Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Aplicación Android para compartir coche basada en la tecnología Firebase

Autor: Sergio Sucino Barrena

Tutor: Francisco José Fernández Jiménez

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

Sevilla, 2019

Trabajo de Fin de Grado
Ingeniería de Telecomunicación

Aplicación Android para compartir coche basada en la tecnología Firebase

Autor:
Sergio Sucino Barrena

Tutor:
Francisco José Fernández Jiménez

Dpto. de Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Trabajo de Fin de Grado: Aplicación Android para compartir coche basada en la tecnología Firebase

Autor: Sergio Sucino Barrena

Tutor: Francisco José Fernández Jiménez

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

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mi familia

A mis maestros y profesores

A mis amigos

Agradecimientos

Hay ocasiones en la vida que sabes a ciencia cierta que van a ocurrir. Incluso sientes que van a ocurrir dentro de poco. Sin embargo, cuando ese momento llega, parece como si no te correspondiera todavía.

Eso está ocurriendo mientras escribo estas líneas. Han sido años largos y duros, más de lo que mi yo del instituto nunca imaginó. Pero aún así, parece extraño finalizar esta etapa como si fuera algo habitual.

El aprendizaje y el crecimiento personal han sido constantes. Más allá de los conocimientos puros de las asignaturas, esta carrera te enseña que todo es relativo. Que lo que parece imposible, un mes después parece sencillo; que lo que parece mucho tiempo, se convierte en un suspiro año tras año. Que la vida es tan amplia como nuestro desconocimiento de las cosas.

Agradecer en primer lugar a mis padres, que nunca han dejado que me desvíe de un camino razonable, pero a la vez me han permitido moverme con libertad. Por haber invertido tanto para convertirme en la persona que soy hoy y de la que me siento tan orgulloso. Solo cabe dar las gracias.

Sin olvidar a mi hermana, por siempre estar ahí para sacarme una sonrisa con nuestras tonterías y por aguantarme todos estos años de luces y sombras.

También a todas las personas maravillosas que he conocido en esta etapa. Aunque muchos de ellos no le tienen mucho apego a la Escuela por lo duro de lo que han pasado, para mi la Universidad ha sido un lugar estimulante y lleno de aventuras gracias a ellos.

Mención especial a un compañero con el que, casi sin quererlo, llevo media vida compartiendo clases y experiencias. Espero que también compartamos un futuro profesional próspero.

Agradezco también a mi pareja, por apoyarme y hacer más llevadera cualquier etapa de estrés y por confiar en mí de la manera en que lo hace.

Gracias también a los profesores por el tiempo dedicado en mi formación, dentro y fuera de clase. En especial a los de la rama que he escogido y que me enamoró desde los primeros años de la carrera. En Telemática todos tenemos en común una tendencia a la mejora y a la autosuficiencia que en parte se desprende de los que nos han enseñado.

Finalmente, a mi tutor, Francisco José Fernández Jiménez, por apoyar desde un primer momento las ideas que me fueron surgiendo y por ayudarme a darles forma estando siempre disponible.

Sergio Sucino Barrena

Sevilla, 2019

Resumen

Con la solución que se plantea en el presente documento, se intenta cubrir el problema de estudiantes que no tienen acceso directo a compañeros que, con un horario compatible, puedan hacer de conductores o pasajeros desde su ciudad hasta su facultad.

Se ha optado por el sistema operativo Android por ser la opción que prima en España entre los usuarios de smartphones. El entorno de desarrollo abierto de Android también ha sido determinante, pues facilita la tarea de crear software para el mismo.

Se ha empleado además la plataforma Firebase para ubicar el backend de la aplicación en la nube, así como ejecutar funciones serverless. Una opción muy demandada hoy en día y que marcará la línea de desarrollo y despliegue de aplicaciones en los próximos años.

También se ha utilizado la plataforma Maps de Google para las funciones de localización que automatizan desde la aplicación el proceso de recogida de pasajeros.

En el presente documento se plantearán los requisitos que se tuvieron en cuenta para construir el proyecto, se analizarán y expondrán todas las partes de la aplicación final, además de desgranar todas las fases de configuración de las plataformas utilizadas para que pueda ser replicado.

En definitiva, se ha pretendido crear un producto sólido y en la vanguardia del desarrollo para dar solución a un problema actual.

Abstract

With the solution presented in this document, we try to cover the problem of students who do not have direct access to classmates who, with a compatible schedule, can act as drivers or passengers from their city to their faculty.

The Android operating system has been chosen as the option that prevails in Spain among smartphone users. The open development environment of Android has also been decisive, as it facilitates the task of creating software for it.

The Firebase platform has also been used to locate the backend of the application in the cloud, as well as to execute serverless functions. A highly demanded option today and that will mark the line of development and deployment of applications in the coming years.

The Google Maps platform has also been used for location functions that automate the passenger pick-up process from the application.

This document will raise the requirements that were taken into account to build the project, analyze and expose all parts of the final application, in addition to explaining all the phases of configuration of the platforms used so that it can be replicated.

In short, it has been intended to create a solid product and at the forefront of development to solve a current problem.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

1 INTRODUCCIÓN

Si hay suficientes ojos encima, cualquier error puede parecer evidente.

- Linus Torvalds -

En la sociedad contemporánea el teléfono móvil se ha convertido en la piedra angular que provoca que incluso las tareas más tradicionales y sencillas puedan ser extendidas a un uso que, hasta un pasado reciente, podrían haber sido tildadas de ciencia ficción.

Mientras todo esto ocurre y avanza a un ritmo difícil de asimilar, surgen soluciones que se integran con las antiguas relaciones sociales para dotarlas de agilidad e inmediatez. Un ejemplo es el objetivo principal del proyecto que se va a tratar.

1.1 Motivación

En el transcurso de las reuniones con el tutor del proyecto, se lanzarón varias ideas sobre aplicaciones Android. Partiendo de la oferta incial de trabajo de fin de grado, se pretendía desarrollar un software útil de cara a los usuarios y que empleara tecnologías punteras en el mercado.

La primera certeza que se tenía es que se trataría de una aplicación Android, pues en el transcurso de la carrera he adquirido un gran interés por esta plataforma y las posibilidades que ofrece. Por esto, una de las motivaciones siempre ha sido desarrollar una aplicación seria y con la calidad de un producto de consumo.

Se llegó a la idea final por medio de una de las dificultades que existen a lo largo del grado: **compartir coche** para ir a la facultad.

Aquí es donde entra la motivación principal del proyecto, que no es otra que poner en contacto a universitarios (en un principio solo de la ETSI) que pueden vivir cerca e incluso coincidir en horario de clase, para compartir coche y ahorrar tiempo, dinero y contaminación. Además, mediante las funciones que implementa la aplicación, puede agilizarse el proceso de recogida de pasajeros.

En cuanto a las tecnologías, la inclusión de **Firebase** en el desarrollo ha supuesto un punto de inflexión. La posibilidad de utilizar tecnologías tan modernas como son el *backend as a service* o el *serverless* que integra Firebase en sus diferentes productos, ha sido alentador. El futuro del desarrollo de aplicaciones móviles pasa por confiar en la nube para delegar funciones complejas y de un ámbito que no controla el desarrollador medio.

1.2 Presentación del problema

Pese a que en Sevilla hay transportes públicos que funcionan de forma razonablemente bien, para los que vivimos en poblaciones de los alrededores, el traslado a la facultad en este medio implica perder entre 40 y 50 minutos.

El ***carpooling*** es la práctica de compartir coche entre personas con una rutina o ruta común. Es la modernización de esta antigua forma de ahorrar en combustible y no viajar solo, adaptada a todas las posibilidades de la web y la telefonía que priman hoy en día.

Muchos de los compañeros de la Escuela no conocen a gran parte de la gente que podría transportarlos, ya sea por estudiar en grados con los que no tiene contacto o simplemente por no haber entablado nunca una relación.

El simple hecho de existir una aplicación que aglutine a los interesados en compartir ruta diaria a la Universidad, puede centralizar y facilitar esta búsqueda.

