Trabajo de Fin de Grado

GRADO DE INGENIERÍA INFORMÁTICA

Facultad de Matemáticas e Informática
Universidad de Barcelona

LocPic: Una app de galería de fotografías geolocalizadas para Android

---

Adrián Pardos López

Director: Sergio Sayago
Realizado en: Departament de Matemàtiques i Informàtica

Barcelona, 28 de enero de 2018

Resumen

En este Trabajo de Fin de Grado presentamos el diseño, desarrollo y evaluación de una app de geolocalización de fotografías para teléfonos Android. Los principales objetivos son desarrollar un proyecto para teléfonos inteligentes que permita a los usuarios almacenar sus fotografías en la nube, tenerlas disponibles en diferentes dispositivos y tener la posibilidad de visualizar estas fotografías distribuidas en un mapa, para así recordar los sitios que hemos visitado o nuestros momentos favoritos.

Para implementar la aplicación, usamos las herramientas más utilizadas en el desarrollo profesional de aplicaciones para Android, como son Android Studio y Firebase. También utilizamos servicios que nos ofrece Google para realizar la implementación e interacción con los mapas. Para el diseño de la aplicación hemos respetado los patrones indicados por Android, conocidos como Material Design, y hemos realizado entrevistas con usuarios para tener su opinión y propuestas. La evaluación de la interfaz se realizó mediante entrevistas con usuarios, en las que realizaron tareas reales de uso y respondieron a una serie de preguntas basadas en esas tareas. Además, hemos realizado una evaluación técnica de la app que nos permite analizar el correcto funcionamiento de las funciones de la aplicación.

Los resultados del Trabajo de Fin de Grado muestran que hemos alcanzado en buena medida los objetivos del proyecto. Los resultados también muestran áreas de mejora para trabajos futuros.

Resum

En aquest Treball de Fi de Grau presentem el disseny, desenvolupament i avaluació d'una app de geolocalització de fotografies per a telèfons Android. Els principals objectius són el desenvolupament d'un projecte per telèfons intel·ligents que permet als usuaris emmagatzemar les seves fotografies al núvol, tenir disponibles aquestes fotografies a diferents dispositius i tenir la possibilitat de visualitzar-les distribuïdes a un mapa, per així recordar els llocs que hem visitat o els nostres moments favorits.

Per implementar l'aplicació, fem servir les eines més utilitzades en el desenvolupament d'aplicacions per Android, com són Android Studio i Firebase. També utilitzem serveis que ens ofereix Google per realitzar la implementació i interacció amb els mapes. Pel disseny de l'aplicació hem respectat els patrons de disseny indicat per Android, coneguts com a Material Design, i hem validat aquest disseny amb entrevistes amb usuaris. L'avaluació de la interfície es va realitzar mitjançant entrevistes amb usuaris, on realitzaven tasques reals d'ús i responien a una sèrie de preguntes basades en les tasques realitzades. A més a més, hem realitzat una avaluació tècnica de l'app que ens permet analitzar el correcte funcionament de les funcions de l'aplicació.

Els resultats del Treball de Fi de Grau mostren que hem aconseguit els objectius del projecte. Els resultats també mostren àrees de millora per a treballs futurs.

Abstract

This project is about the design, development and evaluation of an app of geolocalized pictures for Android phones. The main objectives are to develop an app for smartphones that allows users to (i) save their pictures in the cloud, (ii) to have these pictures available on any Android device, and (iii) to view these pictures displayed on map, which can enable them to remember the places that they have visited or bring back memories.

To implement the app, we have used professional tools, i.e. Android Studio and Firebase. We have also use some services that Google provide us with in order to work with maps. For the design of the app, we respect the patterns indicated by Android, known as Material Design. We have evaluated the app with end-users through semi-structured interviews. We have also conducted a technical evaluation to analyse the performance of the functions of the app.

The results of the TFG have been satisfactory. We have achieved the objectives of the project. The results also show that there is room for improvement, and we indicate a number of future research / development perspective.

Agradecimientos

A mis padres, por todo el apoyo que me han dado durante todo este tiempo y por darme la oportunidad que ellos no tuvieron.

A mis hermanos, que sé que siempre estarán ahí cuando los necesite.

A mis amigos y compañeros de clase, por aguantarme, apoyarme y entenderme en este largo viaje.

Y finalmente a mi tutor, Sergio Sayago, por toda la ayuda, propuestas y guía que me ha aportado a lo largo del trabajo.

Índice de contenidos

Índice de Figuras **9**
Índice de Tablas **10**
1. Introducción **11**
1.1. Motivaciones 11
1.2. Objetivos 11
1.3. Resumen del trabajo desarrollado 12
2. Planificación y presupuesto **13**
2.1. Planificación inicial 13
2.2. Planificación final 13
2.3. Presupuesto 14
3. Tecnologías utilizadas **16**
3.1. Android 16
3.1.1. Sistema operativo Android 16
3.1.2. Android SDK y Android Studio 17
3.2. Componentes principales 17
3.2.1. Activity 17
3.2.2. Intent 18
3.2.3. Fragment 19
3.2.4. GridView y ListView 20
3.2.5. AsyncTask 20
3.3. Firebase 21
3.3.1. Autentificación 21
3.3.2. Realtime Database 21
3.3.3. Cloud Storage 22
3.4. Glide vs Picasso 22
3.5. API Google Maps y Google Places 23
4. Análisis de requerimientos **24**
4.1. Requerimientos 24
4.2. Análisis de otras aplicaciones 25
4.2.1. Inicio 25
4.2.2. Galerías de fotografías 25
4.2.3. Visualización de fotografías en mapas 26
4.2.4. Conclusión de las aplicaciones 27
5. Desarrollo de la aplicación **28**
5.1. Casos de uso 28
5.2. Sincronización del proyecto con Firebase 30
5.3. Paquetes del proyecto 30

5.4. Autentificación 31
5.4.1. Registro 31
5.4.2. Iniciar sesión 32
5.5. Subir fotografía 33
5.5.1. Posición 33
5.5.2. Cámara 35
5.5.3. Galería del dispositivo 36
5.6. Galería 37
5.6.1. Carpetas 37
5.6.2. Fotografías 38
5.7. Mapa 39
5.7.1. Ítems del mapa 39
5.7.2. Visualización de las fotografías 40
5.7.3. Clúster 41
5.8. Otras gestiones 42
5.8.1. Gestión del idioma 42
5.8.2. Gestión de los permisos 42
5.9. Base de datos 43
5.10. Almacenamiento de las fotografías 45
6. Diseño y evaluación de la interfaz de usuario 46
6.1. Interfaz de usuario 46
6.2. Evaluación de la interfaz de usuario 48
7. Evaluación funcional de la aplicación 50
8. Conclusión 52
8.1. Trabajo futuro 52
9. Referencias 54
10. Anexo 55
10.1. Test de usuarios 55
10.2. Manual de usuario 59

Índice de Figuras

Figura 1: Planificación inicial del proyecto....13
Figura 2: Planificación final del proyecto....13
Figura 3: Porcentaje de uso de los principales sistemas operativos (es.statista.com)....16
Figura 4: Datos recopilados hasta diciembre de 2017 (developer.android.com)....17
Figura 5: Ciclo de vida de una Actividad (developer.android.com)....18
Figura 6: Entrega de una Intent para iniciar otra actividad (developer.android.com)....19
Figura 7: Ciclo de vida de un Fragment (developer.android.com)....19
Figura 8: A la izquierda ListView, a la derecha GridView....20
Figura 9: Tamaños de las librerías Glide y Picasso....22
Figura 10: Métodos de las librerías Glide y Picasso....23
Figura 11: Pantalla inicial de Instagram, Twitter y Dropbox....25
Figura 12: Visualización de las carpetas de Google Fotos y QuickPic....26
Figura 13: Distribución de las fotografías en el mapa que nos ofrecía Instagram....26
Figura 14: Caso de uso de un usuario no identificado....28
Figura 15: Caso de uso de un usuario identificado....29
Figura 16: Caso de uso de un usuario identificado para subir una fotografía....29
Figura 17: Contenido de los paquetes del proyecto....30
Figura 18: Diagrama de flujo del registro de un usuario....31
Figura 19: Diagrama de flujo de iniciar sesión mediante correo electrónico....32
Figura 20: Mensaje que recibe el usuario para poder modificar su contraseña....33
Figura 21: Buscador de lugares....35
Figura 22: Diagrama de flujo de la subida de una fotografía realizada con la cámara....36
Figura 23: Diagrama de flujo de la subida de una fotografía de la galería del dispositivo....37
Figura 24: Visualización de las carpetas de la galería....38
Figura 25: Visualización de las fotografías en el mapa....40
Figura 26: Visualización de los clústeres en el mapa....41
Figura 27: Ejemplo de solicitud de permisos al usuario....43
Figura 28: Contenido y estructura de la base de datos para un usuario....43
Figura 29: Control del menú y de la subida de fotografías.....46
Figura 30: Componentes del menú principal....46
Figura 31: Diseño de los botones para subir las fotografías....47
Figura 32: Ejemplos de diálogos en la aplicación....48
Figura 33: Introducción a la LocPic....59
Figura 34: Registro e inicio de sesión en LocPic....59
Figura 35: Permisos necesarios que se solicitan al usuario....60
Figura 36: Menú principal de LocPic....60
Figura 37: Gestión de una fotografía....61
Figura 38: Visualización de una fotografía....61
Figura 39: Botón para subir una fotografía....62
Figura 40: Pasos para subir una fotografía de la galería del dispositivo....62
Figura 41: Selección de la carpeta de destino....63
Figura 42: Visualización del mapa del usuario....63

Índice de Tablas

Tabla 1: Análisis económico del proyecto 14
Tabla 2: Características de las bases de datos de los planes de Firebase (firebase.google.com) 15
Tabla 3: Características de almacenamiento de los planes de Firebase (firebase.google.com) 15
Tabla 4: Características de los dispositivos de pruebas 50
Tabla 5: Resultados de las pruebas de los casos de uso de un usuario no identificado 50
Tabla 6: Resultados de las pruebas de los casos de uso de un usuario identificado 51
Tabla 7: Respuestas del usuario 1 56
Tabla 8: Respuestas del usuario 2 57
Tabla 9: Respuestas del usuario 3 58

1. Introducción

Actualmente, la tecnología móvil y los teléfonos inteligentes tienen una gran importancia en nuestra vida cotidiana, en la que nos encontramos aplicaciones de casi todo tipo, desde aquellas que nos permiten pedir comida a domicilio, hasta otras para chatear con nuestros amigos.

En el presente Trabajo de Fin de Grado (TFG) nos concentramos en el diseño, desarrollo y evaluación de una app (LocPic) de geolocalización de fotografías para la plataforma Android.

LocPic nos permite visualizar nuestras fotografías favoritas en un mapa mediante Google Maps, lo que permite varios escenarios de interacción, como, por ejemplo, recordar nuestros momentos inmortalizados en fotografías de una manera diferente a la que permite la típica galería de fotografías de los teléfonos Android.

Las fotografías se almacenan en la nube, lo que nos permite tener disponible nuestro contenido en diferentes dispositivos mediante nuestra cuenta de LocPic o con nuestra cuenta de Google. Las fotografías las tenemos ordenadas en carpetas, para facilitar su uso y organización. Las fotografías que se visualizan en LocPic son aquellas que se han realizado directamente desde la aplicación o aquellas que se tengan que subir desde la galería del dispositivo. LocPic nos permite gestionar nuestras fotografías en las diferentes carpetas, eliminarlas, visualizarlas y descargarlas para poder tenerlas en nuestro dispositivo. Para utilizar LocPic, es necesario registrarse utilizando un correo electrónico y contraseña, o mediante una cuenta de Google. La interfaz de LocPic se ha diseñado y evaluado con usuarios, y siguiendo las recomendaciones y patrones de diseño de Android.

1.1. Motivaciones

Una de las principales razones que me adentró a la informática y me llevó a cursar el grado en Ingeniería en Informática en la UB fue el sistema operativo Android. Quería entender y aprender cómo funcionaba todo lo relacionado con la informática y Android.

Es cierto que, en el segundo curso del grado, tenemos una asignatura dedicada a Android (Proyecto Integrado de Software). En el año en la que la realicé, hicimos un juego en grupos de 4 personas. La asignatura fue bien, pero sentía que no había aprendido todo lo que me hubiera gustado, así que, decidí que mi Trabajo de Fin de Grado fuera sobre Android.

Concretamente, tenía muchas ganas de realizar un proyecto sobre apps de Android desde cero, pasando por las diferentes fases de desarrollo, aplicando y profundizando los conocimientos adquiridos en el grado.

La idea de la app sobre fotografías geolocalizadas me surgió este verano. Pensé en hacer una aplicación que permitiera recordar los momentos de los diversos sitios que has visitado y que has capturado en tu móvil. Esta idea me pareció muy interesante, ya que engloba muchas funcionalidades que me pueden ayudar a conseguir mi objetivo: gestionar las fotografías en la aplicación y contenido en la nube, gestionar usuarios, utilizar servicios que nos ofrece Google, las bases de datos, la interfaz de usuario…

Estas razones me motivaron a realizar el proyecto y así poder aprender y gestionar el funcionamiento de una aplicación con contenidos en la nube.

1.2. Objetivos

El objetivo principal de este proyecto es la realización de una app de galería de fotografías para Android. Las fotografías se almacenarán en la nube, y serán accesibles desde la aplicación a través

de la cuenta del usuario. Además, los usuarios podrán ver estas fotografías en un mapa (Google Maps), geolocalizadas en el sitio donde se realizaron, de una manera sencilla y visual.

La interfaz de usuario de la app será intuitiva, sencilla y agradable.

Por lo tanto, con la realización de este proyecto se quiere:

- **Objetivo 1**. Desarrollar una aplicación para Android desde 0.
- **Objetivo 2**. Aprender a utilizar una plataforma de desarrollo de aplicaciones para gestionar a los usuarios y el contenido de la aplicación.
- **Objetivo 3**. Aprender a utilizar los servicios que nos ofrece Google a los desarrolladores.
- **Objetivo 4**. Desarrollar una interfaz de usuario siguiendo los patrones de Android.
- **Objetivo 5**. Analizar el comportamiento y opiniones de los usuarios para mejorar las diversas funcionalidades de la aplicación y su diseño.

1.3. Resumen del trabajo desarrollado

En el presente TFG hemos desarrollado una aplicación para Android que permite a los usuarios visualizar sus fotografías distribuidas por el mapa.

El primer objetivo del trabajo era desarrollar una aplicación para Android desde 0. Podemos decir que lo hemos cumplido, tenemos una aplicación funcional que realiza las tareas de los requisitos que planteamos (ver *4.1. Requerimientos*).

Podemos decir que el segundo objetivo también se ha cumplido. La aplicación ha sido desarrollada utilizando Firebase, que nos permite gestionar los usuarios, las bases de datos y el almacenamiento relacionado con nuestra aplicación.

El tercer objetivo era utilizar los servicios que nos ofrece Google a los desarrolladores. Estos servicios los encontramos en el mapa con Google Maps y Google Places o en la creación de un mecanismo de autentificación alternativo, como es el inicio de sesión con una cuenta de Google, que es una combinación de los servicios de Google y Firebase.

El cuarto objetivo era que la interfaz de usuario de la aplicación respetara los patrones de diseño de Android basados en Material Design. Se han seguido esta convención de diseño que presenta Material Design.

El último objetivo era realizar un test con usuarios para poder validar el diseño implementado en la aplicación y obtener datos y opiniones para realizar las modificaciones necesarias sobre el diseño presentado y sus funciones. Destacar que, aunque nos ha servido para validar y añadir mejoras sobre la información que se muestra y el diseño, no todas las mejoras propuestas por los usuarios han sido posible añadirlas al proyecto y ajustarlas a la planificación, y están incluidas como perspectivas de futuro.

Podemos decir que todos los objetivos presentados en el punto anterior los hemos realizado satisfactoriamente.

2. Planificación y presupuesto

A la hora de planificar debemos tener en cuenta el periodo de tiempo que durará el proyecto, las horas de dedicación, el inicio y el final, las personas que participan y los recursos disponibles. También tenemos que considerar las diferentes fases en las que dividiremos el proyecto.

El proyecto lo hemos dividido en 9 fases, que nos permiten planificar con un grado de exactitud mayor el número de actividades, el tiempo de desarrollo y la duración del proyecto. Las fases son: (1) resumen y planificación, (2) análisis y requisitos, (3) prototipo de diseño, (4) aprendizaje, (5) desarrollo, (6) test de la aplicación, (7) test con usuarios, (8) mejoras y gestión de errores y (9) documentación y memoria. La duración del proyecto se ha estimado en 18 semanas y una semana inicial para hacer un breve resumen sobre de lo que tratará el proyecto y para poder hacer la planificación de este.

Esta sección incluye un análisis de los costes del proyecto.

2.1. Planificación inicial

A continuación, mostramos la planificación inicial del proyecto

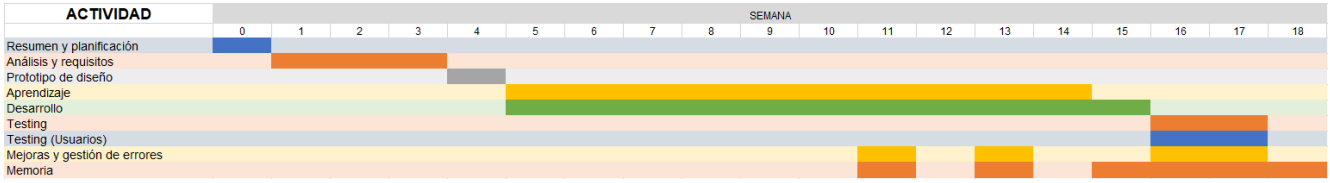


Figura 1: Planificación inicial del proyecto

En la planificación inicial, la duración del TFG era de 18 semanas. En la semana inicial (semana 0), se estima la duración del proyecto y se resume la propuesta. En las primeras semanas de trabajo, realizamos un análisis del proyecto y de las tecnologías que se utilizarán y se establecen los requisitos que tendrá. La cuarta semana se reserva para pensar los diferentes componentes y la distribución de los contenidos de la aplicación. La fase de aprendizaje y la de desarrollo son las que tienen más peso en el proyecto. La fase de desarrollo cuenta con la verificación de las funcionalidades que vamos implementando.

Finalmente, nos encontramos la fase de test con usuarios y funcional de la aplicación, mejoras y gestión de errores y memoria, que se realizarán en las últimas semanas del proyecto.

Cada semana contempla un tiempo de dedicación a documentar las diferentes tareas que se realizan.

2.2. Planificación final

A continuación, podemos ver como se ha distribuido finalmente el tiempo del proyecto.

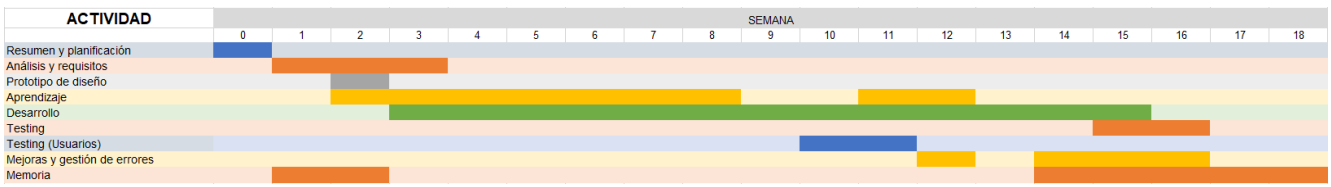


Figura 2: Planificación final del proyecto

Vemos algunos cambios respecto a la planificación inicial. El primer cambio lo apreciamos en las primeras tres semanas, donde no solo realizamos el análisis y los requisitos, sino que realizamos el estudio de los componentes que contiene la aplicación para su diseño (prototipo de diseño) y nos informamos del diseño de las aplicaciones en Android, además del comienzo del aprendizaje y analizando como poder sincronizar las diferentes herramientas utilizadas. El desarrollo del proyecto ocupa la mayor parte del tiempo. El desarrollo incluye tanto la sincronización, como la implementación y el diseño de la aplicación. El test de usuarios finalmente lo realizamos entre las semanas 10 y 11, que corresponden a diciembre. Se pudo adelantar gracias a que disponíamos de un prototipo funcional, lo que nos ha permitido añadir algunas de las sugerencias y mejoras.

