Trabajo Fin de Grado
Ingeniería de Telecomunicación

Aplicación Android para Alquiler de Habitaciones Verificadas

Autor: Gonzalo Carranza Pérez-Tinao
Tutor: José Manuel Fornés Rumbao

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Proyecto Fin de Carrera
Ingeniería de Telecomunicación

Aplicación Android para Alquiler de Habitaciones Verificadas

Autor:
Gonzalo Carranza Pérez-Tinao

Tutor:
José Manuel Fornés Rumbao
Profesor titular

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Proyecto Fin de Carrera: Aplicación Android para Alquiler de Habitaciones Verificadas

Autor: Gonzalo Carranza Pérez-Tinao

Tutor: José Manuel Fornés Rumbao

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia

A mis maestros y profesores

A mis amigos

Agradecimientos

En primer lugar, quería agradecer a mis padres los cuales han sido mi principal apoyo durante todos estos años.

Agradecer también a todos mis compañeros y amigos con los que he compartido todos estos años de universidad, difíciles años, pero completamente satisfactorios.

También quería agradecer a José Manuel Fornés, mi tutor de este Trabajo Fin de Grado, que en los momentos más difíciles ha sabido darme respuesta y buen seguimiento ayudándome en la elaboración de este proyecto pudiendo llegar a las fechas establecidas.

Gonzalo Carranza Pérez-Tinao

Sevilla, 2020

Resumen

Hoy en día, el 67% de la población mundial ya cuenta con un teléfono móvil, siendo su uso mucho más que solo un medio de comunicación hablada. Entre uno de sus usos, la búsqueda de alojamiento, que es de lo que tratamos en este documento.

La solución que se plantea con esta aplicación, es dar la certeza al usuario que quiere alquilar una habitación/inmueble que lo que está alquilando, existe en la realidad y no se trata de una estafa.

En el mundo globalizado en el que vivimos es muy común viajar a diferentes países por un cierto tiempo. En muchas ocasiones no puedes ir a ver una habitación en persona ya que no te encuentras en ese país/ciudad en ese momento. Con esta aplicación se da la seguridad al posible inquilino que la habitación/inmueble que va a alquilar, sea veraz y no tenga la necesidad de visitarlo.

La aplicación está basada en el sistema operativo Android ya que, al ser un entorno de desarrollo abierto, facilita el proceso de creación.

La implementación se ha realizado con el IDE Android Studio, el cual es el entorno de desarrollo integrado oficial para la plataforma Android, brindándonos todas sus ventajas a la hora de crear la aplicación.

Se ha empleado la plataforma Firebase de Google. Se trata de una plataforma ubicada en la nube donde almacenaremos el backend de nuestra aplicación.

Abstract

Nowadays, 67% of the world's population already has a mobile phone, and its use is much more than just a means of spoken communication. Among its uses, the search for accommodation, which is what we are discussing in this document.

The solution proposed with this application is to give the user the certainty that what he is renting exists in reality and that it is not a scam.

In the globalised world in which we live it is very common to travel to different countries for a certain period of time. On many occasions you cannot go and see a room in person as you are not in that country/city at the time. With this application, the possible renter is assured that the room/property he is going to rent is real and he does not need to visit it.

The application is based on the Android operating system, as it is an open development environment that facilitates the creation process.

The implementation has been done with the Android Studio IDE, which is the official integrated development environment for the Android platform, giving us all its advantages when creating the application.

The Google Firebase platform has been used. This is a cloud-based platform where we will store the backend of our application.

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE CÓDIGO

1 INTRODUCCIÓN

Management is about persuading people to do things they do not want to do, while leadership is about inspiring people to do things they never thought they could

- Steve Jobs -

En la sociedad actual, no se contempla la vida sin estar conectados. Con la evolución que ha presentado el teléfono móvil, este se ha convertido en indispensable durante el día a día de cualquier persona, donde podemos hacer todo tipo de tareas, desde llamadas de teléfono, hacer de cámara de fotos y muchas más aplicaciones.

Esta evolución ha llegado a cambiar las relaciones sociales dotándolas de una mayor agilidad. Un ejemplo es el proyecto que presentamos a continuación.

1.1 Motivación

La idea de este proyecto nace por experiencia personal. En septiembre de 2017 estudié por un año en Irlanda gracias al programa Erasmus+. En este país existe una crisis inmobiliaria importante, con lo que buscar una habitación en la que pasar el año desde España no era una tarea sencilla. Varios fueron mis compañeros los cuales sufrieron de estafas ya que lo que estaban alquilando en realidad no existía, perdiendo el dinero de la reserva.

Con esta aplicación quiero presentar una solución a este problema en el que la persona que quiera alquilar una habitación/apartamento sin la necesidad de estar en la ciudad donde alquilar, pueda tener la seguridad que la habitación/apartamento por el que va a pagar, exista en la realidad.

1.2 Objetivos

El objetivo principal de este proyecto es la creación de una aplicación Android que permita a usuarios ofertar habitaciones/apartamentos de una manera sencilla como poder ver listado de las habitaciones disponibles según ubicación.

Para ello, la aplicación se ha ejecutado sobre la plataforma Firebase permitiendo que toda la información se aloje en la nube y tenga un fácil acceso.

1.3 Antecedentes

Existen aplicaciones móviles para el alquiler de habitaciones, estas aplicaciones están disponibles desde las APP Store correspondientes. La novedad que trae esta aplicación es que permite darle la seguridad al usuario de que todo lo que aparece en la aplicación es veraz y no se trata de una estafa. Esto se realiza por medio de una verificación física por parte del administrador de la aplicación. Seguidamente se configuraría el anuncio como verificado apareciendo una imagen con el sello de verificado lo cual permite darle seguridad al usuario final ya

que el anuncio que está viendo en la aplicación está verificado por el administrador.

Encontramos muchas aplicaciones dedicadas a el alquiler de inmuebles ya sean globales como locales. A continuación, detallamos algunos:

- **Idealista:**

Figura 1. Logo Idealista

Idealista es quizás la aplicación más usada para buscar alojamiento en el territorio de España.

Ofrece una plataforma web y móvil donde ofertan en función de tres principales grupos: comprar, alquilar o compartir.

Pero no solo se ofertan lugares donde dormir, puedes encontrar desde garajes, oficinas o incluso terrenos entre otros.

- **Badi**

Figura 2. Logo Badi

Badi es una aplicación enfocada al mundo estudiantil donde podrás encontrar habitaciones en pisos compartidos como también publicar una habitación en piso compartido.

Ofrece una plataforma web y móvil.

- **SpareRoom**

Figura 3. Logo SpareRoom

SpareRoom es la aplicación más usada en Reino Unido para la búsqueda de alojamiento.

Ofrece una plataforma web y móvil donde se ofertan tanto apartamentos completos como habitaciones en pisos compartidos.

2 ENTORNO DE TRABAJO Y TECNOLOGÍAS

A Continuación, se detallan los software y herramientas utilizados para el desarrollo del proyecto.

2.1. Android Studio IDE

Figura 4. Logo Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de apps para Android, basado en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece incluso más funciones que aumentan tu productividad cuando desarrollas apps para Android, como las siguientes:

- Un sistema de compilación flexible basado en Gradle
- Un emulador rápido y cargado de funciones
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android
- Aplicación de cambios para insertar cambios de códigos y recursos a la aplicación en ejecución sin reiniciar la aplicación
- Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra
- Variedad de marcos de trabajo y herramientas de prueba
- Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de la versión, entre otros
- Compatibilidad con C++ y NDK
- Compatibilidad integrada con Google Cloud Platform, que facilita la integración con Google Cloud Messaging y App Engine

Cada proyecto de Android Studio incluye uno o más módulos con archivos de código fuente y archivos de recursos. Entre los tipos de módulos se incluyen los siguientes:

- Módulos de apps para Android
- Módulos de biblioteca
- Módulos de Google App Engine

Los archivos de compilación se pueden ver en el nivel superior de Secuencias de comando de Gradle y cada módulo de app contiene las siguientes carpetas:

- manifests: Contiene el archivo AndroidManifest.xml.
- java: Contiene los archivos de código fuente Java, incluido el código de prueba de JUnit.
- res: Contiene todos los recursos sin código, como diseños XML, strings de IU e imágenes de mapa de bits.

A continuación, podemos ver un ejemplo de la interfaz gráfica:

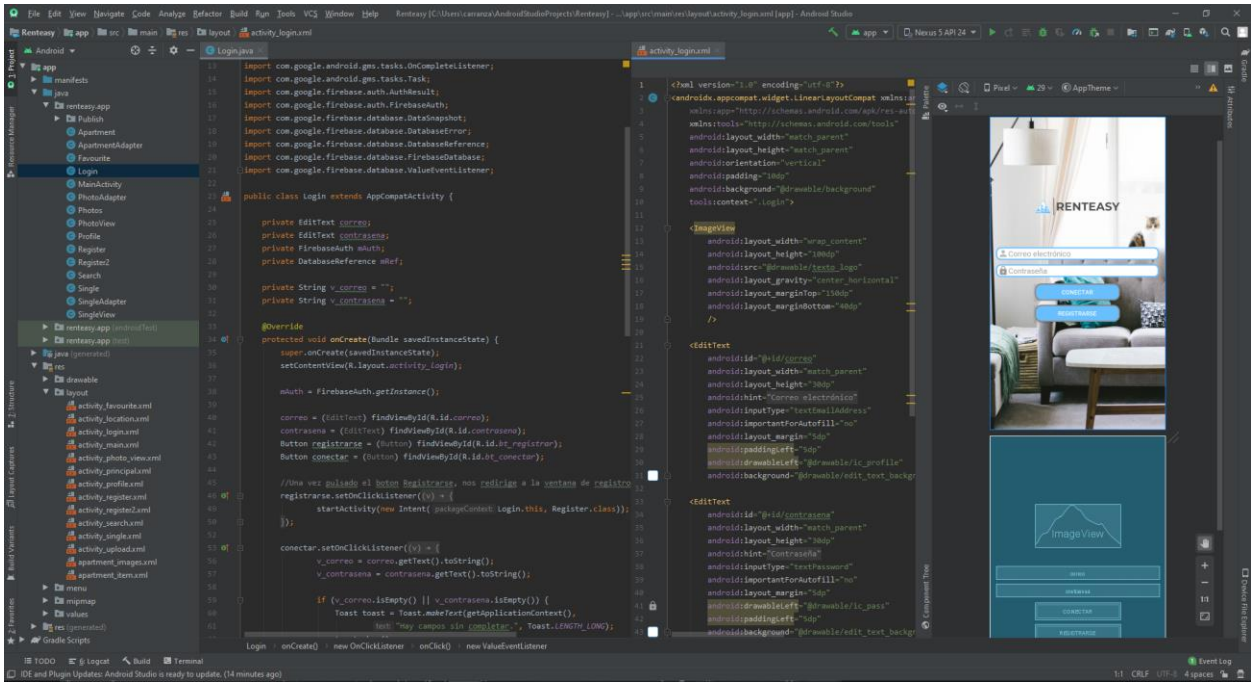

Figura 5. Interfaz gráfica Android Studio

2.2. Firebase

Figura 6. Logo Firebase

Firebase es una plataforma para el desarrollo de aplicaciones web y aplicaciones móviles desarrollada por Google en 2014

Es una plataforma ubicada en la nube, integrada con Google Cloud Platform, que usa un conjunto de herramientas para la creación y sincronización de proyectos que serán dotados de alta calidad, haciendo posible el crecimiento del número de usuarios y dando resultado también a la obtención de una mayor monetización.

Se ha optado por el uso de Firebase ya que todos los datos que se manejan en la aplicación están alojados en la nube y no de manera local en el dispositivo. También ofrece una gran cantidad de servicios los cuales se almacenan y procesan en la propia nube todo ello manejado por la infraestructura de Google Cloud.

Gracias a ello, estamos creando un sistema alojado en la nube en donde lo único que hacemos es manejar las peticiones a los diferentes servicios.

También facilita la sincronización ya que los cambios que realice un usuario se actualizan en tiempo real en la nube viéndose reflejados en el resto de usuarios.

Algunas de ventajas al usar esta plataforma son:

- Sincronizar fácilmente los datos de sus proyectos sin tener que administrar conexiones o escribir lógica de sincronización compleja.
- Usa un conjunto de herramientas multiplataforma: se integra fácilmente para plataformas web como en aplicaciones móviles. Es compatible con grandes plataformas, como IOS, Android, aplicaciones web, Unity y C++.
- Usa la infraestructura de Google y escala automáticamente para cualquier tipo de aplicación, desde las más pequeñas hasta las más potentes.
- Crea proyectos sin necesidad de un servidor: Las herramientas se incluyen en los SDK para los dispositivos móviles y web, por lo que no es necesario la creación de un servidor para el proyecto.

De entre sus desventajas podemos destacar:

- Mala experiencia en redes lentas al depender de la conexión a internet, en redes 2G, 3G o redes WIFI lentas no disfrutaremos de una experiencia de usuario completa al tardar en cargas las diferentes pantallas de la aplicación.
- Genera un mayor consumo de datos móviles al estar constantemente comunicando con los servidores en la nube.
- Si el servidor se encuentra con problemas como por ejemplo caídas en la red, esto afectará directamente a la aplicación ya que el servidor al que se conecta no está operativo.

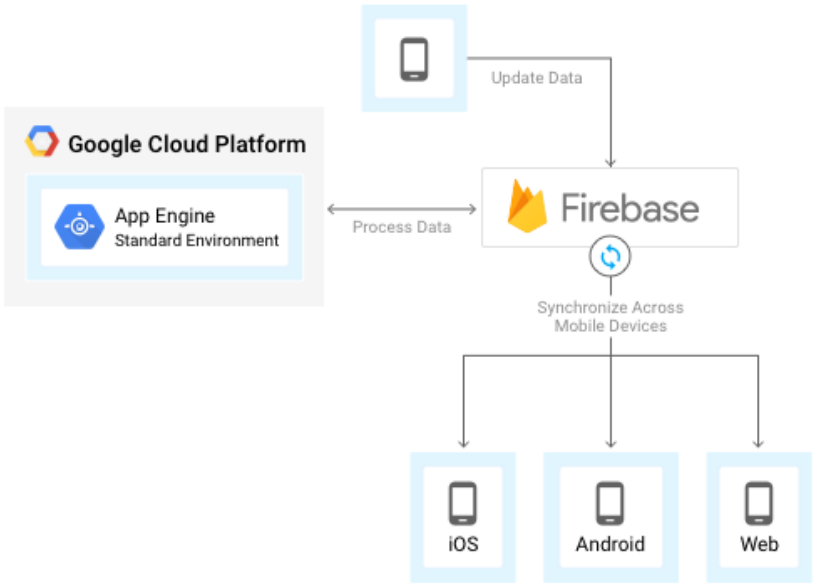

Figura 7. Arquitectura estándar Firebase

Firebase dota a sus usuarios de una gran documentación para crear aplicaciones usando esta plataforma. Aparte de esto, ofrece soporte gratuito mediante correo electrónico para todos sus usuarios, y además sus desarrolladores participan activamente en plataformas como Github y StackOverflow, así como poseen un canal de Youtube explicando el funcionamiento de varias de sus herramientas.

Figura 8. Opciones disponibles en Firebase

Como podemos observar, la plataforma cuenta con muchas opciones diferentes, dependiendo del uso que queramos para nuestra aplicación. Para este proyecto hemos usado: Authentication, Cloud Storage y Realtime Database, los cuales explicamos en detalle a continuación.

Authentication

Firebase Auth es un servicio que puede autenticar los usuarios utilizando únicamente código del lado del cliente. Incluye la autenticación mediante proveedores de inicio de sesión como Facebook, GitHub, Twitter, Google, Yahoo y Microsoft; así como los métodos clásicos de inicio de sesión mediante correo electrónico y contraseña. Además, incluye un sistema de administración del usuario por el cual los desarrolladores pueden habilitar la autenticación de usuarios con email y contraseña que se almacenarán en Firebase.

Este servicio busca facilitar la creación de sistemas de autenticación, a la vez que mejora la incorporación, acceso y seguridad para los usuarios. Gracias a esto, el cliente no tiene que preocuparse por desarrollar métodos de autenticación clásicos, ya que Firebase le aporta de manera sencilla, eficaz y segura métodos para gestionar sus usuarios.

También aporta muchas funcionalidades extra, como la recuperación y verificación de cuentas, tanto por correo electrónico como por SMS, y cuotas de registro para los usuarios, todo esto gestionado mediante los servidores

de la plataforma.

Figura 9. Esquema Firebase Authentication

Cloud Storage

Cloud Firestore es un servicio es un servicio de almacenamiento de objetos potente, simple y rentable construido para la escala de Google. Los SDK de Firebase para Cloud Storage agregan la seguridad de Google a las operaciones de carga y descarga de archivos para tus apps de Firebase, sin importar la calidad de la red.

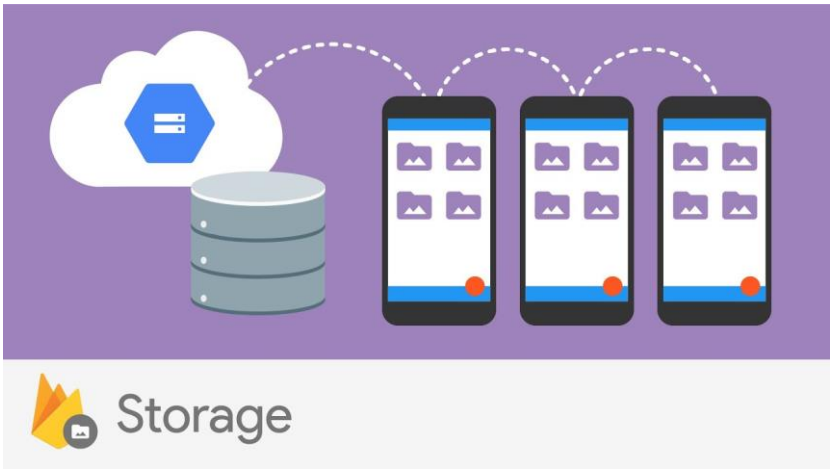

Figura 10. Esquema Firebase Cloud Storage

Similar a Realtime Database, pero enfocado al almacenamiento de contenido generado por los usuarios, como imágenes y videos, lo que te permite integrar contenido de rich media en tus apps. Los datos se almacenan en un depósito de Google Cloud Storage, una solución de almacenamiento de objetos a escala de exabytes con alta disponibilidad y redundancia global. Cloud Storage te permite subir de forma segura estos archivos directamente desde dispositivos móviles y navegadores web, además de administrar las redes irregulares con facilidad.

Realtime Database

Firebase proporciona una base de datos en tiempo real, back-end y organizada en forma de árbol JSON. El servicio proporciona a los desarrolladores de aplicaciones una API que permite que la información de las aplicaciones sea sincronizada y almacenada en la nube de Firebase. Realtime Database habilita integración con aplicaciones Android, iOS, JavaScript, Java, Objective-C, Swift y Node.js. La base de datos es también accesible a través de una REST API e integración para varios sistemas de Javascript como AngularJS, React, Ember.js y Backbone.js. La REST API utiliza el protocolo SSE (del inglés Server-Sent Events), el cual es una API para

crear conexiones de HTTP para recibir notificaciones push de un servidor.

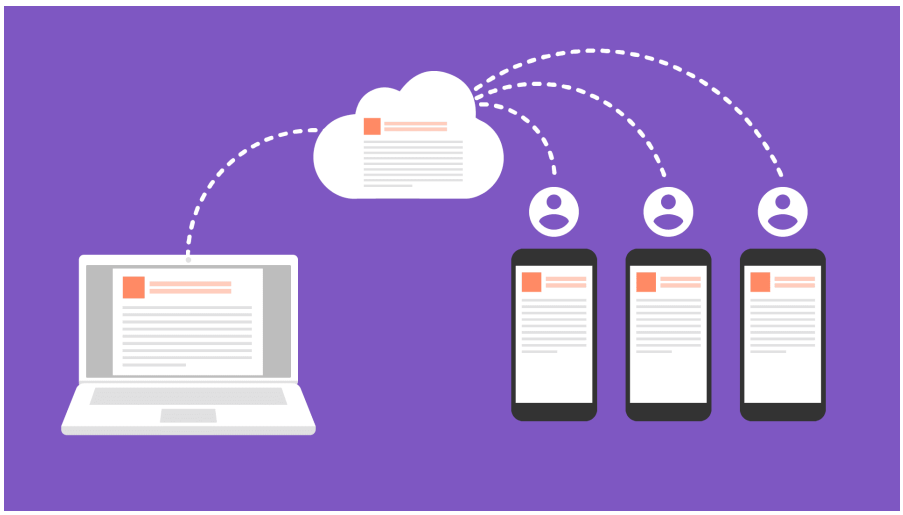

Figura 11. Esquema Firebase Realtime Database

La sincronización en tiempo real de esta base de datos permite que los usuarios accedan a la información de sus datos desde cualquier dispositivo en tiempo real, compartiendo una instancia de Realtime Database, y cada vez que un usuario realice una modificación en esta, se almacena dicha información en la nube y se notifica simultáneamente al resto de dispositivos.

