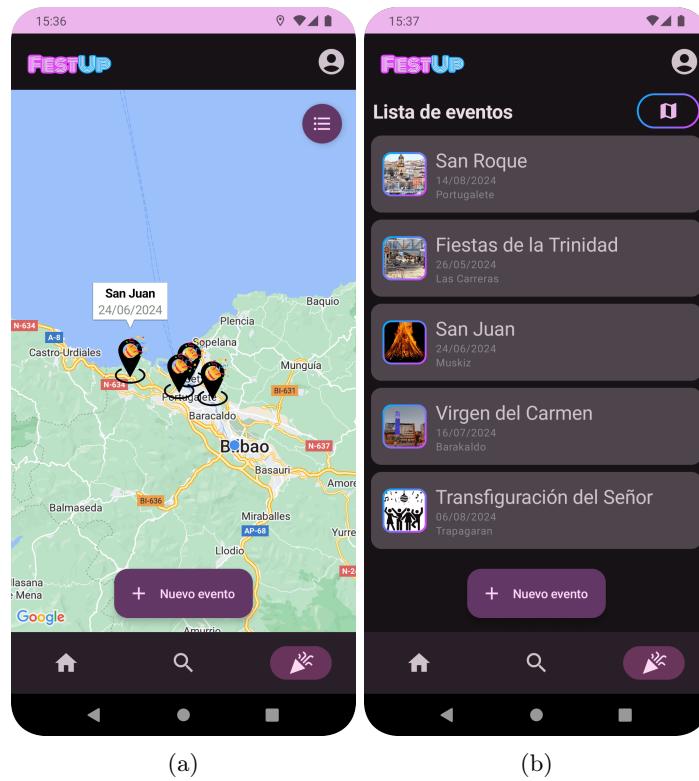


Figura 14: Mapa de eventos (figura 14) y lista de eventos (figura 14b).

4.5. Perfil de usuario

Esta pantalla aparecerá cuando se pulse sobre el icono de perfil de la barra de navegación superior, para mostrar el perfil del usuario identificado (figura 15a), o cuando se pulse sobre otro usuario (figura 15b).

En el perfil del usuario identificado, se mostrará la siguiente información: la foto de perfil (editable mediante el botón del lápiz), el nombre, la edad y el recuento de seguidores y seguidos. También se listarán las cuadrillas a las que pertenece, un botón para crear una nueva cuadrilla y los eventos a los que el usuario está apuntado, tanto individualmente como con sus cuadrillas. Por último, habrá botones para acceder a las preferencias (ícono de engranaje), editar el perfil (ícono de lápiz) y cerrar sesión (ícono de puerta). Es importante destacar que, en modo horizontal, estos botones se mostrarán al pulsar sobre el ícono de tres puntos.

En el perfil de otro usuario, se mostrará la siguiente información: la foto de perfil, el nombre, la edad y el recuento de seguidores y seguidos. Además, se listarán las cuadrillas a las que pertenece y los eventos a los que está apuntado, tanto individualmente como con sus cuadrillas. Por último, habrá un ícono para seguir o dejar de seguir al usuario.

En ambos casos, pulsando sobre los seguidos o seguidores se mostrará una lista con los mismos (figura 15c), y pulsando sobre una cuadrilla o un evento se mostrará una pantalla con más

información sobre estos.