1.3 Objetivos

Uno de los principales objetivos en el desarrollo de la aplicación es que permita la interacción entre usuaios y resulte útil en situaciones reales.

El otro objetivo principal es diseñar un sistema respaldado por la nube mediante Firebase y la adición de servicios de Maps que permitan integrar funcionalidades de localización.

Como objetivo secundario, se ha considerado importante alcanzar unas cotas de calidad y estabilidad de la aplicación altas o comparables a productos ya existentes, teniendo en cuenta los recursos con los que contamos.

1.4 Soluciones existentes en el mercado

Pese a que la aplicación que hemos desarrollado está enfocada a universitarios como punto diferencial, no ofrece nada nuevo que no se haya visto antes. Existen múltiples posibilidades para este tipo de prácticas:

Ilustración 1: Logo de BlaBlaCar

- **BlaBlaCar**

  Blablacar se ha convertido en los últimos años en un referente en el sector en nuestro país.

  Ofrece una plataforma web y móvil donde ofertar cualquier tipo de viaje que vayamos a realizar, además de ofrecer pasarelas de pago y un completo sistema de valoraciones para aportar seguridad a los potenciales pasajeros.

  Aunque es sobre todo conocida para compartir coche en viajes de media o larga distancia, también es usada por muchos usuarios para rutas diarias hacia el trabajo o la universidad.

Ilustración 2: Logo de Amovens

- **Amovens**

  No tan conocida como la anterior, Amovens ofrece una plataforma igual de completa que BlaBlaCar para publicar viajes y unirse a ellos. Además de ello, permite a los usuarios alquilar coches en tu propia ciudad ofertados por particulares, e incluso renting a través de empresas.

Ilustración 3: Logo de Journify

- **Journify**

  Es la aplicación más pequeña que presentamos como alternativa en el mercado, pero la que más coincide con nuestra funcionalidad principal. Journify pone en contacto a universitarios que quieren desplazarse diariamente, en principio, al campus de la Universidad de València. Está desarrollado por un equipo de jóvenes emprendedores y ha ganado varios premios en concursos de este ámbito.

  Sin embargo, actualmente no es muy conocida ni usada en todo el territorio español.

1.5 Solución planteada

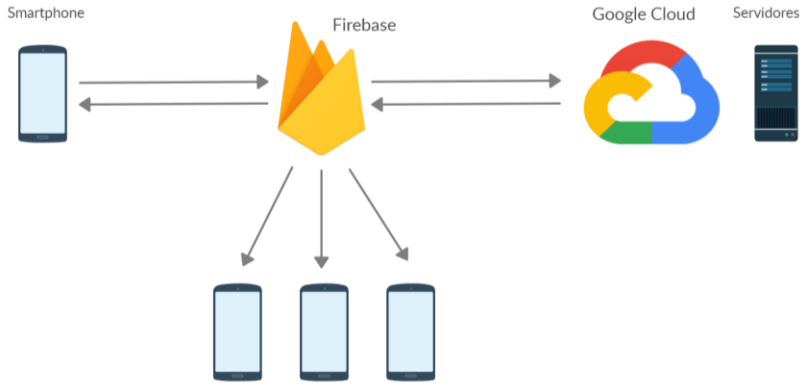


Ilustración 4: Arquitectura simflificada del sistema

En la ilustración adjuntada, podemos observar la solución esquemática que se ha planteado como arquitectura de la aplicación.

El punto principal que hace interesante la solución es que todos los datos que maneja la aplicación se encuentran en la nube. Firebase ofrece una gran cantidad de servicios cuya información y procesado se ubican en la infrestructura de Google Cloud.

De este modo, creamos un sistema liviano y flexible, donde solo tenemos que ocuparnos de las peticiones a los servicios. La base de datos de Firebase, sigue un formato NoSQL JSON, por lo que no existe excesiva dificultad en las consultas o la actualización de información.

La sincronización entre usuarios finales es automática, por lo que cuando se realiza una acción desde un dispositivo, Firebase realiza comunicaciones con el servidor en la nube y notifica en tiempo real al resto de usuarios finales. Estas comunicaciones se realizan a través de peticiones REST sobre HTTP.

Delegar el funcionamiento de nuestra la aplicación en la nube tiene muchas ventajas. Algunas de ellas serían:

- Escalabilidad: es capaz de absorber picos de uso de los usuarios y mantener una tasa de fiabilidad constante.
- Ahorro en administración de infraestructura: no tenemos que configurar ni mantener un servidor dedicado.
- Despliegue mucho más rápido: tan solo es necesario implementar las librerías de los servicios para lanzarlos.

Por contra, tiene algunas desventajas como por ejemplo:

- Experiencia deficiente en redes lentas: depende completamente de la conexión a internet, por lo que

redes 2G o 3G traerán consigo pantallas de carga más largas.

- Mayor consumo de datos: requiere de transferencias constantes de información.
- Dependencia absoluta del proveedor: la aplicación queda atada por el estado de la red de la empresa propietaria, por lo que si hay caídas, no podemos hacer nada para impedir cortes en el servicio.
- Menor control de la privacidad y la seguridad: la facilidad de desarrollo puede hacer que se preste menos antención a la securización de los productos a través de la plataforma. Además de persistir nuestros datos en servidores que pertenecen a terceros.

1.6 Conceptos a conocer para llevar a cabo el proyecto: cloud computing

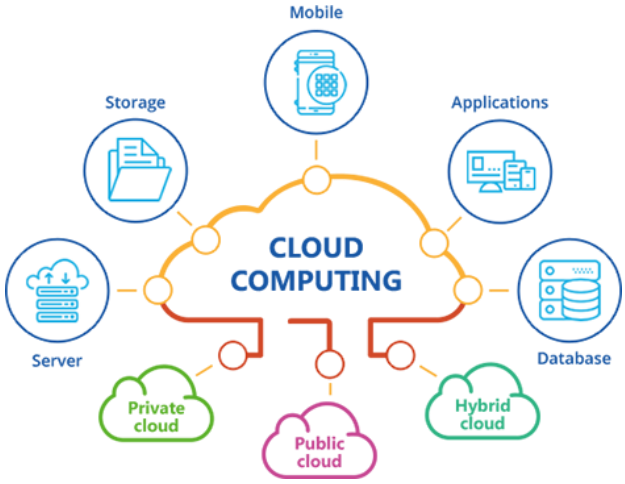


Ilustración 5: Arquitectura de servicios en la nube

La computación en la nube o cloud computing es el paradigma conformado normalmente por servidores que cumplen funciones de almacenamiento, redes, análisis e inteligencia a través de Internet. El término también se utiliza para describir a los centros de datos que se suelen repartir por diferentes ubicaciones geográficas para ofrecer servicios distribuidos.

En la ilustración anterior, se representa una posible arquitectura de computación en la nube con algunos de los servicios más destacados.

La nube puede ser de dominio público en caso de permitir el acceso a muchas organizaciones, privado en el caso de pertenecer a una organización, o híbrido cuando tiene ambas partes. [1]

La computación en la nube ha supuesto una de las mayores revoluciones de la informática en las últimas décadas, desarrollando un gran crecimiento a medida que la tecnología de consumo y de empresas avanza con requisitos de calidad cada vez más altos.

1.6.1 Ventajas frente a computación tradicional

Según la página oficial de Microsoft Azure [2], las principales ventajas de la informática en la nube son:

- "**Costo**: La informática en la nube elimina la inversión de capital que supone la adquisición de hardware y software, y la configuración y ejecución de centros de datos locales (bastidores de servidores, suministro eléctrico ininterrumpido para alimentación y refrigeración y expertos en TI para administrar la infraestructura)."
- "**Velocidad**: La mayoría de los servicios en la nube se proporcionan como autoservicio y a petición, de forma que incluso cantidades enormes de recursos informáticos se pueden aprovisionar en cuestión de minutos."