Una funcionalidad interesante de esta base de datos, es que, si un usuario realiza cambios y pierde a la vez su conexión a Internet, el SDK de la plataforma usa una caché local en el dispositivo donde guarda estos cambios; y una vez que vuelva a tener conexión, automáticamente se sincronizan los datos locales.

2.3. Google Maps Platform

Google Maps Platform es un conjunto de API y SDK que permite a los desarrolladores incrustar Google Maps en aplicaciones para móviles y páginas web, o bien recuperar datos de Google Maps. Hay varias ofertas. Según tus necesidades, es posible que utilices una o una combinación de estas API y SDK.

Los servicios ofrecidos por la plataforma están divididos en tres productos: Maps, Routes y Places

- Maps: Cuenta con dos recursos: Maps y Street View. Estos permiten a los usuarios visualizar el mundo real, por medio de mapas estáticos o interactivos, que pueden ser incorporados a sitios web o aplicaciones.
- Routes: La herramienta Routes ofrece una forma práctica y segura para que los usuarios encuentren el mejor trayecto para el destino seleccionado.
- Places: Ya sea para buscar un restaurante en tu vecindario o un hotel para un viaje en otro país, Places es una gran opción para una buena planeación. Para eso, ofrece información detallada como nombre de los locales, direcciones, evaluaciones y datos de contacto.

2.4. Java

Figura 12. Logo Java

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán a menos que tenga Java instalado y cada día se crean más. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes.

Características

- Lenguaje orientado a Objetos
- Disponibilidad de un amplio conjunto de bibliotecas
- Lenguaje simple
- Distribuido
- Interpretado y compilado a la vez
- Robusto
- Seguro
- Indiferente a la arquitectura
- Portable
- Alto rendimiento
- Multihebra
- Dinámico
-

2.5. Librerías Externas

- Picasso

  Las imágenes añaden mucha información y elegancia a las aplicaciones de Android. Picasso permite cargar imágenes sin problemas en tu aplicación

2.6. Draw.io

Figura 13. Logo Draw.io

Draw.io es un editor de diagramas en línea completamente gratuito construido en torno a Google Drive(TM), que permite crear diagramas de flujo, UML, relación de entidades, diagramas de red, maquetas y más. Sus datos

se almacenan en Google Drive siendo fácil su acceso desde cualquier lugar.

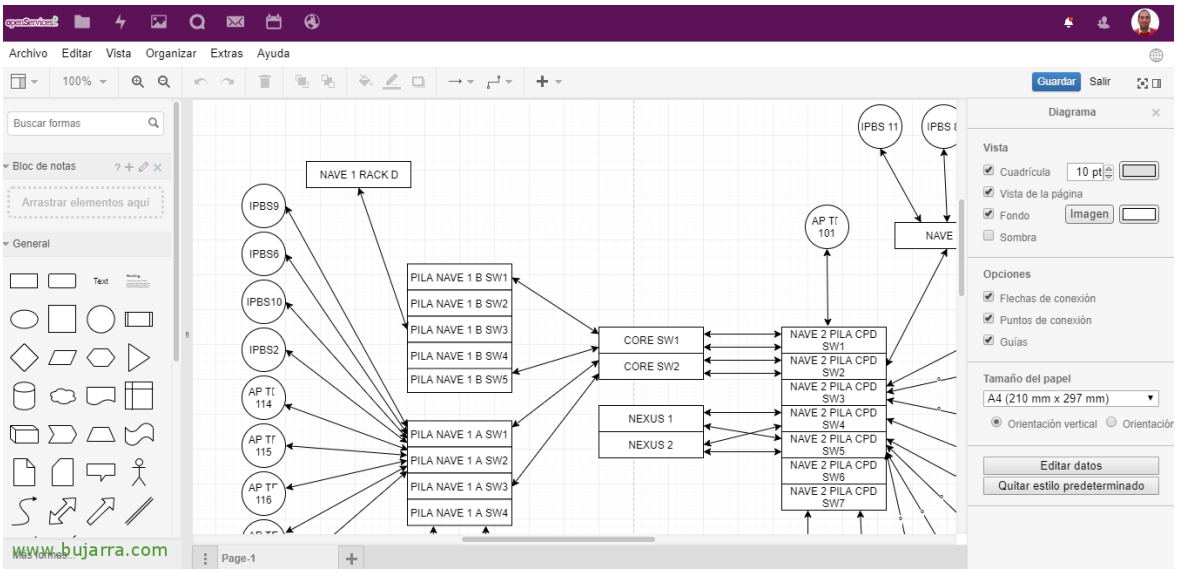

Figura 14. Interfaz gráfica Draw.io

Se ha optado por el uso de Draw.io para la creación de diagramas UML por su sencillez, por ser una herramienta conocida y por ser fácilmente exportable.

2.7. Visual Paradigm

Figura 15. Logo Visual Paradigm

Visual Paradigm es una herramienta UML CASE que admite UML 2, SysML y notación de modelado de procesos empresariales del grupo de gestión de objetos. Además del soporte de modelado, proporciona capacidades de generación de informes e ingeniería de código, incluida la generación de código.

Este programa se ha usado para la creación de diagrama de clases que se verán más adelante en este documento.

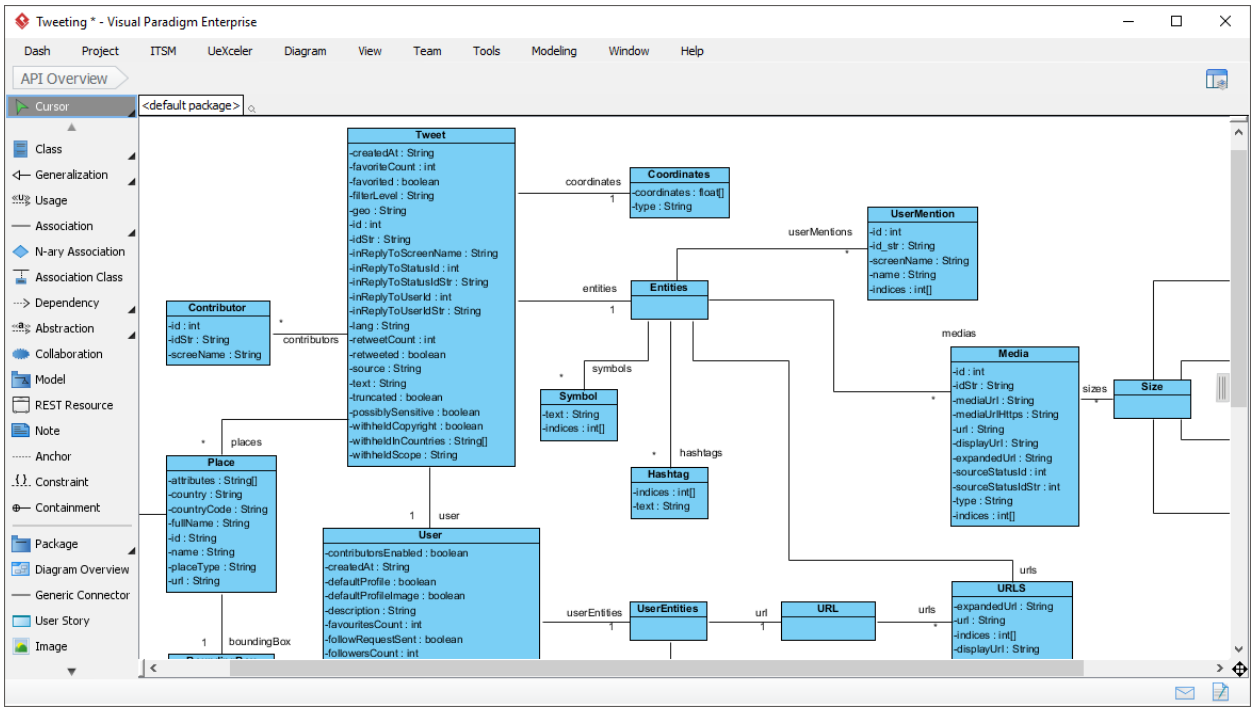

Figura 16. Interfaz gráfica Visual Paradigm

3 INTERFAZ GRÁFICA

En este apartado vamos a describir en detalle el diseño de la interfaz gráfica de usuario de la aplicación. Vamos a analizar cómo se comporta la aplicación, las pantallas que lo componen y las diferentes opciones que tienen los usuarios.

Este documento se dirige a todas las personas que quieran saber el funcionamiento de la aplicación desarrollada.

3.1 Introducción

La aplicación recibe el nombre de Renteasy. Una vez descargada la aplicación, aparecerá en el dispositivo con el siguiente icono:

Figura 17. Logo Renteasy

Para su implementación se ha usado XML como front end de la aplicación. A continuación, mostramos un esquema general de todas las pantallas por las que el usuario puede navegar.

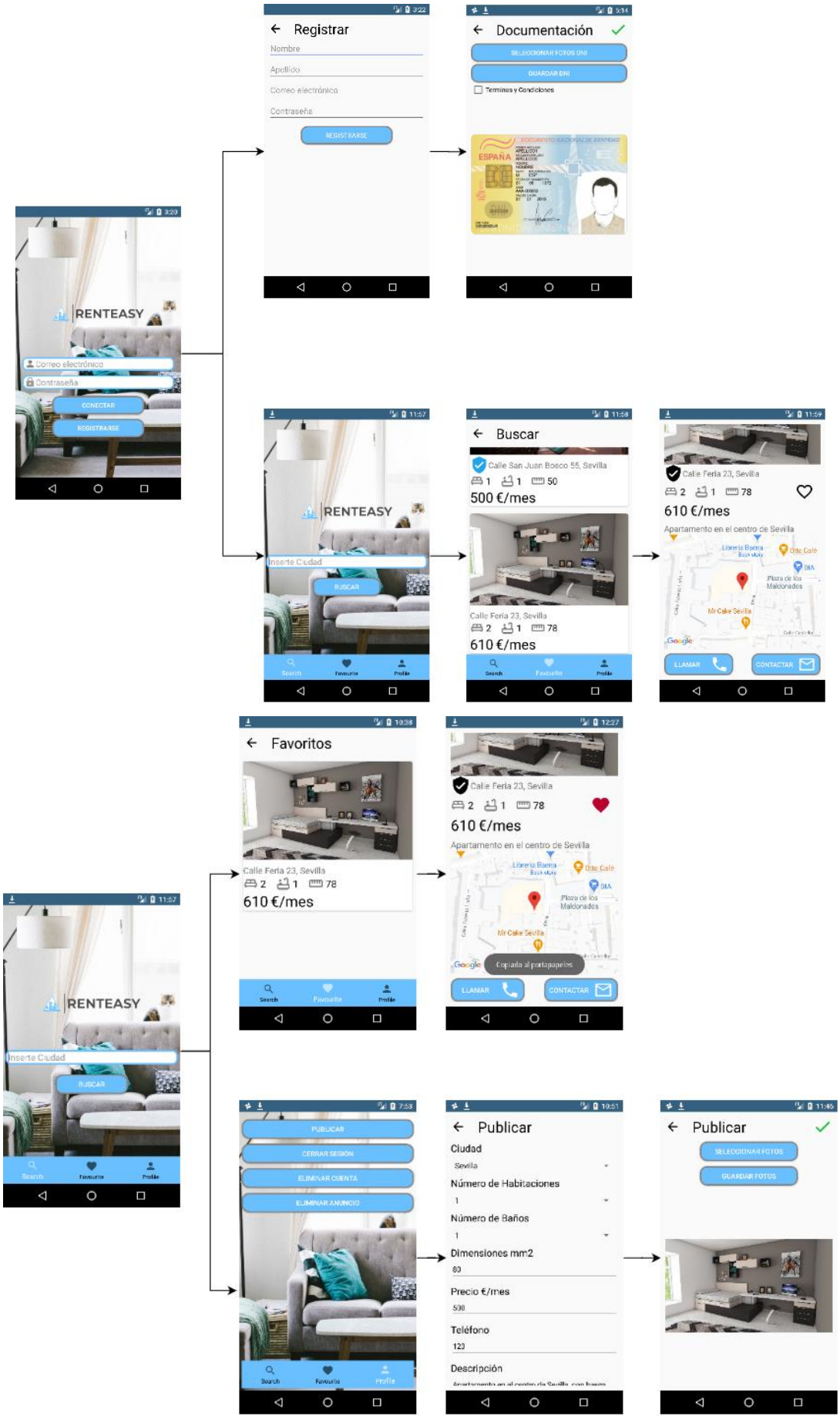

Figura 18. Flujo completo de pantallas

La interfaz gráfica de la aplicación ha sido diseñada según la normativa de diseño Material Desing. Este sistema de diseño fue creado por Google para ayudar a los equipos a construir experiencias digitales de alta calidad. Se ha seguido las líneas de diseño descritas según esta normativa en la implementación de colores y formas de manera que resulte más agradable a la vista.

3.2 Pantalla de Login

En primer lugar, al pulsar sobre el icono de la aplicación, esta nos llevará a dos posibles pantallas. Si hemos iniciado sesión previamente, nos llevará directamente al *Activity* principal Inicio, en caso contrario, nos llevará al *Activity* Login.

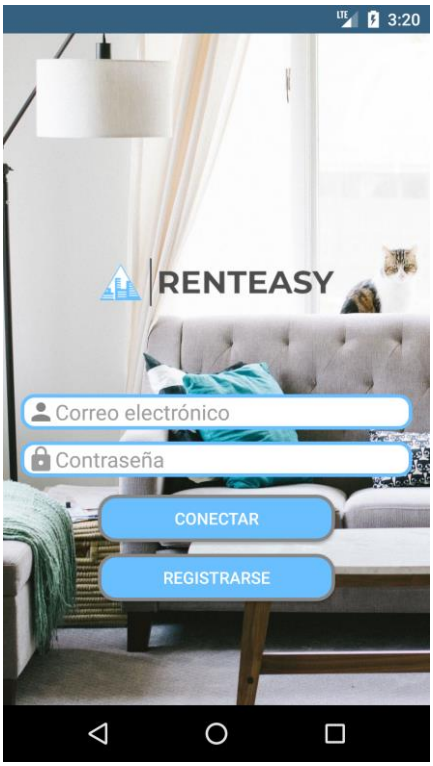

Figura 19. Pantalla Login

En la pantalla de Login tendremos dos opciones. Una vez rellenado los campos de correo y contraseña, la aplicación comprobará que el usuario existe dentro de la base de datos, que la contraseña sea la correcta y que el usuario ha sido verificado (correo de verificación). Completado esto, la aplicación nos llevará al *Activity* Inicio.

3.3 Pantalla de Registro

En el caso de pulsar el botón Registrarse, nos llevará al *Activity* Registro donde se abrirá un formulario con una serie de información de contacto a rellenar.

Figura 20 Pantalla Registro

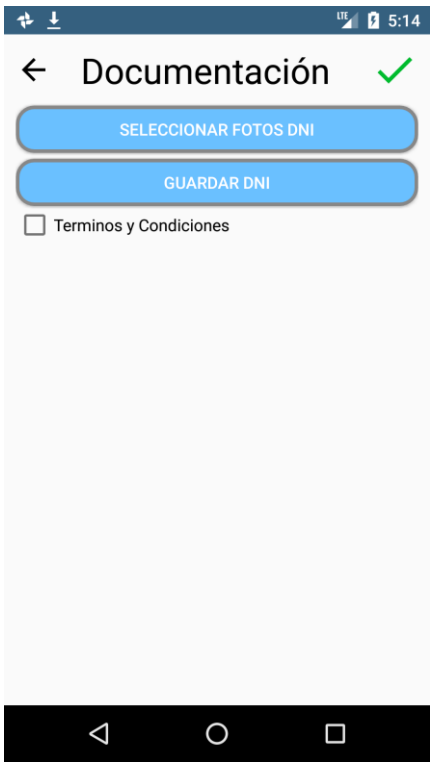

Figura 21 Pantalla Dni 1

Una vez completados todos los campos y pulsado el botón de registrarse, la aplicación comprobará que todos los campos de *EditText* han sido rellenados, sino mandará un *Toast* indicando que faltan campos por rellenar. Si todos los campos se han rellenado, se comprueba en la base de datos que no tengamos ningún usuario con el mismo correo electrónico. Una vez hecho esto pasaremos a la pantalla Dni. En este Activity primero, pulsando sobre el botón Seleccionar Fotos Dni, seleccionamos una foto de la galería del teléfono. Una vez seleccionada una foto, esta se mostrará en la pantalla anterior a través de un *ImageView*. Lo siguiente es pulsar sobre el botón guardar y de esta manera guardar en la base de datos la foto correspondiente a nuestro Dni.

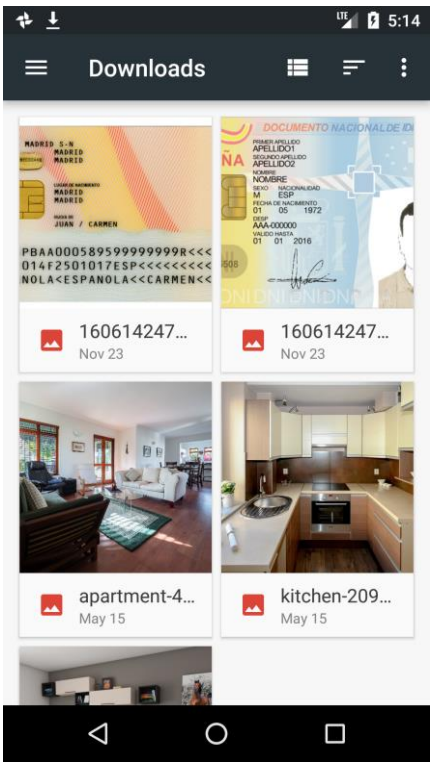

Figura 22. Pantalla galería de fotos

Figura 23. Pantalla Dni 2

Para finalizar, debemos seleccionar la casilla de Términos y Condiciones y tocar sobre el tick verde para finalizar el registro.

Si se cumple todo lo anterior, nos mandará al correo electrónico indicado un correo con un enlace para autenticar la cuenta, y volveremos al *Activity* Login.

3.4 Pantalla Principal

Tras logguearnos con éxito, pasamos a la pantalla de búsqueda. Es la pantalla principal de la aplicación.

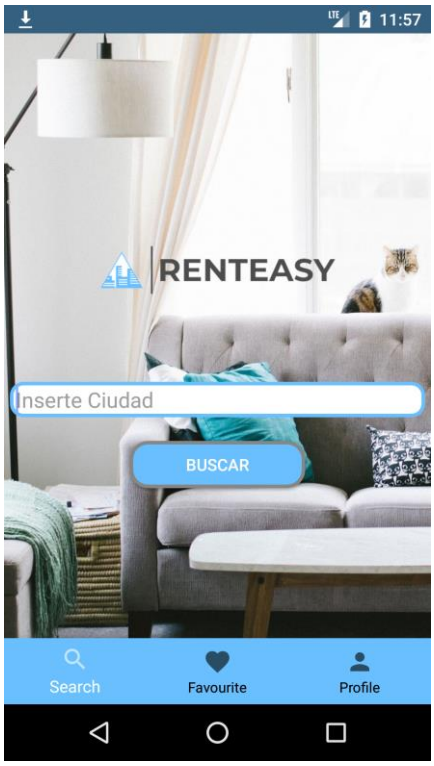

Figura 24. Pantalla Inicio

Como se puede observar está formado por un EditText donde introduciremos el nombre de la ciudad por la que filtrar la búsqueda, un botón que nos llevará al siguiente *Activity* mostrando los anuncios disponibles en la ciudad escrita. En esta pantalla también contamos con un *Bottom Navigation Menu* el cual es una de las herramientas que trae consigo Material Desing con la cual podremos navegar de manera sencilla entre las tres *Activities* principales.

La siguiente pantalla, muestra en un *RecyclerView* los anuncios disponibles para la ciudad indicada en la pantalla Inicio. La información se muestra en formato de *CardView* con la información más relevante de cada anuncio, como podemos comprobar en la siguiente imagen:

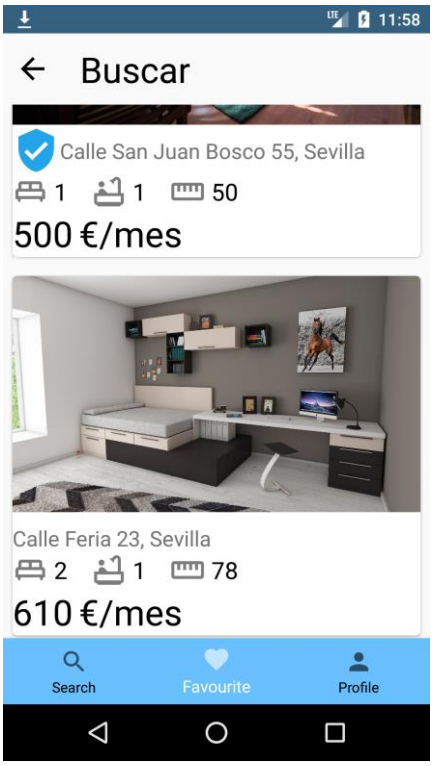

Figura 25. Pantalla de Búsqueda

Tocando sobre uno de los anuncios, pasaremos a otro Activity con toda la información sobre el anuncio seleccionado.