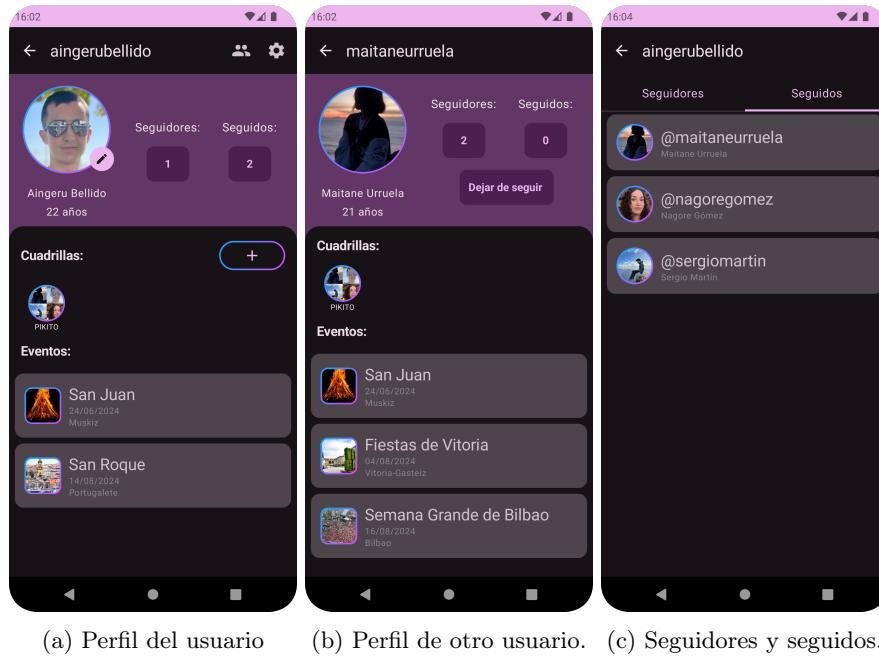


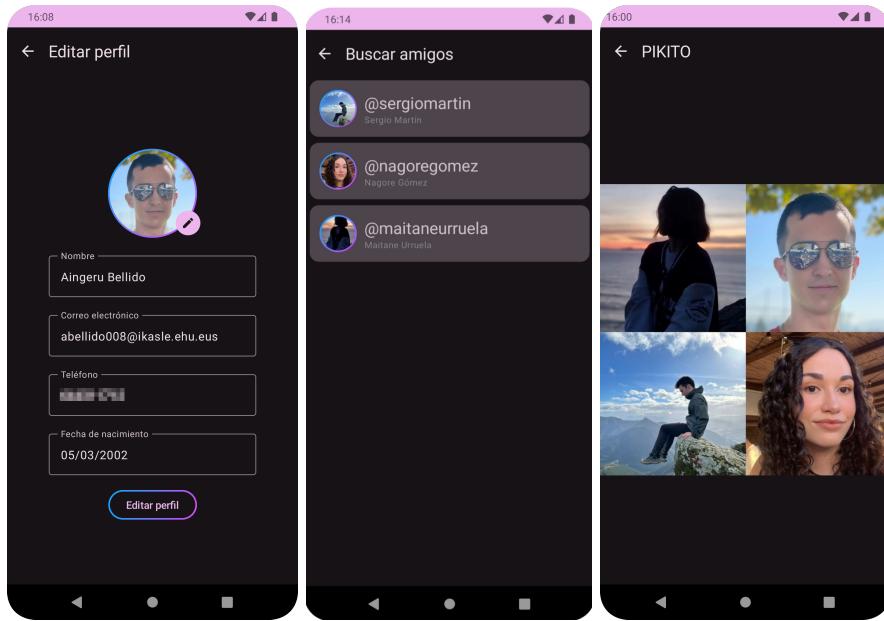
Figura 15: Pefiles de usuario.

En caso de querer editar el perfil de uno mismo, se pulsará el botón de la imagen de perfil, accediendo así a la pantalla de edición del usuario (figura 16a). En esta pantalla, el usuario identificado podrá editar su foto de perfil y actualizar sus datos personales, incluyendo nombre, correo electrónico, número de teléfono y fecha de nacimiento.

Algo a tener muy en cuenta es la calidad de las imágenes que se almacenan en la aplicación. Debido a que los recursos de los que se dispone no son excesivamente avanzados, gestionar imágenes de gran calidad (mayor tamaño) puede suponer tiempos de espera más largos de lo habitual. Por ello, se ha decidido limitar la posibilidad de cargar imágenes de gran tamaño en la aplicación con la idea de mantener una experiencia de usuario fluida y eficiente.

Otra opción disponible desde el perfil del usuario registrado es la de buscar usuarios de la app entre sus contactos. En esta pantalla se mostrarán los contactos del usuario identificado que también tienen una cuenta en la aplicación, facilitando así que los usuarios puedan conectarse fácilmente con sus amigos (figura 16b).

Finalmente, si se pulsa sobre alguna de las fotos en los perfiles de usuario o cuadrilla, se abrirá una pantalla donde se visualizará en pantalla completa la foto de perfil pulsada. (figura 16c).



(a) Pantalla de editar perfil (b) Pantalla de buscar amigos del usuario. (c) Pantalla de visualización de imágenes.