La etapa final, corresponde a principalmente a realizar la memoria, la evaluación de los casos de uso y la gestión de errores, para conseguir una aplicación robusta.

2.3. Presupuesto

Como hemos indicado en la planificación, el proyecto cuenta con 18 semanas de trabajo y una semana adicional inicial para resumir la propuesta y planificar el proyecto.

En total, podemos resumir el proyecto como 19 semanas, con una dedicación diaria de entre 4 y 5 horas (días laborales), lo que en total suman unas 440 horas.

Para realizar el cálculo, es necesario dividir las diferentes tareas realizadas. En total, lo hemos dividido 6 tareas: (1) requisitos y análisis, (2) aprendizaje, (3) diseño, (4) desarrollo y (5) documentación, (6) servicios.

A continuación, mostramos la tabla con los precios por hora brutos y el coste total[1]:

	Horas	Precio (€) / Hora	Coste total
Requisitos y análisis	30 horas	15 €/h	450 €
Aprendizaje	80 horas	0 €/h	0 €
Diseño	60 horas	20 €/h	1200 €
Desarrollo	170 horas	18 €/h	3060 €
Documentación	95 horas	11 €/h	1045 €
Total	440 horas	-	5755 €

Tabla 1: Análisis económico del proyecto

El diseño incluye la evaluación con usuarios y el desarrollo la evaluación de los casos de uso.

Además de estos costes, debemos tener en cuenta los costes que pueden tener los diferentes servicios que hemos utilizado, las herramientas de desarrollo o su publicación en la Google Play Store.

Por lo que respecta al desarrollo y a la realización de pruebas de la aplicación, siendo Android uno de los sistemas operativos más extendidos del mercado, no tendremos problemas en disponer de un terminal para poder realizar pruebas. Además, el entorno de desarrollo que hemos utilizado, Android Studio, dispone de los respectivos emuladores para poder desarrollar las aplicaciones. En estos emuladores encontramos diversos dispositivos, con versiones y características diversas.

[1] Los precios por hora son una aproximación de los precios medios de los diferentes campos.

Respecto a la base de datos, la gestión de las fotografías en la nube y los usuarios hemos utilizado Firebase. En esta plataforma encontramos diversos planes según las necesidades y el alcance del proyecto. Los planes que se ofrecen son: plan Spark, plan Flame y plan Blaze.

Para poder elegir entre los diversos paquetes que se ofrecen, tenemos que fijarnos en las características principales que necesitamos. El plan Spark es gratuito, el plan Flame tiene un coste de 25 dólares mensuales y el plan Blaze es un plan personalizado que se adapta al uso del servicio.

En la siguiente tabla, mostramos las características que nos ofrecen los diferentes planes sobre las bases de datos:

	Spark	**Flame**	**Blaze**
Conexiones simultáneas	100	100k	100k/BBDD
GB almacenados	1 GB	2.5 GB	5 USD/GB
GB descargados	10 GB/mes	20 GB/mes	1 USD/GB
Múltiples bases de datos	No	No	Si

Tabla 2: Características de las bases de datos de los planes de Firebase (firebase.google.com)

A continuación, mostramos la tabla de características que nos ofrecen los diferentes planes sobre el almacenamiento:

	Spark	**Flame**	**Blaze**
GB almacenados	5 GB	50 GB	0.026 USD/GB
GB descargados	1 GB/día	50 GB/día	0.12 USD/GB
Operaciones de carga	20k/día	100k/día	0.05 USD/10k
Operaciones de descarga	50k/día	250k/día	0.004 USD/10k

Tabla 3: Características de almacenamiento de los planes de Firebase (firebase.google.com)

El plan Spark está recomendado para aplicaciones que están comenzando, el plan Flame para aplicaciones que están en expansión y el plan Blaze para aplicaciones a gran escala. Por lo tanto, inicialmente utilizaremos el plan Spark para el proyecto, ya que, aunque sea gratuito, ofrece un servicio más que suficiente y teniendo en cuenta que Firebase siempre permite el cambio de plan, podremos modificarlo según nuestras necesidades.

Respecto al uso de las APIs de Google Maps y Google Places, encontramos dos tipos de planes: el plan estándar y el plan Premium. El plan estándar, que será el utilizado por ser gratuito, dispone de uso ilimitado de Google Maps y 1.000 solicitudes predeterminadas por día en Google Places, que podemos aumentar a 150.000 solicitudes si verificamos nuestra identidad. Los servicios del plan Premium hay que acordarlos con el departamento de ventas de Google, ya que son personalizados y están pensados para aplicaciones a gran escala y que requieran de funcionalidades personalizadas de los servicios de Google.

Por último, si finalmente se decide publicar la aplicación en Google Play Store, el coste de la suscripción es un único pago de 25 dólares, poco más de 20 euros.

3. Tecnologías utilizadas

En esta sección, analizamos las tecnologías utilizadas en el proyecto, desde el sistema operativo y el entorno de desarrollo, así como los componentes principales que conforman la aplicación o las principales librerías y APIs utilizadas.

3.1. Android

3.1.1. Sistema operativo Android

Android es un sistema operativo móvil que fue lanzado en Septiembre de 2008 por Google con núcleo Linux, con su primera versión 1.0 Apple Pie. Actualmente tenemos disponible la versión 8.1 para algunos dispositivos del mercado.

Una de las características de este sistema operativo es que utiliza una máquina virtual para ejecutar las aplicaciones Java. A partir de la versión 4.4, Google, aparte de tener disponible la máquina virtual Dalvik (que era la utilizada hasta el momento), añadió ART (Android Runtime). Las diferencias entre estas dos máquinas virtuales es que Dalvik interpreta el código en el tiempo que inicia la aplicación. En cambio, ART hace una pre-compilación en el momento de instalar la aplicación, con lo cual, en el momento de ejecutarla, es más eficiente a la hora de interpretarla.

El año pasado, Android, superó a Windows como sistema operativo más utilizado, contando teléfonos inteligentes, tabletas y ordenadores.

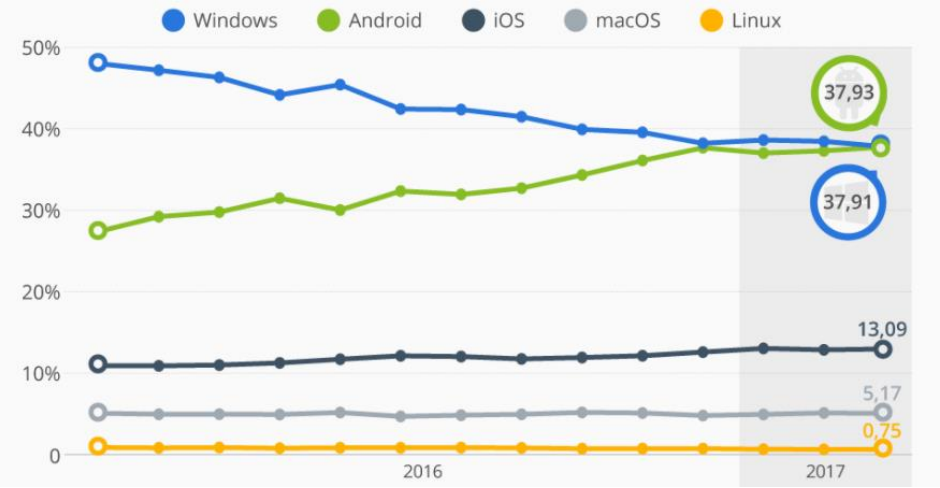


Figura 3: Porcentaje de uso de los principales sistemas operativos (es.statista.com)

Vemos como la utilización de Windows ha disminuido en los últimos años y la de Android ha aumentado. Por lo que respecta a los demás sistemas operativos, vemos un uso bastante equilibrado, sin aumentar o disminuir de forma notable.

Android lo encontramos en diferentes dispositivos de diferentes marcas, lo que hace que cada marca tenga su propia política de actualizaciones. Esto hace que cada dispositivo tenga una versión diferente, ya que excepto los dispositivos propios de Google (actualmente Pixel, antes Nexus), no reciben actualizaciones en un plazo de tiempo corto, si es que finalmente, estas llegan.

Por lo tanto, al desarrollar una aplicación para Android se ha de tener en cuenta que tiene que funcionar en diferentes dispositivos, con diferentes versiones de sistema operativo y diferentes características. Esto influye también en el diseño de la aplicación, ya que trabajar sobre pixeles por pulgada puede afectar si la aplicación se ejecuta en un dispositivo con una resolución menor.

A continuación, mostramos los datos sobre la cantidad relativa de dispositivos que usan una versión determinada de la plataforma Android. No se tienen en cuenta versiones con una distribución inferior al 0,1%.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	2.0%
4.2.x		17	3.0%
4.3		18	0.9%
4.4	KitKat	19	13.4%
5.0	Lollipop	21	6.1%
5.1		22	20.2%
6.0	Marshmallow	23	29.7%
7.0	Nougat	24	19.3%
7.1		25	4.0%
8.0	Oreo	26	0.5%

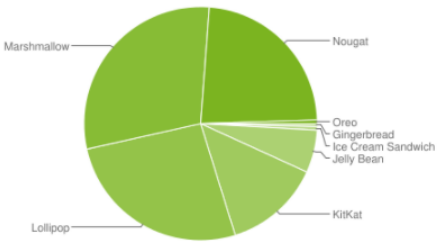


Figura 4: Datos recopilados hasta diciembre de 2017 (developer.android.com)

Respecto a los datos que mostramos sobre las versiones de Android, vemos como la mayoría de los dispositivos utilizan versiones entre 5.1 y 7.0. La última versión, 8.0 o Android Oreo, se lanzó en el Q4 de 2017 y no tiene mucha presencia actualmente.

3.1.2. Android SDK y Android Studio

El entorno de desarrollo integrado oficial para Android, Android Studio, nos proporciona las herramientas necesarias para el desarrollo del proyecto de una manera sencilla y con soporte para todo tipo de dispositivos Android. Iniciamos el proyecto con la versión 2.3.3, y actualizamos en octubre a su versión 3.0, con mejoras de rendimiento, mejoras en el emulador y en la compilación y ejecución del proyecto, entre otras.

Por lo que respecta a la versión SDK, la versión mínima es la 15, hasta la versión 27, lo que nos permite desarrollar la aplicación entre las versiones de Android 4.0.3 y 8.1.

3.2. Componentes principales

Android Studio y el SDK de Android nos ofrecen una gran variedad de componentes para poder desarrollar nuestras aplicaciones. A continuación, mostramos los componentes principales utilizados para el desarrollo de LocPic.

3.2.1. Activity

Una actividad (Activity) es un componente de la aplicación que permite a los usuarios interactuar con esta para realizar una acción. A cada actividad se le asigna un archivo XML, que contiene la interfaz de usuario. Cada actividad, puede iniciar a su vez otra actividad, para poder realizar diferentes acciones. Por lo tanto, una aplicación es un conjunto de actividades funcionando entre sí.

Cada actividad tiene su ciclo de vida. Gestionar el ciclo de vida de las actividades es fundamental para el desarrollo de la aplicación.

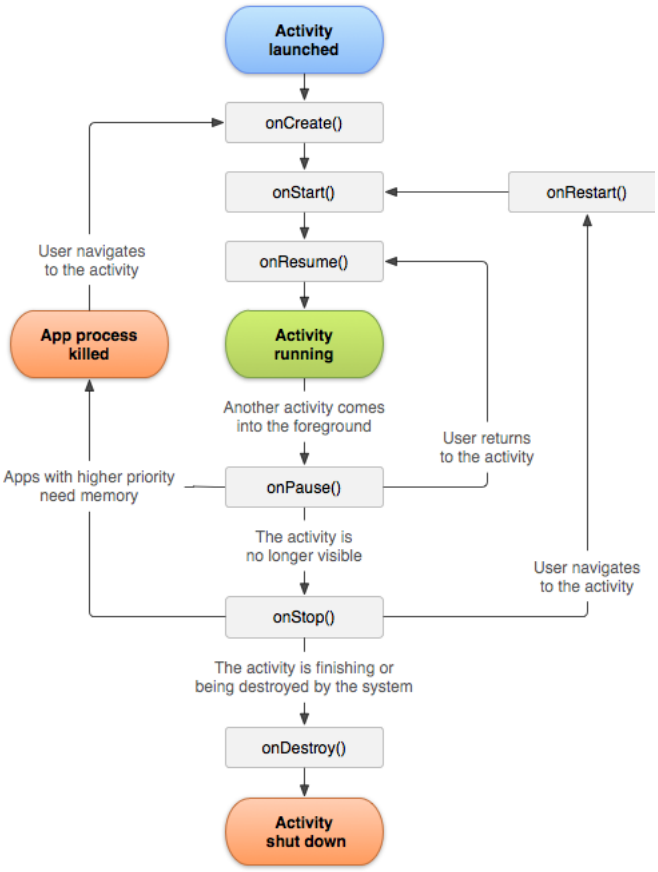


Figura 5: Ciclo de vida de una Actividad (developer.android.com)

En la figura anterior podemos ver el ciclo de vida de una actividad. Tenemos diferentes métodos "callback" en el ciclo de vida:

- **onCreate ()**. Recibe la llamada cuando se crea la actividad por primera vez. Lo utilizamos para realizar todo tipo de inicializaciones, crear vistas, enlazar datos con listas, etc.
- **onStart ()**. Recibe la llamada justo antes de que la actividad se vuelva visible para el usuario.
- **onResume ()**. Recibe la llamada justo antes de que la actividad comience a interactuar con el usuario.
- **onPause ()**. Recibe la llamada cuando el sistema está a punto de reanudar otra actividad. Este método se utiliza para confirmar los cambios sin guardar como datos persistentes, para detener animaciones u otras tareas que podrían estar consumiendo recursos de la CPU, etc.
- **onStop ()**. Recibe la llamada cuando la actividad ya no es visible para el usuario.
- **onDestroy ()**. Recibe la llamada antes de que se destruya la actividad, es la última llamada que recibirá la actividad.
- **onRestart ()**. Recibe la llamada después de que se detiene la actividad, justo antes de que vuelva a iniciarse.

3.2.2. Intent

Una Intent es un objeto de acción que se puede utilizar para solicitar una acción de otro componente de la aplicación. También los podemos utilizar para "compartir" información entre componentes.

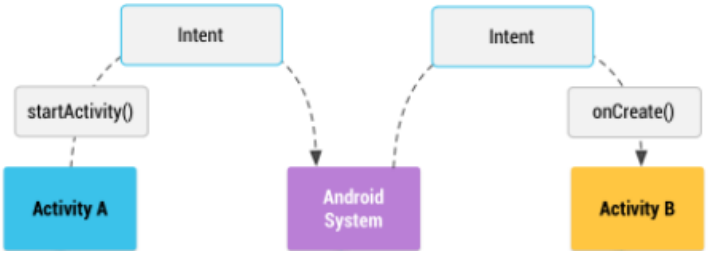


Figura 6: Entrega de una Intent para iniciar otra actividad (developer.android.com)

En la figura anterior podemos ver como una actividad A crea una Intent con una descripción de acción, la cual pasa a **startActivity**(). El sistema inicia la actividad coincidente con la Intent, en este caso actividad B, invocando su método **onCreate()**.

Desde la actividad A podríamos pasar algún dato a través del Intent, y recogerlo en el método onCreate de la actividad B para poder utilizarlo.

3.2.3. Fragment

Un fragment representa un comportamiento o una porción de la interfaz de usuario en una actividad. En una actividad podemos tener múltiples fragments, lo que nos permite tener una interfaz de usuario multi-panel. Podemos considerar que un fragment es una “subactividad” que podemos utilizar en diferentes actividades.

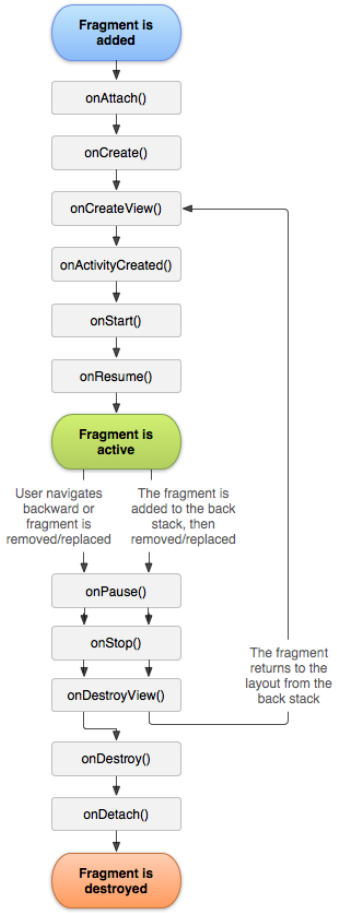


Figura 7: Ciclo de vida de un Fragment (developer.android.com)

Un fragment tiene su propio ciclo de vida. Los fragments tienen que estar integrados en actividades, con lo cual el ciclo de vida del fragment se ve directamente afectado por el ciclo de

vida de la actividad a la que pertenece. Por lo tanto, al finalizar la actividad que contiene el fragment, este también finalizará.

Cada fragment tiene su propio archivo XML y su archivo Java con su interfaz y sus métodos propios para realizar las tareas necesarias.

El uso principal de los fragments en la aplicación será en nuestro menú principal, donde tenemos una interfaz multi-panel con diferentes fragments en la misma actividad.

3.2.4. GridView y ListView

Para poder mostrar la información al usuario de una manera simple y funcional, Android nos facilita dos clases con las que podemos trabajar, la clase ListView (Lista) y la clase GridView (Cuadrícula).

Un ListView en Android es un contenedor para organizar la información en forma vertical y con la capacidad de desplazamiento (scroll) para simplificar su visualización. La clase ListView permite mostrar al usuario un conjunto de datos de forma práctica y accesible. Por ejemplo, esta clase resulta muy útil para poder mostrar las carpetas que tiene el usuario y pueda seleccionar una para mover o guardar el contenido.

Similar a ListView, un GridView en Android es una vista que contiene otras vistas en forma de cuadrícula, cuya orientación puede ser vertical u horizontal. Esta clase la podemos encontrar normalmente en las galerías de fotografías, por lo tanto, es otro componente principal que utilizamos en la aplicación.

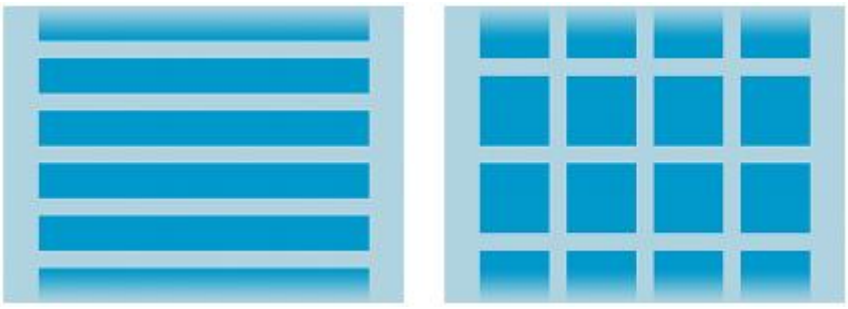


Figura 8: A la izquierda ListView, a la derecha GridView

Para ambas clases, tenemos la opción de modificar nuestro contenido de manera personalizada gracias a los Adapters. Con ellos, podremos mostrar nuestro contenido y gestionar el Array de datos a mostrar. Para ello creamos un archivo XML con el diseño con el que queremos mostrar cada elemento de nuestra lista o cuadrícula.