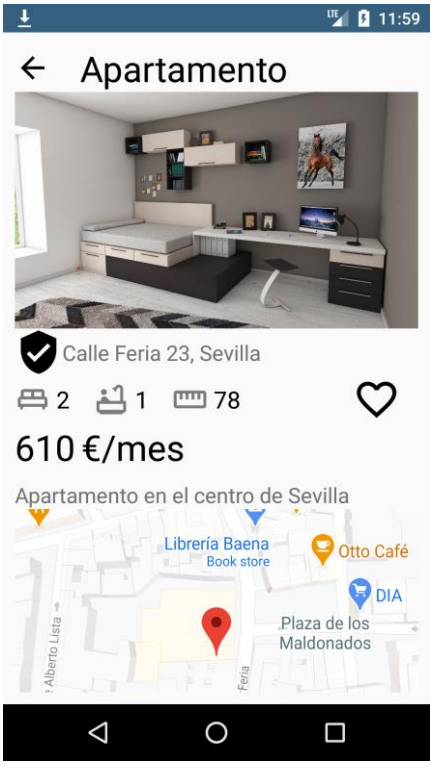

Figura 26. Pantalla Anuncio 1

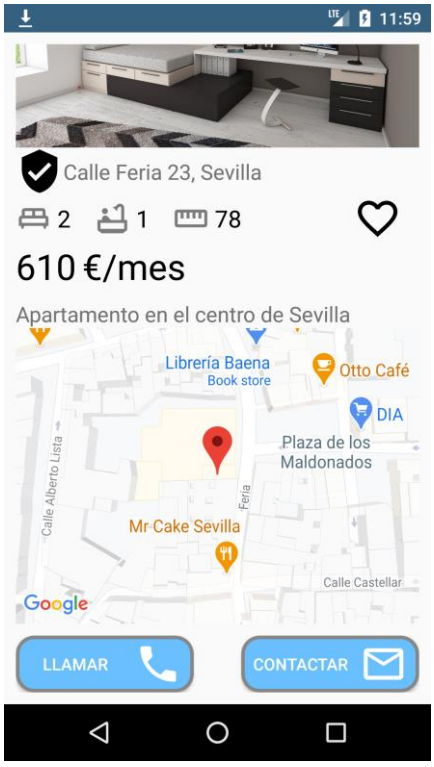

Figura 27. Pantalla Anuncio 2

En esta *Activity* podremos hacer diferentes acciones:

- Tocando sobre la imagen, se nos abrirá otra pantalla con todas las imágenes que el propietario haya subido. Estas imágenes se desplegarán en un *RecyclerView* como podemos observar en la imagen.
- Tocando sobre el *ImageButton* corazón, guardaremos este anuncio en nuestros favoritos, cambiando a su vez este de color a rojo.
- Tocando sobre el botón Llamar, si no se han aceptado previamente los permisos de llamada, la aplicación nos solicitará permisos para su uso. A continuación, pasaremos a llamar al teléfono que el propietario añadió al anuncio.
- Tocando sobre el botón Contactar, se copiará al portapapeles el correo del propietario.

Figura 28. Pantalla Imágenes

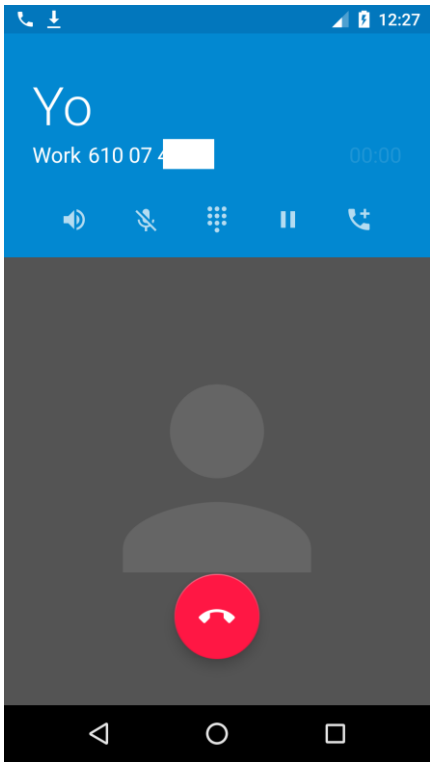

Figura 29. Pantalla Llamar

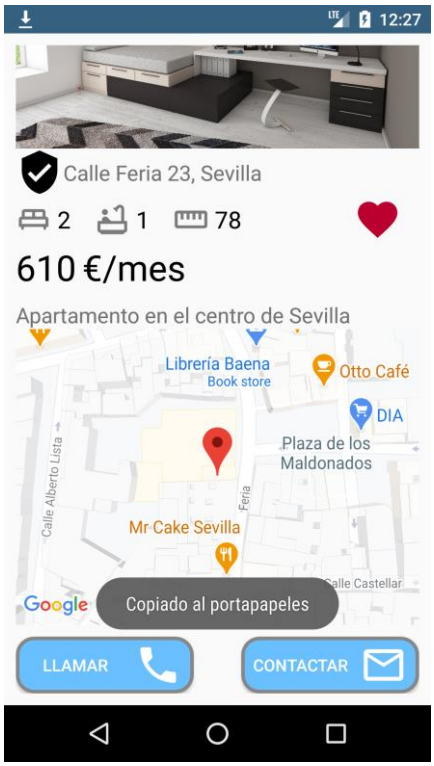

Figura 30. Pantalla Contactar

Aquí también destacamos uno de los puntos diferenciales de esta aplicación. Una vez se ha creado un anuncio, nos aparecerá el icono de verificado en negro. Este icono trabaja con un dato tipo *Boolean* el cual se iniciará en estado falso al crear el anuncio. El administrador, en este caso el creador de la aplicación, cambiará este estado de falso a verdadero una vez haya podido verificar la veracidad del anuncio publicado. El usuario podrá distinguir que un anuncio ha sido verificado gracias al icono que aparece en cada anuncio:

Figura 31. Icono Verificado

3.5 Pantalla de Favorito

A continuación, mostramos la pantalla Favorito, esta *Activity* se muestra de la misma manera que la pantalla de

búsqueda, pero mostrando los anuncios que han sido marcados como favoritos. Al igual que la pantalla de búsqueda, al tocar sobre un anuncio, este abrirá el mismo *Activity* usado en el apartado anterior para mostrar la información sobre el anuncio.

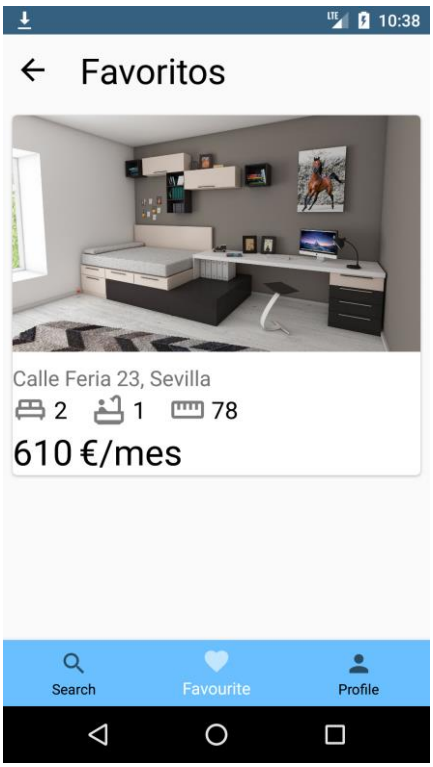

Figura 32. Pantalla Favoritos

3.6 Pantalla de Usuario

Por último, tenemos la pantalla de Usuario. En este *Activity* nos encontramos con cuatro botones.

Al pulsar el botón de cerrar sesión, como su propio nombre indica, el usuario cerrará la sesión abierta volviendo a la pantalla de Login.

El botón eliminar anuncio, borrará el anuncio que tengamos publicado.

El botón eliminar cuenta, borrará permanentemente nuestro usuario. Una vez completado volveremos a la pantalla de Login.

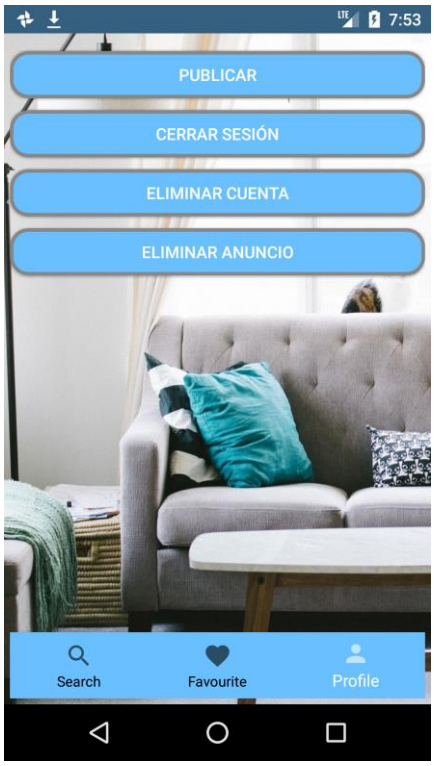

Figura 33. Pantalla de Usuario

El botón de publicar nos llevará a un primer *Activity* con un formulario a rellenar con la información del anuncio.

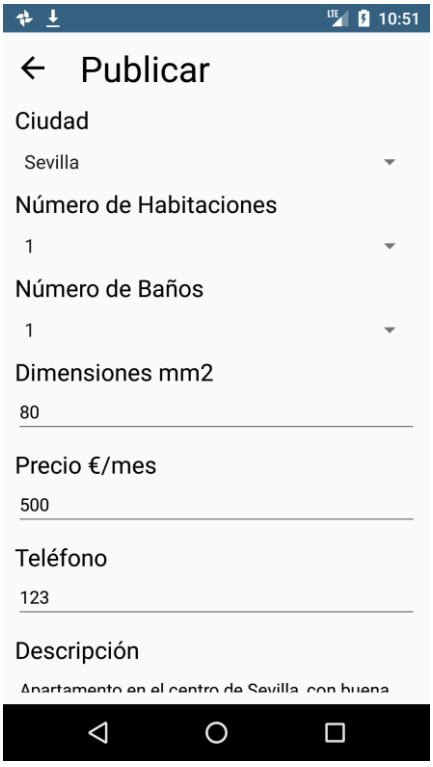

Figura 34. Pantalla Publicar 1

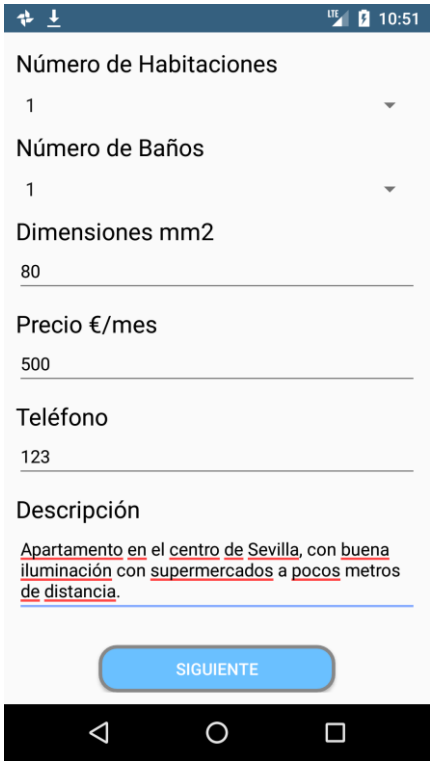

Figura 35. Pantalla Publicar 2

En ella podemos ver como varios *Spinners* y *EditText*. Destacar que en esta versión de proyecto se ha decidido que la entrada de ciudad sea en formato Spinner para acotar la solución. Una vez tengamos todos los campos completados, pulsaremos el botón siguiente guardando esta información en la base de datos de Firebase. Si hay

algún campo sin completar, el sistema no dejará avanzar a la siguiente pantalla, solicitando por medio de un *Toast* que hay campos sin completar.

La siguiente pantalla, nos aparecerá a un *Activity* en el cual se implementan funcionalidades de Google Maps.

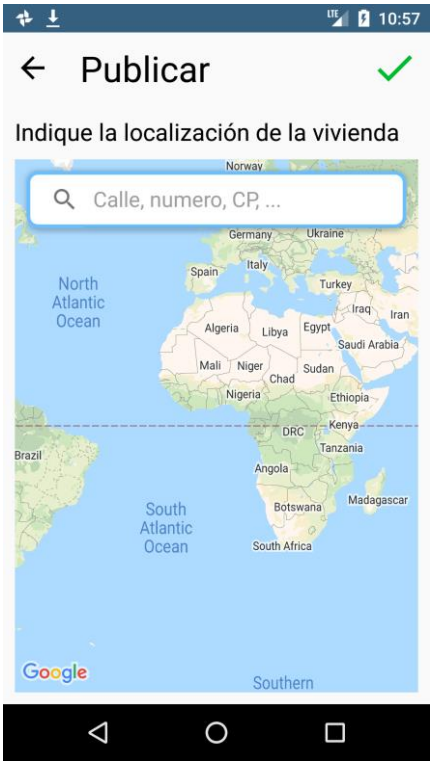

Figura 36. Google Maps 1

Figura 37. Google Maps 2

La pantalla está formada por una barra superior *SearchView* que proporciona una interfaz de usuario para que el usuario introduzca una consulta de búsqueda y envía una solicitud al proveedor de búsqueda, en este caso Google Maps. Del nombre introducido se obtiene la latitud y la longitud que son los parámetros que usaremos para marcar en el mapa el punto indicado. Tocando en el tick verde, guardaremos el nombre de la calle en la base de datos y pasaremos a la siguiente pantalla.

La siguiente pantalla es la de seleccionar fotos de nuestra galería para añadirlas al anuncio.

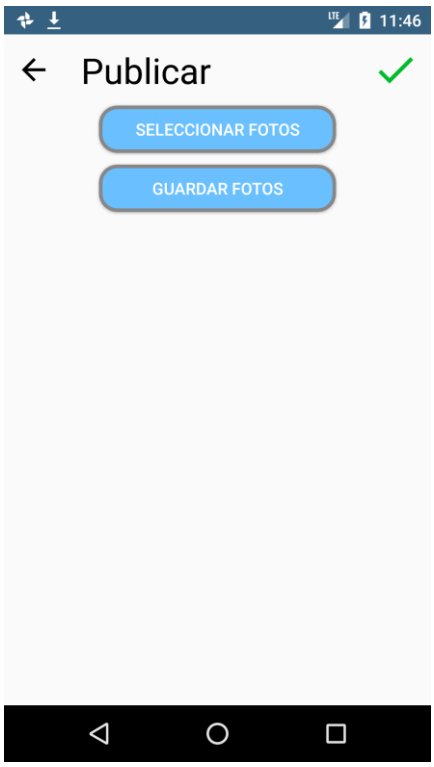

Figura 38. Pantalla Publicar 1

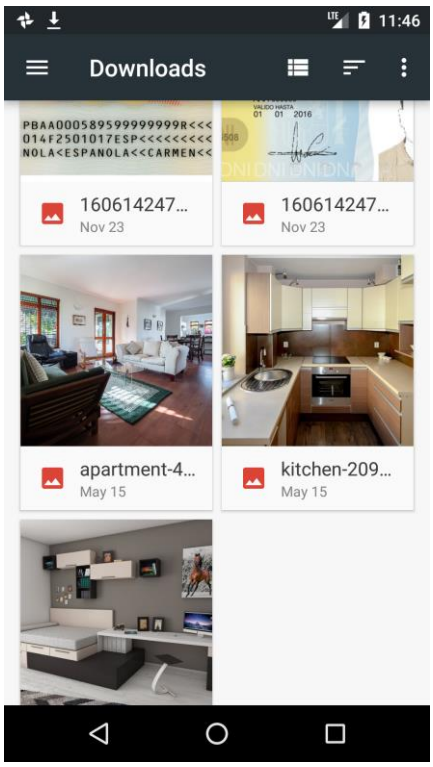

Figura 39. Pantalla galería fotos

Podemos comprobar que se nos dan dos opciones, una para seleccionar fotos. Al tocar sobre este botón se nos abrirá nuestra galería mostrándonos las diferentes fotos que tenemos almacenadas en nuestro teléfono.

Al seleccionar sobre una imagen, volveremos a la pantalla anterior, mostrándose la imagen seleccionada en grande sobre un *ImageView* como podemos ver en la imagen.

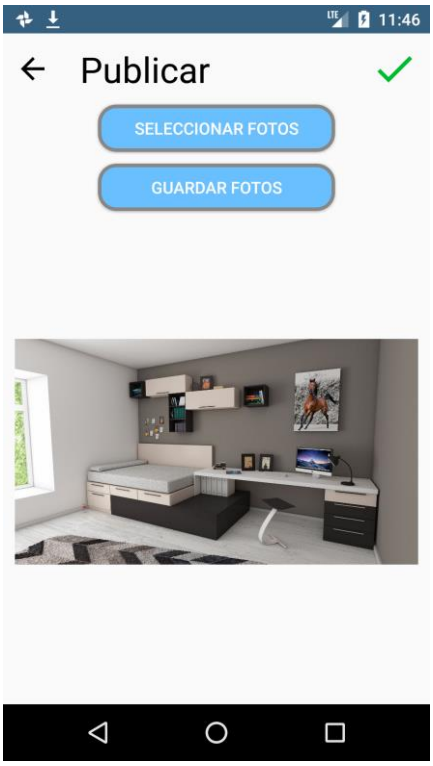

Figura 40. Pantalla Publicar 2

Tocando sobre el botón Guardar Fotos, guardaremos en la base de datos la foto mostrada en el *ImageView*.

Una vez subidas las fotos que queramos, le daremos al tick verde para finalizar la publicación del anuncio, volviendo a la pantalla de Inicio.

4 DISEÑO DEL SISTEMA

En este apartado vamos a analizar cómo se comporta la aplicación, la estructura en la que se desarrolla el sistema y todas las funcionalidades que presenta.

Este documento se dirige a todas las personas que quieran saber el funcionamiento de la aplicación desarrollada.

4.1 Diagramas de Flujo

En este apartado veremos una serie de diagramas de flujo donde se detallan las acciones disponibles para el usuario y el comportamiento de la aplicación.

En la siguiente tabla, se describen las diferentes vistas de la aplicación.

Vistas de la aplicación	
Vista	**Descripción**
Login	Vista donde tendremos dos opciones, Loggearnos con un usuario y contraseña, o registrarnos
Registro	Vista donde se muestra un formulario con los datos a rellenar del usuario y nos solicita que subamos foto de DNI
Inicio	Vista donde podremos buscar anuncios escribiendo el nombre de la ciudad
Buscar	Vista donde se muestran las habitaciones disponibles para la ciudad indicada
Habitación	Vista donde se muestra toda la información sobre un anuncio
Favoritos	Vista donde se muestran todas las habitaciones que previamente hemos guardado en favoritos
Usuario	Vista donde tendremos cuatro opciones, publicar anuncio, cerrar sesión, eliminar anuncio y eliminar cuenta
Publicar	Vista donde se muestra un formulario con datos a rellenar del anuncio, ubicación del anuncio y nos solicita que subamos imágenes del anuncio

A continuación, se muestra un esquema general de todas las vistas y cómo estas interaccionan con la base de datos de Firebase.