- "**Escala global**: Entre las ventajas de los servicios informáticos en la nube, se incluye la capacidad de escalar los recursos de forma elástica. En términos de nube, esto significa ofrecer la cantidad adecuada de recursos de TI (por ejemplo, mayor o menor capacidad de proceso, almacenamiento y ancho de banda) en el momento justo en el que se necesitan y desde la ubicación geográfica adecuada."
- "**Productividad**: la informática en la nube elimina la necesidad de muchas tareas, de forma que los equipos de TI pueden dedicar su tiempo a lograr objetivos más importantes para su negocio."
- "**Rendimiento**: Los mayores servicios informáticos en la nube se ejecutan en una red mundial de centros de datos seguros, que se actualizan periódicamente con el hardware más rápido y eficiente de última generación."
- **"Confiabilidad**: La informática en la nube facilita y abarata la creación de copias de seguridad de los datos, la recuperación ante desastres y la continuidad empresarial, ya que los datos se pueden reflejar en varios sitios redundantes en la red del proveedor de servicios en la nube."
- **"Seguridad**: Muchos proveedores de nube ofrecen un conjunto completo de directivas, tecnologías y controles que refuerzan la situación general de seguridad, ayudando a proteger los datos, las aplicaciones y la infraestructura frente a posibles amenazas."

1.6.2 Situación actual del mercado

El mercado está segmentado por gran cantidad de empresas potentes que ofrecen servicios en cloud. Sin embargo, Amazon Web Services (AWS) ha tomado la delantera desde hace unos años en el negocio. Según estimaciones de Canalys, en 2018 AWS contaba con una cuota del 31,5%, seguido de Microsoft Azure con un 13,3% y Google Cloud con un 6,4% [3].

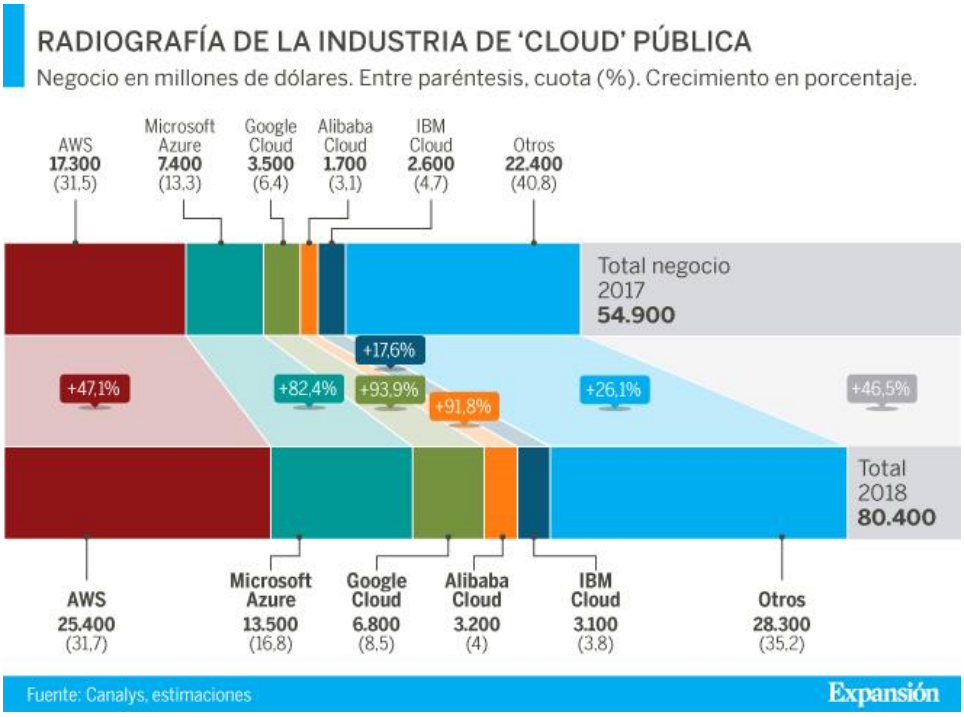


Ilustración 6: Situación de la industria de *cloud*

El sistema operativo dominante en la nube es Linux, incluso por la que pertenece a Microsoft, según reveló la empresa en 2019.

1.6.3 Servicios principales en la nube

Los principales modelos de servicios que existen en el mercado son [4]:

- **IaaS**: Infraestructura como servicio (Digital Ocean, OVH o AWS Lightsail). Basado en la provisión de máquinas virtuales donde, después de instalar y configurar el sistema operativo, podemos desplegar y ejecutar nuestras aplicaciones.
- **CaaS**: Contenedores como servicio (Google Cloud Engine, AWS EC2 Container Service o Microsoft Azure Container Service). Nuestra unidad de trabajo mínima es el contenedor docker que desplegamos en el servicio del proveedor.
- **PaaS**: Plataforma como servicio (Google App Engine, AWS Beanstalk o RedHat OpenShift). Su característica principal es la capacidad que ofrece de desplegar en el Cloud aplicaciones soportadas por el proveedor. El cliente no gestiona ni controla la infraestructura subyacente, incluyendo la red, los servidores, los sistemas operativos o el almacenamiento, sino que tiene control sobre las aplicaciones desplegadas y, posiblemente, sobre los ajustes de configuración del entorno de alojamiento de aplicaciones.
- **BaaS** o **MBaaS**: *Backend as a Service* o *Mobile Backend as a Service* (Parse, Firebase, Auth0 o AWS Cognito). Modelo en el que podemos integrar nuestra aplicación con backends de distinto tipo disponibles en el Cloud y que nos ofrecen un servicio especializado que podemos consumir como si de un API tradicional se tratara. Algunas de las características que proporcionan pueden ser: Gestión de usuarios, notificaciones push, integración de servicios de redes sociales, etc.
- **FaaS:** *Function as a Service* (AWS Lambda, Google Cloud Functions). Permite el despligue de métodos individuales que son invocados en respuesta a eventos. FaaS se engloba también dentro del término *serverless*.

En el proyecto se ha utilizado el servicio de BaaS (Firebase) y de FaaS (Google Cloud Functions).

1.7 Descripción de la memoria

- **Capítulo 1: Introduccion.**

  En este capítulo se encuentra la motivación para llevar a cabo el proyecto, así como el problema y la solución planteada.

  También se encuentran alternativas similares en el mercado y una introducción del concepto de cloud.

- **Capítulo 2: Entorno de trabajo y herramientas**

  En este capítulo se recopilan los recursos hardware y software empleados para llevar a cabo el proyecto.

- **Capítulo 3: Requisitos**

  En este capítulo se realiza el proceso conocido como Ingeniería de requisitos.

- **Capítulo 4: Tecnologías**

  En este capítulo se desglosan todos los productos de las plataformas Firebase y Maps que se han utilizado en el proyecto.

- **Capítulo 5: Aplicación Android**

  En este capítulo se explicará con detalle el funcionamiento de la aplicación y todas sus pantallas, así como la estructura del proyecto y los problemas encontrados al implementarlo.

- **Capítulo 6: Plan de pruebas**

  En este capítulo se enumeran las pruebas que debe pasar la aplicación para considerarse que funciona como se espera de ella.

- **Capítulo 7: Conclusión**

  En este capítulo haremos un cierre del documento comentando la experiencia de llevar a cabo el proyecto, así como un resumen del tiempo dedicado.

- **Anexos**

  En este capítulo se encuentran las guías de despliegue de las plataformas utilizadas y una muesra de política de privacidad válida para la aplicación.

2 ENTORNO DE TRABAJO Y HERRAMIENTAS

En las últimas décadas, el desarrollo de software ha tenido una transformación muy importante. En gran medida por el avance en software, que permite avanzar más rápido y de forma más segura al llevar a cabo un proyecto. A continuación se exponen las herramientas utilizadas para llevar a cabo el proyecto.

2.1 Recursos humanos

Las personas que han intervenido en la realización del proyecto han sido por un lado, el autor del mismo, y por otro, como apoyo fundamental, el tutor mencionado en las primeras páginas de este documento.