También dispones de los diferentes métodos para poder gestionar cuando un elemento es seleccionado para poder realizar las correspondientes tareas según la elección del usuario.

3.2.5. AsyncTask

Cuando se desarrolla una aplicación en Android, todos los componentes y tareas son ejecutados en el hilo principal. En muchos casos, son necesarios más hilos para poder realizar las diferentes tareas de una manera eficiente.

En muchas ocasiones necesitamos ejecutar varias tareas que deben presentar cambios en el hilo principal. Para ello, Android dispone de la clase AsyncTask, cuyo objetivo principal es liberar al programador del uso de hilos, la presentación de resultados en el hilo primario y la sincronización entre ellos. Esta clase nos permite gestionar de forma asíncrona la ejecución de las tareas.

Debemos extender una nueva clase con las características de AsyncTask e implementar los métodos necesarios para la ejecución en segundo plano de la tarea y la presentación de los resultados en el hilo principal. Los métodos principales son **doInBackground()**, que recibe los parámetros de entrada necesarios y nos permite ejecutar las instrucciones que irán en segundo plano y **onPostExecute(Resultados…)**, que nos permite publicar todos los resultados retornados por **doInBackground()** en el hilo principal. Los otros métodos son: **onPreExecute()**, donde irán las instrucciones necesarias antes de iniciar la tarea en segundo plano, **onProgressUpdate()**, que se utiliza para mostrar el progreso de la tarea en segundo plano y **onCancelled()**, para poder ejecutar las instrucciones que se requieran al cancelar la tarea asíncrona.

AsyncTask, por ejemplo, nos será realmente útil en el momento de descargar y mostrar las fotografías en el mapa o descargar una fotografía y guardarla en el teléfono del usuario.

3.3. Firebase

Firebase es una plataforma para el desarrollo de aplicaciones móviles y aplicaciones web. Firebase nos ayuda en el desarrollo de nuestras aplicaciones. Utiliza la infraestructura de Google y se escala automáticamente con las aplicaciones. Nos permite también ver los datos y las estadísticas de los usuarios de una manera sencilla e intuitiva.

Entre los servicios que nos ofrece Firebase, encontramos tres que serán los utilizados en la aplicación. Estos servicios son: Firebase Authentication (Autenticar), Realtime Database (Base de datos en tiempo real) y Firebase Cloud Storage (Almacenamiento de archivos en la nube).

3.3.1. Autentificación

Existen muchas aplicaciones que necesitan identificar a sus usuarios. Gestionar a los usuarios a través de la identidad nos permite que una aplicación guarde sus datos en la nube, poder gestionar el contenido y dar una experiencia personalizada en todos los dispositivos del usuario.

Con Firebase Authentication podemos gestionar a los usuarios de la aplicación. Nos proporciona servicios de “backend”, SDK y bibliotecas de IU para autenticar a los usuarios. Nos permite autenticar a los usuarios de una manera simple y segura, además, nos permite el inicio de sesión a través de proveedores externos, como por ejemplo Google, Facebook o Twitter.

Disponemos de un panel de control para poder gestionar manualmente, configurar y modificar las diferentes opciones y usuarios.

3.3.2. Realtime Database

La base de datos nos permite almacenar y sincronizar los datos de la aplicación. Los datos se sincronizan con los usuarios en tiempo real y se mantienen disponibles cuando la aplicación está sin conexión. Estos datos se almacenan en una base de datos NoSQL alojada en la nube, en formato JSON. Para gestionar los cambios de la base de datos, Firebase utiliza Listeners, que debemos gestionar en nuestra aplicación.

NoSQL es una clase de sistemas de gestión de base de datos. Cuando aparecen los servicios o aplicaciones con millones de usuarios, las bases de datos tradicionales comienzan a tener problemas de escalabilidad, y aunque estas se pueden adaptar, la complejidad de estas bases de datos aumenta. Por esta razón, aparecen las bases de datos NoSQL, que intentan solucionar el problema con una estructura de almacenamiento más versátil, aunque perdemos algunas funcionalidades, como las operaciones con más de una colección de datos. Las características principales de las bases de datos NoSQL son la ausencia de esquemas en los registros de datos

(los datos no tienen una definición de atributos fija, por lo tanto, cada registro puede contener información con diferente formato), escalabilidad horizontal y son mucho más rápidas.

3.3.3. Cloud Storage

Firebase nos permite almacenar y publicar el contenido de los usuarios, como fotos o videos.

Los SDK de Firebase para Cloud Storage realizan las operaciones de descarga y las operaciones de carga sin importar la calidad de la red. Estas operaciones son robustas, con lo cual, si en algún caso ocurre un problema y estas se interrumpen, se logra ahorrar tiempo y ancho de banda gracias a que se reinician en el punto en el que se interrumpieron.

Este servicio nos permite guardar las diferentes fotografías del usuario. Tendremos que gestionar que contenido pertenece a cada usuario y enlazar el contenido con la base de datos.

3.4. Glide vs Picasso

Disponemos de diferentes librerías que permiten la carga de imágenes de manera eficiente. En este apartado comparamos dos de ellas: Glide y Picasso.

Glide y Picasso son las librerías de carga de imágenes más utilizadas en el mundo de las aplicaciones Android. Ambas librerías proporcionan características similares, rápidas y optimizadas. Pero tienen algunas diferencias.

Para poder escoger una de las dos librerías, hemos realizado una comparación entre las librerías para saber la que debemos utilizar en el proyecto.

Primero, veamos el peso de los archivos .jar de cada una y el número de métodos disponibles.

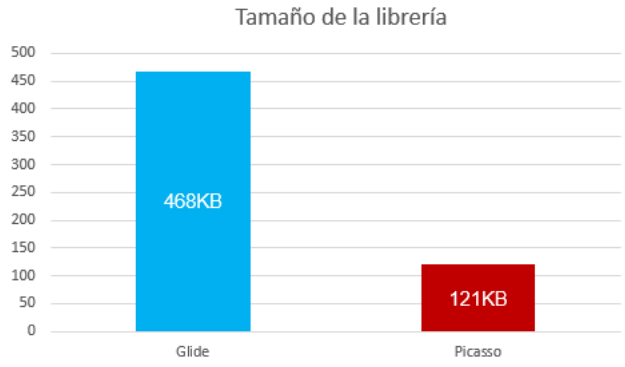


Figura 9: Tamaños de las librerías Glide y Picasso

El tamaño de la librería Glide es casi cuatro veces superior al de la librería Picasso.

Respecto al número de métodos disponibles, Picasso tiene 849 métodos, mientras que Glide tiene un total de 2678 métodos.

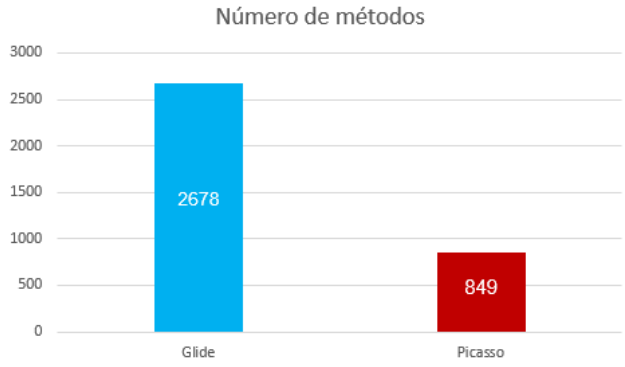


Figura 10: Métodos de las librerías Glide y Picasso

Una de las ventajas de Glide, es que nos permite encajar su funcionamiento en el ciclo de vida de la actividad y del fragment. Esto significa que, si la actividad se detiene, también lo hará la carga de la imagen, y cuando reanude su estado, lo hará también la carga de la imagen.

Respecto al almacenamiento en caché, Glide guarda la imagen en dos tamaños, una con el tamaño original y otra con el tamaño del ImageView, que es el componente de la aplicación donde se inserta la imagen. En cambio, Picasso guarda la imagen en un único tamaño, el original de la imagen. Al guardar la imagen en dos tamaños, Glide está ocupando más memoria caché que Picasso. Pero esto tiene una ventaja. Picasso, carga las imágenes la primera vez más rápido, seguramente por la razón comentada anteriormente, que únicamente guarda una imagen. Sin embargo, al volver a cargar la imagen desde caché, Glide gana la partida, ya que tiene los dos tamaños, sin embargo, Picasso tendrá que modificar el tamaño de la imagen para adaptarlo al ImageView. Esto puede beneficiar si tenemos diversas imágenes en la actividad.

Glide, es la recomendada por Google, la cual se presentó en un evento de ellos. También destacar que Glide está integrado con Firebase.

Teniendo en cuenta las diferentes características y las herramientas que vamos a utilizar en el proyecto, Glide es la opción que escogida.

3.5. API Google Maps y Google Places

Para poder utilizar los mapas y gestionar los lugares, hemos utilizado las APIs de Google Maps y Google Places. La API de Google Maps está enfocada a los mapas y su funcionalidad y la API de Google Places está enfocada a los sitios y puntos de interés.

La API de Google Maps nos permite construir un mapa personalizado en nuestra aplicación Android. Con ella, podemos utilizar el mapa y añadir los marcadores con las fotografías. Para poder utilizar la API de Google Maps tenemos que sincronizar nuestro proyecto a través de una clave, que nos dará acceso a la API, y podremos gestionar todo lo relacionado con los mapas y sus funcionalidades.

Como LocPic proporciona la posibilidad de subir fotografías realizadas en el pasado, a través de la galería del dispositivo, hemos utilizado la API de Google Places, que nos permite obtener los datos de la misma base de datos que usa Google Maps y así obtener los lugares. La API de Google Places presenta más de 100 millones de puntos de interés que se actualizan regularmente. Con ella, podremos crear nuestro buscador de ubicaciones, para que el usuario pueda buscar una dirección introduciendo el nombre de esta. También nos proporciona información de los lugares de interés. Como hemos comentado, esta funcionalidad nos será útil en el caso de subir una fotografía realizada anteriormente, ya que con esta API podemos crear el buscador de ubicaciones.

4. Análisis de requerimientos

Una tarea importante en la Ingeniería de Software es el análisis de requerimientos. En este TFG, antes del desarrollo de LocPic, hemos analizado las diferentes funcionalidades que contendrá, así como un análisis de propuestas de usuarios potenciales y un análisis de los componentes principales de otras aplicaciones que son necesarios en LocPic.

4.1. Requerimientos

Con tal de poder tener un listado de todas las opciones que tendremos disponibles en la aplicación, hemos realizado un análisis de los requerimientos principales, para conseguir los objetivos de uso. Para ello, hemos tenido en cuenta el contexto de la aplicación, así como que opciones o funcionalidades necesitamos para poder realizar las tareas que dispone el usuario.

También mostramos algunas propuestas realizadas por usuarios potenciales de la aplicación, que mostramos en este apartado, para poder añadir a nuestra lista de requisitos principales y obtener más funcionalidades.

Los requerimientos principales de la aplicación son los siguientes:

- **Registro**. Permite a los usuarios crear una cuenta para acceder al servicio. Como cada usuario tendrá su contenido, necesitamos crear un sistema de autentificación.
- **Iniciar sesión**. Una vez el usuario esté registrado, debe poder acceder a la aplicación desde cualquier dispositivo mediante sus datos. La sesión se mantendrá activa en el dispositivo hasta que el usuario cierre sesión manualmente o elimine la aplicación.
- **Cámara**. El usuario podrá utilizar la cámara de su dispositivo para poder realizar una fotografía.
- **Subir fotografía desde el dispositivo**. El usuario tendrá la posibilidad de subir una fotografía realizada anteriormente y que el usuario tenga almacenada en el dispositivo.
- **Ver las fotografías en el mapa**. Opción principal de la aplicación. Permite al usuario ver el mapa con las fotografías distribuidas en él, según la ubicación donde se realizaron.
- **Almacenar las fotografías en la nube**. Todas las fotografías estarán almacenadas en la nube, lo que permite al usuario, a través de su cuenta, tener las fotografías disponibles en cualquier dispositivo. Por lo tanto, el usuario necesitará conexión a internet para poder utilizar la aplicación.

Para explorar otros requerimientos y validar los que nos parecían más importantes (anteriormente listados), nos reunimos con un grupo de potenciales usuarios de la aplicación. La reunión se realizó con 5 personas. En este grupo de usuarios podemos encontrar “heavy users” (usuarios que utilizan mucho el teléfono móvil) y usuarios de uso medio. La edad media de los usuarios es de 21 años. Entre los usuarios, algunos de ellos viajan y visitan otras ciudades con más regularidad que otros, asisten a ferias o actos, como por ejemplo ferias de videojuegos, etc.

Antes de comenzar, les comentamos la idea de la app y los requisitos mencionados anteriormente, y les pedimos su opinión y sugerencias.

Entre las propuestas de los usuarios destacamos las siguientes, que las incorporamos en el diseño y desarrollo de la app:

- **Método de acceso alternativo**. Dar la posibilidad al usuario de acceder a la aplicación, por ejemplo, a través de su cuenta de Google o su cuenta de Facebook.
- **Galería**. Tener la posibilidad de visualizar las fotografías realizadas en forma de galería tradicional. Es decir, mostrar todas las fotografías que tiene el usuario en su cuenta.
- **Diseño.** Diseño minimalista e intuitivo.

4.2. Análisis de otras aplicaciones

Para obtener información e ideas para la implementación y el diseño de la aplicación, e identificar rasgos característicos, realizamos un análisis de una serie de aplicaciones similares. El análisis lo realizamos en los tres aspectos que vemos a continuación.

4.2.1. Inicio

Para poder utilizar LocPic, el usuario debe registrarse o iniciar sesión. Debemos mostrar al usuario las diferentes opciones que tiene para poder acceder a ella. Para poder ver la información mostrada por pantalla y el ciclo que hacen los usuarios para poder registrarse o iniciar sesión, se han analizado tres aplicaciones: Instagram, Twitter y Dropbox. Las tres son aplicaciones muy utilizadas.

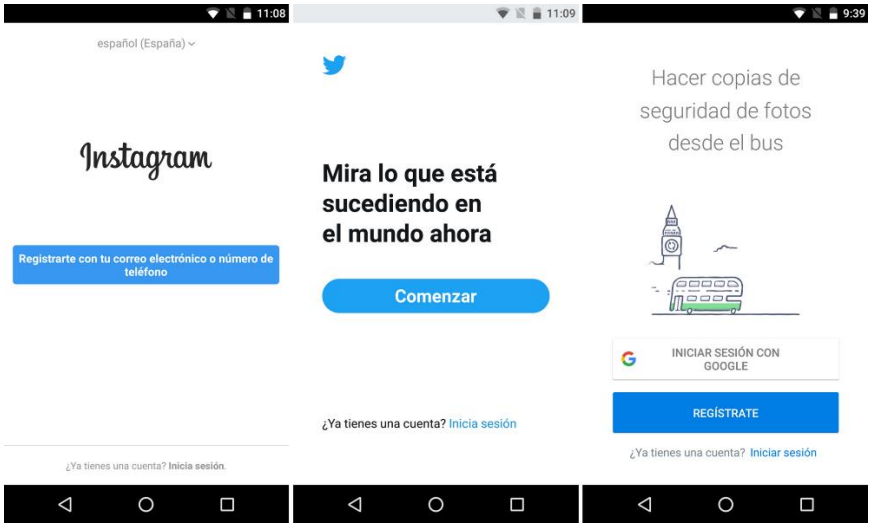


Figura 11: Pantalla inicial de Instagram, Twitter y Dropbox

Como podemos ver en la figura anterior, las tres aplicaciones muestran al usuario las diferentes opciones que tienen para poder acceder a la aplicación, en el caso de que tengan una cuenta o no la tengan. También se puede apreciar como la opción más visible es la de registrarse. La opción de iniciar sesión en los tres casos se encuentra en la parte inferior de la actividad.

En el caso de querer iniciar sesión, las tres aplicaciones solicitan el identificador del usuario (correo electrónico, nombre de usuario o número de teléfono) y la contraseña, datos que se solicitan en el apartado de registrarse, para poder identificar al usuario y mostrar el contenido que le corresponde.

4.2.2. Galerías de fotografías

La implementación de la galería con las fotografías del usuario es otro de los puntos básicos de la aplicación. Para ello, veamos cómo se distribuyen las imágenes en dos de las mejores galerías para Android, Google Fotos (la aplicación de galería de Google) y QuickPic (aplicación de galería de Cheetah Mobile), ambas con millones de descargas en la Google Play Store.

Primero, vamos a ver la distribución de las carpetas de ambas aplicaciones.

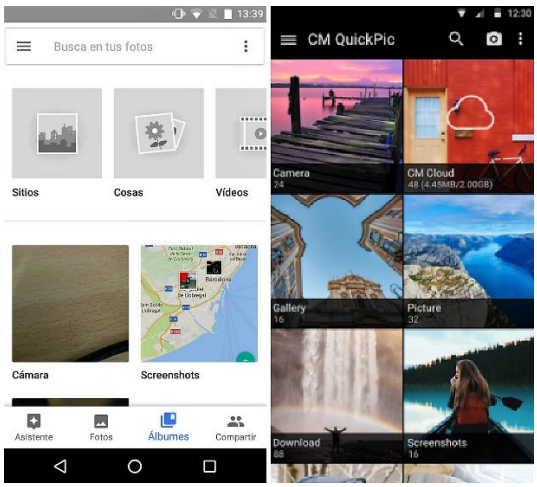


Figura 12: Visualización de las carpetas de Google Fotos y QuickPic

Vemos como el contenido se muestra con un patrón parecido, aunque tengamos más opciones en pantalla en Google fotos. En ambas las carpetas se muestran en forma cuadriculada, con una portada en cada carpeta y el nombre de la carpeta en la parte inferior. También destacar el panel de navegación inferior que nos ofrece Google Fotos, que nos permite navegar en diferentes opciones de una manera muy sencilla y visual.

Al entrar en la carpeta, el contenido se muestra de la misma forma, en formato cuadriculado. Destacar que Google Fotos nos muestra también una organización por fechas dentro de cada carpeta. Dentro de la carpeta, si mantenemos presionado una fotografía, nos permite gestionarla. Entre estas opciones encontramos mover o eliminar la fotografía.

Al seleccionar la fotografía que queremos visualizar, esta se muestra en una nueva pantalla con un tamaño superior. QuickPic tiene una opción para ver la ubicación donde se realizó una fotografía, pero solo nos marca el lugar donde se realizó dicha fotografía.

4.2.3. Visualización de fotografías en mapas

Finalmente, respecto a la característica principal de poder visualizar las fotografías en el mapa, nos centraremos en una opción que encontrábamos en Instagram hace un par de años, ya que, haciendo una búsqueda en la tienda de aplicaciones de Google, tenemos aplicaciones que no funcionan correctamente o no aportan ideas a la realización del proyecto.

Figura 13: Distribución de las fotografías en el mapa que nos ofrecía Instagram

En versiones anteriores de Instagram, teníamos la posibilidad de poder visualizar las fotografías que hemos realizado distribuidas en un mapa, para así ver los lugares que hemos visto. Esta es la

característica principal del proyecto, pero de carácter personal. Esta opción que nos ofrecía Instagram fue desapareciendo poco a poco entre las versiones actualizadas de la aplicación. La eliminación de la opción fue a causa de que querían encarrilar más la aplicación al compartir fotografías y ceñirse más a lo que hacia la competencia. Esto lo pudimos ver claramente con las historias, que se introdujeron recientemente y que ya podíamos ver anteriormente en otra aplicación, Snapchat.