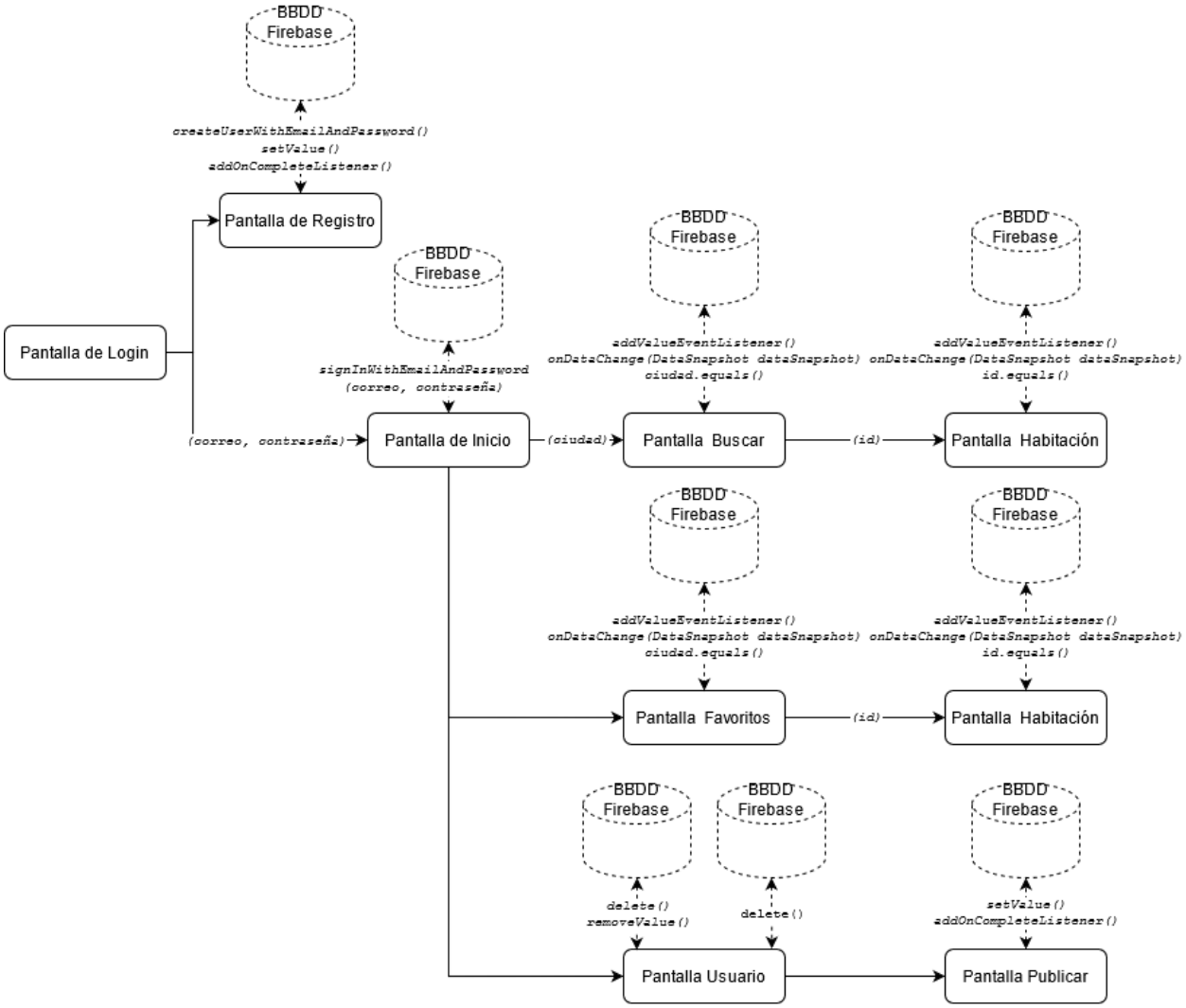

Figura 41. Esquema General App-Firebase

A continuación, representamos el inicio de la aplicación. Una vez abrimos la aplicación y esta detecta que no se ha iniciado sesión previamente, pasaremos a la pantalla de Login. Si el usuario hubiese iniciado sesión previamente, nos iríamos directamente a la pantalla de Inicio.

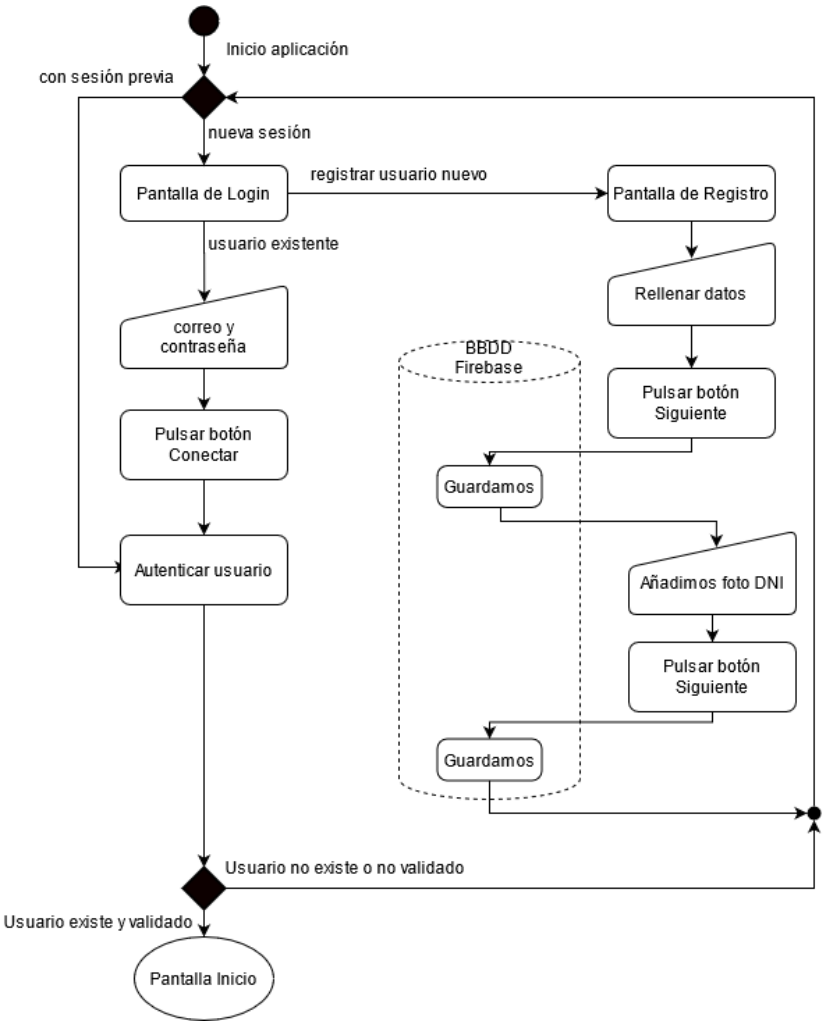

Figura 42. Diagrama de flujo subsistema Login

En la pantalla de Login tendremos dos opciones, registrar un usuario nuevo o iniciar sesión con un usuario registrado previamente.

El segundo diagrama muestra la pantalla principal Inicio de la aplicación. Esta es la pantalla de búsqueda en donde en función de la ciudad que introduzcamos en el buscador, nos mostrará los anuncios disponibles. Estos anuncios se mostrarán en formato de tarjetas con la información más relevante. Al tocar sobre una de ellas nos abrirá una nueva pantalla con toda la información de la misma.

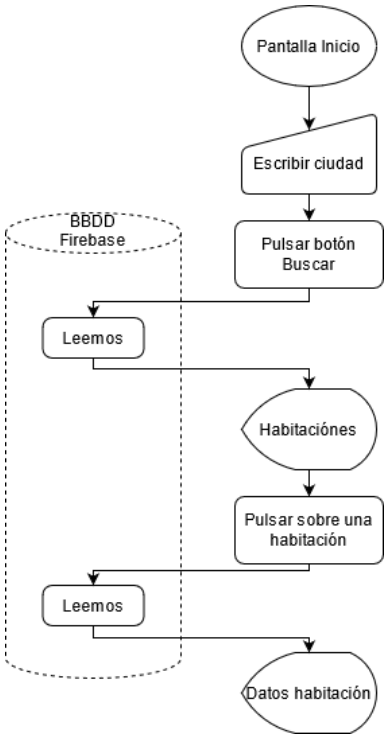

Figura 43. Diagrama de flujo subsistema Inicio

El tercer diagrama muestra la pantalla de favoritos la cual mostrará de la misma forma que el diagrama anterior, pero en este caso mostrando únicamente los anuncios guardados.

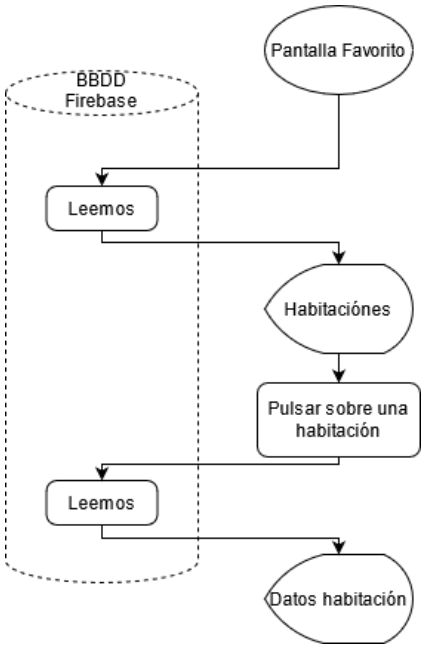

Figura 44. Diagrama de flujo subsistema Favoritos

El cuarto diagrama muestra la pantalla de Usuario. En ella tendremos tres opciones, hacer una publicación, editar información y cerrar sesión.

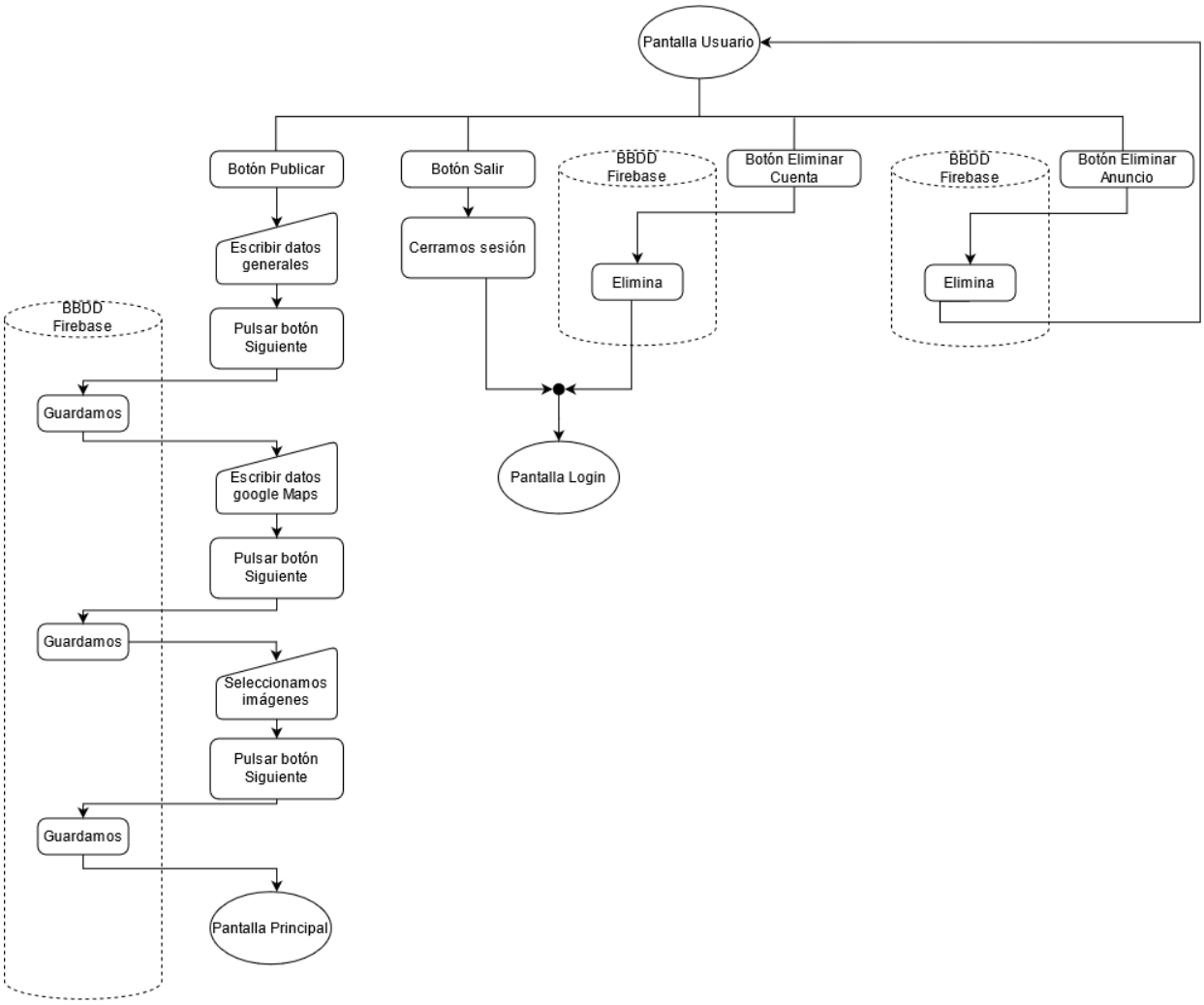

Figura 45. Diagrama de flujo subsistema Usuario

4.2 Diagramas de Clases

El diagrama de clases es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, métodos y relación entre objetos. A continuación, mostramos los diagramas de clases de nuestra aplicación. Para una mejor comprensión se han añadido las clases principales y se ha dividido en cuatro partes.