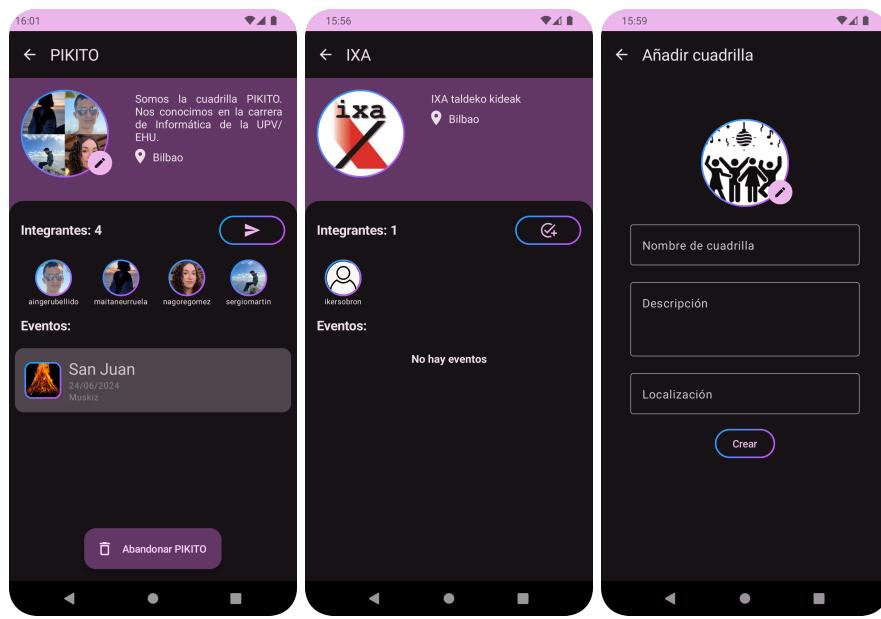
Figura 16: Otras opciones del perfil del usuario identificado.

4.6. Perfil de cuadrilla

En esta pantalla se mostrará la descripción y la localización, los integrantes y los eventos a los que está apuntada la cuadrilla.

En caso de pertenecer a la cuadrilla (figura 17a) aparecerá un botón (ícono de avión de papel) para compartir un código de acceso, mediante el canal de mensajería de preferencia (WhatsApp o Telegram) o mediante un código QR. También en esta pantalla se mostrará un botón para abandonar la cuadrilla.

Por el contrario, si no se es parte de la cuadrilla (figura 17b), aparecerá un botón (ícono de círculo con "check") que desplegará una alerta para insertar el código de acceso para unirse a la misma, mediante texto o escaneando el código QR.



(a) Perfil de una cuadrilla (b) Perfil de una cuadrilla (c) Pantalla para crear una del usuario. de otro usuario. nueva cuadrilla.

Figura 17: Perfiles de cuadrilla.

Al pulsar sobre el botón para crear una cuadrilla (en el perfil del usuario identificado) se mostrará la pantalla (figura 17c), que contará con un formulario con las siguientes especificaciones:

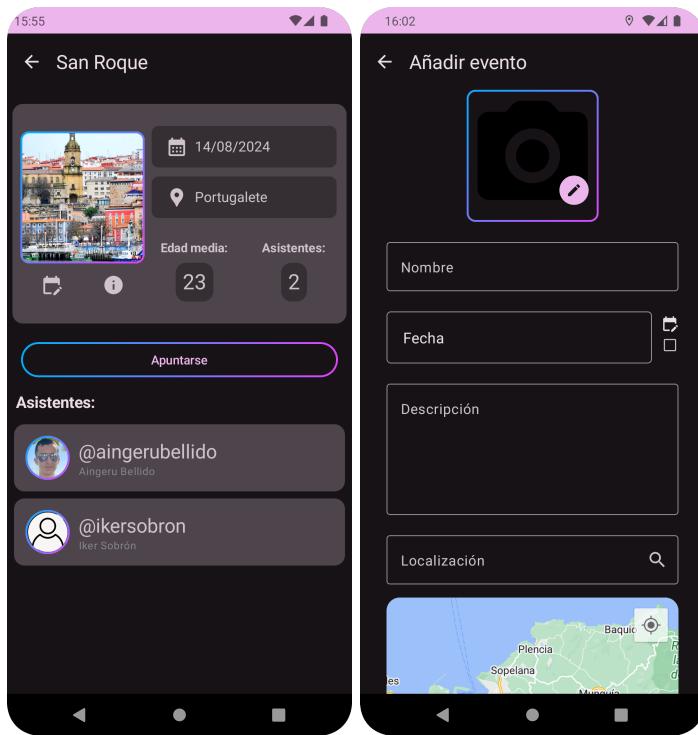
- **Foto del perfil:** Foto de perfil para la cuadrilla, esta se podrá cambiar más adelante.
- **Nombre de cuadrilla:** Este es el nombre que se utiliza para identificar a la cuadrilla en el resto de la aplicación, por lo que deberá ser único y sin espacios.
- **Descripción:** Una pequeña descripción con datos sobre el grupo.
- **Ubicación:** La zona en la que viven, o si son de zonas diferentes, donde suelen estar; por ejemplo.