Respecto a la visualización de las fotografías en el mapa, vemos como se muestran en forma cuadricular y con un marco. Los números que aparecen en la parte superior, corresponde al número de fotografías que tenemos en una determinada zona, ya que, al alejar la cámara del mapa, estás se agrupan en clústeres.

4.2.4. Conclusión de las aplicaciones

Tras analizar y visualizar como otras aplicaciones solicitan y muestran la información al usuario, seguiremos una estética basada en las aplicaciones analizadas en los puntos anteriores.

Para estar seguros de que el contenido que mostramos y la información que solicitamos es la correcta y necesaria, realizamos un test con usuarios una vez disponíamos de un prototipo funcional de la aplicación. En este test, los usuarios indican si la información que mostramos y solicitamos es la correcta, el tamaño y la posición de los componentes es la correcta, y que información añadirían o eliminarían de la aplicación para facilitar su uso (Ver *6.2 Evaluación de la interfaz de usuario*).

5. Desarrollo de la aplicación

En este apartado, veremos las partes más importantes de la etapa de desarrollo. Lo primero que mostramos son los diferentes casos de uso, según si el usuario está identificado (ya dispone de una cuenta en LocPic) o no está identificado.

Tras los casos de uso, abordaremos el desarrollo y gestión las funcionalidades principales de la aplicación, la estructura de la base de datos y la gestión del almacenamiento.

5.1. Casos de uso

Los diagramas de casos de uso nos permiten ver de una forma visual las acciones que puede realizar un usuario.

Los usuarios pueden tener dos roles diferentes: pueden estar identificados o no estar identificados. Por lo tanto, tenemos dos tipos de actores. Según el rol en el cual se encuentre el usuario, este tendrá diferentes opciones.

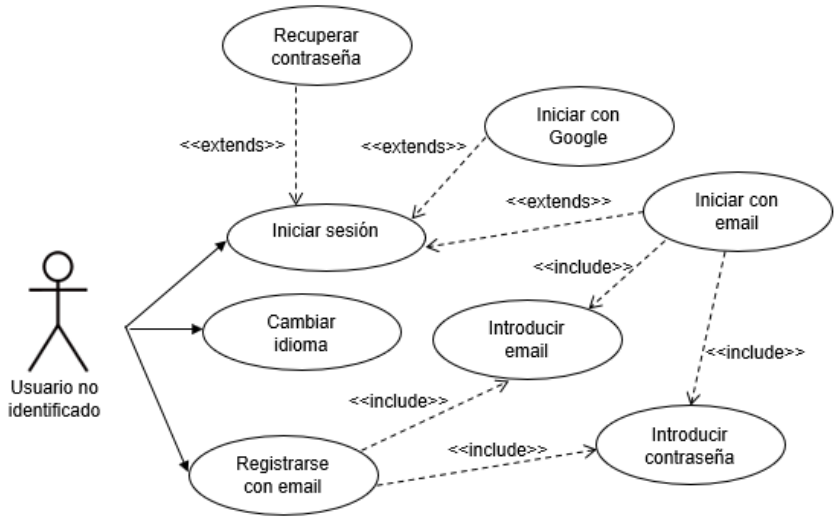


Figura 14: Caso de uso de un usuario no identificado

En el caso de que el usuario no esté identificado, tiene la opción de registrarse para poder hacer uso de la aplicación o de iniciar sesión si ya tiene una cuenta.

Para registrarse debe introducir su correo electrónico y su contraseña, con las cuales después puede iniciar sesión. Si el usuario lo prefiere, puede iniciar sesión directamente a través de su cuenta de Google.

Además, si el usuario tiene una cuenta y no recuerda su contraseña, tiene la opción de recuperarla. Para ello, debe introducir el correo electrónico con el cual se registró en la aplicación.

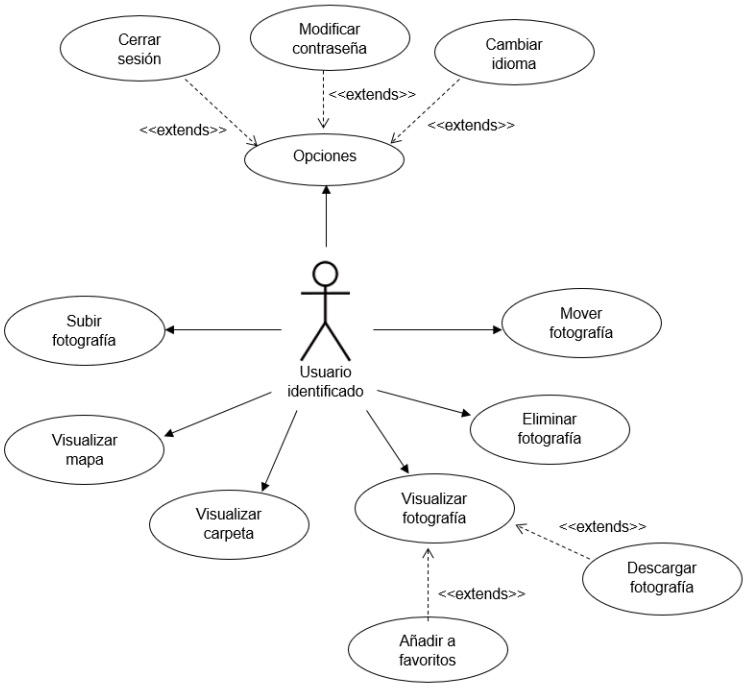


Figura 15: Caso de uso de un usuario identificado

En la figura anterior podemos ver los casos de uso de un usuario que ya está registrado y ha accedido a la aplicación. Este usuario puede moverse entre las diferentes opciones que tiene disponibles en la aplicación.

El caso de uso de *Subir fotografía* podemos descomponerlo, ya que el usuario dispone de dos opciones. El caso de uso quedaría de la siguiente forma:

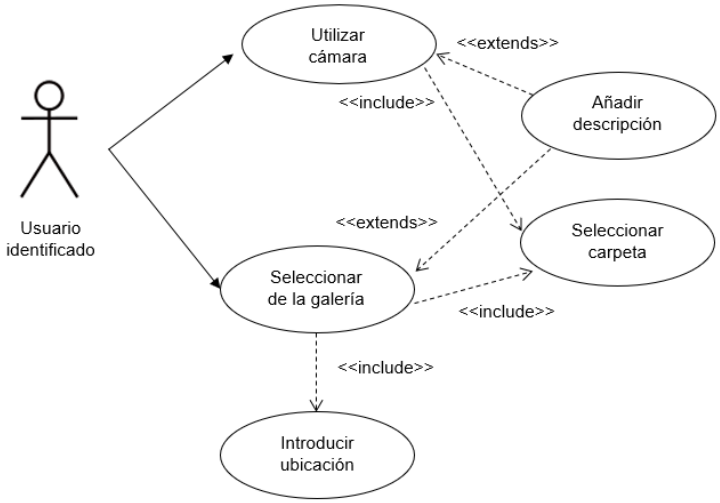


Figura 16: Caso de uso de un usuario identificado para subir una fotografía

Como podemos apreciar en la figura anterior, en el caso de que el usuario seleccione la opción de realizar una fotografía mediante la cámara del dispositivo, tendrá la opción de añadir una descripción y tendrá que elegir o crear la carpeta de destino de la fotografía. En el caso de que el usuario decida subir una fotografía de la galería del dispositivo, además de tener la posibilidad de añadir una descripción o tener que elegir o crear la carpeta de destino, debe introducir la ubicación en la cual se realizó la fotografía.

5.2. Sincronización del proyecto con Firebase

Antes de comenzar a desarrollar el proyecto, debemos sincronizar nuestro proyecto creado en Android Studio, con Firebase.

Para ello, tras iniciar sesión en Firebase, añadimos un nuevo proyecto. Al crear el nuevo proyecto, nos ofrece sincronizar nuestra aplicación Android, iOS o Web.

Lo primero que nos solicita es el nombre del paquete de Android, el cual debe coincidir con el que hemos asignado a nuestro proyecto. También nos da la posibilidad de introducir un apodo, para poder diferenciarlo en nuestra cuenta de Firebase.

El segundo paso consiste en descargar el archivo de configuración, que es un archivo JSON que debemos incluir en la carpeta *app* de nuestro proyecto.

Finalmente, debemos añadir el SDK de Firebase. Para ello tenemos que incluir la dependencia en el archivo *build.gradle* del proyecto y el plugin que nos permite utilizar los servicios de Google en el *build.gradle* de la aplicación.

Además, debemos añadir las bibliotecas con las diferentes funciones de Firebase. Las bibliotecas que necesitamos son la biblioteca de autentificación, la biblioteca de la base de datos y la biblioteca del almacenamiento, que nos proporcionan las funciones necesarias para poder gestionar el contenido.

Tras estos pasos, ya tendremos nuestro proyecto sincronizado con Firebase.

5.3. Paquetes del proyecto

A continuación, mostramos como hemos organizado las clases del proyecto.

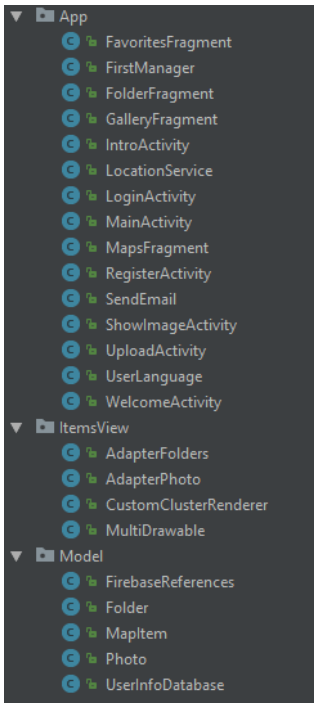


Figura 17: Contenido de los paquetes del proyecto

Hemos divido el contenido en 3 paquetes, que comentamos a continuación.

- **Model**. Contiene la representación de los datos que maneja el sistema. Son las clases que denominaríamos lógica de negocio. Encontramos las clases que permiten utilizar los objetos

que guardamos en la base de datos de Firebase con sus mismos atributos, la clase para representar un elemento del mapa y las referencias de Firebase.

- **ItemsView**. Contiene todas las clases que permiten la representación de los elementos de los componentes de la aplicación. Encontramos los Adapters para las fotografías y las carpetas, la clase *CustomClusterRenderer* que permite la representación de las fotografías y los clústeres en el mapa y la clase *MultiDrawable* que permite representar las fotografías dentro de un clúster con más de un elemento.
- **App**. Contiene todas las actividades y fragments de la aplicación, además de clases para la gestión del contenido.

La división de los paquetes está adaptada al proyecto pensando en una aproximación que nos permita facilidad para mantener, escalar y testear la aplicación.

5.4. Autentificación

Para poder utilizar la aplicación, hemos creado un sistema de autentificación. Para ello utilizamos Firebase. Antes de comenzar a desarrollar este apartado, debemos activar los métodos de inicio de sesión que utilizaremos en el panel de control de Firebase. En nuestro caso, habilitamos el proveedor de correo electrónico/contraseña y el de Google.

Una vez configurado, podemos comenzar a diseñar nuestro registro de usuarios y el inicio de sesión.

5.4.1. Registro

Hemos creado una actividad llamada *RegisterActivity* para gestionar el registro del usuario. Para ello, mostramos tres componentes para que el usuario pueda introducir los datos necesarios, en este caso, el correo electrónico, la contraseña y la confirmación de la contraseña.

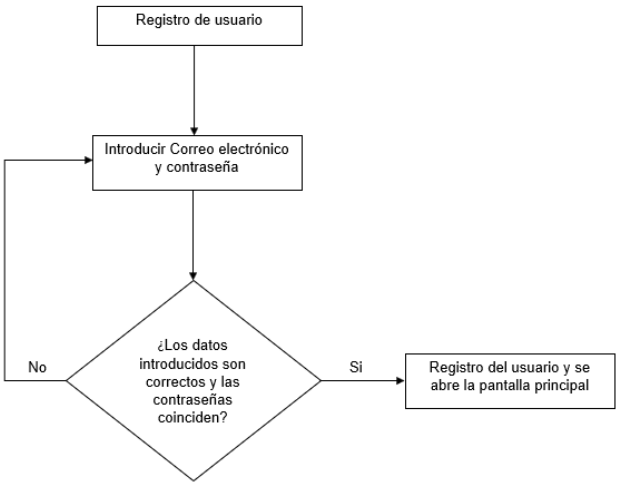


Figura 18: Diagrama de flujo del registro de un usuario

Los componentes tienen una serie de controles para verificar que se haya introducido un correo electrónico o que la contraseña cumpla con unos requisitos de complejidad. Para el correo electrónico, se comprueba que el usuario haya introducido el símbolo @ en el campo de introducción de texto. Para la contraseña, se exige un mínimo de 6 caracteres para que sea válida.

Además, el usuario debe confirmar la contraseña. En caso de no cumplir con alguno de los requisitos, se muestra al usuario que sus datos introducidos no son válidos en el componente de introducción de texto correspondiente y no se procede con el registro.

Si los datos son correctos, podemos proceder con el registro. Para ello, se obtiene una instancia del objeto FirebaseAuth. Para crear la cuenta, llamamos al método que nos proporciona Firebase, **createUserWithEmailAndPassword()**, al cual pasamos como parámetros el correo electrónico y la contraseña del usuario. Debemos gestionar si la tarea se ha completado o no. Si la respuesta es afirmativa, se procede a entrar a la aplicación y gestionar los datos. En caso contrario, se muestra un error al usuario.

Al crear la cuenta del usuario, tendremos un UID de usuario, que es un identificador único que nos permite posteriormente gestionar todo el contenido, la base de datos y el almacenamiento de las fotografías para ese usuario determinado. También podemos inicializar los valores en base de datos para este usuario.

5.4.2. Iniciar sesión

Para el inicio de sesión hemos creado la actividad *LoginActivity*. A la hora de iniciar sesión, el usuario tiene disponibles dos opciones: iniciar sesión mediante la cuenta creada a partir de su correo electrónico o iniciar sesión mediante su cuenta de Google.

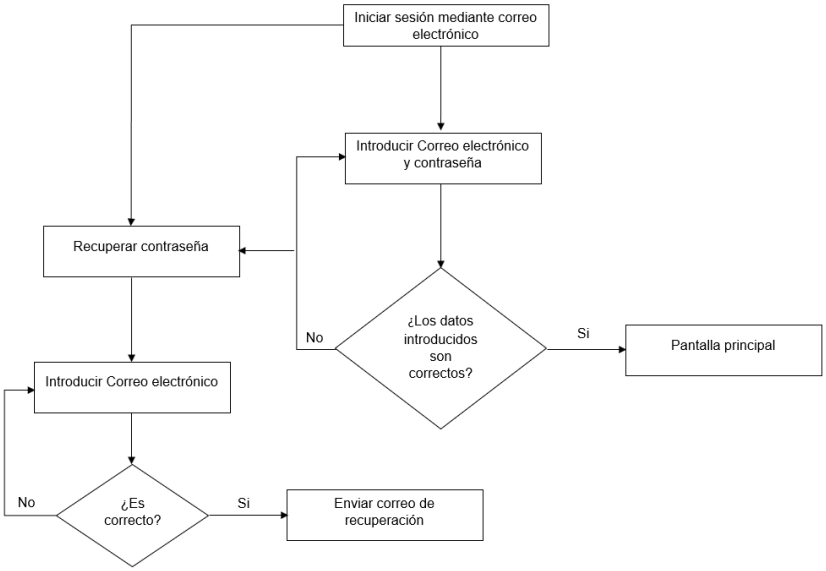
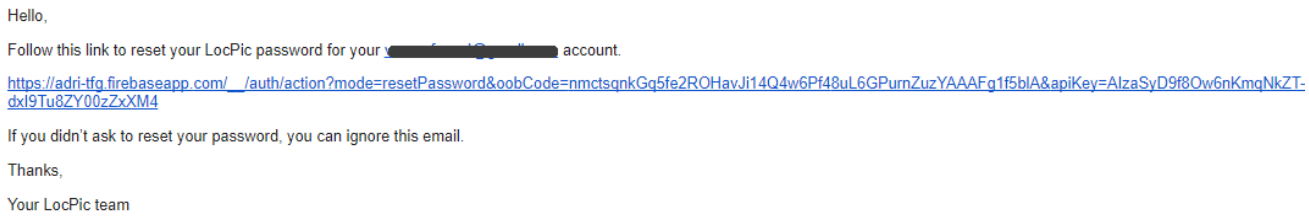


Figura 19: Diagrama de flujo de iniciar sesión mediante correo electrónico

En el caso de querer iniciar sesión mediante su correo electrónico, el usuario debe antes registrarse. Se muestra al usuario dos componentes para poder introducir su correo electrónico y su contraseña, con los mismos controles de datos que en el registro. La comunicación con Firebase es similar. Obtenemos la instancia de FirebaseAuth y la utilizamos para llamar al método **signInWithEmailAndPassword()**, al cual le pasamos como parámetros los datos introducidos por el usuario. En el caso de que los datos sean correctos, mostramos la pantalla principal y si no lo son, informamos al usuario de que ha ocurrido un error.

Si el usuario no recuerda la contraseña, tiene la opción de modificarla. Para ello, debe introducir su correo electrónico con el cual se registró en la aplicación. A partir de este correo electrónico, se envía un formulario de modificación de contraseña utilizando Firebase. El método utilizado es **sendPasswordResetEmail()**, y como en las interacciones anteriores de registro o iniciar sesión, para utilizar esté método se necesita una instancia de FirebaseAuth. Debemos gestionar nosotros el resultado de la tarea, así como la información del mensaje que recibirá el usuario. En la siguiente figura, vemos un ejemplo del correo que recibe el usuario para poder modificar la contraseña.


Hello,

Follow this link to reset your LocPic password for your [redacted] account.

https://adri-tfg.firebaseapp.com/__/auth/action?mode=resetPassword&oobCode=nmctsqnkGq5fe2ROHavJi14Q4w6Pf48uL6GPurnZuzYAAAFg1f5blA&apiKey=AIzaSyD9f8Ow6nKmqNkZT-dxI9Tu8ZY00zZxXM4

If you didn't ask to reset your password, you can ignore this email.

Thanks,

Your LocPic team

Figura 20: Mensaje que recibe el usuario para poder modificar su contraseña

En el caso de querer iniciar sesión mediante su cuenta de Google, este debe tener una cuenta asociada en el dispositivo o crear una nueva cuenta de Google. Para hacer uso del servicio de autentificación de usuarios mediante la cuenta de Google, debemos configurar el proyecto en Firebase.

Debemos obtener nuestra huella digital de certificado SHA1 de nuestro proyecto Android, y añadirla a la configuración del proyecto en Firebase. Esta huella digital se utiliza para crear un cliente de OAuth2 (protocolo de autentificación) y una clave de API para la aplicación, para poder utilizar las herramientas que nos ofrece la API de Google para la autentificación de usuarios. La huella SHA1 la podemos obtener fácilmente en el apartado de Gradle de nuestro proyecto Android, entrando en la carpeta *Tasks*. En la carpeta *Tasks*, accedemos a la carpeta *Android*, hacemos clic derecho en el archivo *signingReport* y seleccionamos Run. En la consola de Android Studio nos mostrará la clave SHA1, entre otros datos del proyecto.