2.2 Herramientas hardware

Los recursos hardware son aquellos dispositivos o equipos electrónicos que han sido empleados durante alguna fase del proyecto.

2.2.1 Ordenadores personales y pantallas de visualización

El **equipo informático principal** consta de un PC de sobremesa tipo "torre" personalizado por piezas:

- Procesador: Intel Core i7-8700K x @ 3.7Ghz (2x6 hilos / 12 MB caché)
- Memoria RAM: Corsair Vengeance 2x8GB DDR4 @ 3000MHz
- Tarjeta gráfica: Gigabyte GeForce GTX 970 G1 4GB GDDR5
- Disco duro: SSD M.2 Samsung 870 EVO 500GB
- Sistema Operativo: Windows 10 Pro 64bits

Las **pantallas de visualización** son dos monitores en modo escritorio extendido:

- Monitor principal: Dell U2417H (24", 1080p, 60Hz)
- Monitor secundario: Asus VX229H (21,5", 1080p, 60Hz)

Se ha utilidado un equipo portátil secundario que consta de la siguiente configuración:

- Modelo: Asus A55A-SX465H
- Procesador: Intel Core i7-3630QM @ 2,40GHz (4x2 hilos / 6 MB caché).
- Memoria RAM: 2x4GB DDR3
- Tarjeta gráfica: Intel HD Graphics 4000 (integrada).
- Disco duro: SSD Samsung 850 EVO 250GB
- Sistema Operativo: Windows 10 Home 64bits

2.2.2 Dispositivos moviles

Los teléfonos móviles inteligentes utilizados en las tareas de pruebas fueron los siguientes:

2.2.2.1 Nokia 7 Plus

- Android 9.0 Pie

- Pantalla de 6” LCD IPS (18:9, 2160x1080 px)
- Procesador Qualcomm Snapdragon 660 Octa-Core @ 2.2Ghz
- Memoria RAM de 4GB
- Memoria interna de 64GB eMMC
- Batería de 3800 mAh
- Wi-Fi 802.11n 5GHz, Bluetooth 5.0

2.2.2.2 Sony Xperia Z

- Android 5.0 Pie
- Pantalla de 5” LCD IPS (16:9, 1920x1080 px)
- Procesador Qualcomm Snapdragon S4 Pro Quad-Core @ 1.5GHz
- Memoria RAM de 2GB
- Memoria interna de 16GB eMMC
- Batería de 2400 mAh
- Wi-Fi 802.11b/g/n, Bluetooth 4.0

2.2.2.3 Oneplus 7

- Android 9.0 Pie
- Pantalla de 6.41” Optic AMOLED (19.5:9, 2340x1080 px)
- Procesador Qualcomm Snapdragon 855 Octa-Core @ 2.84Ghz
- Memoria RAM de 8GB
- Memoria interna de 256GB UFS 3.0
- Batería de 3700 mAh
- Wi-Fi 802.11n 5GHz, Bluetooth 5.0

2.3 Recursos software

2.3.1 IDE: Android Studio

Ilustración 7: Logo de Android Studio

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014.

Está basado en el software **IntelliJ IDEA** de JetBrains y ha sido publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, macOS y GNU/Linux. Ha sido diseñado específicamente para el desarrollo de Android [5].

La totalidad del código de la aplicación ha sido desarrollado en Android Studio 3.x, con el sistema de

construcción de dependencias Gradle. Las pruebas se han realizado en mayor medida en la máquina virtual de Android de la que dispone Android Studio.

2.3.2 Control de versiones: Git con GitHub

Ilustración 8: Logo de git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

En el proyecto que nos ocupa, hemos utilizado la **integración de git en Android Studio** para realizar el control de versiones de manera rápida y visual.

Ilustración 9: Logo de GitHub

GitHub es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git.

Se ha empleado GitHub como repositorio remoto de código para las funciones de backup, interoperabilidad entre los equipos en los que se ha desarrollado y gestión de cambios.

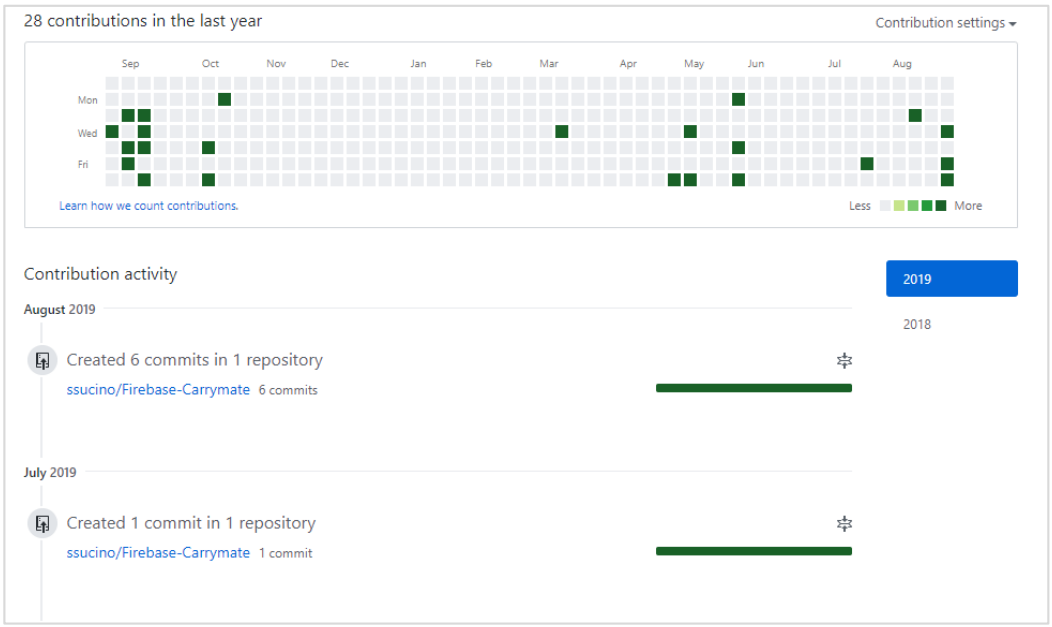


Ilustración 10: Control de versiones en GitHub

2.3.3 StarUML

Ilustración 11: Logo de StarUML

StarUML es una herramienta UML creada por MKLab. El software hizo uso de una licencia derivada de GNU GPL hasta 2014, cuando se lanzó una versión con licencia propietaria.

En la ilustración siguiente se incluye una muestra del aspecto de StarUML.