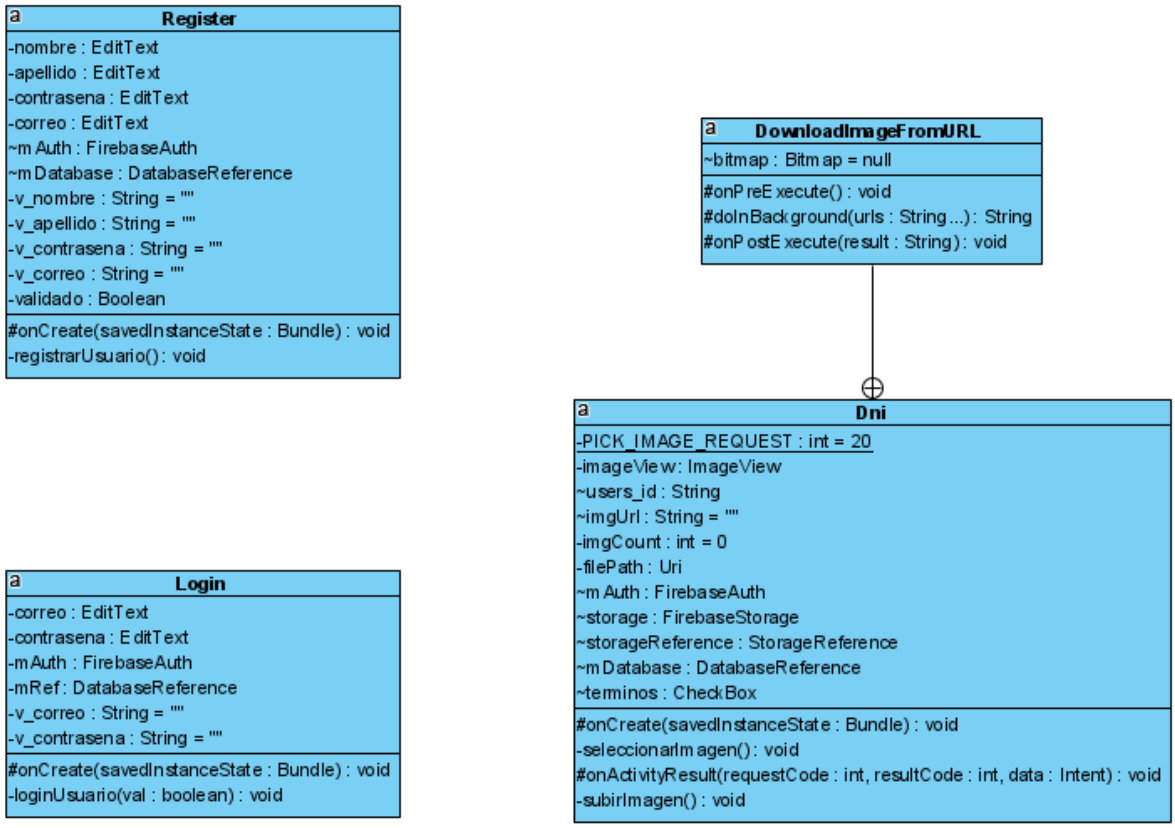

Figura 46. Diagrama de clases Login y Register

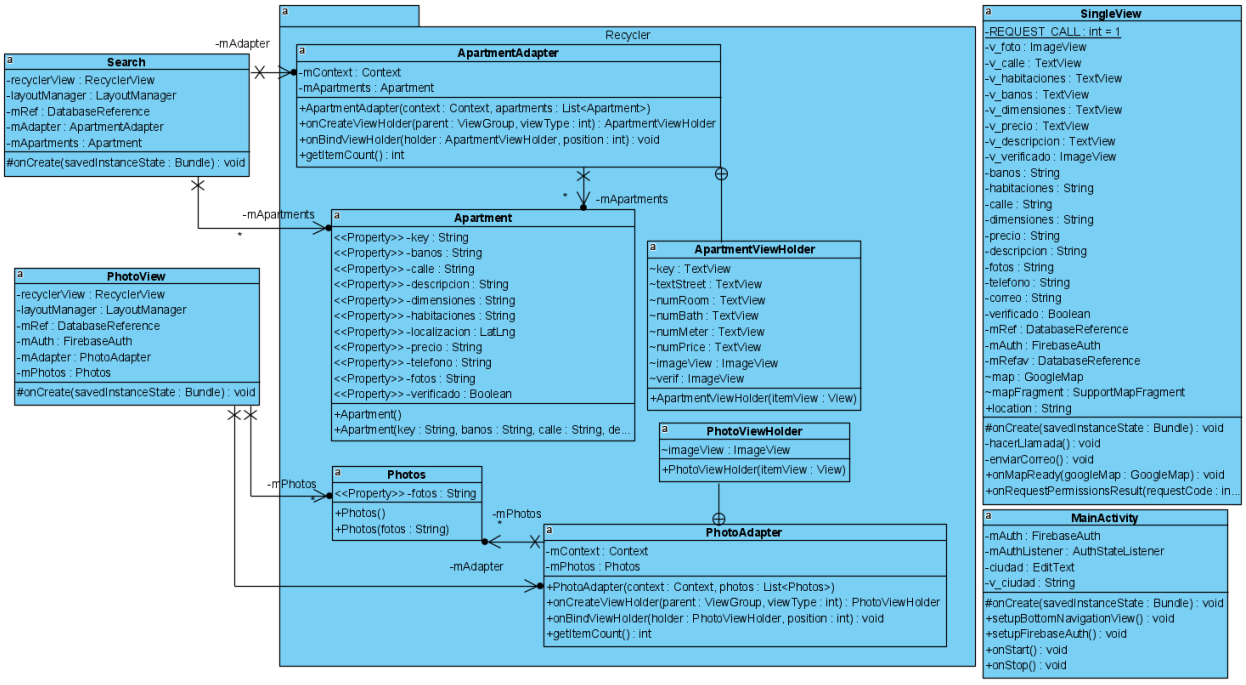

Figura 47. Diagrama de clases Buscar

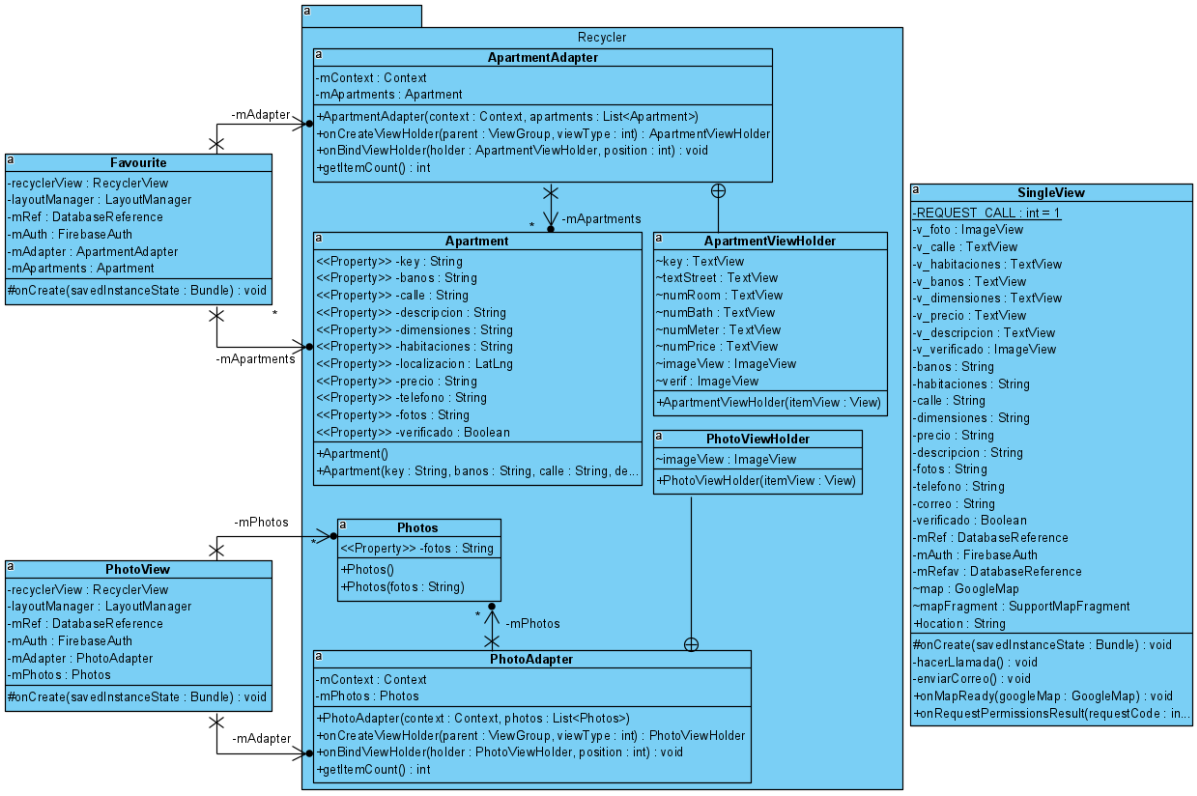

Figura 48. Diagrama de clases Favoritos

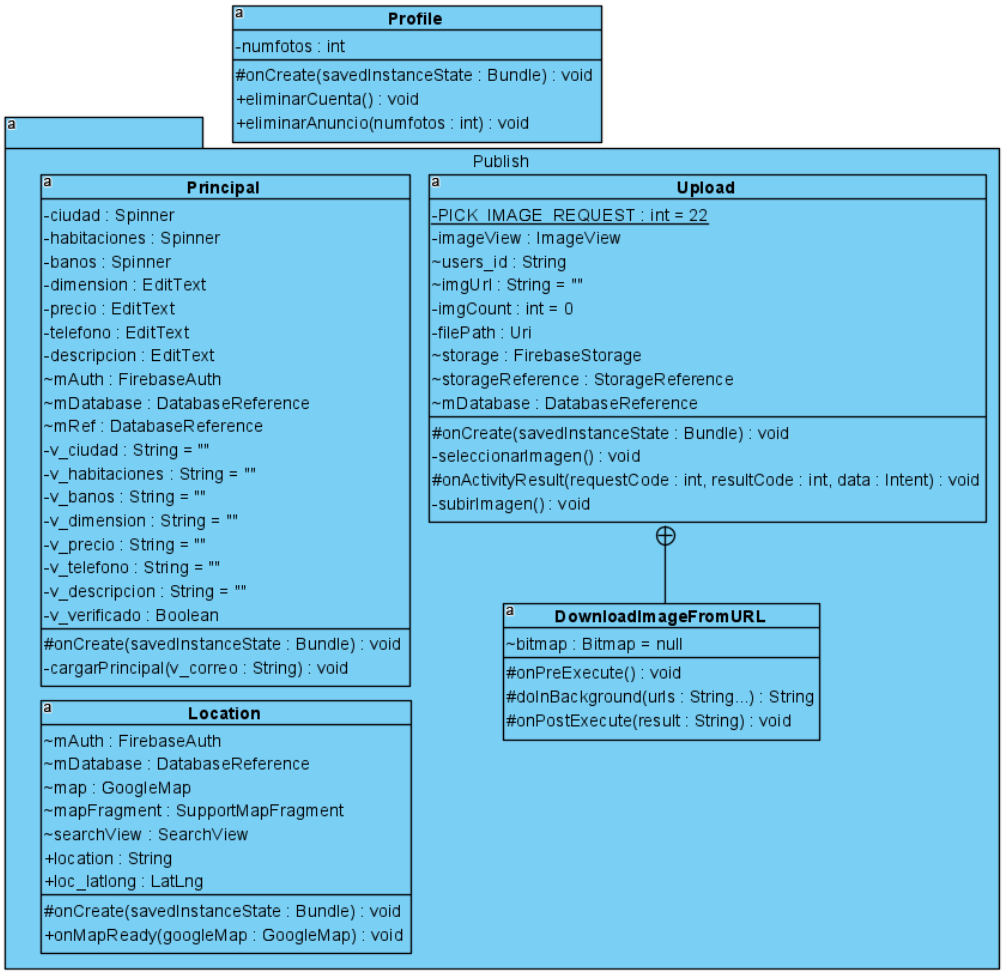

Figura 49. Diagrama de clases publicar

5 IMPLEMENTACIÓN

En este apartado, analizaremos la implementación del sistema correspondientes al desarrollo y la programación software del proyecto.

Dividiremos en dos partes: aspectos relacionados con Android Studio y aspectos relacionados con Firebase.

5.1 Android Studio

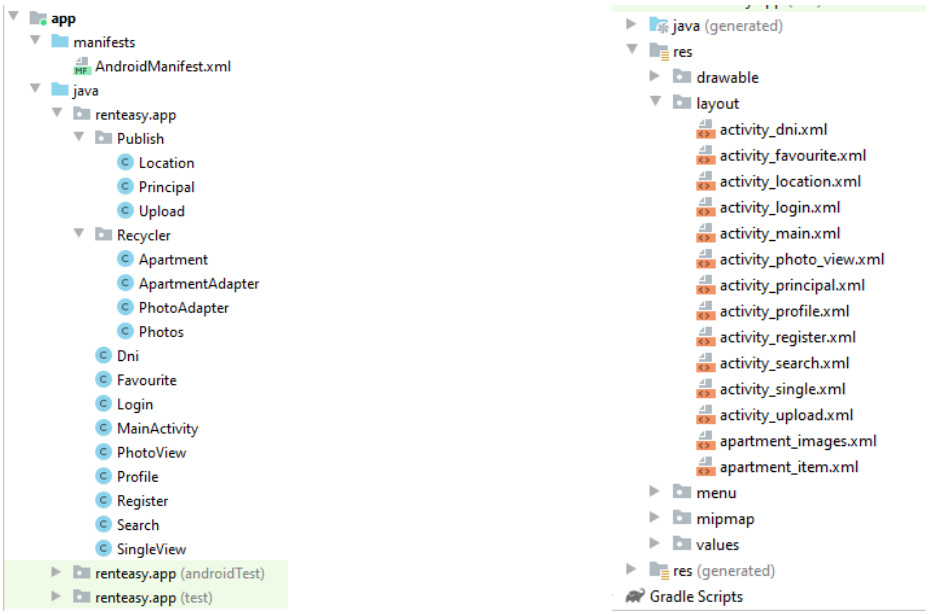

Figura 50. Estructura del Proyecto de Android Studio

Esta es la arquitectura de nuestro proyecto de Android Studio.

En la carpeta *java* se encuentran todos los archivos de código fuente Java, conformando la parte lógica de nuestro proyecto.

La carpeta *Recycler*, contiene las clases tipo *Adapter* y *Objetos* necesarias para la implementación de listas de ítems dinámicas, *recyclerview*.

La carpeta *Publish* agrupa las *activities* dedicadas a la subida de un anuncio a la plataforma.

Por otro lado, tenemos la carpeta *res* la cual contiene todos los recursos sin código, como los diseños XML, strings imágenes e iconos. Estos recursos son almacenados en diferentes subcarpetas dependiendo del tipo de recurso:

- *Drawable:* esta carpeta almacena aquellos recursos que son de tipo imagen y sus descriptores en XML.
- *Layout:* en esta carpeta se almacenan los ficheros XML que equivalen a las vistas de nuestra aplicación. Estas vistas permiten configurar diferentes pantallas que juntas forman las diferentes pantallas de interfaz de usuario.
- *Menu:* en esta carpeta se encuentra el archivo XML con el *Bottom Navigation Menu* que aparece en la parte inferior de algunas de las pantallas con acceso a los tres *activities* principales (*MainActivity*,

Favourite y *Profile*)

- *Mipmap:* en esta carpeta se ubica el icono de lanzamiento de nuestra aplicación. Similar a la carpeta *drawable,* pero en este caso las imágenes no son escalables.
- *Values:* en esta carpeta se encuentran los diferentes ficheros XML que indican diferentes valores usados en la aplicación. Encontramos cuatro ficheros: *colors.xml* en la que se definen los diferentes colores de la aplicación. *Strings.xml* en el que se definen los diferentes textos que se encuentran en nuestra aplicación. *Styles.xml* en el que se definen el estilo y tema de nuestra aplicación.

En la carpeta manifest, tenemos el documento AndroidManifest.xml el cual es un archivo de configuración donde aplicamos las configuraciones básicas de nuestro proyecto. En él se definen:

- El nombre de la aplicación, los paquetes Java, estilos de la aplicación, etc.
- Librerías incluidas
- Características hardware
- Versión mínima del API de Android necesaria para poder ser ejecutada, la versión de la aplicación, etc.
- Permisos que necesita la aplicación para su correcto funcionamiento, como son:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.CALL_PHONE" />
```

Código 1. Permisos de la aplicación

La carpeta Grandle Scripts, se almacenan determinados ficheros Grandle, los cuales permiten compilar y construir la aplicación.

5.2 Firebase

En este apartado, vamos a analizar los usos y como está configurado nuestra base de datos.

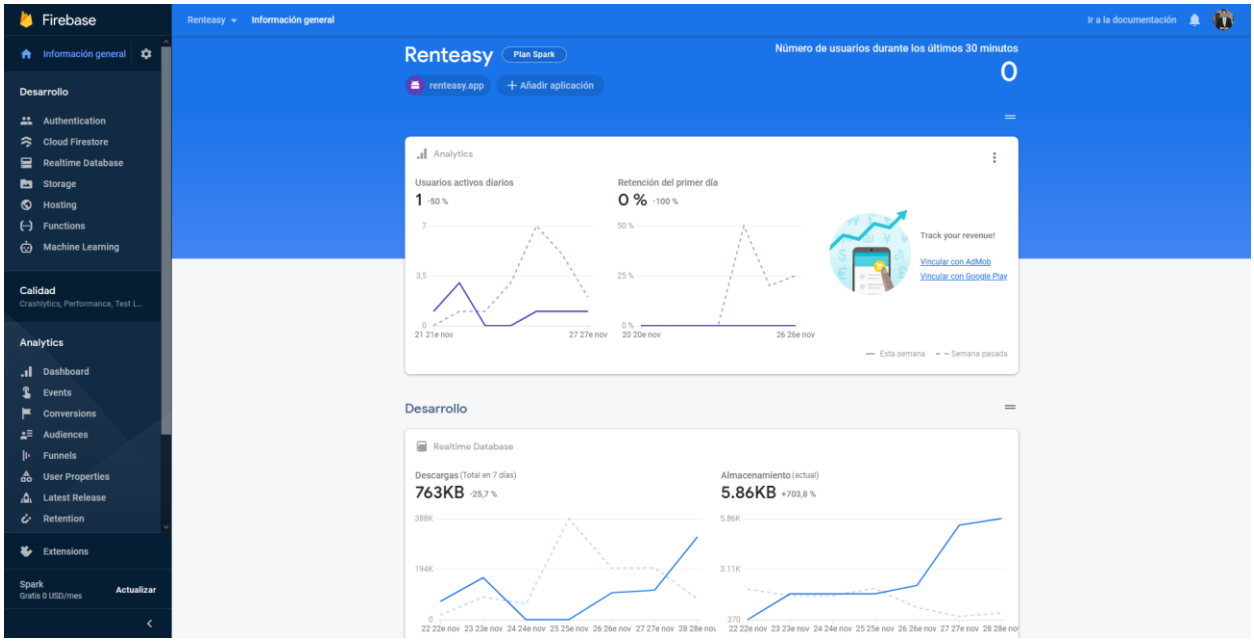

Figura 51. Pantalla principal Firebase

Como comentamos anteriormente, los productos usados en este proyecto han sido: Authentication, Realtime Database y Storage.

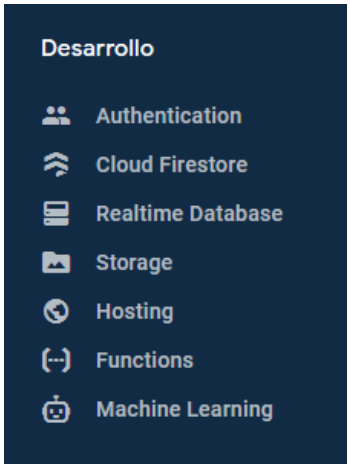

Figura 52. Herramientas de desarrollo Firebase

- Authentication

En authentication se almacenan las credenciales de los usuarios registrados en la aplicación.

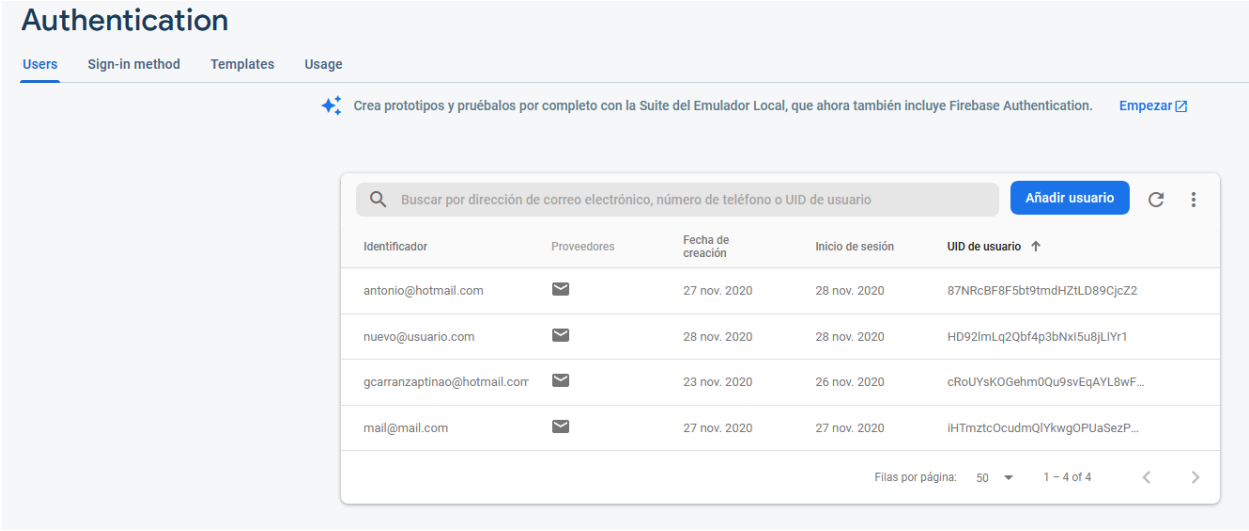

Figura 53. Vista Authentication Firebase

Aquí podemos comprobar los diferentes usuarios que se han registrado en nuestra aplicación y su ID de usuario asignado que utilizaremos como identificador.

Esta herramienta nos permite los diferentes métodos de creación de usuario.

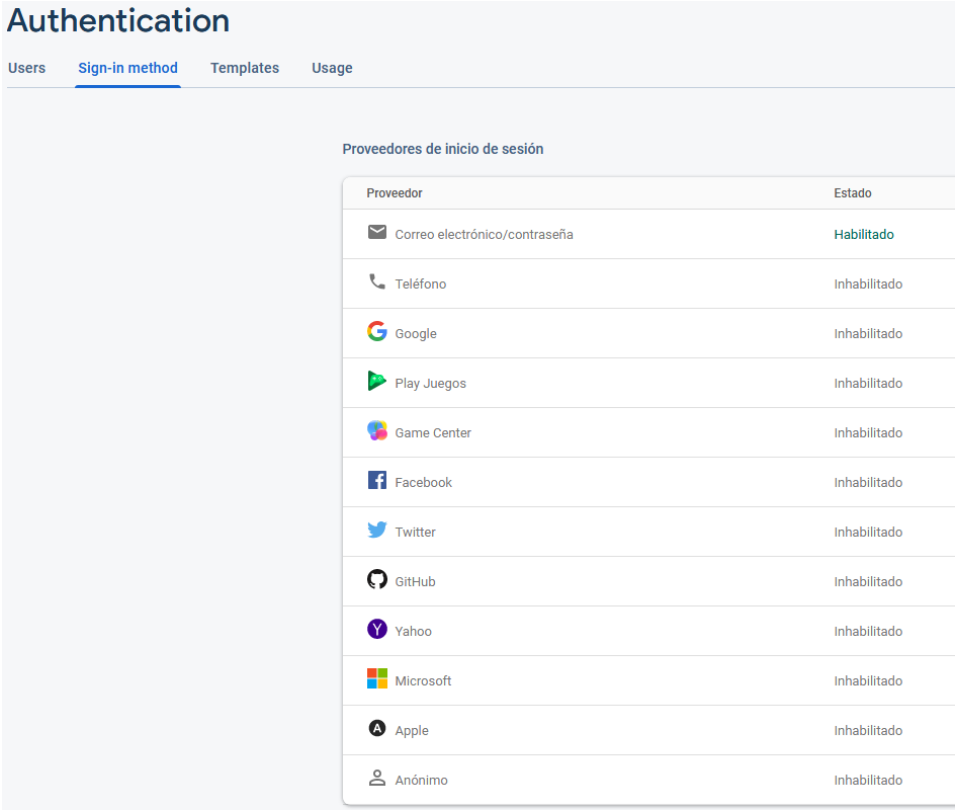

Figura 54. Métodos de registro de usuario Firebase

Para simplificar en este proyecto y evitar spam innecesario, hemos optado por únicamente habilitar la primera la cual pide un correo y una contraseña.

Otra ventaja de esta herramienta es la que hemos usado para el envío de un correo de verificación de correo electrónico el cual se lanza desde firebase con el método *sendEmailVerification()*.

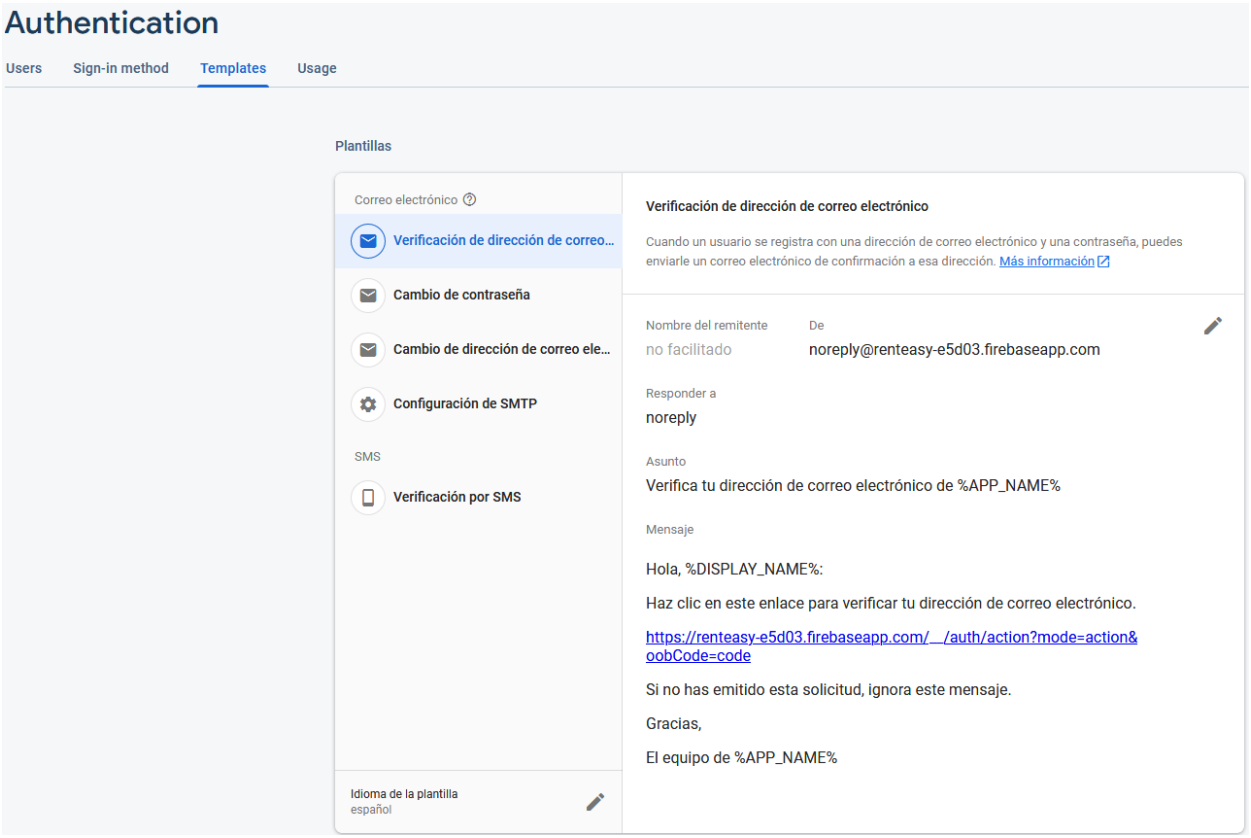

Figura 55. Plantilla email de verificación

- Realtime Database

En este apartado vamos a ver nuestra base de datos en tiempo real, back-end, la cual está organizada en forma de árbol JSON.

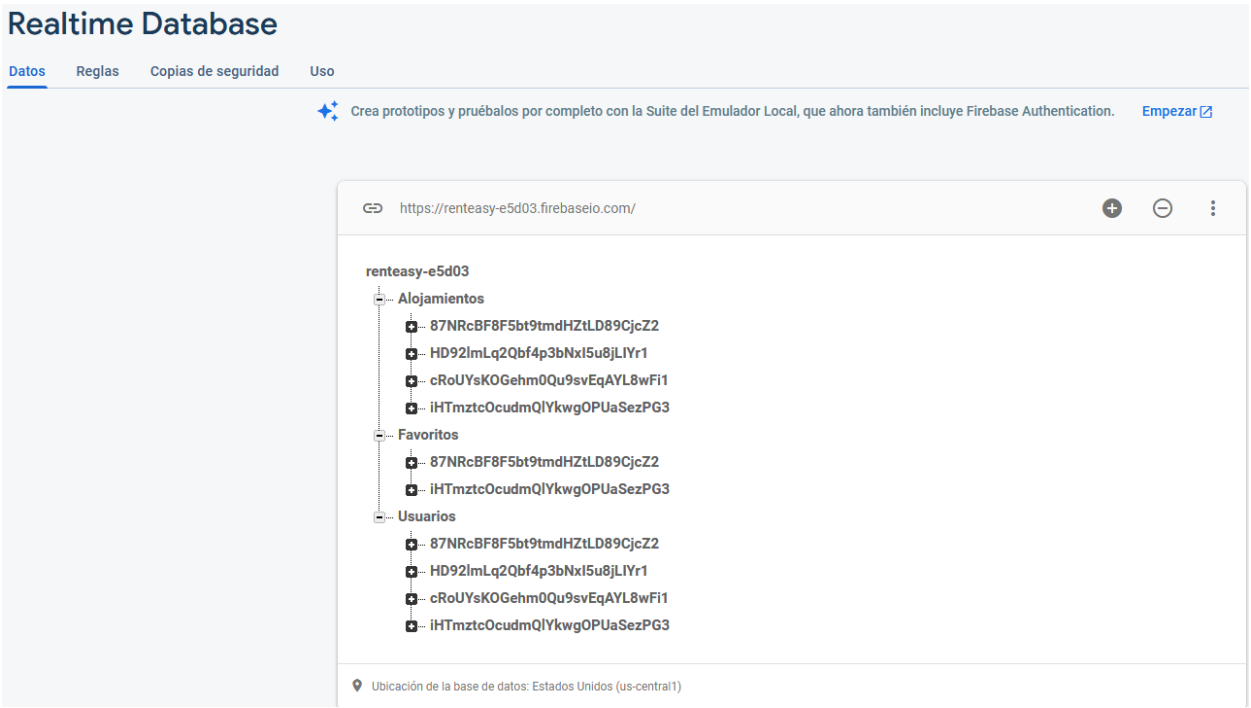

Figura 56. Realtime Database Firebase

Nuestra base de datos se estructura en tres grupos: Alojamientos, Favoritos y Usuarios.

- Usuarios: es donde almacenamos la información de cada usuario. Cada usuario se identifica por su ID dado al generar una cuenta en nuestra aplicación.

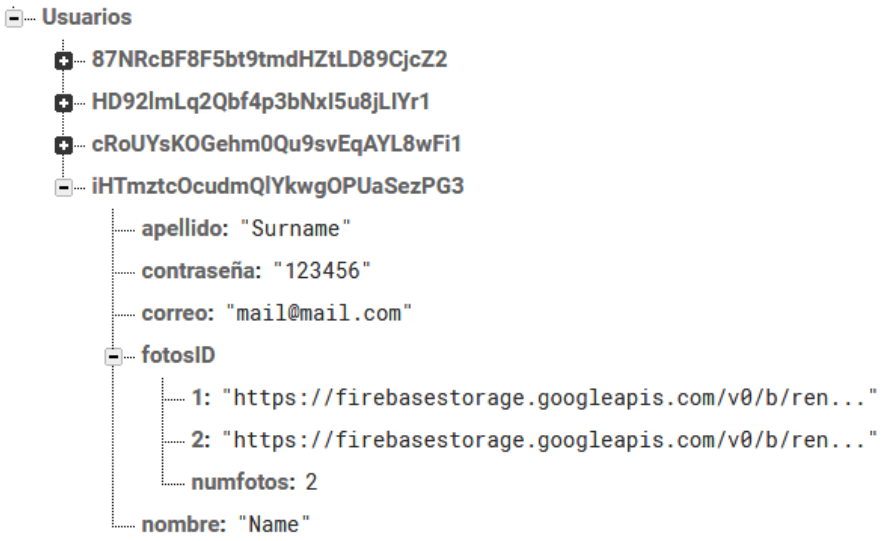

Figura 57. Realtime Database Usuarios

Como podemos observar, almacenamos la información que nos solicita la aplicación al registrar un nuevo usuario. El almacenamiento de las imágenes está en la herramienta de Storage que veremos a continuación. En Realtime Database lo que almacenamos es el enlace URL donde está almacenado la imagen en Storage

- Alojamientos: es donde almacenamos toda la información sobre los anuncios que cada usuario ha subido a la aplicación.

Figura 58. Realtime Database Alojamientos

Como podemos observar, aquí almacenamos la información que se nos mostrará de cada anuncio publicado.

Aquí destacamos uno de los puntos diferenciales de esta aplicación. Una vez se ha creado un anuncio, se crea el valor *Boolean* verificado. El administrador, en este caso el creador de la aplicación, cambiará este estado de falso a verdadero una vez haya podido verificar la veracidad del anuncio publicado. El usuario podrá distinguir que un anuncio ha sido verificado gracias al icono que aparece en cada anuncio:

Figura 59. Icono Verificado

- Storage

En este apartado vamos a ver cómo está organizado nuestro Cloud Storage.

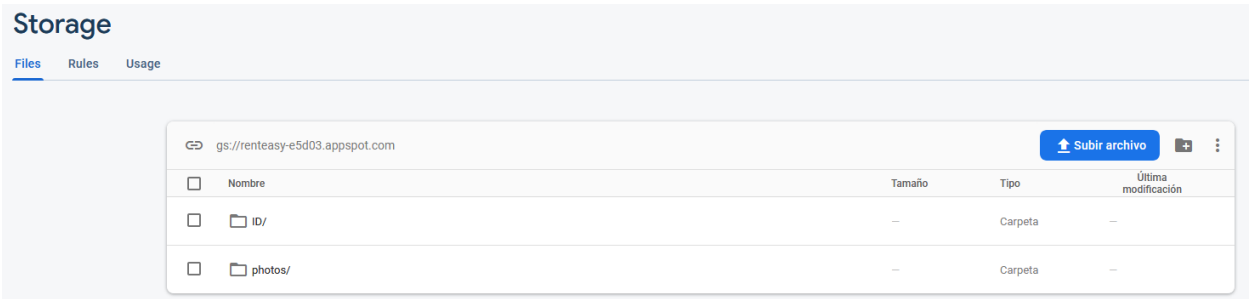

Figura 60. Storage Firebase

El almacenamiento de imágenes está repartido en dos subcarpetas: ID y potos.

- ID: aquí es donde almacenamos las fotos de DNI de cada usuario.

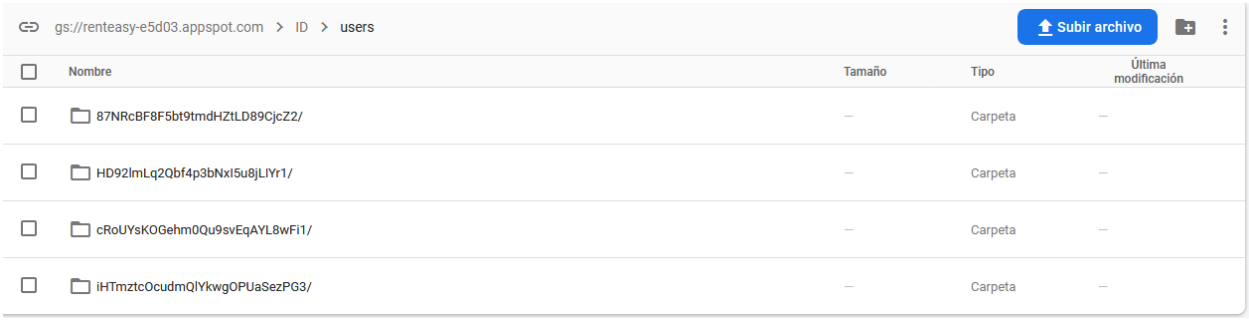

Figura 61. Storage ID

A su vez esta carpeta tiene una subcarpeta llamada *users* en la cual tendremos una carpeta por cada usuario. Los nombres de estas carpetas se identifican por el ID asignado a cada usuario.

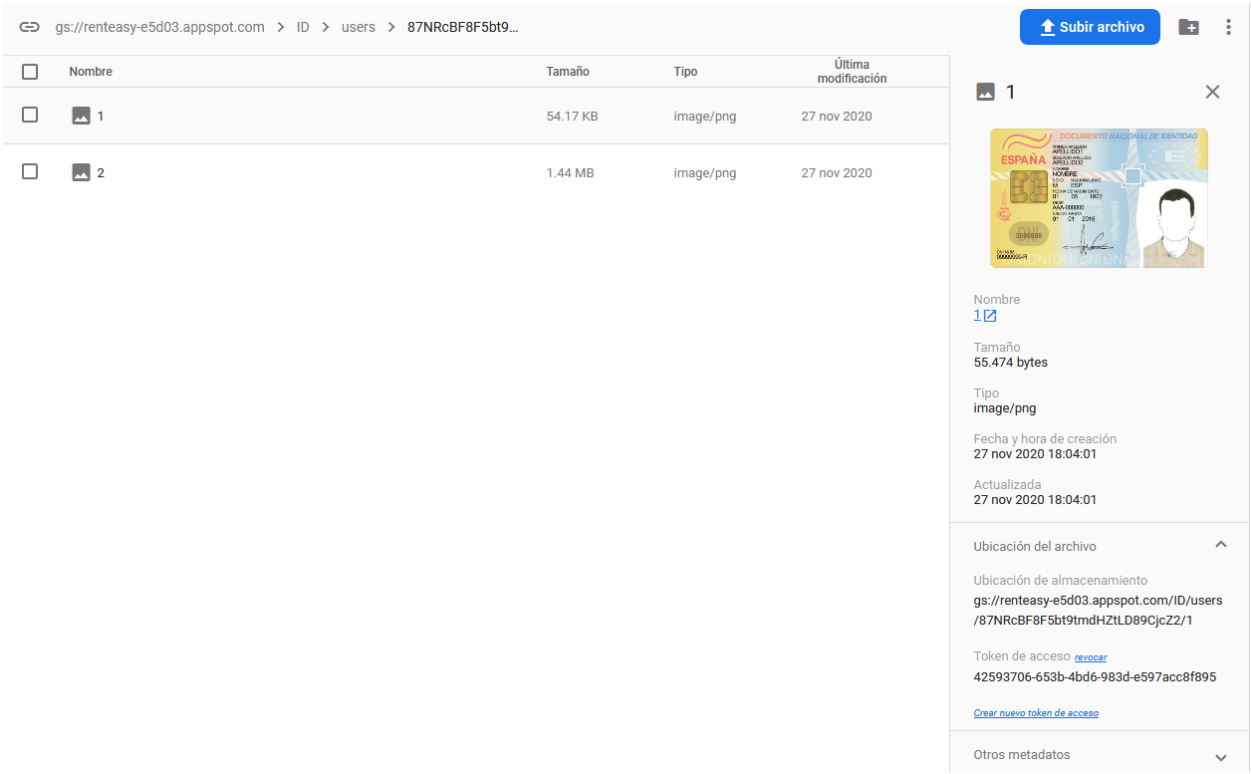

Figura 62. Imagen DNI en Storage

Aquí podemos comprobar las dos fotos que corresponden al usuario (frontal y reverso DNI). Abajo a la derecha se encuentra el enlace con el que se ubica esta foto dentro de firebase. Este es el enlace que usaremos en Realtime Database para acceder a esta imagen.

La subcarpeta photos se almacena de la misma manera que se almacenan las imágenes de DNI de los usuarios.

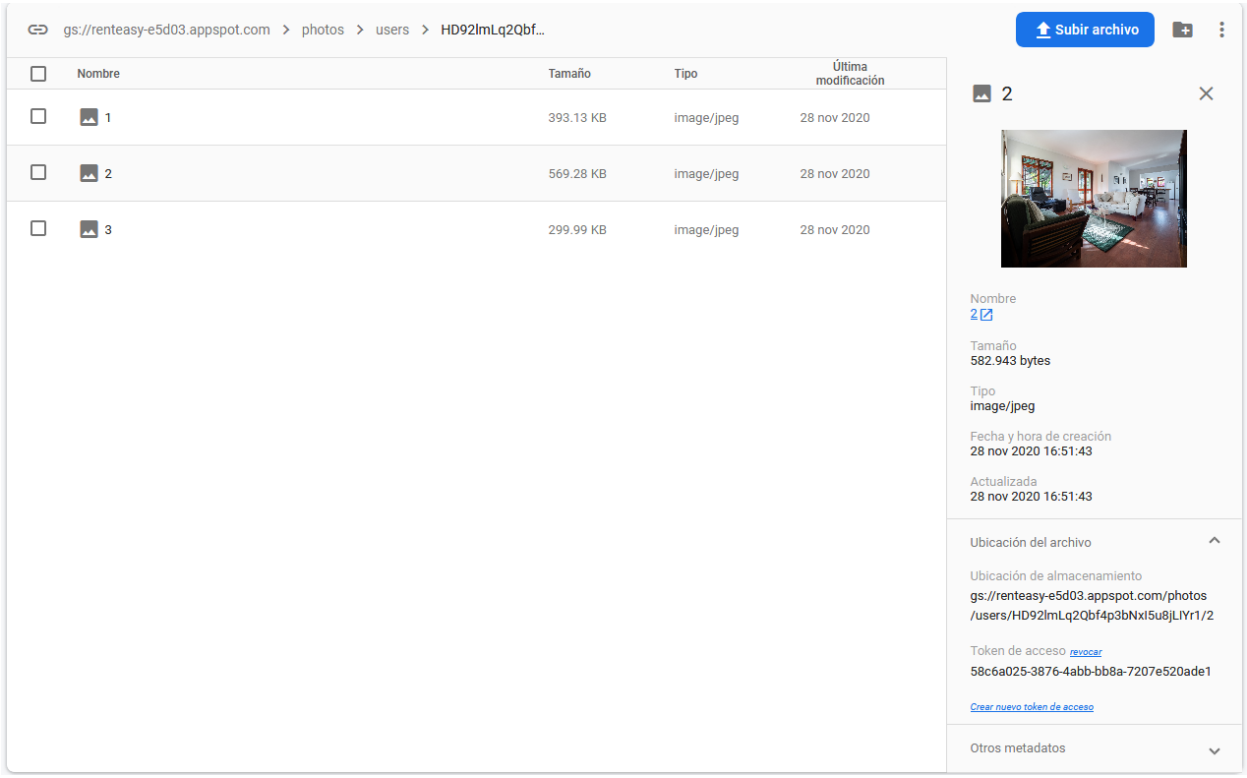

Figura 63. Imagen anuncio Storage

El nombre con el que se identifica cada imagen ha sido dado de esta manera para su fácil acceso desde Android Studio.

6 CONCLUSIONES

En este apartado, expondremos las conclusiones finales, las principales dificultades encontradas durante el proceso de creación de la aplicación. También indicaremos las posibles vías de mejora para futuro de nuestra aplicación.

6.1 Conclusiones

La realización de este trabajo fin de grado ha supuesto un gran reto para mí ya que la rama por la que opté durante el grado fue la electrónica. Desde hacía un tiempo se despertó en mi la curiosidad de saber cómo implementar una aplicación móvil en la que pudiese plasmar ideas que tenía y poderlas ver en un formato que está a diario con nosotros. El momento de elección de un trabajo estaba completamente perdido, a medida que pasaban los días y estaba con otro trabajo fin de grado, esta curiosidad me hizo decidirme y cambié de proyecto lanzándome a este mundo. Partiendo de absolutamente nada, empecé a verme un tutorial en Youtube de nombre *¨Curso de programación Android desde cero"* por el autor *La Geekipedia de Ernesto.* Esta lista de reproducción cuenta con 68 videos explicativos desde como instalar y configurar Android Studio en nuestro ordenador, hasta accesos a bases de datos y muchas más. Una vez entendí lo básico, comencé con mi aplicación *Renteasy.*

La realización del proyecto ha sido dura, ya que a medida que avanzaba, nuevos problemas aparecían. Con el uso de más y más métodos nuevos, esto requería de mucha documentación para saber la funcionalidad de cada uno y la forma en la que se tenía que implementar.

Uno de los puntos más complicados durante la creación del proyecto, fue sin lugar a duda todo lo relacionado con la base de datos de Firebase. Esta plataforma tiene una gran cantidad de servicios, de los que solo he usado unos pocos. Con cada servicio se trabaja de una determinada manera, con lo que para tener un buen dominio de cómo usarlo, he necesitado de leer mucha documentación (tanto oficial como no oficial), videos de ayuda en la red el foro de ayuda de Stackoverflow y muchas páginas más.

Para concluir, creo que he conseguido realizar la aplicación que tenía en mente al empezar. Me ha llevado bastante tiempo y esfuerzo, pero tengo la satisfacción de haber creado lo que quería y he aprendido como funciona este mundo.

6.2 Mejoras para futuro

A continuación, se exponen algunas de las posibles mejoras del proyecto que se podrían implementar a futuro en la aplicación:

- Una de las opciones que vemos en algunas de las aplicaciones con el mismo objetivo que la nuestra es que nos permiten navegar por la aplicación, pudiendo ver las ofertas de anuncios publicados en nuestra aplicación, sin la necesidad de requerir un usuario. Esto puede ser un punto de posible mejora en donde nos pidan un usuario en el momento en el que queramos contactar por un anuncio, publicar un anuncio o la lista de favoritos.
- Opciones de autenticación. En esta primera versión de la aplicación, solo hemos considerado la opción de registrarnos por medio de un correo electrónico y contraseña. En la actualidad y como está el mercado, sería razonable considerar otras formas de autenticación como puede ser por medio de Google o Facebook lo cual da una mayor facilidad al usuario.

- Añadir foto de DNI. Este punto lo hemos localizado en el momento de crear un usuario nuevo ya que queremos dar la seguridad tanto al arrendador como al arrendatario. Esto se podría cambiar y solicitarnos esta información en el momento en el que queramos crear un anuncio o contactar con la persona que ha publicado el anuncio.
- Aumentar el número de filtros para la búsqueda de anuncios. En este momento el filtro con el que cuenta nuestra aplicación es por ciudad. Una posible mejora es aumentar el número de filtros para acotar la búsqueda y así dar rápidamente con lo que queremos. En esta versión de la aplicación se ha acotado a tres ciudades, en un futuro sería más que razonable que esto no exista y que se puede publicar un anuncio para cualquier ciudad del mundo.
- Medios de contacto. Las opciones que tiene un usuario de contactar a la persona que ha publicado un anuncio es, o bien vía llamada telefónica, o vía email. Una posible mejora podría ser la incorporación de un chat dentro de la aplicación.
- Mayor número de anuncios por usuario. Hasta ahora se ha pensado que un usuario solo pueda subir un único anuncio. Una posible mejora es poder permitir que un usuario pueda publicar el número de anuncios que quiera.
- Aplicación web. Para dar un mayor acceso a usuarios, se podría implementar una aplicación hibrida en donde ofrecer los mismos servicios tanto en el navegador web como en la aplicación móvil
- Aplicación iOS. En relación con el punto anterior. Para poder ofrecer un servicio completo, se debería de desarrollar la aplicación en la plataforma de Apple.
- Idiomas. Para poder llegar a un mayor número de personas, sería interesante añadir más idiomas a parte del español
- Funcionalidades de la plataforma Firebase. La plataforma de Firebase nos ofrece muchas otras funcionalidades que no se han explorado. Algunas de estas funcionalidades son por ejemplo *Crashlytics* la cual prioriza y resuelve problemas de estabilidad. Performance, con la que obtener información valiosa sobre el rendimiento de la aplicación. *Resize Images* para comprimir de manera óptima las imágenes almacenadas en *Cloud Storage.*

6.3 Problemas encontrados

A continuación, vamos a describir algunos de los problemas encontrados durante la creación de nuestra aplicación.

- **Android Studio.** Al comenzar el desarrollo, una de las primeras dificultades encontradas fue el entorno de programación. Para ello y dado que nunca había programado ninguna aplicación, me ayudé de varios tutoriales en Youtube de uso de Android Studio y así poder familiarizarme con el entorno y con el desarrollo de aplicaciones Android.
- **Base de datos**. Uno de los primeros problemas encontrados fue a la hora de guardar información en la base de datos. En un primero momento se optó por *Sqlite* ya que es la base de datos que conocía. A medida que iba avanzando en la aplicación me di cuenta de que esta base de datos estaba muy limitada para la función que quería desempeñar. Fue entonces cuando, consultando por la red, encontré Firebase y todas las ventajas que este nos ofrecía al estar alojada en la nube de Google.
- **Firebase**. Al pasar de Sqlite a Firebase, ya que nunca había usado esta herramienta, me ayudé de la documentación oficial y de tutoriales en Youtube. Tras analizar su funcionamiento y realizar muchas pruebas, obtuve un cierto conocimiento que me permitió seguir con el desarrollo de la aplicación.
- **Manejo de imágenes**. Otro de los problemas encontrados fue en el manejo de imágenes. En un principio se iba a utilizar el método *getBitmap*, el cual Android Studio nos lo marcaba tachado y con el mensaje *deprecated,* ya que este método solo era válido para versiones API anteriores. La alternativa encontrada fue el uso de la librería externa *Picasso*. Esta herramienta sencilla de usar nos permite mostrar sin problemas imágenes en nuestra aplicación.

ANEXO 1: MANUAL DE INSTALACIÓN

Este anexo hace referencia a como se tiene que configurar y vincular la plataforma de Firebase y de Google Maps a nuestro proyecto de Android Studio.

1. Firebase

A continuación, se describen los pasos que hay que seguir para configurar en Android Studio la plataforma de Firebase:

1. En Firebase console, pulsamos en *añadir proyecto* y escribimos el nombre del proyecto, pulsamos en continuar. A continuación, elegiremos si queremos habilitar Google Analytics en este proyecto. Confirmamos y continuamos. Ahora se empezará a crear nuestro proyecto.
2. En Firebase console, pulsamos sobre el icono de Android para inicial el flujo de trabajo de configuración como se pude ver a continuación:

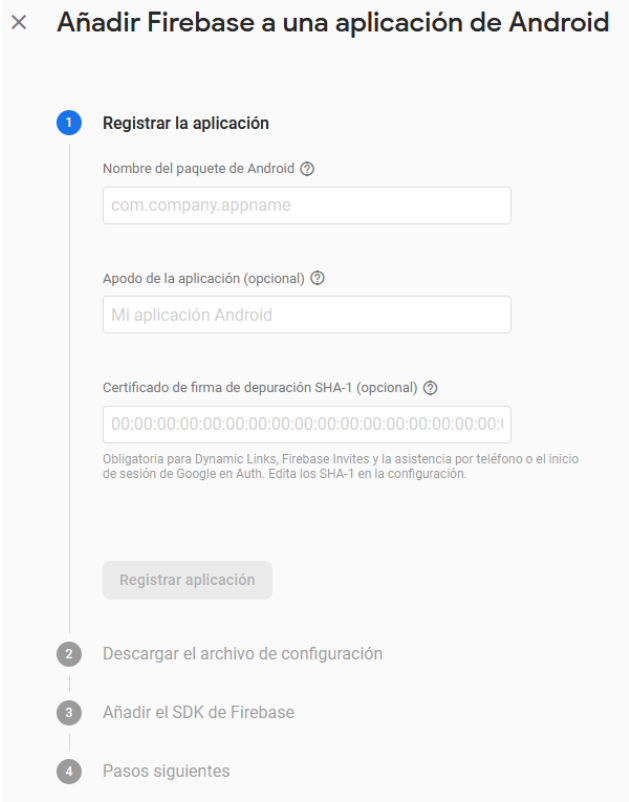

Figura 64. Registro Firebase

Rellenamos el nombre y picamos en Registrar aplicación.

3. Descargamos el archivo JSON que nos indican y lo ubicamos dentro de la carpeta app de nuestro proyecto.

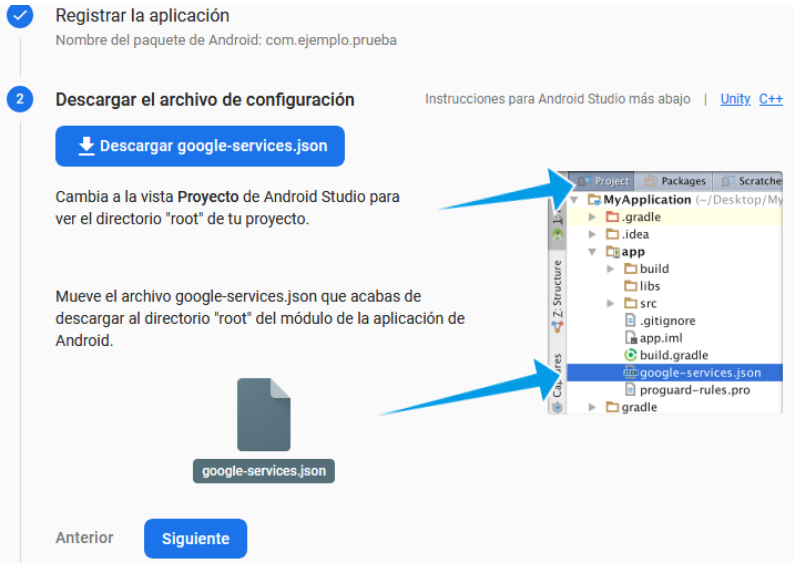

Figura 65. Archivo de configuración Firebase

4. Añadimos el SDK de Firestore añadiendo en Build.gradle lo que nos indican:

Build.gradle de nivel de proyecto (<proyecto>/build.gradle):