Una vez realizado este paso, ya podemos añadir el botón de inicio de sesión con Google a nuestra interfaz. Destacar, que esté botón tiene unos patrones de diseño que no podemos modificar.

Para dar funcionalidad al botón, debemos crear un nuevo Intent utilizando la API de Google de la siguiente forma:

```
Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(googleApiClient)
```

Iniciamos la actividad que se encarga de mostrar el fragment con las cuentas del dispositivo de Google y la opción de crear una nueva cuenta de Google. Una vez el usuario interactúe con las opciones disponibles, gestionamos el resultado de la actividad. Si el resultado es correcto, añadimos la cuenta de inicio de sesión mediante una cuenta de Google a Firebase. Tras gestionar los datos, mostramos la pantalla principal o en el caso de que haya ocurrido un problema, informamos al usuario de que no ha sido posible iniciar sesión.

5.5. Subir fotografía

Como hemos visto en el apartado de casos de uso, el usuario tiene dos formas de poder subir sus fotografías, realizando una fotografía con la cámara del dispositivo desde la aplicación o seleccionándola de la galería. Dependiendo del método, tendremos que gestionar la fotografía y su subida de una manera diferente. También debemos tener en cuenta la ubicación donde se realizó la fotografía, ya que la necesitamos para poder ubicarla en el mapa. La clase que se encarga de la gestión de la subida de la fotografía y de mostrar los datos correspondientes es *UploadActivity*.

5.5.1. Posición

Tenemos dos formas de obtener la ubicación del usuario: a través de la red o a través del GPS del dispositivo. A través de la red nos permite obtener la ubicación de una forma rápida, aunque un poco menos precisa que si utilizamos la opción del GPS, que es más precisa, pero puede llegar a tardar más de 30 segundos en obtener la ubicación. La opción de obtener la ubicación a través de

GPS es más utilizada, por ejemplo, en aplicaciones de monitorizar rutas a la hora de hacer running, gracias a su precisión. En cambio, para aplicaciones que quieren obtener la posición del usuario en un momento determinado, es más utilizada la opción de la red. Además, la opción GPS no funciona correctamente en interiores. Por lo tanto, hemos utilizado la opción de obtener la ubicación mediante la red.

Hemos definido una clase, llamada *LocationService*, que implementa a LocationListener y nos permite, no solo gestionar la ubicación del usuario, sino también el estado del proveedor de la ubicación o si el usuario tiene o no activada la ubicación.

Para obtener la ubicación del usuario creamos una instancia de LocationManager. Esta clase nos permite el acceso a los servicios de ubicación. Permite hacer una única petición o peticiones periódicamente. Nosotros realizamos una única petición. Debemos indicar también el servicio que deseamos utilizar (GPS o Red) y pasar como parámetro también nuestro *LocationService*, que se encargará de gestionar la localización.

A continuación, mostramos un ejemplo para realizar una petición.

```
locManager.requestSingleUpdate(LocationManager.NETWORK_PROVIDER),
                        locListener, null);
```

En el caso de querer múltiples peticiones, debemos llamar a **requestLocationUpdates()** e indicarle el tiempo de actualización entre peticiones, además del tipo de servicio y nuestro *LocationService*.

Para trabajar con la ubicación utilizamos la latitud y la longitud. Estas, es importante que sean de tipo double, ya que estas pueden ser largas y negativas.

A partir de la latitud y la longitud, obtenemos el nombre de la dirección, para así mostrar al usuario el nombre donde está ubicado, y no los valores de latitud y la longitud. Para ello creamos un Geocoder, que es una clase que nos proporciona Android, y obtenemos la lista de direcciones asignadas a partir de la latitud y longitud que hemos obtenido del dispositivo.

Además, como el usuario puede subir una fotografía que realizó anteriormente, debemos crear un buscador de ubicaciones, para que pueda introducir la ubicación de la fotografía. Esta forma de introducir la ubicación únicamente la necesitaremos cuando el usuario decida subir una fotografía de su dispositivo.

Mediante la API de Google Places, podemos crear este buscador. Para ello, necesitamos crear un nuevo Intent. Tenemos la opción de lanzar el buscador como un fragment (método utilizado) o creando una nueva actividad.

```
                Intent intent = new
PlaceAutocomplete.IntentBuilder(PlaceAutocomplete.MODE_OVERLAY)
                .build(UploadActivity.this);
startActivityForResult(intent, PLACE_AUTOCOMPLETE_REQUEST_CODE);
```

El segundo parámetro que asignamos al comienzo de la actividad es el código que nos permite gestionar posteriormente su resultado.

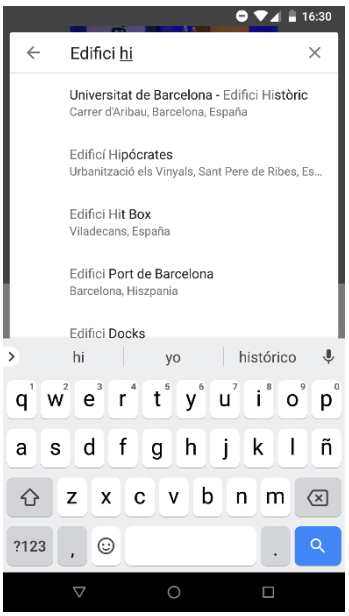


Figura 21: Buscador de lugares

Una vez el usuario busque y seleccione un lugar, debemos gestionar el resultado. Obtenemos la latitud y la longitud del lugar que ha seleccionado el usuario y que nos devuelve la API y obtenemos la dirección del lugar, para mostrarla al usuario, y actualizamos los valores de latitud y longitud.

Finalmente, recordar que necesitamos añadir los permisos de ubicación al proyecto para su funcionamiento.

5.5.2. Cámara

El usuario puede utilizar la cámara de su dispositivo para realizar una fotografía desde la aplicación y subirla a la nube. Para ello, creamos un Intent que permite utilizar la cámara del dispositivo de la siguiente forma:

```
Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
```

Una vez creado el Intent, antes de lanzar la actividad, creamos un archivo temporal, donde guardamos la fotografía para poder trabajar con ella una vez el usuario la realice. Para crear el archivo temporal utilizamos el método **createTempFile()** de la clase File de Android, que recibe como parámetros el nombre del archivo (en nuestro caso generamos un nombre a partir de la fecha actual), la extensión del archivo y su directorio. Una vez guardamos la fotografía en este archivo temporal, no es visible para el usuario en la galería de su dispositivo, pero se puede visualizar a través de un gestor de archivos y nosotros podremos trabajar con ella.

Destacar que, no es necesario que el usuario haya aceptado los permisos de almacenamiento para crear y guardar este archivo.

Cuando el usuario realice la fotografía, pasamos a una nueva actividad (*UploadActivity*) para mostrar la fotografía realizada y las opciones que tiene el usuario antes de subir su fotografía. A esta actividad, le pasamos como parámetro la ruta del archivo temporal creado, donde tendremos la fotografía capturada por el usuario, además de un código que nos permite diferenciar si la fotografía ha sido realizada con la cámara o seleccionada del dispositivo, para poder gestionar en la nueva actividad el proceso de subida, ya que como veremos, varía según el método utilizado.

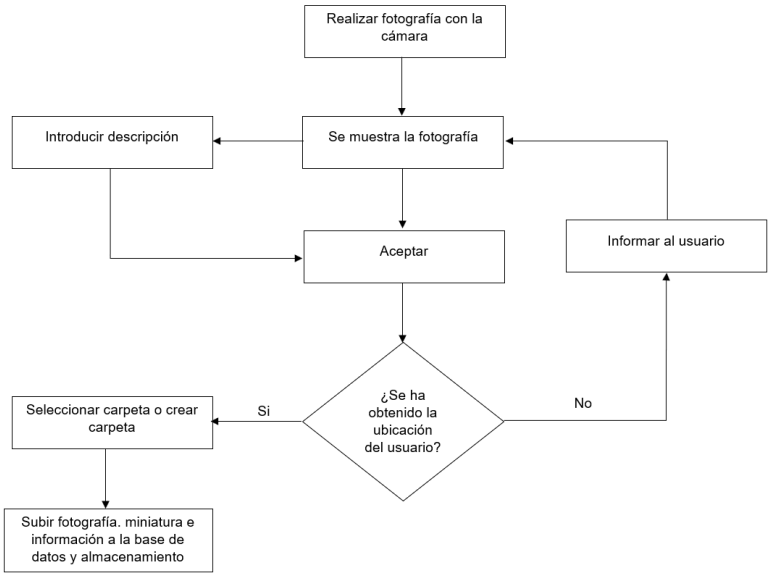


Figura 22: Diagrama de flujo de la subida de una fotografía realizada con la cámara

Si la fotografía ha sido realizada con la cámara, el usuario tendrá disponible la opción de escribir una descripción. Obtenemos la localización automáticamente, por lo cual el usuario necesita tener activada la ubicación. Si el usuario no tiene la ubicación activada o los permisos, se le indica con un mensaje, ya que, no puede proceder con el siguiente paso sin la localización. Para saber si tiene la ubicación activada utilizamos nuestra clase *LocationService*.

Cuando el usuario decida subir la fotografía, le aparecerá un listado de las carpetas que tiene creadas, o la posibilidad de crear una nueva. El usuario debe seleccionar o crear la carpeta de destino de la fotografía. Una vez seleccionada, tendremos los datos necesarios y podemos subir la fotografía al servidor y la información a la base de datos. Generamos un nombre con la fecha actual y un número aleatorio para poder almacenar la fotografía en el servidor y en la base de datos.

Además, se almacena otra fotografía de baja resolución, que utilizamos para las miniaturas de la galería y en el mapa. Esto nos permite utilizar menos red al cargar las miniaturas de las fotografías y mejorar la velocidad de carga del contenido, ya que, si el usuario tiene por ejemplo 30 fotografías, no es lo mismo cargar 30 fotografías de 5MB cada una, que 30 fotografías de 100KB.

Destacar también, que en este caso debemos subir la fotografía y la miniatura en bytes, ya que el almacenamiento temporal no nos permite subir la fotografía a través de la ruta donde está almacenada.

5.5.3. Galería del dispositivo

Como alternativa, el usuario puede subir una fotografía que tenga en su dispositivo y realizó anteriormente. Al escoger esta opción, mostramos al usuario las galerías disponibles en su dispositivo y tras seleccionar una, esta se abrirá. El usuario puede navegar y buscar la fotografía que desea subir.

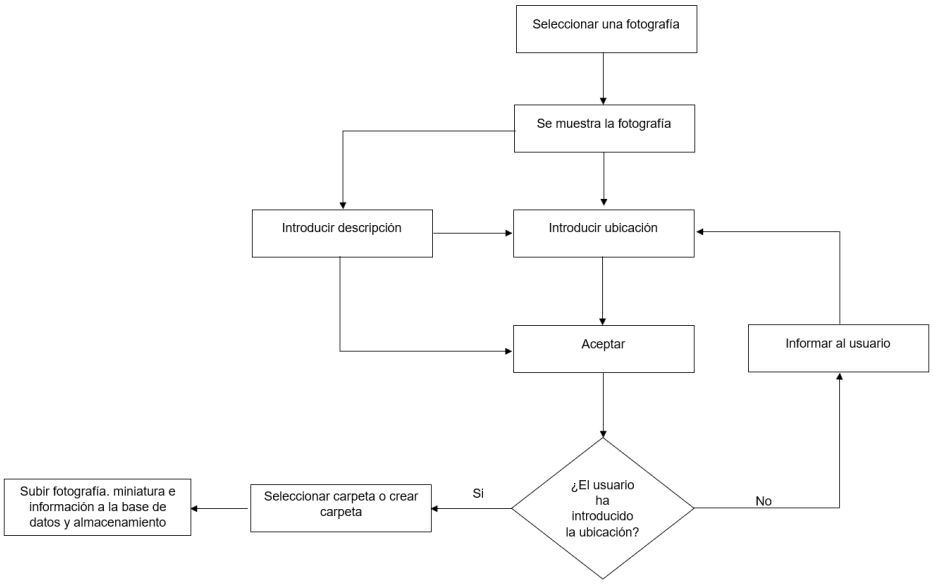


Figura 23: Diagrama de flujo de la subida de una fotografía de la galería del dispositivo

La fotografía seleccionada pasa a una nueva actividad (*UploadActivity*), para mostrar la fotografía seleccionada y las opciones que tiene el usuario antes de subir su fotografía. El usuario dispone de un botón que permite abrir el buscador de lugares, para poder introducir la ubicación.

Sabemos si tenemos que mostrar o no el botón de búsqueda según el código que enviamos a esta nueva actividad, para diferenciar una fotografía seleccionada de la galería y una realizada con la cámara desde la aplicación.

Una vez introducidos los datos, como en el caso de subir una fotografía realizada con la cámara, se mostrarán las carpetas que tiene creadas, o la posibilidad de una nueva. Al seleccionar o crear la carpeta, se procederá a subir la fotografía al servidor y la información a la base de datos. Generamos un nombre con la fecha actual y el nombre que tenía la fotografía en el dispositivo para guardarla en la base de datos y el almacenamiento.

Además, guardamos también una miniatura de la fotografía como en el caso anterior, que nos permitirá agilizar la carga de las fotografías y optimizar el consumo de red.

5.6. Galería

Para visualizar las fotografías, el usuario dispone de una galería con sus fotografías, distribuidas en las carpetas que ha creado. Además, ofrecemos la posibilidad de visualizar sus fotografías favoritas de una forma rápida.

5.6.1. Carpetas

Lo primero que se muestra al usuario son sus carpetas. Como tenemos un panel dinámico, el fragment *GalleryFragment* se encarga de mostrar este contenido. Para poder mostrar las carpetas de un usuario, accedemos a la base de datos y hacemos una petición para obtener las carpetas del usuario. Para ello, necesitamos el UID del usuario, para obtener la información de ese usuario determinado.

Realizamos la petición a Firebase (ver *5.9. Base de datos*) e iteramos sobre los datos recibidos, que son las carpetas del usuario. Las carpetas las guardamos en un Array de objetos de tipo *Folder,* que es la clase que hemos creado y contiene los atributos que tiene una carpeta. Además, también accedemos al almacenamiento para obtener la portada de la carpeta.

Figura 24: Visualización de las carpetas de la galería

Una vez tenemos las carpetas en nuestro Array, necesitamos crear un Adapter. Este objeto Adapter permite el acceso a los elementos de datos y también es responsable de crear una vista para cada elemento de la colección. La clase Adapter de las carpetas es *AdapterFolders.* Por lo tanto, debemos crear nuestro Adapter, donde configuramos la forma en que se visualizan las carpetas. Para cada carpeta, necesitamos su nombre y su portada. Para el diseño del contenido necesitamos definir un nuevo archivo XML con el contenido para mostrar las carpetas.

Una vez creado, lo asignamos al componente que hemos creado para mostrar los datos, en este caso un GridView, que nos permite mostrar las diferentes carpetas en una lista cuadriculada. Las carpetas se muestran ordenadas alfabéticamente.

5.6.2. Fotografías

El usuario podrá ver sus fotografías ordenadas en las carpetas que haya creado, o a través del apartado de favoritos, además claro, del mapa.

Cuando el usuario accede a una carpeta, mostramos las fotografías que esta contiene. A partir del nombre de la carpeta, accedemos a la base de datos y obtenemos las fotografías que pertenecen a esa carpeta. Cada fotografía, en su información, contiene la carpeta a la que pertenece. Recordar que, se necesita una clase con los mismos atributos que tiene cada objeto fotografía en la base de datos. Para obtenerlas, iteramos sobre el resultado de la petición a Firebase y guardamos en una Array de objetos *Photo* las que pertenecen a la carpeta.

Una vez tenemos la lista de fotografías, como en el caso de las carpetas, necesitamos crear un Adapter (*AdapterPhoto*) y un archivo XML con los componentes necesarios para mostrar la fotografía. Este Adapter será diferente al de las carpetas, ya que el contenido que se muestra no es el mismo. Una vez tengamos creado el Adapter y asignado el Array de objetos de fotografías, lo añadimos a nuestro componente, que en este caso también será un GridView. Las fotografías se ordenan por orden de subida.

Además, desde el interior de una carpeta podemos gestionar las fotografías, moviendo la fotografía de carpeta, o eliminándola. Para ello hacemos una pulsación larga, lo que nos desplegará las dos opciones que tenemos en una ventana.

Para mover las fotografías, mostramos al usuario sus carpetas y la opción de crear una nueva. Tras la selección del usuario, realizamos un cambio de información de la base de datos sobre las carpetas afectadas y la fotografía. En el caso de querer eliminar la fotografía, no solo eliminamos la información de la base de datos, sino que también necesitamos eliminar la fotografía en el almacenamiento y su miniatura. Además, estas acciones requieren el control de las carpetas que las contienen.

Si el usuario desea visualizar la fotografía, debe seleccionarla. Al seleccionar una fotografía, esta pasa a una nueva actividad (*ShowImageActivity*) para verla con mejor tamaño y con su respectiva información. La fotografía que mostramos es la de tamaño real, ya no la miniatura. En esta nueva actividad el usuario puede visualizar la información de la fotografía, un botón para añadirla a favoritos y la opción de descargar la fotografía.

Realizamos una petición al almacenamiento de Firebase para obtener la dirección de descarga de la fotografía. Para esta descarga, definimos una nueva clase que hereda de AsyncTask donde creamos el archivo para guardar la fotografía y realizamos la descarga. A continuación, mostramos como obtener la dirección de descarga de la fotografía a partir de la dirección donde está almacenada (*imgUrl*, obtenida de la información de la fotografía)

```
storageReference.child(imgUrl).getDownloadUrl().addOnSuccesListener(...)
```

La llamada anterior nos devuelve la Uri de descarga de la fotografía que pasamos a nuestro AsyncTask. Para proceder con la descarga hacemos lo siguiente:

```
Glide.with(getApplicationContext()).load(uri.toString()).asBitmap()
                        .into(-1, -1).get();
```

El resultado lo guardamos en un Bitmap. El *into(-1,-1)* permite descargar la fotografía a tamaño completo.

Para la fotografía generamos un nombre a partir de la fecha de descarga, que será el nombre de la fotografía en el dispositivo y creamos el archivo *File* para guardarla en el dispositivo. Las fotografías descargadas se almacenan en la carpeta *descargas* del dispositivo, obteniendo la dirección de esta carpeta del terminal. Recordar que, el usuario debe aceptar los permisos de almacenamiento para realizar esta acción.

5.7. Mapa

La característica principal de LocPic, es la posibilidad de visualizar las fotografías distribuidas en un mapa. Hemos utilizado la API de Google Maps para realizar esta característica. El mapa lo mostramos en un fragment (*MapsFragment*). En este apartado, se presenta como se posicionan, se visualizan y se agrupan las fotografías en el mapa.

5.7.1. Ítems del mapa

Un ítem (también conocidos como marcadores o “markers”) es cada elemento que visualizamos en el mapa. Hemos definido una clase llamada *MapItem* que contiene la información necesaria para la implementación de cada ítem. Necesitamos crear esta clase ya que nuestros marcadores son personalizados, no son los marcadores por defecto de Google. Como trabajamos con clústeres, está clase implementa a la clase ClusterItem de la API de Google.

Cada ítem hace referencia a una fotografía y contiene su descripción, su posición y un bitmap, que es la miniatura de la fotografía. Para crear los ítems, realizamos una petición a Firebase para obtener todas las fotografías del usuario. Definimos una clase llamada *ImageTask* que hereda de

AsyncTask y se encarga de la descarga de las miniaturas para cada una de las fotografías. Por lo tanto, iteramos sobre las fotografías del usuario y creamos un ImageTask para descargar la miniatura de cada fotografía. La utilización de AsyncTask es necesaria para evitar los problemas que conlleva la carga de las fotografías mediante la red, porque recordamos, que todas las fotografías están en el servidor, y no utilizarlo genera problemas, sobre todo en dispositivos con memoria y/o procesador con menos capacidades y esto puede producir el famoso “Forzar cierre”.