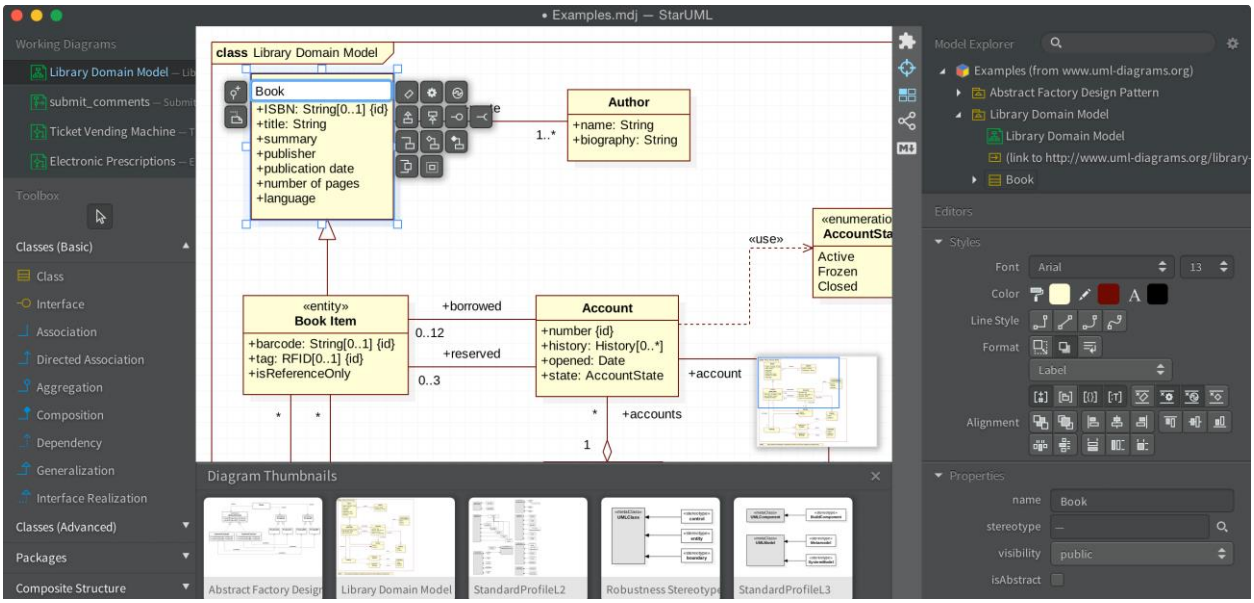


Ilustración 12: Ejemplo de uso de StarUML

Tras un breve análisis de las principales opciones del mercado para la creación de diagramas UML, se optó por StarUML por su facilidad de uso, su interfaz gráfica cuidada y por poseer una versión de prueba que contiene todas las características necesarias para el diseño del sistema que nos ocupa.

2.3.4 Librerías externas

```
implementation 'cn.carbs.android:AvatarImageView:1.0.4' //Imagen circular con letra incial
implementation 'com.github.bumptech.glide:glide:4.6.1' //Recortar Imagen Descargar Imagen
implementation 'org.parceler:parceler-api:1.1.12' //Serializar objetos entre actividades
annotationProcessor 'org.parceler:parceler:1.1.12' //Anotaciones para Parceler
```

Código 1: Dependencias de librerías externas en build.gradle

En la captura anterior del archivo build.gradle puede verse el conjunto de librerías externas que serán descargadas y compiladas por Gradle.

La lista de dependencias es la siguiente:

- **AvatarImageView** (v1.0.4) [6]

  Librería que implementa una vista derivada de *ImageView* pero con una gran cantidad de añadidos enfocados en la representación avatares de perfil de usuario.

  Su principal función es la de mostrar imágenes con forma circular, así como representar texto en su interior en caso de no especificar un archivo de imagen. También incluye la posibilidad de establecer un color de fondo aleatorio tomando una cadena de texto como semilla.

```
AvatarImageView fotoPerfilView = findViewById(R.id.fotoPerfil);

fotoPerfilView.setTextAndColorSeed(
        usuario.getNombre().substring(0, 1), //Iniciales
        usuario.getNombre()); //Semilla de color de fondo aleatorio
```

Código 2: Dependencias de librerías externas en build.gradle

Se ha empleado AvatarImageView para el diseño de la interfaz gráfica en todos los casos en los que interviene una foto de perfil de usuario.

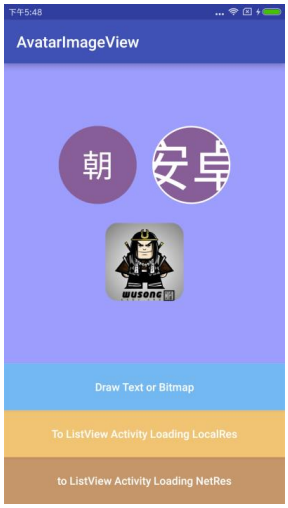


Ilustración 13: Muestra de AvatarImageView

- **Glide** (v4.6.1) [7]

  Desarrollada por Bump Technologies, una empresa adquirida por Google en 2014, es la librería más utilizada y recomendada para la gestión de carga de imágenes en Android. Entre sus funciones se encuentra la descarga de imágenes a través de URL, el cacheo personalizable o la decodificación y transformación de recursos de vídeo y GIF.

  Glide se centra en la facilidad de uso de cara a los desarrolladores y en el rendimiento, siendo la librería más indicada en el caso de, por ejemplo, largas listas de datos con imágenes.

  Se ha empleado Glide para la carga de imágenes de perfil de usuario sobre los AvatarImageView, puesto que estos archivos se encuentran en servidores remotos.

```
Glide
  .with(myFragment)
  .load(url)
  .centerCrop()
  .placeholder(R.drawable.loading_spinner)
  .into(myImageView);
```

Código 3: Ejemplo de uso de Glide

- **Parceler** (v1.1.12) [5]

  En Android, los objetos de tipo Parcelable son una buena manera de serializar objetos Java entre actividades. En comparación con la serialización tradicional, Parcelable toma del orden de 10 veces menos tiempo para serializar y deserializar. Sin embargo, hay un problema importante, los parcelables contienen gran cantidad de redundancia de código o *boilerplate*.

  Parceler es una librería que facilita la creación de Parcelables sin necesidad de implementarlo de la manera tradicional. A través de las anotaciones Java, es capaz de generar el código necesario para serializar objetos fácilmente.

```
@Parcel
public class Example {
    String name;
    int age;

    public Example() {}

    public Example(int age, String name) {
        this.age = age;
        this.name = name;
    }

    public String getName() { return name; }

    public int getAge() { return age; }
}
```

Código 4: Ejemplo de creación de objeto con Parceler

```
Parcelable wrapped = Parcels.wrap(new Example("Andy", 42));

Example example = Parcels.unwrap(wrapped);
example.getName(); // Andy
example.getAge(); // 42
```

Código 5: Ejemplo de serialización con Parceler

3 REQUISITOS DEL SISTEMA

En el presente capítulo vamos a acometer el proceso conocido como Ingeniería de requisitos. Esto nos dará una idea de la estructura del sistema, así como el comportamiento de cada una de sus partes y sus relaciones, en consonancia con las restricciones existentes. Se incluyen los casos de uso, requisitos funcionales y requisitos no funcionales.

Este documento va dirigido a todas las personas interesadas en entender el funcionamiento de la aplicación que se quiere desarrollar.

3.1 Actores

Código	Nombre de actor	Descripción
ACT-01	Usuario	Representa a las personas que harán uso de la aplicación. Pueden poseer dos roles: • Usuario administrador de anuncio: es el creador del anuncio, por lo que tiene funciones activas en las funcionalidades de chat y de localización. • Usuario cliente: es el que se une a un determinado chat asociado a un anuncio, por lo que tiene funciones pasivas en las funcionalidades de chat y de localización.
ACT-02	Administrador	Representa a la/s persona/s que realizarán las tareas de gestión, puesta en marcha y monitorización de los servicios de Firebase que utilizaremos en la aplicación móvil.
ACT-03	Servidor en cloud	Representa el sistema de servidores de Google Cloud al que podemos acceder a través de los servicios de Google Firebase.

Tabla 1: Actores del sistema

3.2 Requisitos generales

Esta sección contiene la especificación de los requisitos generales del sistema, también denominados características del sistema u objetivos del sistema.