```
buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
  }
  dependencies {
    ...
    // Add this line
    classpath 'com.google.gms:google-services:4.3.4'
  }
}

allprojects {
  ...
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
    ...
  }
}
```

Java Kotlin

Build.gradle de nivel de aplicación (<proyecto>/<app-module>/build.gradle):

```
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:26.1.0')

  // Add the dependency for the Firebase SDK for Google Analytics
  // When using the BoM, don't specify versions in Firebase dependencies
  implementation 'com.google.firebase:firebase-analytics'

  // Add the dependencies for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

Código 2. Instalación de servicios

Una vez completados estos pasos, habremos vinculado con éxito Firebase con nuestro proyecto.

2. Firebase Authentication

Desde Firebase console, nos dirigimos a Authentication. Lo primero que debemos de hacer es habilitar los métodos de inicio de sesión con los que queremos que nuestra aplicación cuente:

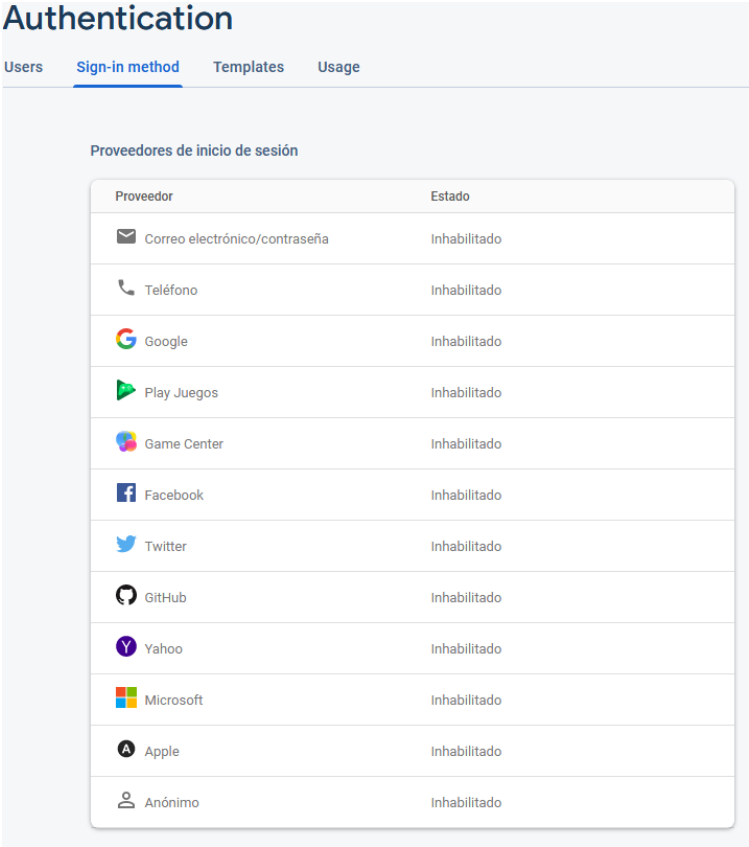

Figura 66. Proveedores de inicio de sesión Firebase

Para habilitar un proveedor de inicio de sesión, solo tendremos que seleccionar el que queramos, y habilitarlo:

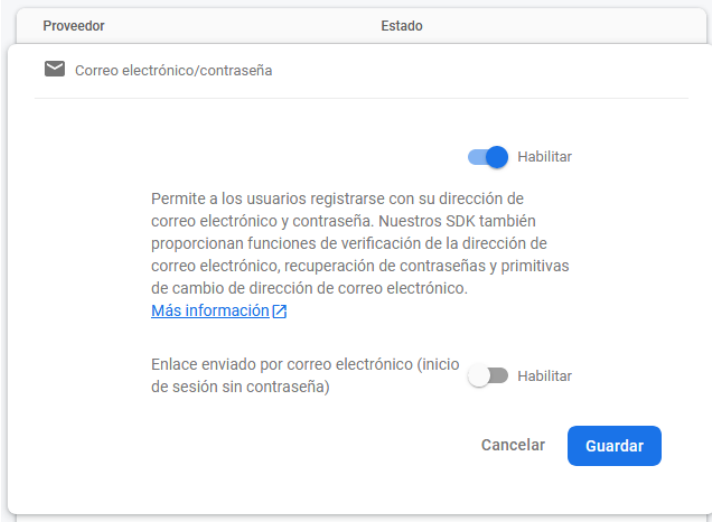

Figura 67. Habilitar proveedor de inicio de sesión

Este servicio de la plataforma de Firebase nos ofrece una serie de plantillas con las que podremos verificar y administrar nuestras cuentas por medio de enlaces que nos envían al correo electrónico o incluso vía SMS.

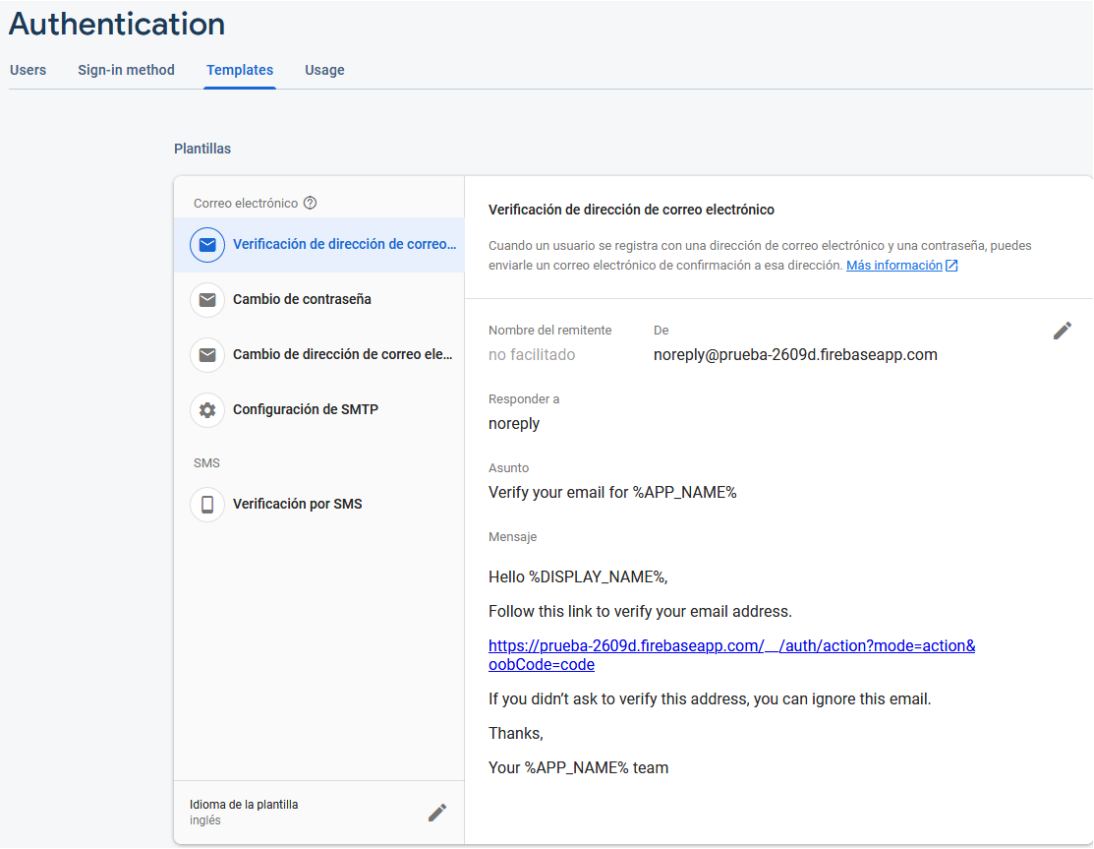

Figura 68. Plantillas Firebase Authentication

Para su implementación en Android Studio, en primer lugar, debemos añadir lo siguiente en nuestro archivo Gradle

```
dependencies {
    // Import the BoM for the Firebase platform
    implementation platform('com.google.firebase:firebase-bom:26.0.0')

    // Declare the dependency for the Firebase Authentication library
    // When using the BoM, you don't specify versions in Firebase library dependencies
    implementation 'com.google.firebase:firebase-auth'
}
```

Código 3. Implementación en Gradle para Authentication

Con las siguientes líneas de código se puede comprobar cuál es el estado de autenticación actual:

```
private FirebaseAuth mAuth;
// Initialize Firebase Auth
mAuth = FirebaseAuth.getInstance();
@Override
public void onStart() {
    super.onStart();
    // Check if user is signed in (non-null) and update UI accordingly.
    FirebaseUser currentUser = mAuth.getCurrentUser();
    updateUI(currentUser);
}
```

Código 4. Comprobar estado de autenticación actual

Para registrar un nuevo usuario, creamos un nuevo método *createAccount* que recibe una dirección de correo electrónico y una contraseña, las valida y, luego, crea un nuevo usuario con el método *createUserWithEmailAndPassword.*

```
mAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    // Sign in success, update UI with the signed-in user's information
                    Log.d(TAG, "createUserWithEmail:success");
                    FirebaseUser user = mAuth.getCurrentUser();
                    updateUI(user);
                } else {
                    // If sign in fails, display a message to the user.
                    Log.w(TAG, "createUserWithEmail:failure", task.getException());
                    Toast.makeText(EmailPasswordActivity.this, "Authentication failed.",
                            Toast.LENGTH_SHORT).show();
                    updateUI(null);
                }

                // ...
            }
        });
```

Código 5. Registro de nuevo usuario

3. Firebase Realtime Database

Lo primero que debemos de configurar en Realtime Database es el modo de inicio para las reglas de seguridad de Firebase.

Figura 69. Reglas Firebase Realtime Database

Hay dos modos, el modo de prueba y el modo bloqueado:

- Modo de prueba: Es el modo recomendado si recién comienzas a usar las bibliotecas cliente para dispositivos móviles y la Web, pero permite que todos lean y reemplacen tus datos.
- Modo bloqueado: Rechaza todas las lecturas y escrituras de clientes móviles y web. Tus servidores de aplicaciones autenticados aún pueden acceder a tu base de datos.

La opción de Copias de seguridad solo está disponible en la versión de pago.

La herramienta nos da la opción de monitorizar el uso de Realtime Database.

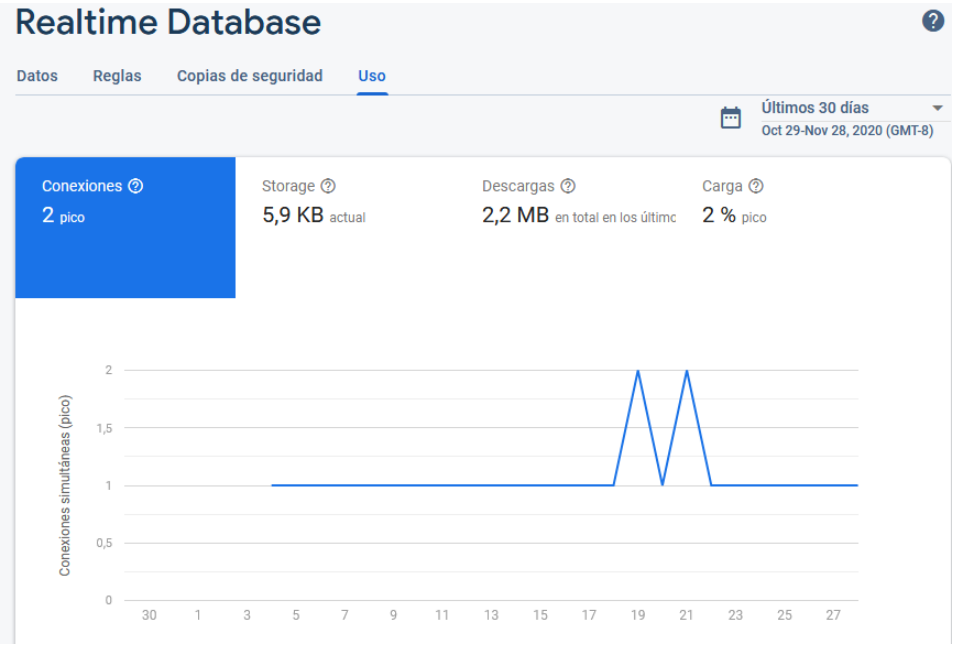

Figura 70. Usos de Realtime Database

Desde la pestaña de datos, tenemos acceso directo a los datos almacenados. Desde aquí, el administrador podrá ver y validar puntos, los cuales se verán reflejados en el momento en el que los usuarios naveguen por la aplicación.

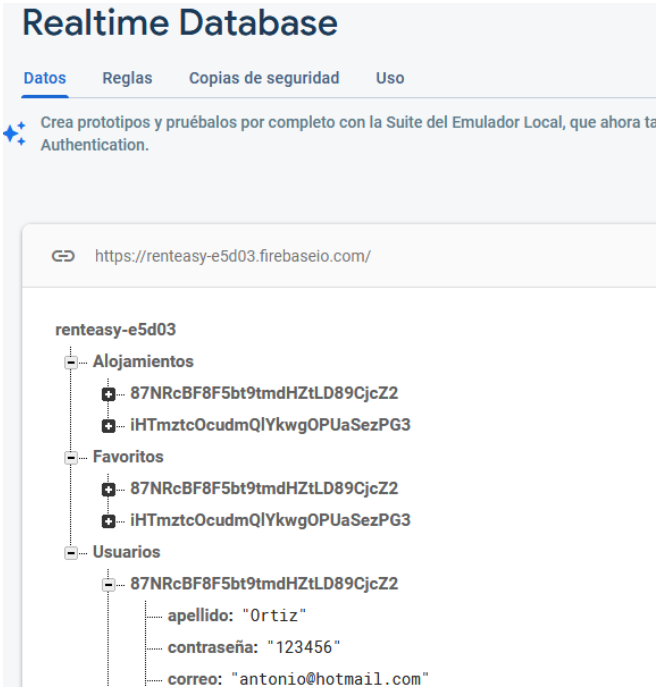

Figura 71. Interfaz de base de datos Realtime Database

Realtime Database nos brinda la opción de importar y exportar archivos JSON, descargando o sobrescribiendo los datos actuales de nuestra base de datos.

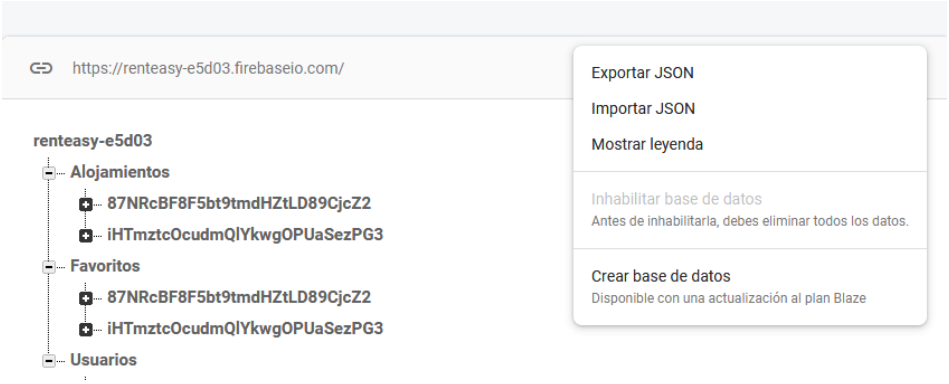

Figura 72. Opciones de datos Realtime Database

Podemos añadir nuevas instancias de Realtime Database, pero esta opción no está permitida con el plan gratuito que nos ofrecen.

Para su implementación en Android Studio, en primer lugar, debemos añadir lo siguiente en nuestro archivo Gradle

```
dependencies {
    // Import the BoM for the Firebase platform
    implementation platform('com.google.firebase:firebase-bom:26.0.0')

    // Declare the dependency for the Realtime Database library
    // When using the BoM, you don't specify versions in Firebase library dependencies
    implementation 'com.google.firebase:firebase-database'
}
```

Código 6. Implementación en Gradle para Realtime Database

A continuación, se emplean diferentes instancias para realizar operaciones de escritura en la base de datos:

```
mDatabase.child("Usuarios").child(id).setValue(map).
        addOnCompleteListener(new OnCompleteListener<Void>() {
    @Override
    public void onComplete(@NonNull Task<Void> task2) {
        if (task2.isSuccessful()){
            user.sendEmailVerification()
            Intent i = new Intent( packageContext: Register.this, Dni.class);
            startActivity(i);
        }
```

Código 7. Ejemplo de escritura en Realtime Database

De esta manera, se pueden guardar diversos tipos de datos en la base de datos, incluso objetos Java. Cuando guardas un objeto, las respuestas de cualquier método de obtención se guardan como elementos secundarios de esta ubicación.