Una vez se realiza la descarga de cada miniatura de la fotografía, la asignamos a un Bitmap y en el método **onPostExecute()** de nuestra clase *ImageTask*, creamos un ítem del mapa con los datos necesarios de la fotografía (descripción, posición y miniatura) y lo añadimos al ClusterManager.

```
clusterManager.addItem
(new MapItem(new LatLng(photo.getLatitud(),photo.getLongitude()),
photo.getDescription(), bitmap));
```

La API de Google Maps nos proporciona esta clase ClusterManager, que se encarga de esta información.

5.7.2. Visualización de las fotografías

Para visualizar las fotografías, definimos una nueva clase, *CustomClusterRenderer*, que nos permite gestionar la visualización de los ítems del mapa. Esta clase hereda de DefaultClusterRenderer, que nos ofrece la API de Google Maps.

En nuestro fragment del mapa (*MapsFragment*), asignamos al ClusterManager un nuevo objeto de tipo *CustomClusterRenderer* mediante el método **setRenderer()**.

Antes de representar un ítem, se invoca el método **onBeforeClusterItemRenderer()** de nuestra clase *CustomClusterRenderer*. Este método tiene como parámetros el ítem que tenemos que representar y un objeto MarkerOptions, que se encarga de gestionar las diferentes opciones de cada ítem o marcador posicionado en el mapa. Como nuestro ítem tiene la información necesaria para su representación, asignamos la miniatura a un ImageView, previamente creado y modificado para la visualización de la miniatura de la fotografía y un marco que hemos creado, además de asignar como título la descripción del ítem al objeto MarkerOptions.

A continuación, mostramos un ejemplo de la visualización de las fotografías:

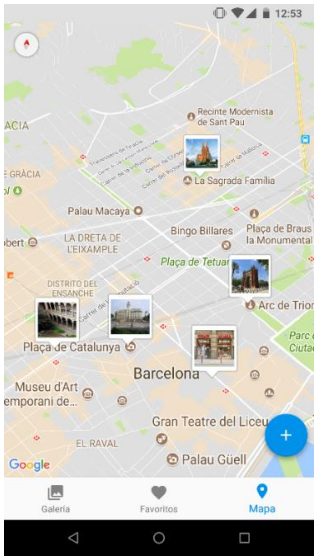


Figura 25: Visualización de las fotografías en el mapa

Las miniaturas de las fotografías se ajustan para obtener un resultado uniforme, en el cual todas las fotografías tienen el mismo tamaño.

5.7.3. Clúster

Cuando tenemos más de una fotografía en una zona, para evitar el solapamiento de fotografías hemos implementado los clústeres. Estos clústeres permiten agrupar las fotografías y evitar que encontrar una gran cantidad de fotografías en una zona determinada. Los clústeres se modifican según la aproximación de la cámara del mapa. A continuación, mostramos un ejemplo:

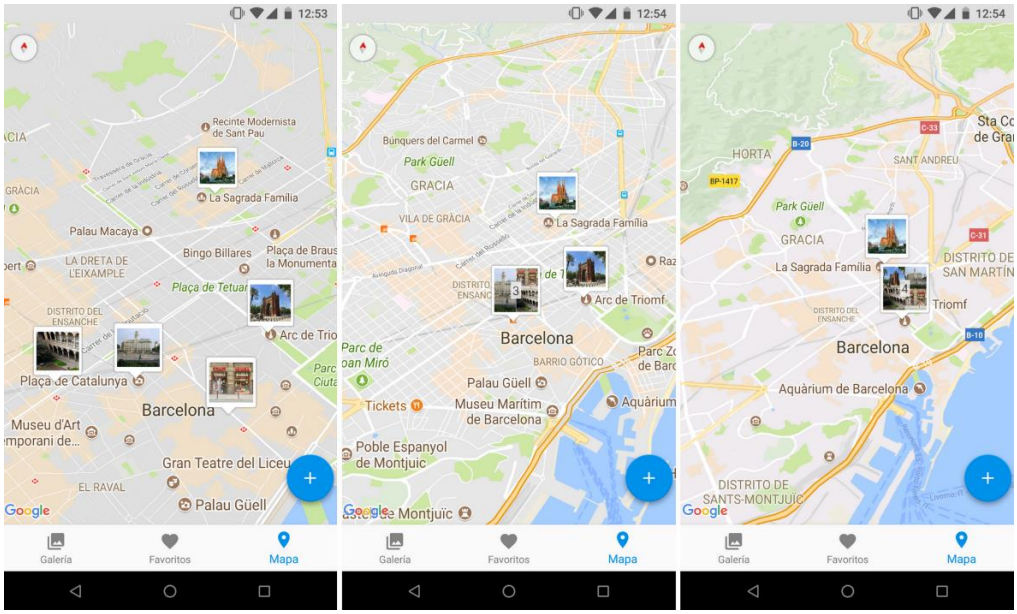


Figura 26: Visualización de los clústeres en el mapa

En la figura anterior vemos como se agrupan según su proximidad. Al agruparse, vemos como en el centro del clúster visualizamos el número de fotografías que están agrupadas. También se visualizan hasta 4 fotografías en la miniatura del clúster.

Para lograrlo, seguimos un procedimiento parecido al de la visualización de las fotografías. En nuestra clase *CustomClusterRenderer*, tenemos un método llamado **onBeforeClusterRenderer()**, que se invoca antes de la representación de un clúster. Como hemos indicado, los clústeres se ejecutan según la proximidad. Este método se ejecuta cuando tenemos fotografías próximas entre ellas, donde el método **souldRenderAsCluster()** nos informa si tenemos más de una fotografía sobre una misma zona según la visualización en ese momento. El método recibe como parámetros el clúster, que contiene los ítems que lo componen, y el objeto MarkerOptions. A partir del clúster, obtenemos hasta un máximo de 4 ítems y los guardamos en una lista para poder crear la miniatura que se muestra en el clúster.

Como en un clúster se muestra en el ítem de dos a cuatro fotografías, la visualización del clúster está dividida en 4 partes y en cada una se muestra una fotografía, en el caso de tener dos o tres fotografías, para evitar tener alguna parte en blanco, se aprovecha la parte vacía para ampliar una fotografía. Para realizar la visualización comentada, adaptamos una clase llamada *MultiDrawable* ofrecida por Google. Antes, debemos modificar el tamaño las fotografías que se mostrarán en el clúster y guardarlas en una lista, que será la lista de las fotografías que se muestran.

Mostramos también la cantidad de fotografías que tiene el clúster, que es la longitud de la lista que tenemos como parámetro en el método **onBeforeClusterRenderer()**.

5.8. Otras gestiones

Además de los puntos mencionados anteriormente, tenemos otras características en cuenta en el desarrollo de LocPic. A continuación, explicamos como se gestiona el idioma de la aplicación y los permisos que necesitamos para su funcionamiento.

5.8.1. Gestión del idioma

LocPic permite modificar el idioma de la aplicación. Tenemos disponibles tres idiomas: español, catalán e inglés. La clase que se encarga de modificar y guardar el idioma es *UserLanguage*.

Para ello, creamos tres archivos XML (uno para cada idioma) que contienen los Strings de que se muestran en la aplicación. Cada archivo contiene las traducciones de los Strings en su respectivo idioma. A continuación, mostramos como definir un String:

```
<string name="loc_enabled">Ubicación activada</string>
```

El nombre del String es “loc_enabled”, y el contenido (el texto que se muestra) es “Ubicación activada”.

Al iniciar por primera vez la aplicación, obtenemos el idioma del dispositivo del usuario. Si el idioma del dispositivo es catalán o español, asignamos el archivo XML con las correspondientes traducciones. En caso de ser inglés o cualquier otro idioma, asignamos el archivo XML con las traducciones en inglés.

El usuario tiene la posibilidad de modificar manualmente el idioma. Damos la posibilidad de cambiar el idioma antes de registrarse o iniciar sesión y una vez dentro de la aplicación. Como necesitamos guardar está configuración, usamos SharedPreferences. SharedPreferences nos proporciona métodos simples para leer o escribir datos. Un objeto SharedPreferences tiene asociado un archivo que contiene pares clave-valor. Por lo tanto, usaremos la clave “Language” para obtener o guardar su valor, que nos indicará el idioma del usuario.

5.8.2. Gestión de los permisos

En el archivo *Manifest* de nuestra aplicación debemos indicar los permisos necesarios para el uso de la aplicación y los servicios del dispositivo. Entre estos permisos encontramos el uso de red, el acceso a la ubicación, cámara o escribir en el dispositivo.

Según el grado de confidencialidad del permiso, el sistema otorga automáticamente el permiso o es el usuario el que lo debe conceder. Por ejemplo, si la aplicación solicita el permiso de la linterna, este se concede automáticamente. En cambio, si necesitamos utilizar la ubicación, es el usuario el que debe conceder el permiso. Dependiendo de la versión del dispositivo, el usuario concede los permisos al descargar la aplicación (versiones Android 5.1 o anterior) o mientras la aplicación se ejecuta (versiones Android 6.0 o posterior).

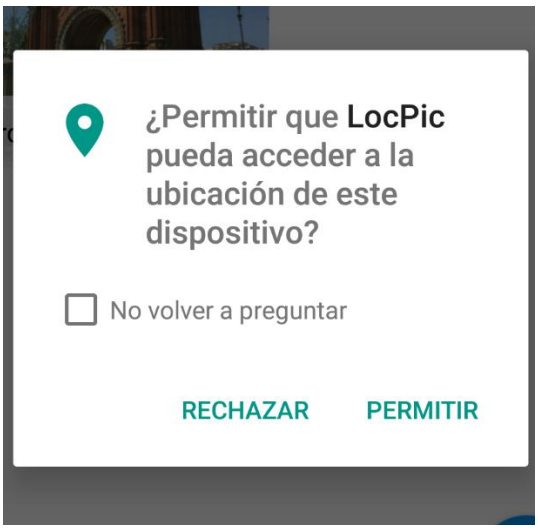


Figura 27: Ejemplo de solicitud de permisos al usuario

Nosotros necesitamos solicitar dos permisos al usuario: permiso de ubicación y permiso de almacenamiento. Ambos permisos se solicitan al iniciar la aplicación. Si el usuario decide no conceder los permisos, no tendrá disponibles las opciones en las que estos se necesitan. Si no los acepta, informamos al usuario en cada acción que necesitemos estos permisos para realizar la tarea. El usuario siempre puede gestionar los permisos en los ajustes del dispositivo. El permiso de almacenamiento es necesario a la hora de guardar una fotografía descargada en el dispositivo.

5.9. Base de datos

Como hemos indicado en apartados anteriores, hemos utilizado Firebase para la implementación de la base de datos. Utilizamos la base de datos para guardar la información de las fotografías de cada usuario y sus carpetas.

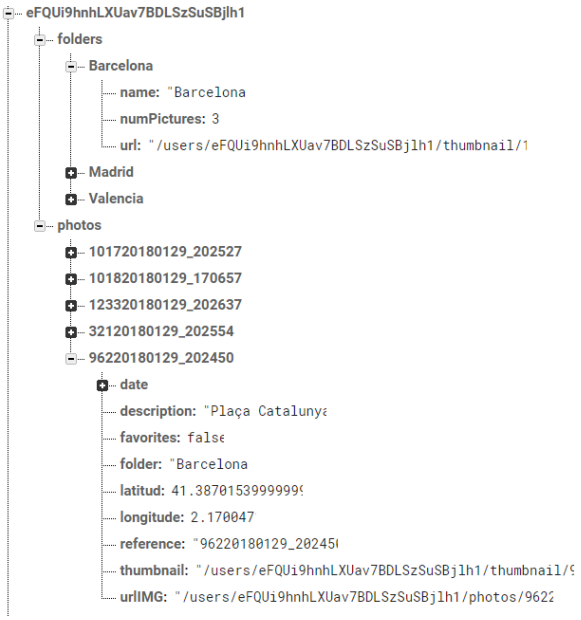


Figura 28: Contenido y estructura de la base de datos para un usuario

La base de datos contiene un nodo raíz, que corresponde al proyecto de Firebase, un nodo *app* y finalmente un nodo *users.* Dentro del nodo *users*, tenemos todos los usuarios. La figura anterior muestra el contenido que tendría cada uno de los usuarios. Vemos que principalmente tiene dos nodos inferiores que corresponden a las carpetas y a las fotografías.

El nodo principal de cada usuario es su identificador único de usuario. Esto nos asegura, una vez el usuario ha iniciado sesión, que accede únicamente a su contenido.

En el nodo *folders* (carpetas) se alojan todas las carpetas que tiene el usuario. Cada carpeta contiene:

- **name**. Nombre de la carpeta.
- **numPictures**. Número de fotografías que hay en la carpeta.
- **url**. Referencia al almacenamiento de la portada de la carpeta.

En el nodo *photos* (fotografías) se alojan todas las fotografías del usuario. Cada fotografía contiene:

- **date**. Objeto Date con la fecha de subida de la fotografía.
- **description**. Descripción de la fotografía.
- **favorites**. Booleano que nos indica si la fotografía es favorita o no lo es.
- **folder**. Carpeta a la que pertenece la fotografía.
- **latitude**. Latitud de la ubicación de la fotografía.
- **longitude**. Longitud de la ubicación de la fotografía.
- **reference**. Referencia de base de datos de la fotografía.
- **thumbnail**. Referencia al almacenamiento de la miniatura de la fotografía.
- **urlIMG**. Referencia al almacenamiento de la fotografía.

Hemos definido una clase para cada componente que puede tener un usuario, *Folder* y *Photo.* Es necesario que estas clases contengan los mismos atributos que cada elemento de *folders* y *photos*, además de un constructor vacío y los getters, si no, no será posible obtener los objetos.

Finalmente, hemos configurado las reglas de acceso a la base de datos para que solo usuarios registrados en la aplicación puedan acceder a ella.

Para realizar una consulta, necesitamos una instancia de DatabaseReference que creamos de la siguiente forma:

```
DatabaseReference dataRef = FirebaesDatabase.getInstance()
    .getReference(FirebaseReferences.APP_REFERENCE);
```

FirebaseReferences.APP_REFERENCE, es la referencia al nodo *app* de nuestra base de datos.

Una vez tenemos la referencia, accedemos al nodo que deseamos para obtener información utilizando el método **child()**.

A continuación, mostramos un ejemplo de cómo realizar una petición a base de datos para obtener las carpetas de un usuario:

```
dataRef.child(FirebaseReferences.USERS_REFERENCE)
        .child(currentUser.getUid())
.child(FirebaseReferences.FOLDERS_REFERENCE).addValueEventListener(…)
```

FirebaseReferences.USERS_REFERENCE, es la referencia al nodo *users*. El siguiente nodo es el identificador único del usuario, el cual obtenemos a partir de la instancia de FirebaseUser utilizando el método **getCurrentUser()** y posteriormente obtenemos su identificador único con el método **getUid()**. Finalmente, la última referencia, en este caso, es la que pertenece al nodo *folders*.

Una vez nos encontramos en el nodo correspondiente, utilizamos un Listener para atender a la petición. Disponemos de:

- **addValueEventListener(ValueEventListener)**. Este Listener se llamará cada vez que los datos cambian y recibimos el evento en **onDataChange()**.
- **addChildEventListener(ChildEvenetListener)**. Este Listener actúa cuando se agregan, eliminan, cambian o mueven los datos. A diferencia del caso anterior, se lanza un método diferente según el cambio.
- **addListenerForSingleValueEvent(ValueEventListener)**. Este Listener se active una vez con el valor de los datos de la ubicación.

Como en este caso queremos obtener todas las carpetas, podemos iterar sobre la petición de la siguiente forma:

```
for( DataSnapshot child : dataSnapshot.getChildren() ){}
```

Dentro del bucle, podemos hacer las gestiones pertinentes para cada carpeta. Cada objeto carpeta será un DataSnapshot, que contiene los datos de una ubicación de la base de datos de Firebase, sobre los cuales iteramos para obtener, en este caso, cada una de las carpetas del usuario.

5.10. Almacenamiento de las fotografías

La implementación del almacenamiento es similar a la base de datos. Hemos creado una carpeta *users* que contiene en su interior otras carpetas con los usuarios. Como en la base de datos, distinguimos los diferentes usuarios con su identificador único.

En el interior de las carpetas de cada usuario encontramos otras dos, una carpeta que hemos llamado *photos* que contiene las fotografías del usuario y una carpeta que hemos llamado *thumbnail* que contiene las miniaturas de las fotografías del usuario.

Para acceder al almacenamiento tenemos que crear una instancia de FirebaseStorage. Como Glide está integrado con Firebase, podemos cargar las fotografías directamente con ello. Un ejemplo de carga de una fotografía es el siguiente:

```
Glide.with(activity).using(new FirebaseImageLoader())
.load(storageReference.child(dir.getThumbnail())).asBitmap()
.thumbnail(0.1f).fitCenter().into(imageView);
```

El código anterior corresponde a la carga de una fotografía del usuario.

Si por el contrarío queremos obtener el enlace de descarga de una fotografía, podemos acceder a los diferentes nodos y elementos de la misma manera que en la base de datos, utilizando **child()** y los métodos que nos proporciona la librería de FirebaseStorage.

Finalmente, como en la base de datos, configuramos las reglas de acceso al almacenamiento para que ningún solo usuarios registrados en la aplicación puedan acceder a ella.

6. Diseño y evaluación de la interfaz de usuario

En esta sección, mostramos el diseño de la interfaz de usuario de la app, que ha seguido las directrices de diseño de Android. La evaluación de la interfaz/app corresponde a una prueba con usuarios.

6.1. Interfaz de usuario

En el año 2014, Google anunciaba su normativa de diseño enfocada en la visualización del sistema operativo Android, llamada Material Design [16]. Se integró en Android Lollipop (Android 5.0) como reemplazo de Holo, que es la anterior normativa de diseño que utilizaba Android. Es un diseño donde la profundidad, las superficies, los bordes, las sombras y los colores juegan un papel principal. El IDE Android Studio que utilizamos proporciona las herramientas necesarias para crear este estilo de diseño.

También es importante seguir un estándar de colores en la aplicación y un estándar en el diseño de los componentes como podrían ser los botones.

LocPic sigue esta normativa de diseño implantada por Google. A continuación, veremos la distribución del contenido e interfaz de usuario de la aplicación.

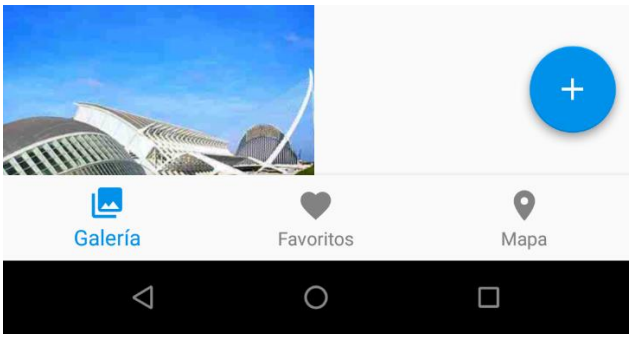


Figura 29: Control del menú y de la subida de fotografías.

En la figura anterior, vemos la parte inferior de la aplicación una vez identificados. El usuario dispone de la mayoría de los controles en esta parte.

Para navegar entre las diferentes opciones hemos implementado un BottomNavigationView. Para respetar la normativa de Material Design, este componente ha de tener un mínimo de tres opciones, si no, se considera innecesario.