RGEN-01	Mostrar pantalla inicial de carga
Versión	1.0
Descripción	La aplicación deberá mostrar una pantalla estática que se oculte cuando la precarga de la aplicación se haya completado.
Prioridad	Media

Tabla 2: Mostrar pantalla inicial de carga

RGEN-02	Acceso a datos de servidor en la nube
Versión	1.0
Descripción	La aplicación deberá ser capaz de conectarse mediante internet a la instancia de Firebase que contiene sus datos para descargarlos o modificarlos.
Prioridad	Esencial

Tabla 3: Acceso a datos de servidor en la nube

RGEN-03	Actualizar datos en tiempo real
Versión	1.0
Descripción	La aplicación deberá ser capaz de obtener datos que estén siendo creados o modificados en tiempo real por los usuarios.
Prioridad	Esencial

Tabla 4: Actualizar datos en tiempo real

RGEN-04	Mostrar visualmente los datos
Versión	1.0
Descripción	La aplicación deberá mostrar visualmente los datos descargados de la nube que han sido proporcionados por los usuarios mediante la interacción con los diferentes apartados de la aplicación.
Prioridad	Esencial

Tabla 5: Mostrar visualmente los datos

RGEN-05	Permanecer autenticado
Versión	1.0
Descripción	La aplicación deberá autenticar al usuario automáticamente al iniciarse si la última vez no cerró sesión.
Prioridad	Esencial

Tabla 6: Permanecer autenticado

RGEN-06	**Permitir múltiples formas de autenticación**
Versión	1.0
Descripción	La aplicación dará opción a autenticarse mediante la pasarela de cuenta de Google o bien mediante un correo y contraseña registrados anteriormente.
Prioridad	Esencial

Tabla 7: Permitir múltiples formas de autenticación

RGEN-07	**Mostrar animación de carga de datos**
Versión	1.0
Descripción	La aplicación deberá mostrar animaciones que indiquen que los datos que se encuentran en Firebase Database y Firebase Storage están siendo cargados. Al dibujar la interfaz con los datos deberá ocultar esta animación.
Prioridad	Media

Tabla 8: Mostrar animación de carga de datos

RGEN-08	**Notificar de nuevos datos**
Versión	1.0
Descripción	La aplicación deberá mostrar notificaciones nativas de Android para avisar de que hay nuevos datos disponibles de su interés.
Prioridad	Esencial

Tabla 9: Notificar de nuevos datos

RGEN-09	**Obtener información de localización**
Versión	1.0
Descripción	La aplicación deberá ser capaz de obtener la información localización del usuario.
Prioridad	Esencial

Tabla 10: Obtener información de localización

RGEN-10	**Calcular y dibujar rutas**
Versión	1.0

Descripción	La aplicación deberá ser capaz obtener la ruta entre dos localizaciones de un mapa y representarla.
Prioridad	Esencial

Tabla 11: Calcular y dibujar rutas

3.3 Casos de uso

En este apartado indicaremos la especificación de los casos de uso del sistema, incluyendo diagramas y especificaciones de los propios casos de uso. Estos describen como se comportará el sistema de manera funcional desde el punto de vista de los actores que intervienen.

Como introducción, mostraremos un diagrama de caso de uso generalista.

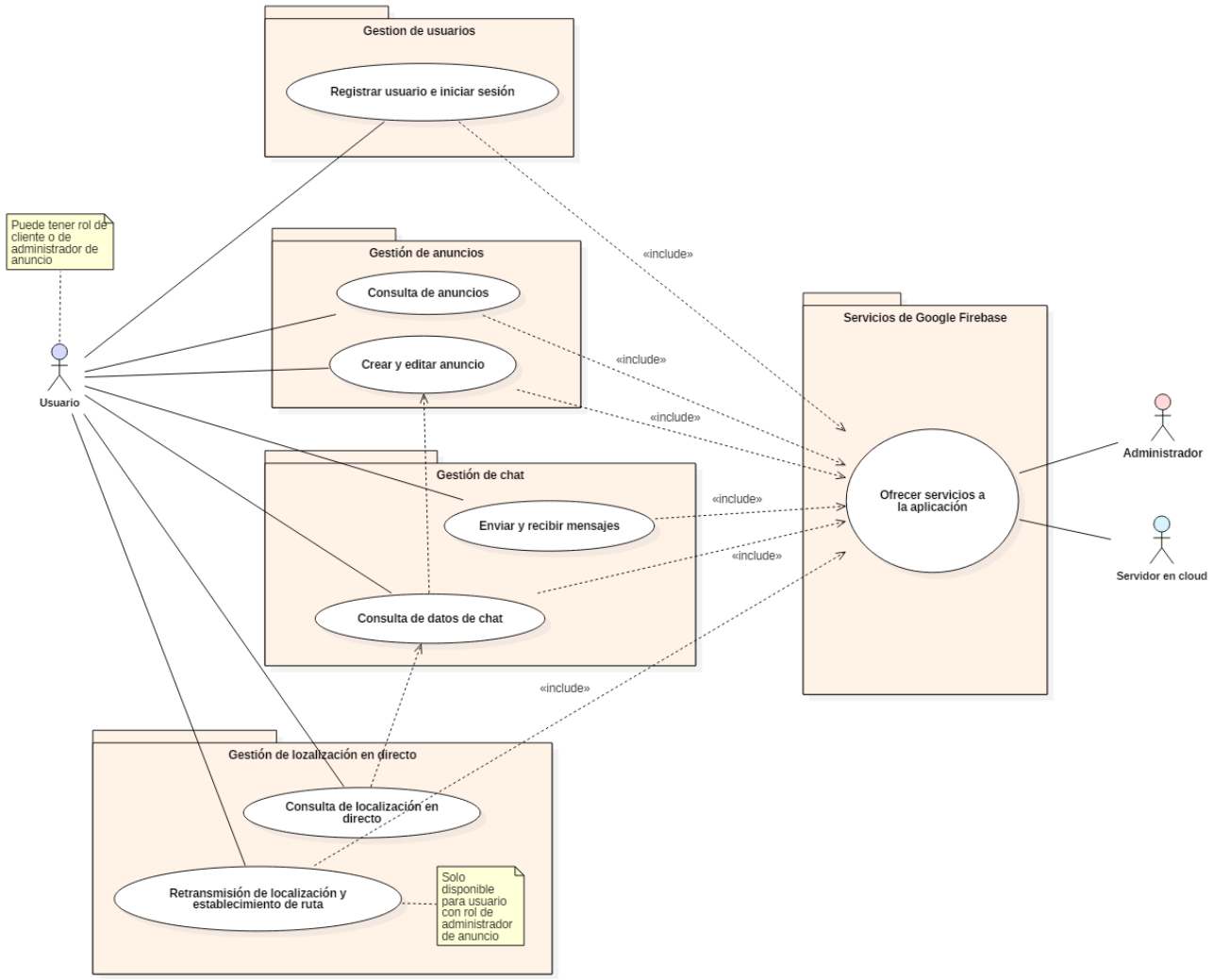


Ilustración 14: Diagrama general de casos de uso

A continuación indicaremos los casos de uso de cada uno de los subsistemas en los que se ha modelado el sistema con mayor profundidad.

3.3.1 Servicios de Google Firebase

En primer lugar, expondremos los casos de uso relacionados con la plataforma Firebase. Podemos ver un diagrama en la imagen que sigue al texto.

Estas funciones se realizan físicamente en el Servidor en cloud de Google, pero podemos acceder a ellas de forma sencilla y flexible a través de los distintos servicios que encontramos en Firebase.

Para el funcionamiento de la plataforma sobre la aplicación, ha sido necesaria una configuración previa por el Administrador del sistema.