4.7. Pantalla del evento

Esta pantalla (figura 18) proporcionará la siguiente información sobre el evento: imagen, fecha, ubicación, número de asistentes y su edad promedio.

También se mostrarán dos botones: uno con el icono de un calendario para añadir el evento al calendario, y otro con el icono de información para ver la descripción del evento.

Además, para apuntarse y desapuntarse de dicho evento se deberá pulsar sobre el botón "Apuntarse". Este botón desplegará un cuadro donde se mostrarán con un *switch* el usuario y sus cuadrillas para añadirlos o eliminarlos del evento. En caso de apuntar una cuadrilla, todos los integrantes de la misma serán añadidos al conteo de asistentes.



(a) Información del evento. (b) Pantalla para nuevo evento.

Figura 18: Pantallas de evento.

Al pulsar sobre el botón para crear un nuevo evento, en las pantallas de mapa y listado de eventos, se mostrará esta pantalla (figura 18b), que contendrá un formulario con las siguientes especificaciones:

- **Foto del perfil:** A diferencia de en los perfiles de usuario y cuadrilla, esta imagen **no se podrá actualizar más adelante**.
- **Nombre del evento:** El nombre debe ser claro y conciso de forma que los usuarios sepan de qué trata el evento.
- **Fecha:** La fecha en la que se celebrará el evento.
- **Descripción:** Una pequeña descripción con datos más concretos con respecto al evento.
- **Localización:** El lugar donde se planea llevar a cabo el evento, ya sea una ciudad, un local, o similares. Se mostrará un mapa para facilitar la búsqueda, en el que aparecerá la localización una vez pulsado el icono de la lupa.

4.8. Ajustes

Esta pantalla (figura 19) se mostrará al pulsar sobre el icono de engranaje en el perfil del usuario identificado. Todas las preferencias seleccionadas valdrán por cada usuario en cada dispositivo, es decir, si el mismo usuario se conecta en otro dispositivo, deberá volver a guardar sus preferencias en este.

Las preferencias a elegir son las siguientes:

- **Idioma:** El idioma podrá cambiarse de castellano a euskera y viceversa.
- **Tema de colores:** Se podrá alternar entre modo oscuro y modo claro.
- **Recibir notificaciones:** Por defecto, las notificaciones estarán habilitadas, lo que permitirá recibir notificaciones de los usuarios a los que se sigue, así como notificaciones personalizadas. Las notificaciones disponibles de la aplicación son: el aviso de un evento al que se está apuntado si este es mañana, el aviso de que una persona a la que se sigue se ha apuntado a un evento y notificaciones propias de la app. Sin embargo, al cerrar sesión, el usuario dejará de recibir notificaciones.

La pantalla cuenta también con el botón para cerrar la sesión actual.

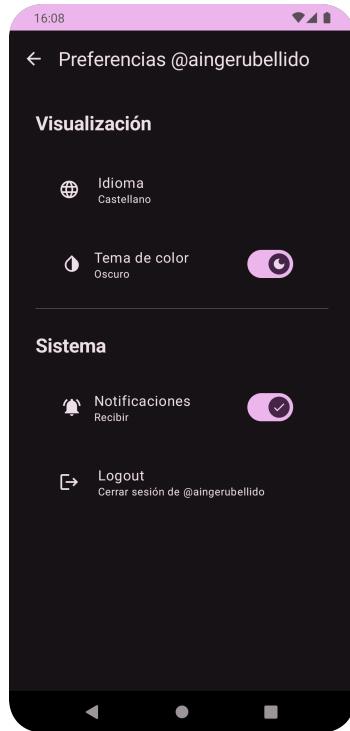


Figura 19: Pantalla de preferencias.