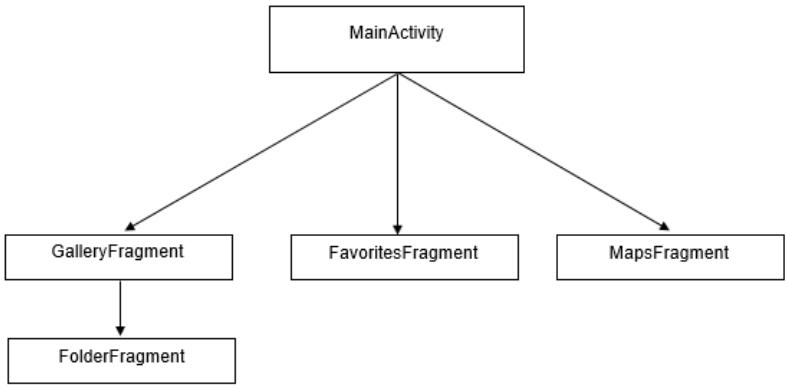


Figura 30: Componentes del menú principal

En este componente encontramos tres opciones: galería, favoritos y mapa. Cada una de estas opciones es un fragment que nos permite mostrar el contenido de cada una de las opciones. La opción galería se componente de otro fragment para mostrar el contenido de la carpeta. Para informar al usuario de que opción tiene seleccionada, mostramos esta de color azul.

Para la opción de subir una fotografía hemos implementado un Floating Action Button. Estos botones se crearon para destacar una acción de la aplicación. Se caracteriza por tener una forma circular, con un icono interno que informa al usuario de la acción que puede realizar.

Al presionar el botón, como el usuario tiene la posibilidad de subir una fotografía utilizando la cámara o seleccionándola de la galería, mostramos las dos opciones disponibles. Esta forma de mostrar las opciones que tiene para subir el contenido es muy utilizada actualmente en cualquier aplicación Android.

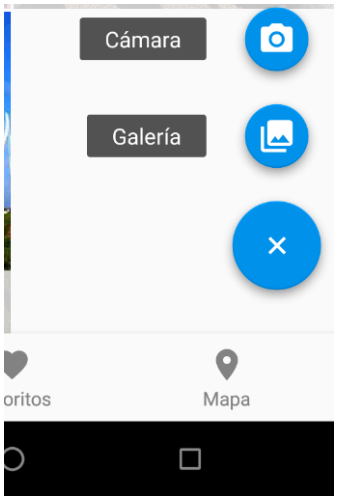


Figura 31: Diseño de los botones para subir las fotografías

Los botones con las opciones que tiene el usuario se muestran con un tamaño más pequeño, tal y como indica Material Design. Al seleccionar alguna de las opciones, se procede a realizar la tarea que correspondiente.

El usuario, en diferentes momentos, debe seleccionar entre diferentes opciones que tiene disponible para realizar una tarea. Para ello, utilizamos AlertDialogs o cuadros de diálogo. Los cuadros de diálogo son ventanas que indican al usuario que deben tomar una decisión o introducir algún dato para completar una tarea. Estos cuadros de diálogo, los encontramos en diferentes momentos de la aplicación, como cuando el usuario debe introducir el correo electrónico para recuperar la contraseña, para seleccionar una carpeta, para seleccionar que acción quiere el usuario realizar sobre una fotografía o para solicitar los permisos.

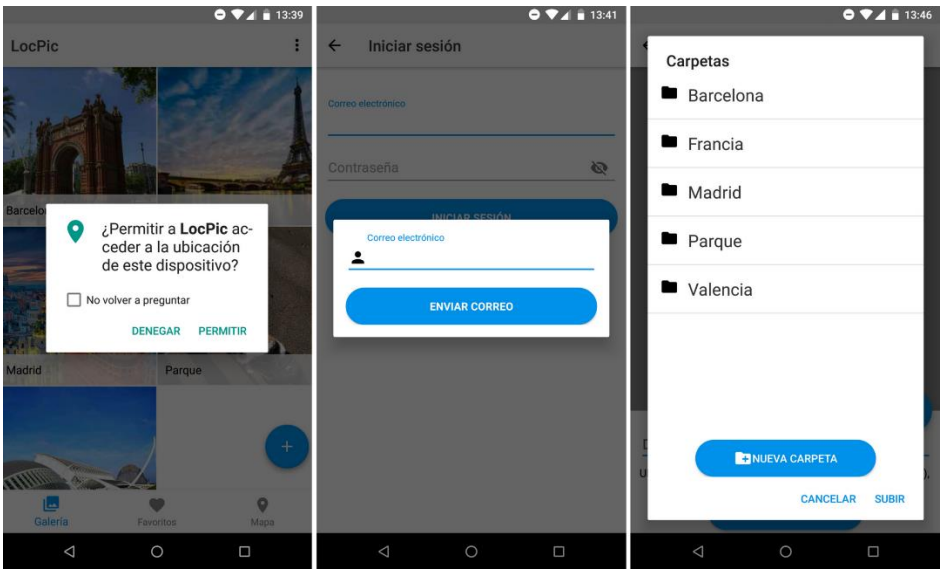


Figura 32: Ejemplos de diálogos en la aplicación

En la figura anterior, apreciamos también como el fondo del diálogo toma un tono más oscuro, lo que permite visualizar mejor el diálogo que se muestra al usuario.

Como hemos indicado en apartados anteriores, la galería, las fotografías de una carpeta y la sección de favoritos se muestra mediante un GridView de forma cuadricular.

Finalmente, cuando un usuario se registra, inicia sesión o sube una fotografía, se muestra el estado de la tarea con un ProgressBar, que indica al usuario que se está realizando la tarea.

6.2. Evaluación de la interfaz de usuario

Para validar el diseño y la distribución del contenido de la aplicación, realizamos un test de usuarios con el objetivo de conseguir recopilar información y mejoras sobre el diseño y validar la distribución de los contenidos.

El test tuvo una duración aproximada de unos 10-15 minutos. Los usuarios deben realizar una serie de tareas para validar el contenido de la aplicación. El test realizado no fue una prueba con la versión de la aplicación definitiva, aunque ya presentábamos la mayoría de las funcionalidades y distribución del contenido. El objetivo principal del test, como hemos indicado, era conseguir recopilar la información necesaria para acabar de definir el diseño y/o añadir información o modificaciones al diseño presentado.

El test lo hemos realizado con 3 usuarios. Antes de comenzar el test, hicimos una breve introducción de que consiste LocPic y expusimos al usuario las diversas tareas que debía realizar.

A continuación, se presentan las tareas que debe realizar el usuario:

- Abrir la aplicación y modificar el idioma.
- Registrarse.
- Subir una fotografía (Opción cámara o galería).
- Utilizar la galería, navegar y seleccionar una fotografía para visualizarla.
- Visualizar el mapa.
- Cerrar sesión.

Al finalizar las tareas, realizamos una serie de preguntas basadas en las tareas que el usuario realizó para validar el diseño actual y las mejoras que podríamos aplicar (Ver *9.1 Test de usuarios*).

Tras la realización de todas las pruebas con los usuarios, analizamos sus respuestas para extraer la información necesaria y aplicarla al desarrollo. Basándonos en la planificación y el tiempo disponible, se añadieron diversas mejoras basadas en sus respuestas. Las mejoras o modificaciones que incluimos son:

- Posibilidad de visualizar la contraseña introducida.
- El usuario visualizará la ubicación, descripción y la fecha de subida de la fotografía seleccionada.
- Zoom de la cámara al acceder al mapa.
- Visualización de la descripción al seleccionar una fotografía en el mapa.
- Agrupación de fotografías en el mapa.
- Confirmación de contraseña.

7. Evaluación funcional de la aplicación

Para comprobar el funcionamiento de las diferentes opciones que disponemos en LocPic, realizamos pruebas de funcionamiento de los diferentes casos de uso que tiene un usuario. Estas pruebas las realizamos en dos dispositivos con características diferentes. Además, las pruebas las realizamos con conexión Wi-Fi y con red móvil.

A continuación, mostramos la versión, resolución, tamaño de pantalla, memoria RAM y resolución de la cámara de los dispositivos.

	Nexus 5	**Nexus 5x**
Versión del dispositivo	6.0.1	8.1.0
Resolución de pantalla	1920x1080	1920x1080
Tamaño de la pantalla	4,95 pulgadas	5,2 pulgadas
Memoria RAM	2 GB	2GB
Cámara	8 megapíxeles	12.3 megapíxeles

Tabla 4: Características de los dispositivos de pruebas

Utilizar diferentes versiones en las pruebas nos permite validar el funcionamiento en versiones antiguas y recientes.

En la siguiente tabla mostramos los diferentes casos de uso de un usuario no identificado y el resultado de las pruebas.

Caso de uso	**Resultado**
Registrarse con correo electrónico	Correcto
Iniciar sesión con correo electrónico	Correcto
Iniciar sesión con una cuenta Google	Correcto
Cambiar idioma	Correcto
Recuperar contraseña	Correcto

Tabla 5: Resultados de las pruebas de los casos de uso de un usuario no identificado

Las pruebas que hemos realizado en los casos de uso de un usuario no identificado incluyen probar la robusteza de la aplicación y la gestión de introducción de texto de las opciones.

En la siguiente tabla mostramos los diferentes casos de uso de un usuario identificado y el resultado de las pruebas.

Caso de uso	**Resultado**
Cerrar sesión	Correcto
Modificar contraseña	Correcto
Cambiar Idioma	Correcto

Subir fotografía (cámara)	Correcto
Subir fotografía (galería)	Correcto
Visualizar mapa	Correcto
Visualizar carpeta	Correcto
Visualizar fotografía	Correcto
Añadir a favoritos	Correcto
Descargar fotografía	Correcto
Eliminar fotografía	Correcto
Mover fotografía de carpeta	Correcto

Tabla 6: Resultados de las pruebas de los casos de uso de un usuario identificado

Las pruebas que hemos realizado en los casos de uso de un usuario identificado incluyen probar la robusteza de la aplicación y la gestión de introducción de texto de las opciones.

Las pruebas Wi-Fi las realizamos con una conexión de 100 Mb de subida y 30 Mb de descarga, y las pruebas con conexión red móvil con 4G.

Debemos tener en cuenta que, dependiendo de la calidad de la cámara, las fotografías tienen mayor o menor peso, y se puede apreciar que una fotografía realizada con el dispositivo Nexus 5x tarda ligeramente un poco más que el dispositivo Nexus 5.

Esta evaluación nos ha servido para validar el funcionamiento de las opciones y tareas que puede realizar un usuario y comprobar el funcionamiento general de la aplicación.

8. Conclusión

En este Trabajo de Fin de Grado hemos diseñado, implementado y evaluado LocPic, una app de galería de fotografías geolocalizadas para Android.

Para desarrollar LocPic, hemos adaptado el modelo de desarrollo de software iterativo y creciente según las características de este TFG. Cada 2-3 semanas realizamos reuniones que permitían validar lo realizado en la iteración y marcar los siguientes pasos basándonos en los requisitos iniciales, además de añadir otras funcionalidades, como podría ser el idioma, la gestión de las fotografías o la creación de las carpetas, que inicialmente no las contemplamos.

LocPic proporciona un mecanismo de autentificación, para que el usuario pueda registrarse e iniciar sesión, además de tener un método alternativo de inicio de sesión con su cuenta de Google.

LocPic permite al usuario dos formas de poder subir una fotografía, utilizando la cámara o seleccionando una fotografía de la galería. Almacenamos todas las fotografías y su información en la nube, lo que permite al usuario tener su contenido en sus dispositivos utilizando su cuenta de usuario.

LocPic permite visualizar todas las fotografías del usuario distribuidas por el mapa de una manera sencilla y visual. Las fotografías también se agrupan en clústeres para facilitar la visualización del contenido.

Hemos realizado un test con usuarios para validar el diseño y las funcionalidades implementadas, y añadir algunas de las sugerencias de mejora para hacer la aplicación más completa y usable. También hemos realizado una evaluación más técnica, centrándonos en las funcionalidades principales de LocPic.

Los resultados de las evaluaciones han sido positivos, ya que el test con usuarios nos ha permitido añadir algunas características que no teníamos presentes en el diseño y desarrollo de la aplicación, y la evaluación funcional nos ha permitido confirmar el correcto funcionamiento de todas las características implementadas.

Como conclusión global del proyecto, hemos alcanzado los objetivos propuestos y expuesto en la sección *4.1. Requerimientos* y consolidado los conocimientos de diferentes asignaturas que encontramos en el Grado. Por lo tanto, podemos decir que el resultado es satisfactorio.

La app tiene posibles mejoras o perspectivas de futuro. En la siguiente sección, mostramos algunas de estas mejoras y perspectivas de futuro.

8.1. Trabajo futuro

Partiendo del estado actual en que se encuentra el proyecto, encontramos diferentes direcciones para desarrollo futuro o ampliaciones.

A continuación, se presentan algunas de las ampliaciones y mejoras que se pueden realizar en el proyecto.

- **Portabilidad.** Al tratarse de una aplicación para Android, un posible trabajo futuro sería realizar la aplicación para dispositivos iOS o para plataformas Web, para abarcar más dispositivos.

- **Temas**. Esta mejora surgió en el test con usuarios de la aplicación. Un usuario propuso la posibilidad de poder editar el diseño (colores) de la aplicación.

- **Contenido social**. Otra posibilidad sería la de cambiar la dirección del proyecto a una aplicación social, que en la actualidad tiene mucha importancia. El usuario podría tener la posibilidad de visualizar los mapas de sus amigos, o de compartir un mapa entre diferentes usuarios para visualizar fotos que han realizado en común.

- **Mejoras del test de usuarios.** Otra posibilidad sería añadir las mejoras propuestas por los usuarios en el test y que no se incluyeron en el desarrollo por la planificación, como por ejemplo, que cada usuario disponga de un perfil, la posibilidad de tener un ID de usuario (apodo) personalizado, etc.

- **Visualización:** El proyecto no tenía como objetivo investigar en los diferentes tipos de visualización de mapas (vista satélite, mapa…) y la geolocalización de las fotografías, pero una línea interesante de trabajo futuro sería investigar qué visualizaciones son más útiles, accesibles… para los usuarios.

9. Referencias

[1] AlertDialogs. http://www.hermosaprogramacion.com/2015/06/como-crear-dialogos-en-android/ (2/12/2017)

[2] Android developers. https://developer.android.com (4/01/2018)

[3] Apuntes Disseny de Software, Ingeniería informática UB. http://www.ub.edu/grad/plae/AccesInformePD?curs=2017&codiGiga=364303&idioma=CAT&recurs=publicacio (25/10/2017)

[4] Apuntes Factors Humans i Computació, Ingeniería informática UB. http://www.ub.edu/grad/plae/AccesInformePD?curs=2017&codiGiga=364323&idioma=CAT&recurs=publicacio (29/11/2017)

[5] Apuntes Projecte Integrat de Software, Ingeniería informática UB. http://www.ub.edu/grad/plae/AccesInformePD?curs=2017&codiGiga=364304&idioma=CAT&recurs=publicacio (10/10/2017)

[6] Clústeres, Anton Shkurenko. https://medium.com/@tonyshkurenko/work-with-clustermanager-bdf3d70fb0fd (17/12/2017)

[7] Firebase Android. https://www.firebase.com/docs/android/ (9/01/2018)

[8] Firebase Authentication. https://firebase.google.com/docs/auth (5/12/2017)

[9] Firebase Realtime Database. https://firebase.google.com/docs/database/ (12/01/2018)

[10] Firebase Storage. https://firebase.google.com/docs/storage/ (3/11/2017)

[11] Floating action button. https://github.com/futuresimple/android-floating-action-button (27/11/2017)

[12] Glide. https://github.com/bumptech/glide (7/11/2017)

[13] Google developers. https://developers.google.com (15/12/2017)

[14] Google Maps. https://developers.google.com/maps/ (9/01/2018)

[15] Google Places. https://developers.google.com/places/ (9/01/2018)

[16] Material Design. https://material.io/guidelines/ (7/01/2017)

[17] Picasso. http://square.github.io/picasso/ (7/11/2017)

[18] Stackoverflow. https://stackoverflow.com (20/12/2017)

10. Anexo

En esta sección, se presentan las preguntas y respuestas que realizaron los usuarios en la prueba de la aplicación con usuarios y el manual de usuario de la aplicación.

10.1. Test de usuarios

Tras realizar las tareas presentadas, el usuario debe responder a una serie de preguntas que nos permiten recopilar la información necesaria.

Preguntas:

1. Abrir aplicación.
   - 1.1. ¿Crees que falta algún dato o funcionalidad para entrar a la aplicación?
   - 1.2. ¿Ves correctamente la selección del idioma de la aplicación?
   - 1.3. ¿Puedes visualizar fácilmente las opciones que tienes inicialmente?
2. Registro.
   - 2.1. ¿Crees que es fácil encontrar esta opción?
   - 2.2. ¿Visualizas la información necesaria y clara para realizar la tarea?
   - 2.3. ¿Añadirías o modificarías algún tipo de información para realizar la tarea de una forma más intuitiva?
3. Subir una fotografía.
   - 3.1. ¿Visualizas claramente las opciones que tienes para subir una fotografía a tu cuenta?
   - 3.2. ¿Cómo crees que se tendría que mostrar esta información por pantalla?
   - 3.3. Una vez seleccionada o realizada la fotografía, ¿Crees que se muestra la información necesaria por pantalla antes de subir esta al servidor? ¿Añadirías, eliminarías o modificarías algún dato que se muestra?
4. Galería y visualizar una fotografía.
   - 4.1. ¿Te gusta el tamaño de las carpetas de la galería? En caso de que no, indica cómo te gustaría que se visualizasen.
   - 4.2. ¿Visualizas claramente la fotografía? ¿Con que tamaño te gustaría que se mostrasen?
   - 4.3. ¿Qué información ves necesaria que se muestre en la pantalla de visualizar la fotografía? (Opciones: Lugar donde se realizó la fotografía, fecha, carpeta, descripción)
   - 4.4. ¿Crees que las opciones que dispones son las necesarias? En el caso de negativo, indica cual encuentras a faltar. (Opciones: Descargar, eliminar).
5. Visualizar el mapa.
   - 5.1. ¿Se muestra la fotografía que has subido anteriormente?
   - 5.2. ¿Como te gustaría que se mostrará la fotografía en el mapa?
   - 5.3. ¿Visualizas la información que se muestra al pulsar el marcador? ¿Qué información mostrarías?
   - 5.4. ¿Te gustaría que las fotografías se agruparan para visualizar mejor el contenido?
6. Cerrar sesión.
   - 6.1. ¿Encuentras la opción fácilmente? En caso de negativo, indica cómo te gustaría acceder a esta opción.
7. Otras.
   - 7.1. ¿Qué información/contenido te gustaría que se mostrara por pantalla al abrir la aplicación?

A continuación, mostramos las respuestas de los usuarios a las preguntas del test.