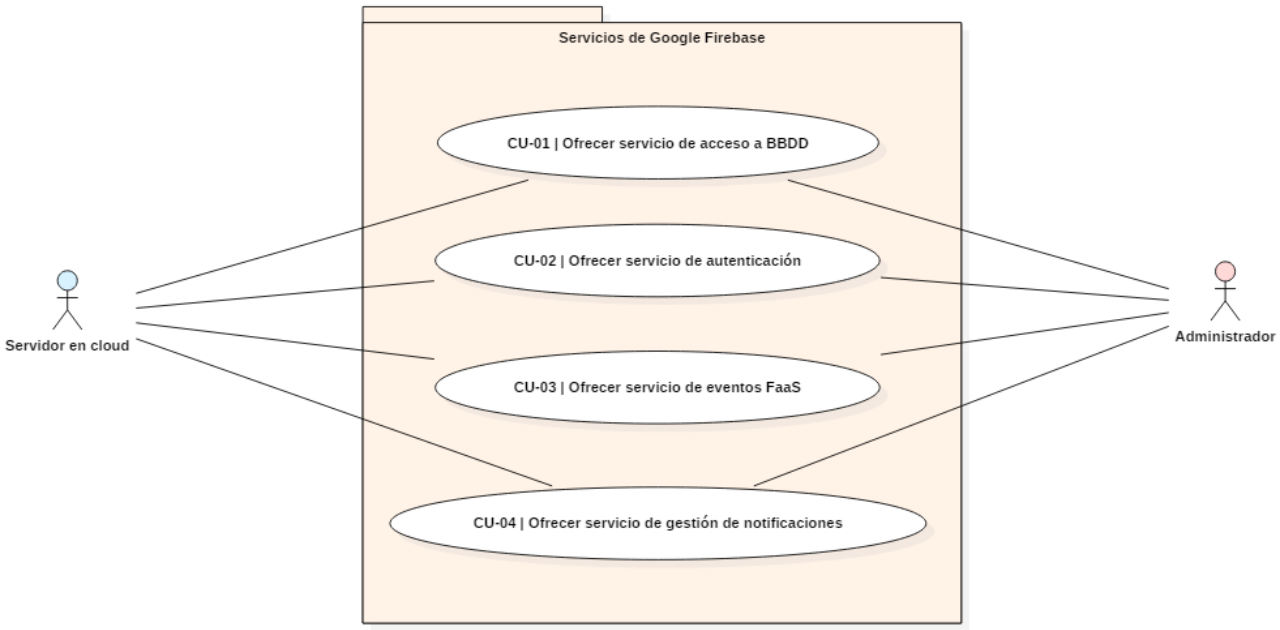


Ilustración 15: Diagrama de caso de uso de subsistema Servicios de Firebase

CU-01	**Ofrecer servicio de acceso a BBDD**
Precondición	El Administrador debe haber configurado y securizado la instancia de Firebase Realtime Database del sistema.
Descripción	Permite la conexión con la base de datos en el servidor en la nube de Google mediante peticiones de conjuntos de datos.
Postcondición	No procede.

Tabla 12: Ofrecer servicio de acceso a BBDD

CU-02	**Ofrecer servicio de autenticación**
Precondición	El Administrador debe haber configurado y habilitado los diferentes métodos de autenticación deseados en su instancia de Firebase.

Descripción	Permite la autenticación de usuarios a través de la pasarela de Firebase Auth. Los métodos de autenticación disponibles serán: • Correo y contraseña. • Cuenta de Google.
Postcondición	No procede.

Tabla 13: Ofrecer servicio de autenticación

CU-03	**Ofrecer servicio de eventos FaaS**
Precondición	El Administrador debe haber configurado en su instancia de Firebase el acceso a Google Cloud Functions y haber publicado mediante el Firebase SDK Admin los scripts que se emplearán.
Descripción	Permite la ejecución de métodos serverless cuando se producen determinadas condiciones preestablecidas. La ejecución puede ser monitorizada por el Administrador desde Firebase Functions. Estos eventos pueden interactuar con otros servicios de Firebase o realizar procesados de datos que puedan resultar útiles.
Postcondición	No procede.

Tabla 14: Ofrecer servicio de eventos FaaS

CU-04	**Ofrecer servicio de gestión de notificaciones**
Precondición	Tener configurada la instancia de Firebase para el sistema.
Descripción	Permite la distribución de mensajes con carga de datos hacia un grupo de dispositivos o un dispositivo en concreto que tengan la aplicación instalada.
Postcondición	No procede.

Tabla 15: Ofrecer servicio de gestión de notificaciones

3.3.2 Gestión de usuarios

A continuación expondremos los casos de uso relacionados con sesiones y datos de usuario, representado por el siguiente diagrama:

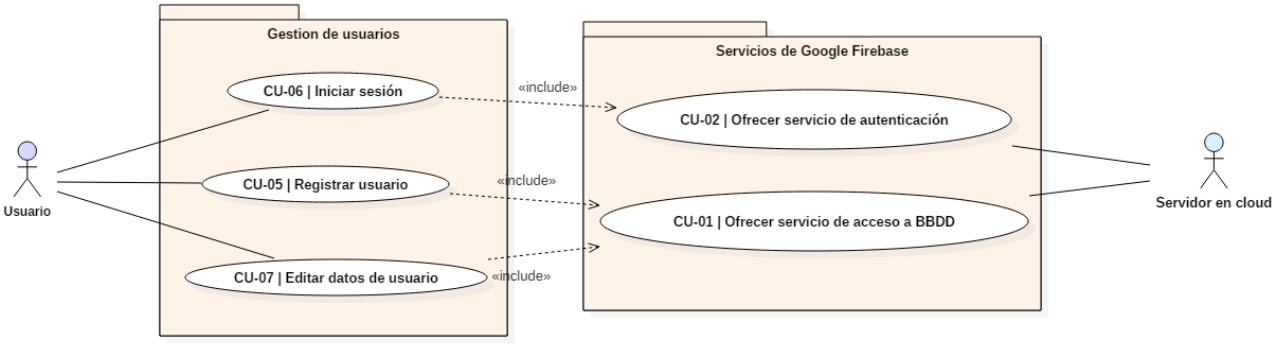


Ilustración 16: Diagrama de caso de uso de subsistema de Gestión de usuarios

CU-05	**Registrar usuario**
Precondición	El dispositivo debe poseer conexión a internet activa.
Descripción	La aplicación deberá poder registrar un usuario que aún no tenga datos personales en base de datos. Tenemos dos casos según tipo de autenticación: • A través de pasarela de Google: los datos se obtienen de la propia cuenta Gmail del usuario. • A través de correo y contraseña: se muestra un formulario que debe ser rellenado para registrar los datos de interés del usuario.
Postcondición	Una vez registrado, el log-in debe completarse de forma exitosa para confirmar que el registro es correcto.

Tabla 16: Registrar usuario

CU-06	**Iniciar sesión**
Precondición	El dispositivo debe poseer conexión a internet activa, el usuario debe estar registrado en la aplicación.
Dependencias	• CU-05: Registrar usuario
Descripción	La aplicación deberá autenticar al usuario por cualquiera de los métodos disponibles, a través del servicio Firebase Auth.
Postcondición	Después de iniciar sesión, la aplicación debe descargar los datos del usuario.

Tabla 17: Iniciar sesión

CU-07	Editar datos de usuario
Precondición	El dispositivo debe poseer conexión a internet activa, el usuario debe estar logueado en la aplicación.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión
Descripción	La aplicación deberá permitir al usuario modificar los datos de la cuenta del usuario: nombre, foto de perfil.
Postcondición	Después de guardar los cambios, deben mostrarse actualizados en la pantalla principal de la aplicación.

Tabla 18: Editar datos de usuario

3.3.3 Gestión de anuncios

A continuación expondremos los casos de uso relacionados con creación y modificación de anuncios, representado por el siguiente diagrama:

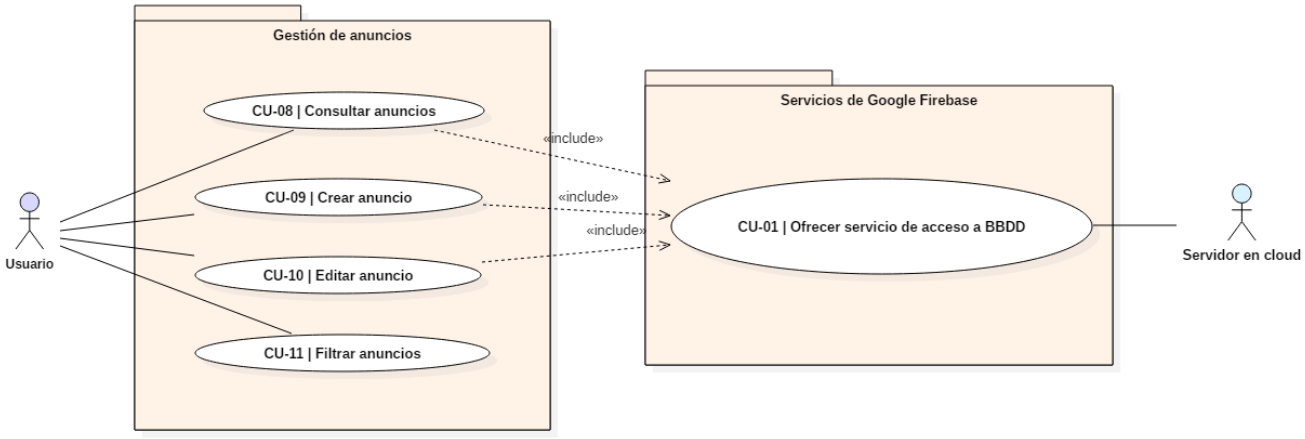


Ilustración 17: Diagrama de caso de uso de subsistema Gestión de anuncios

CU-08	Consultar anuncios
Precondición	El dispositivo debe poseer conexión a internet activa, el usuario debe estar logueado en la aplicación.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión
Descripción	La aplicación deberá mostrar en pantalla una lista con todos los anuncios de coches que han sido publicados por los usuarios y que están guardados en base de datos.