4.9. *Widget*

El *widget* de la aplicación, sirve para mostrar los próximos eventos a los que el usuario o una de sus cuadrillas está apuntado.

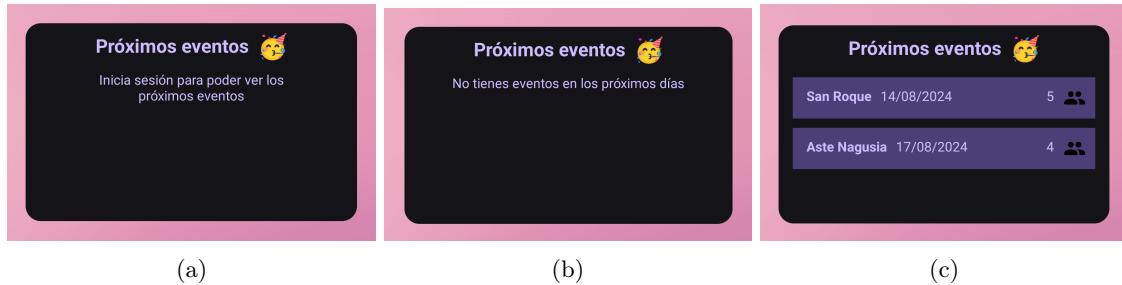


Figura 20: Diferentes estados del *widget*.

El *widget* no mostrará ninguna información hasta que el usuario haya iniciado sesión en la aplicación. Una vez dentro de la aplicación, el *widget* mostrará los eventos del usuario (figura 20c), o un mensaje en caso de que el usuario no esté apuntado a ningún evento (figura 20b).

REFERENCIAS

Referencias

- [1] Adminer. *Adminer*. URL: <https://www.adminer.org/>.
- [2] COIL. *COIL*. URL: <https://coil-kt.github.io/coil/>.
- [3] Developers. *Cómo guardar datos en una base de datos local usando Room*. URL: <https://developer.android.com/training/data-storage/room>.
- [4] Android Developers. *Alarm Manager*. URL: <https://developer.android.com/reference/android/app/AlarmManager>.
- [5] Android Developers. *Capa de dominio*. URL: <https://developer.android.com/topic/architecture/domain-layer?hl=es-419>.
- [6] Android Developers. *Corrutinas de Kotlin en Android*. URL: <https://developer.android.com/kotlin/coroutines>.
- [7] Android Developers. *Descripción general del proveedor de calendario*. URL: <https://developer.android.com/guide/topics/providers/calendar-provider?hl=es-419>.
- [8] Android Developers. *Geocoder*. URL: <https://developer.android.com/reference/android/location/Geocoder>.
- [9] Android Developers. *Glance*. URL: <https://developer.android.com/develop/ui/compose/glance>.
- [10] Android Developers. *Guía de arquitectura de apps*. URL: <https://developer.android.com/topic/architecture?hl=es-419>.
- [11] Android Developers. *Loading images*. URL: <https://developer.android.com/develop/ui/compose/graphics/images/loading#coil>.
- [12] Android Developers. *Selector de fotos*. URL: <https://developer.android.com/training/data-storage/shared/photopicker?hl=es-419>.
- [13] Docker. *Docker*. URL: <https://www.docker.com/>.
- [14] FastApi. *FastApi Framework*. URL: <https://fastapi.tiangolo.com/>.
- [15] Firebase. *Envía mensajes a varios dispositivos*. URL: <https://firebase.google.com/docs/cloud-messaging/android/send-multiple?hl=es-419>.
- [16] Ktor. *Ktor*. URL: <https://ktor.io/>.
- [17] Google Maps Platform. *Biblioteca de Maps Compose*. URL: <https://developers.google.com/maps/documentation/android-sdk/maps-compose?hl=es-419>.
- [18] PostgreSQL. *PostgreSQL*. URL: <https://www.postgresql.org/>.
- [19] SQLAlchemy. *SQLAlchemy*. URL: <https://www.sqlalchemy.org/>.
- [20] SQLite. *SQLite*. URL: <https://www.sqlite.org/>.
- [21] ZXing. *ZXing*. URL: <https://github.com/journeyapps/zxing-android-embedded>.