Para la lectura de datos de la base de datos y que los datos de tu app se actualicen en tiempo real, agregamos un *ValueEventListener* a la referencia que acabamos de crear.

El método *onDataChange*() de esta clase se activa cuando se adjuntamos el objeto de escucha y cada vez que cambian los datos, incluidos los secundarios.

```
mRef = FirebaseDatabase.getInstance().getReference( path: "Alojamientos");

mRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        for (DataSnapshot apartmentSnapshot : dataSnapshot.getChildren()){
            String key = apartmentSnapshot.getKey();
            String ciudad_int = getIntent().getStringExtra( name: "ciudad");
            String ciudad = apartmentSnapshot.child("ciudad").getValue(String.class);
```

Código 8. Ejemplo de lectura de Realtime Database

4. Firebase Cloud Storage

Al igual que con Realtime Database, lo primero que deberemos de configurar son las reglas de seguridad de Firebase.

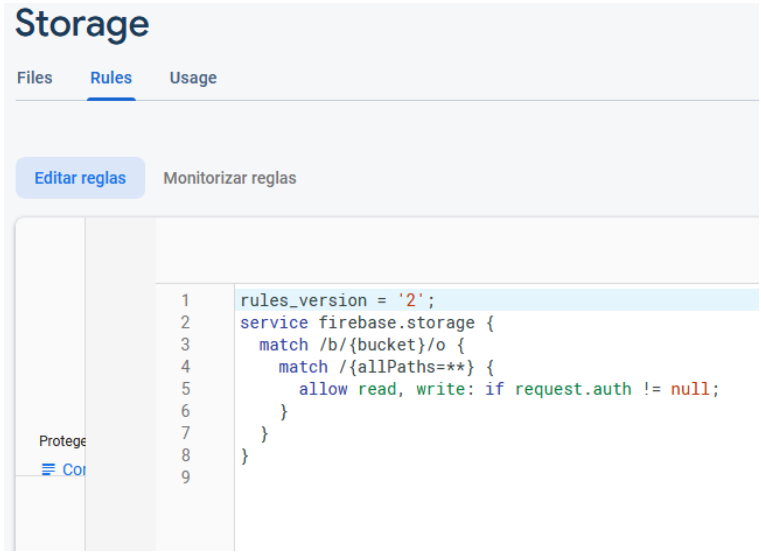

Figura 73. Reglas de seguridad de Storage

Al igual que con Realtime Database, también podemos monitorizar el uso de esta herramienta.

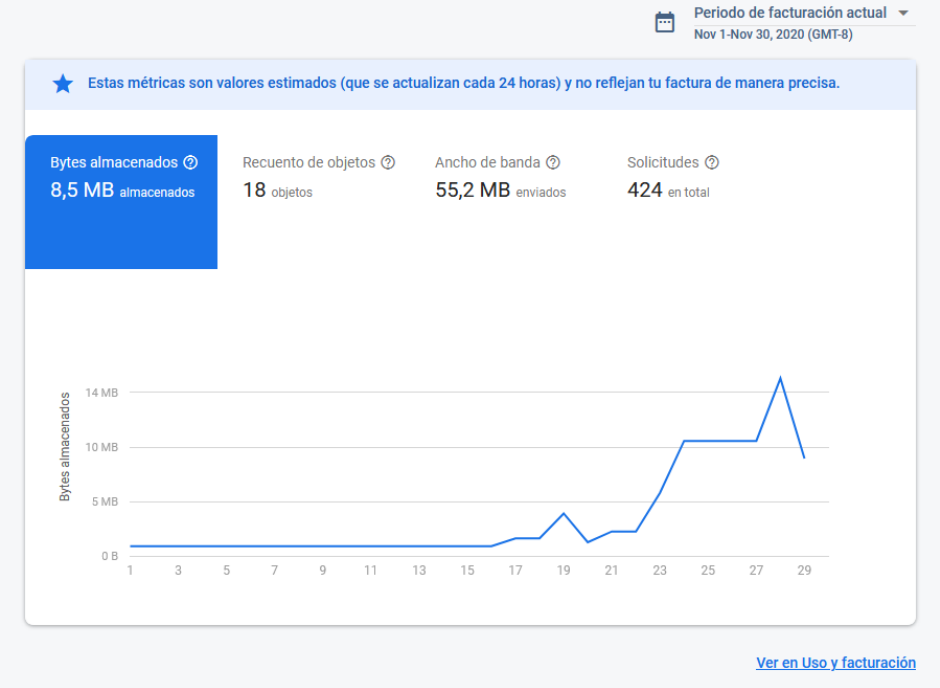

Figura 74. Uso de Cloud Storage

Desde la pestaña de files, tenemos acceso a los datos almacenados. Desde aquí, el administrador podrá ver y eliminar archivos, ver las direcciones URL donde se encuentran alojados los archivos y subir nuevos archivos.

Se puede organizar subdirectorios al directorio raíz como podemos observar:

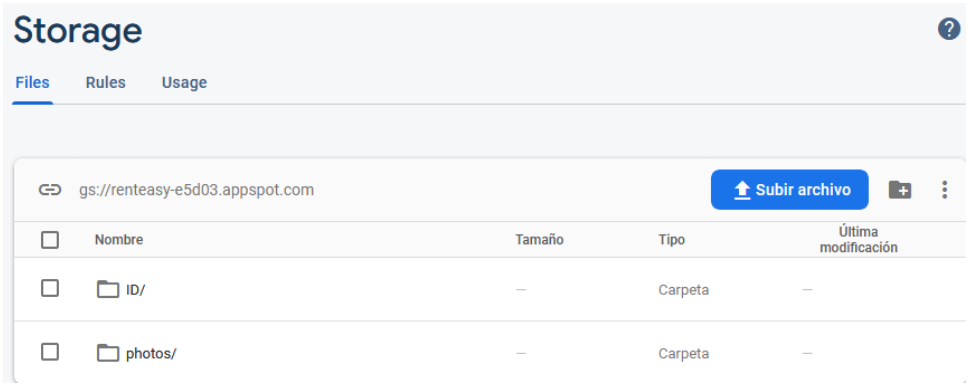

Figura 75. Organización en Storage

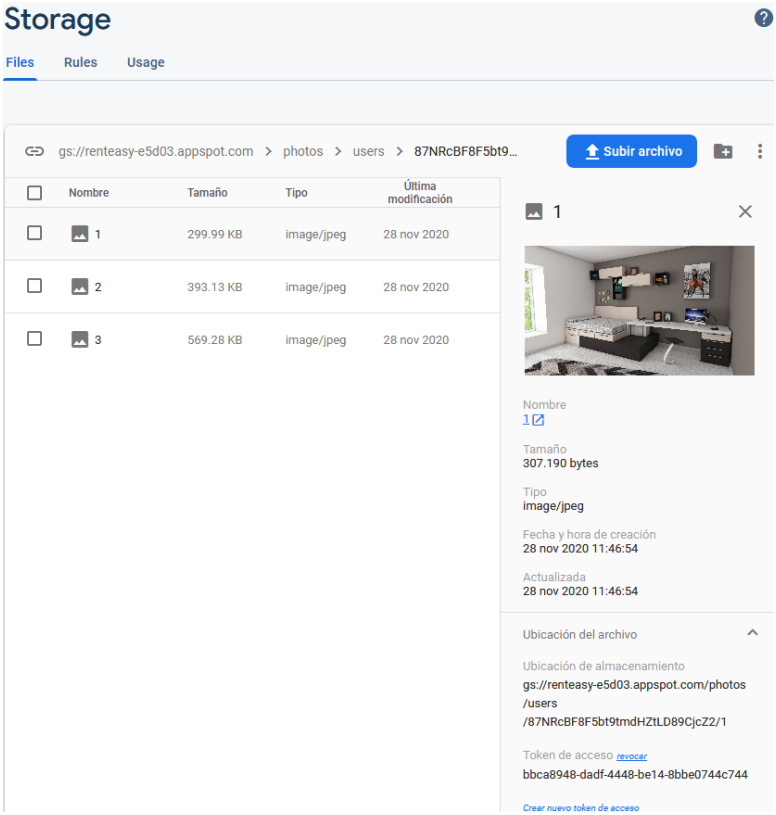

Figura 76. Datos en Storage.

Para su implementación en Android Studio, en primer lugar, debemos añadir lo siguiente en nuestro archivo Gradle

```
dependencies {
    // Import the BoM for the Firebase platform
    implementation platform('com.google.firebase:firebase-bom:26.0.0')

    // Declare the dependency for the Cloud Storage library
    // When using the BoM, you don't specify versions in Firebase library dependencies
    implementation 'com.google.firebase:firebase-storage'
}
```

Código 9. Implementación en Gradle para Cloud Storage

A continuación, se emplean diferentes instancias para realizar operaciones subida de archivos a Cloud Storage:

```
if (filePath != null) {
    final StorageReference ref = storageReference.
            child("photos/users/" + "/" + users_id + "/" + (imgCount + 1));

    UploadTask uploadTask = ref.putFile(filePath);
    Task<Uri> uriTask = uploadTask.continueWithTask((task) -> {
            if (!task.isSuccessful()) {
                throw task.getException();
            }
            return ref.getDownloadUrl();
    }).addOnCompleteListener((task) -> {
            if (task.isSuccessful()) {
                Uri downloadUri = task.getResult();
                filePath = null;
                imgUrl = downloadUri.toString();
                String imgName = ref.getName();

                Map<String, Object> update = new HashMap<>();
                update.put(imgName, imgUrl);
                update.put("numfotos", imgCount);

                mDatabase.child("Alojamientos").child(users_id).child("fotos")
                        .updateChildren(update);
            } else {
                Toast.makeText( context: Upload.this,  text: "Error al subir imagen",
                        Toast.LENGTH_SHORT).show();
            }

            DownloadImageFromURL downloadImageFromURL = new DownloadImageFromURL();
            downloadImageFromURL.execute("");
    }).addOnFailureListener((e) -> {
            Toast.makeText( context: Upload.this,  text: "Error" + e.getMessage(),
                    Toast.LENGTH_SHORT).show();
    });
}
```

Código 10. Subir archivo a Cloud Storage

Observamos que, aparte de añadir la imagen a Storage, también guardamos el enlace URL de esta imagen en Realtime Databas, para obtener de manera rápida esta imagen.

5. Google Maps Platform

Lo primero que deberemos de hacer es añadir los servicios de Google Play a nuestro proyecto en Android Studio:

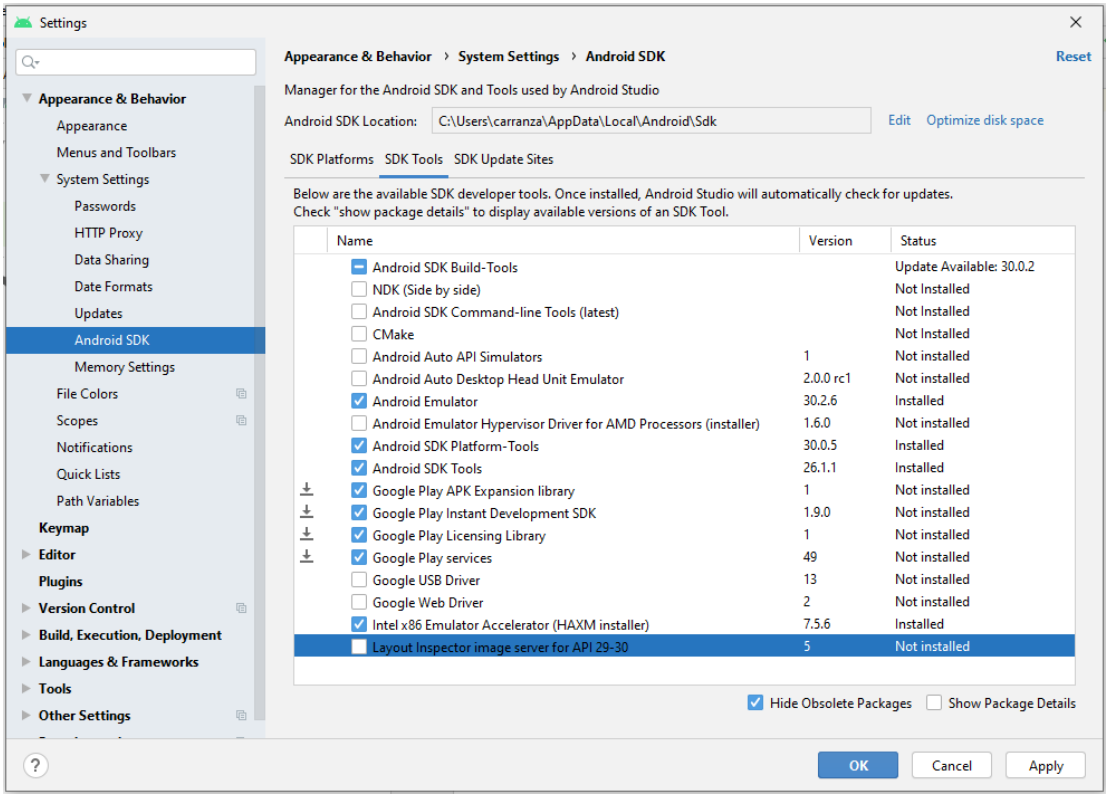

Figura 77. Añadir servicios de Google Play

Lo siguiente para su implementación en Android Studio es añadir lo siguiente en nuestro archivo Gradle

```
implementation 'com.google.android.gms:play-services-maps:17.0.0'
```

Código 11. Implementación en Gradle para Google Maps Platform.

Para implementar la vista de google maps, debemos añadir el siguiente *fragment* en la vista layout:

```
<fragment
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/google_map"
    android:name="com.google.android.gms.maps.SupportMapFragment" />
```

Código 12. Fragment en vista de mapa en layout

En nuestro caso, a través de un Searchview obtenemos el nombre de la calle que queremos marcar en el mapa.

```
//Busqueda de la direccion escrita en el buscador, y muestra la ubicacion en el mapa
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                location = searchView.getQuery().toString();
                List<Address> addressList = null;

                if (location != null || !location.equals("")) {
                    Geocoder geocoder = new Geocoder( context: Location.this);
                    try {
                        addressList = geocoder.getFromLocationName(location,  maxResults: 1);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    Address address = addressList.get(0);
                    LatLng latLng = new LatLng(address.getLatitude(), address.getLongitude());
                    map.addMarker(new MarkerOptions().position(latLng).title(location));
                    map.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng,  v: 18));
                    loc_latlong = latLng;
                }

                return false;
            }

            @Override
            public boolean onQueryTextChange(String newText) { return false; }
        });

        mapFragment.getMapAsync( onMapReadyCallback: this);
```

Código 13. Marcar un punto en el mapa

REFERENCIAS

[1] *Cómo mostrar la dirección de una ubicación*. (n.d.). Retrieved May 7, 2020, from https://developer.android.com/training/location/display-address?hl=es

[2] *Address | Desarrolladores de Android | Android Developers*. (n.d.). Retrieved May 7, 2020, from https://developer.android.com/reference/android/location/Address

[3] *Guarda datos | Firebase Realtime Database*. (n.d.). Retrieved May 11, 2020, from https://firebase.google.com/docs/database/admin/save-data?hl=es#section-update

[3] *How to Upload Images to Firebase from an Android App*. (n.d.). Retrieved May 13, 2020, from https://code.tutsplus.com/tutorials/image-upload-to-firebase-in-android-application--cms-29934

[4] *Easily Adding Nested Recycler View in Android - AndroidPub*. (n.d.). Retrieved May 26, 2020, from https://android.jlelse.eu/easily-adding-nested-recycler-view-in-android-a7e9f7f04047

[5] *Implement a Nested Recycler view Android — Steemit*. (n.d.). Retrieved May 26, 2020, from https://steemit.com/utopian-io/@ideba/implement-a-nested-recycler-view-android

[6] *Autentica con Firebase mediante un vínculo de correo electrónico en Android*. (n.d.). Retrieved November 19, 2020, from https://firebase.google.com/docs/auth/android/email-link-auth?hl=es

[7] *java - Displaying values in recyclerview (based on boolean value) returns same result for all items - Stack Overflow*. (n.d.). Retrieved November 23, 2020, from https://stackoverflow.com/questions/49230420/displaying-values-in-recyclerview-based-on-boolean-value-returns-same-result-f

[8] *Introducción a Android Studio | Desarrolladores de Android*. (n.d.). Retrieved November 23, 2020, from https://developer.android.com/studio/intro?hl=es

[9] *Firebase - Wikipedia, la enciclopedia libre*. (n.d.). Retrieved November 23, 2020, from https://es.wikipedia.org/wiki/Firebase

[10] *Programación en Java/Características del lenguaje - Wikilibros*. (n.d.). Retrieved November 23, 2020, from https://es.wikibooks.org/wiki/Programación_en_Java/Características_del_lenguaje

[11] *Introduction - StarUML documentation*. (n.d.). Retrieved November 23, 2020, from https://docs.staruml.io/

[12] *Herramientas case Star UML*. (n.d.). Retrieved November 23, 2020, from https://sites.google.com/site/herramientascasestaruml/

[13] *diagrams.net - Google Workspace Marketplace*. (n.d.). Retrieved November 24, 2020, from https://workspace.google.com/marketplace/app/diagramsnet/671128082532

[14] *Mobile app backend services | Solutions | Google Cloud*. (n.d.). Retrieved November 25, 2020, from https://cloud.google.com/solutions/mobile/mobile-app-backend-services#firebase-appengine-standard

[15] *Creación de componente Autenticación y Registro con Angular & Firebase # 2 | by Sergio Mesa | Medium*. (n.d.). Retrieved November 25, 2020, from https://medium.com/@mesasergio/creación-de-componente-autenticación-y-registro-con-angular-firebase-2-92c8926fd09d

[16] *Primeros pasos en la Web | Firebase*. (n.d.). Retrieved November 25, 2020, from https://firebase.google.com/docs/storage/web/start?hl=es

[17] *Google Maps Platform: Coloca tu empresa en el mapa*. (n.d.). Retrieved November 25, 2020, from https://maplink.global/es/blog/que-es-google-maps-platform/

[18] *In App Phone Call - Coding in Flow*. (n.d.). Retrieved November 26, 2020, from https://codinginflow.com/tutorials/android/in-app-phone-call

[19] *clipboard - How to Copy Text to Clip Board in Android? - Stack Overflow*. (n.d.). Retrieved November 26, 2020, from https://stackoverflow.com/questions/19253786/how-to-copy-text-to-clip-board-in-android

[20] *Diagrama de clases - Wikipedia, la enciclopedia libre*. (n.d.). Retrieved November 29, 2020, from https://es.wikipedia.org/wiki/Diagrama_de_clases

[21] *Picasso*. (n.d.). Retrieved November 29, 2020, from https://square.github.io/picasso/

[22] *Primeros pasos con Firebase Authentication en Android*. (n.d.). Retrieved November 30, 2020, from https://firebase.google.com/docs/auth/android/start?hl=es

[23] *Instalación y configuración en Android | Firebase Realtime Database*. (n.d.). Retrieved November 30, 2020, from https://firebase.google.com/docs/database/android/start?hl=es

[24] *Comienza a usar Cloud Storage en Android | Firebase*. (n.d.). Retrieved November 30, 2020, from https://firebase.google.com/docs/storage/android/start?hl=es

[25] *Get Started | Maps SDK for Android | Google Developers*. (n.d.). Retrieved November 30, 2020, from https://developers.google.com/maps/documentation/android-sdk/start?hl=es

[26] *Cómo conocer la ubicación más reciente | Desarrolladores de Android*. (n.d.). Retrieved December 6, 2020, from https://developer.android.com/training/location/retrieve-current?hl=es

[27] *Bottom navigation - Material Design*. (n.d.). Retrieved December 6, 2020, from https://material.io/develop/android/components/bottom-navigation

[28] *BottomNavigationView | Desarrolladores de Android | Android Developers*. (n.d.). Retrieved December 6, 2020, from https://developer.android.com/reference/android/support/design/widget/BottomNavigationView.html

[29] *Cómo programar una barra de navegación inferior para una aplicación de Android*. (n.d.). Retrieved December 6, 2020, from https://code.tutsplus.com/es/tutorials/how-to-code-a-bottom-navigation-bar-for-an-android-app--cms-30305

[30] *Salón Sofá Diseño De Interiores - Foto gratis en Pixabay*. (n.d.). Retrieved December 6, 2020, from https://pixabay.com/es/photos/salón-sofá-diseño-de-interiores-2569325/