Preguntas	Respuestas
1..1.	No, todo bien
1.2.	Sí, pero siempre se pueden poner más idiomas
1.3.	Sí
2.1	Sí
2.2.	Se ve claro, pero le falta la opción de ver la contraseña por si te equivocas
2.3.	Misma respuesta que 2.2
3.1.	Sí
3.2.	Está bien así
3.3.	No
4.1.	Sí, pero estaría bien que se vieran 4 en miniatura en la portada (Me enseñó a que se refería. El reproductor de iOS que la música salen las portadas de los discos en una cuadricula en la portada de la carpeta)
4.2.	Grande está bien
4.3.	Ubicación, descripción y fecha
4.4.	Sí
5.1.	Sí
5.2.	Estaría bien que se vea grande cuando pulsas
5.3.	Únicamente la foto ni información ni nada
5.4.	Sí
6.1.	Sí
7.1.	Que se mostrara como un perfil de usuario
Apuntes extras del usuario	Cuando se abre el mapa que haga zoom (Se informó al usuario que está característica será implementada) Que se puedan cambiar los colores (tema)

Tabla 7: Respuestas del usuario 1

Preguntas	Respuestas
1..1.	No
1.2.	Sí
1.3.	Sí
2.1	Sí
2.2.	Sí
2.3.	Me dijo que se pudiera poner un apodo, para no tener que usar el correo electrónico para iniciar sesión
3.1.	Sí
3.2.	Tal y como está lo veo bien
3.3.	No, está bien
4.1.	Está bien así, como la mayoría
4.2.	Me da igual
4.3.	La descripción que he puesto a la fotografía, lo demás me da igual
4.4.	Modificar texto
5.1.	Sí
5.2.	Está bien así
5.3.	La fotografía, y si clicas que muestre lo que has escrito
5.4.	Sí
6.1.	Sí, pero estaría bien que preguntará si estás seguro
7.1.	Las carpetas de las fotografías
Apuntes extras del usuario	Nada *NOTA (Como estaba la opción de elegir carpeta, ocurrió un error, y es que al poner “Intro”, hizo salto de línea y al subir la fotografía y después cargarla en la app, se quedaba en bucle cargando muchas iguales, es un bug que permitiré nombres de carpeta en una única línea para evitar el salto de línea). Se anotó para poder corregirlo.

Tabla 8: Respuestas del usuario 2

Preguntas	Respuestas
1..1.	No
1.2.	Sí
1.3.	Sí
2.1	Sí
2.2.	Sí
2.3.	Le gustaría la letra más clara y confirmar la contraseña (se refería a los hints de introducción de texto)
3.1.	Sí
3.2.	Así está bien
3.3.	Me gustaría más interacción con la foto (como Instagram)
4.1.	Está bien así
4.2.	Grande
4.3.	Fecha, comentario y lugar
4.4.	Sí
5.1.	Sí
5.2.	Que se visualice la fotografía
5.3.	-
5.4.	Sí
6.1.	Sí
7.1.	Poner una animación de entrada, por si las carpetas tardan en cargar
Apuntes extras del usuario	Ninguna anotación extra del usuario

Tabla 9: Respuestas del usuario 3

Todas las respuestas de los usuarios se anotaron y posteriormente se valoraron para ayudar con la parte final del trabajo. (Ver *6.2.1. Evaluación con usuarios*).

10.2. Manual de usuario

Al abrir por primera vez la aplicación, visualizarás una bienvenida a LocPic. Puedes cambiar de pantalla de presentación u omitir esta presentación con los controles en la parte inferior de la pantalla.

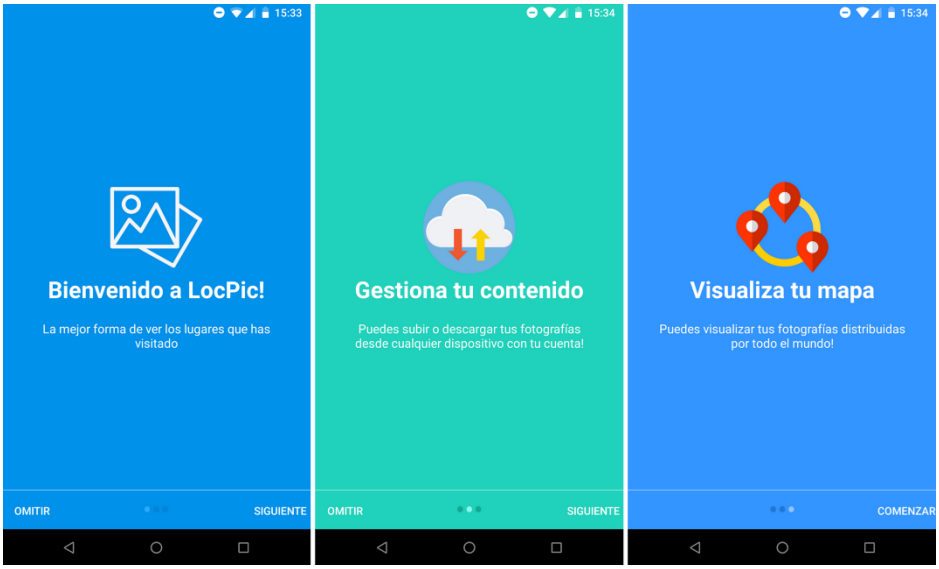


Figura 33: Introducción a la LocPic

Tras la bienvenida, se muestran las opciones que dispones para iniciar sesión o registrarte. Si decides iniciar sesión mediante una cuenta de Google, accederás directamente sin la necesidad de realizar más pasos.

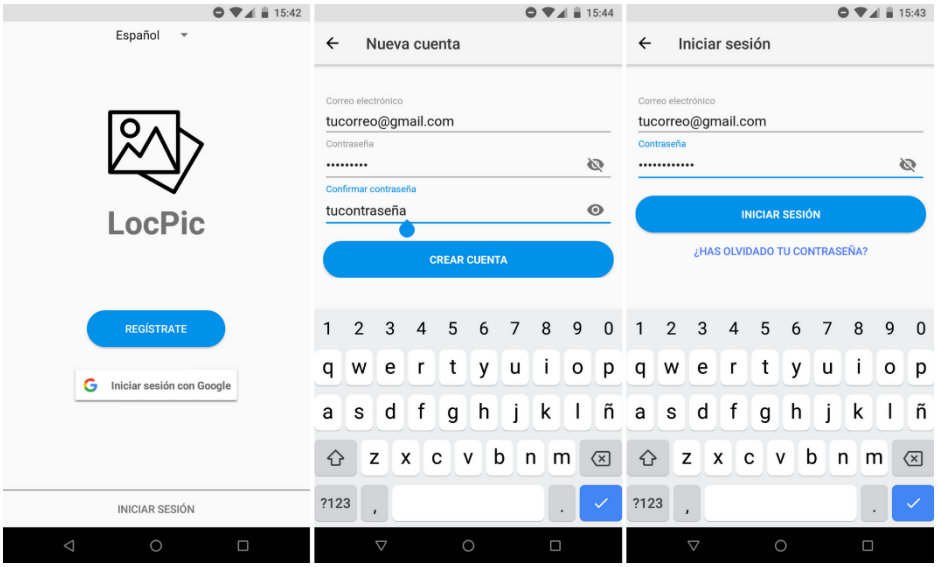


Figura 34: Registro e inicio de sesión en LocPic

Si decides registrarte mediante un correo electrónico y contraseña, debes seleccionar la opción de *Regístrate*. Visualizarás una pantalla que solicita el correo electrónico y la contraseña. Además, necesitas escribir la contraseña una segunda vez para confirmar. Recuerdas que puede visualizar la contraseña que has introducido. Una vez introducido los datos, puedes registrarte. Si el registro es correcto, accederás directamente a la aplicación. En caso contrario, revisa los datos que has introducido y vuelve a intentarlo.

Si ya dispones de una cuenta en LocPic, puedes iniciar sesión accediendo a la opción de la parte inferior de la pantalla. Se solicitan el correo electrónico con el cual te registraste y la contraseña. Si no recuerdas la contraseña, tienes la opción de modificarla. Para ello, selecciona la opción *¿Has olvidado tu contraseña?* e introduce tu correo electrónico. Recibirás un correo electrónico con el enlace que te permite crear la nueva contraseña.

Cuando te registres o accedas a la aplicación, te solicitamos los permisos de ubicación y de almacenamiento si es la primera vez que accedes a la aplicación o si todavía no los has aceptado.

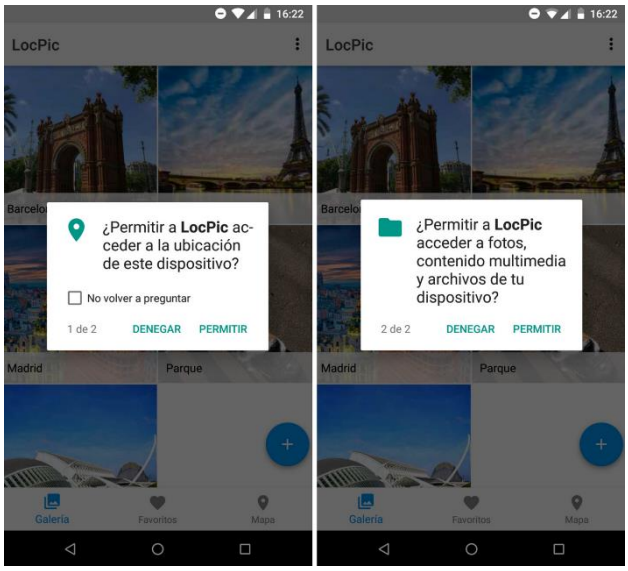


Figura 35: Permisos necesarios que se solicitan al usuario

Necesitamos el permiso de ubicación para obtener tu localización cuando realices una fotografía con la cámara desde LocPic y el permiso de almacenamiento para descargar las fotografías y guardarlas en tu dispositivo. Siempre puedes gestionar los permisos manualmente en el apartado de ajustes de tu dispositivo Android.

Una vez estés en el menú principal, puedes navegar entre las diferentes opciones utilizando la barra de la parte inferior. Tienes disponible tu galería donde se muestran las carpetas que has creado, tus fotografías favoritas y el mapa con tus fotografías.

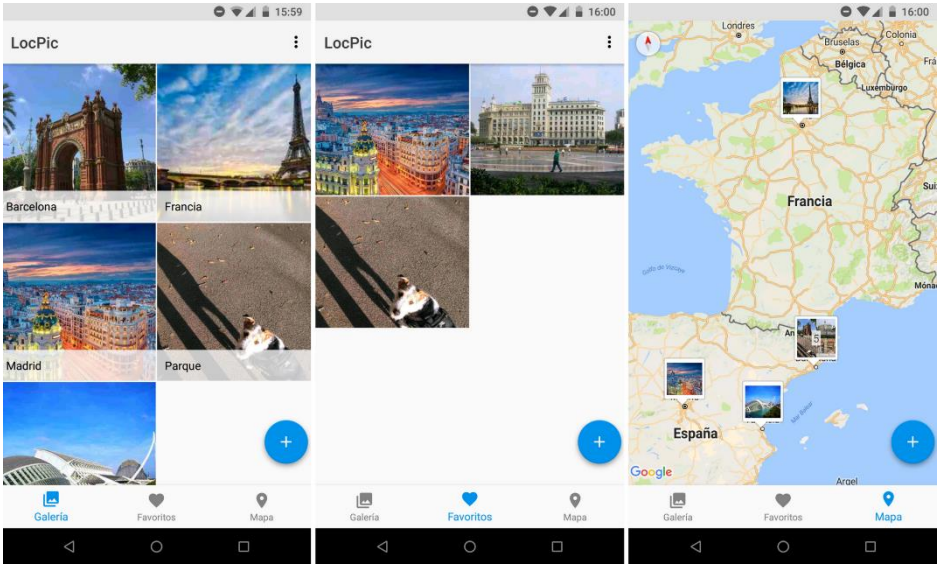


Figura 36: Menú principal de LocPic

Si quieres acceder a una carpeta para ver su contenido, solo debes seleccionarla. Una vez dentro de la carpeta, puedes gestionar tus fotografías. Tienes dos interacciones disponibles, realizar una pulsación larga o seleccionar una fotografía.

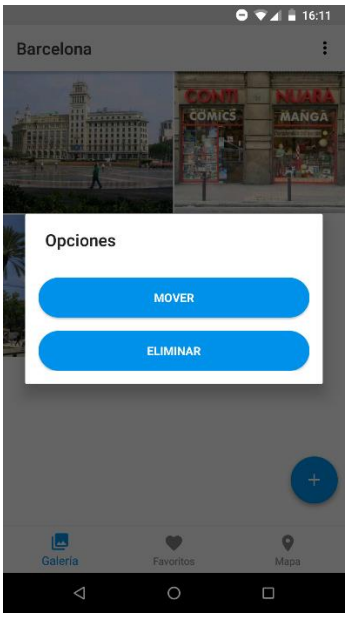


Figura 37: Gestión de una fotografía

Si realizas una pulsación larga, te mostramos las opciones que tienes disponibles para gestión de la fotografía. Podrás mover de carpeta o eliminar la fotografía seleccionada. Si eliges la opción de mover la fotografía de carpeta, te mostramos todas tus carpetas para que selecciones la de destino. También te damos la posibilidad de crear una nueva si lo deseas.

Si seleccionas una fotografía, te mostramos la fotografía con un tamaño mayor y visualizarás la información de la fotografía.

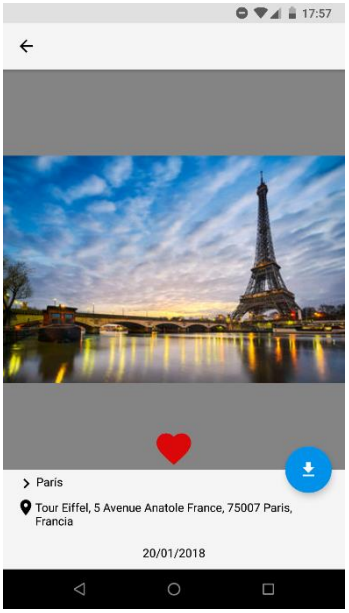


Figura 38: Visualización de una fotografía

Además, puedes descargar la fotografía para guardarla en la galería de tu dispositivo o añadir la fotografía al apartado de favoritos. Para descargar la fotografía utiliza el botón de color azul que encuentras en la pantalla de visualización de la fotografía. Recuerda que debes conceder los

permisos de almacenamiento a la aplicación. Para agregar o eliminar la fotografía de favoritos, debes utilizar el corazón que se muestra en la pantalla.

Para poder subir una fotografía de la galería o realizar una con la cámara, debes utilizar el botón que tienes en la parte inferior derecha en el menú principal.

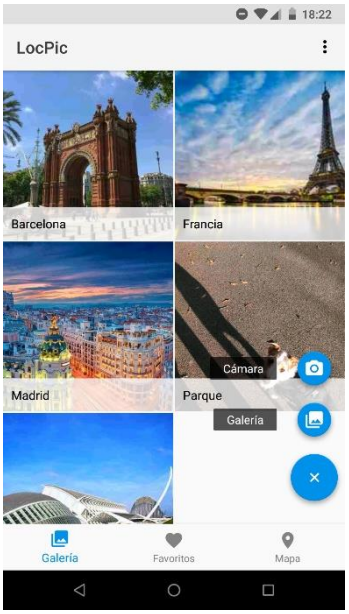


Figura 39: Botón para subir una fotografía

Se mostrarán las dos opciones. En el caso de seleccionar la opción de *galería*, te mostraremos todas las galerías que tienes en el dispositivo. Debes seleccionar una y elegir la fotografía que quieres subir. Si prefieres utilizar la cámara del dispositivo, selecciona la opción de *cámara,* la cual iniciará la cámara para que puedas realizar la fotografía.

Una vez seleccionada o realizada la fotografía, esta se mostrará en una nueva pantalla con las opciones de subida.

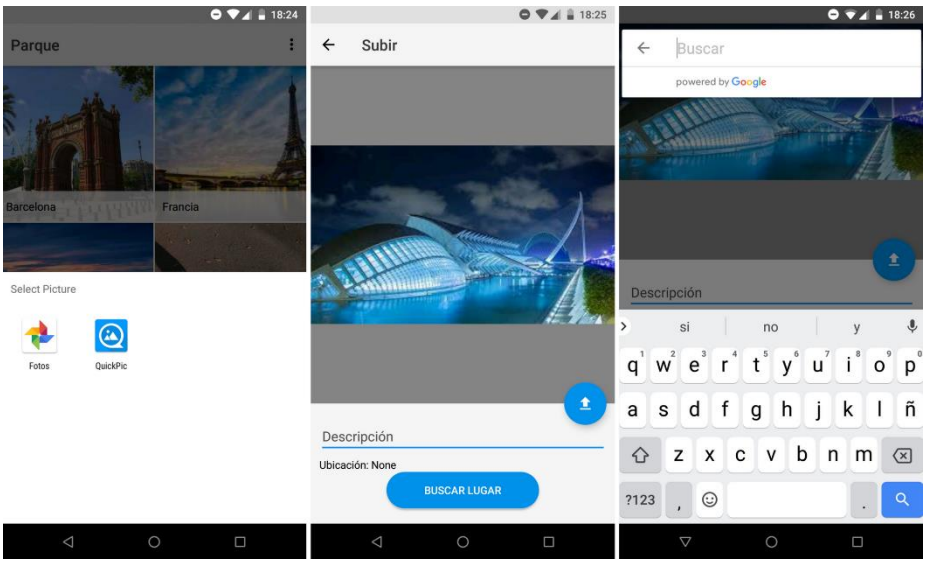


Figura 40: Pasos para subir una fotografía de la galería del dispositivo

En el caso de seleccionar una fotografía del dispositivo, debes introducir manualmente la ubicación donde se realizó la fotografía utilizando el botón *BUSCAR LUGAR*. Además, tienes la posibilidad de introducir una descripción de la fotografía. Esta descripción tiene un límite de longitud.

Si la opción seleccionada para subir la fotografía es la cámara, automáticamente se mostrará la posición donde has realizado la fotografía, por lo tanto, no debes introducirla. Como en el caso anterior, puedes añadir una descripción a la fotografía.

En ambos casos, para poder realizar la tarea, tienes que seleccionar la carpeta de destino de la fotografía.

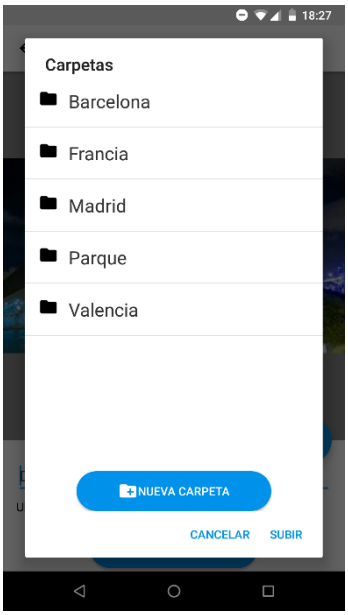


Figura 41: Selección de la carpeta de destino

Te mostramos todas las carpetas creadas anteriormente y la posibilidad crear una nueva. Una vez seleccionada o introducido el nombre de la nueva carpeta, selecciona la opción subir para finalizar el proceso y proceder con la subida de la fotografía.

Para visualizar el mapa y todas tus fotografías distribuidas en él, selecciona la opción de *Mapa* del menú principal.

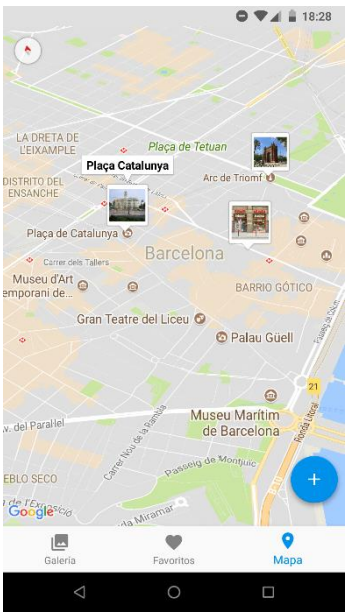


Figura 42: Visualización del mapa del usuario

Puedes interactuar con él y visualizar tus fotografías.

Finalmente, en la parte superior derecha del menú principal, tienes las diferentes opciones de la aplicación, donde encontrarás la posibilidad de cambiar el idioma, de cambiar la contraseña o de cerrar sesión.