Postcondición	No procede.

Tabla 19: Consultar anuncios

CU-09	**Crear anuncio**
Precondición	El dispositivo debe poseer conexión a internet activa, el usuario debe estar logueado en la aplicación.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión
Descripción	La aplicación deberá permitir al usuario publicar un anuncio de coche mediante un formulario donde se indicarán los datos: grado, ciudad, zona, horario seminal.
Postcondición	Después de guardar los datos, la aplicación deberá mostrar la pantalla principal donde aparecerá el anuncio que acabamos de publicar.

Tabla 20: Crear anuncio

CU-10	**Editar anuncio**
Precondición	El dispositivo debe poseer conexión a internet activa, el usuario debe estar logueado en la aplicación y poseer un anuncio publicado.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión • CU-09: Crear anuncio
Descripción	La aplicación deberá permitir al usuario editar un anuncio de coche mediante el mismo formulario de registro donde se indicarán los datos: grado, ciudad, zona, horario seminal.
Postcondición	Después de guardar los datos, la aplicación deberá mostrar la pantalla principal donde aparecerá el anuncio que acabamos de editar.

Tabla 21: Editar anuncio

CU-11	**Filtrar anuncios**
Precondición	El usuario debe encontrar en la pantalla de anuncios con la lista ya cargada.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión • CU-08: Consultar anuncios
Descripción	La aplicación deberá permitir al usuario establecer un filtro en la lista de anuncios según los criterios: ciudad, grado. El filtrado se realizará de forma local y sin

	necesidad de conexión.
Postcondición	No procede.

Tabla 22: Filtrar anuncios

3.3.4 Gestión de chat

A continuación expondremos los casos de uso relacionados con un chat de anuncio, representado por el siguiente diagrama:

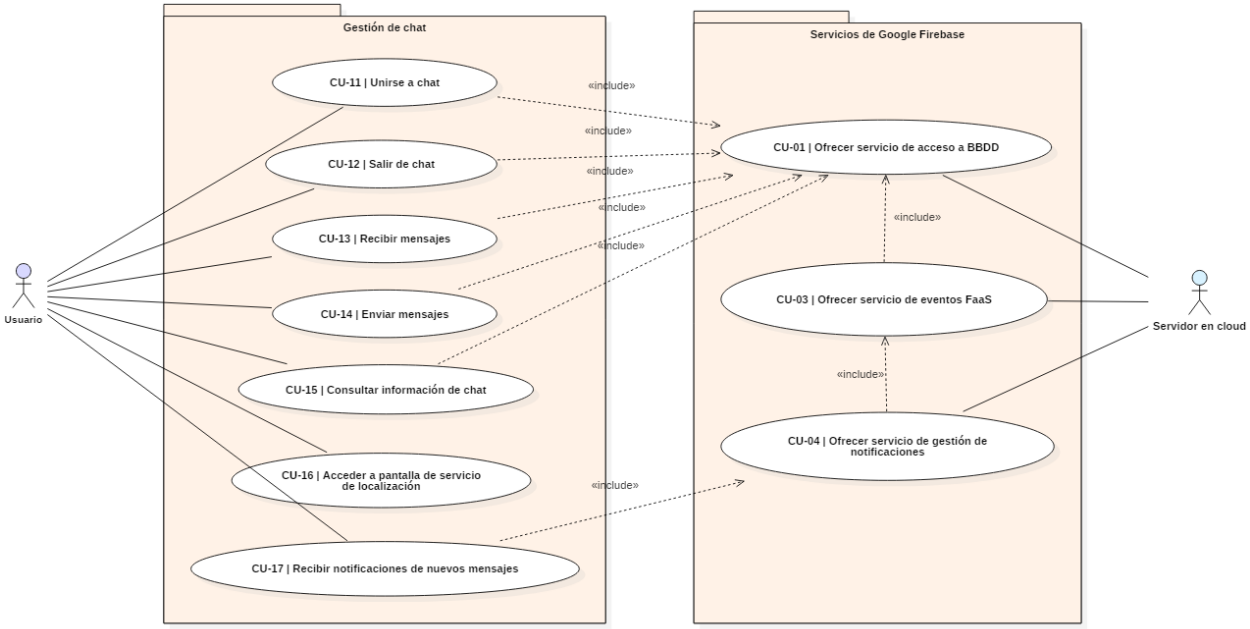


Ilustración 18: Diagrama de casos de uso de subsistema de Gestión de chat

CU-12	**Unirse a chat**
Precondición	El dispositivo debe poseer conexión a internet activa, el usuario debe estar logueado en la aplicación.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión • CU-08: Consultar anuncios
Descripción	La aplicación deberá permitir unirnos al chat de un anuncio pulsando sobre uno de la lista.
Postcondición	Al entrar en el chat el usuario no debe poder visualizar los mensajes anteriores al momento de la unión.

Tabla 23: Unirse a chat

CU-13	**Salir de chat**
Precondición	El dispositivo debe poseer conexión a internet activa, el usuario debe estar logueado en la aplicación y formar parte de un chat de anuncio.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión • CU-12: Unirse a chat
Descripción	La aplicación deberá permitir desvincularnos de un chat de anuncio.
Postcondición	En la pantalla principal de anuncios no deberá mostrarse el chat del que salimos en la lista de chats recientes.

Tabla 24: Salir de chat

CU-14	**Intercambiar mensajes de chat**
Precondición	El dispositivo debe poseer conexión a internet activa, el usuario debe estar logueado en la aplicación y formar parte de un chat de anuncio.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión • CU-12: Unirse a chat
Descripción	La aplicación deberá permitir al usuario recibir mensajes de texto de otros usuarios miembros del chat, así como enviar los suyos propios; todo ello en tiempo real.
Postcondición	No procede.

Tabla 25: Intercambiar mensajes de chat

CU-15	**Consultar información de chat**
Precondición	El usuario debe estar logueado en la aplicación y formar parte de un chat de anuncio.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión • CU-12: Unirse a chat
Descripción	La aplicación deberá permitir al usuario consultar la información detallada del anuncio vinculado al chat.
Postcondición	No procede.

Tabla 26: Consultar información de chat

CU-16	Acceder a pantalla de servicio de localización
Precondición	El dispositivo debe poseer conexión a internet activa, el usuario debe estar logueado en la aplicación.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión • CU-08: Consultar anuncios • CU-12: Unirse a chat
Descripción	La aplicación deberá permitir el acceso al servicio de retransmisión de localización desde la pantalla de chat.
Postcondición	No procede.

Tabla 27: Acceder a pantalla de servicio de localización

CU-17	Recibir notificaciones de nuevos mensajes
Precondición	El dispositivo debe poseer conexión a internet activa, el usuario debe estar logueado en la aplicación.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión • CU-08: Consultar anuncios • CU-12: Unirse a chat • CU-14: Intercambiar mensajes de chat
Descripción	La aplicación deberá lanzar notificaciones nativas de Android para los nuevos mensajes de los grupos de chat donde el usuario es miembro.
Postcondición	No procede.

Tabla 28: Recibir notificaciones de nuevos mensajes

3.3.5 Gestión de localización en directo

A continuación expondremos los casos de uso relacionados con la funcionalidad de localización, representado por el siguiente diagrama:

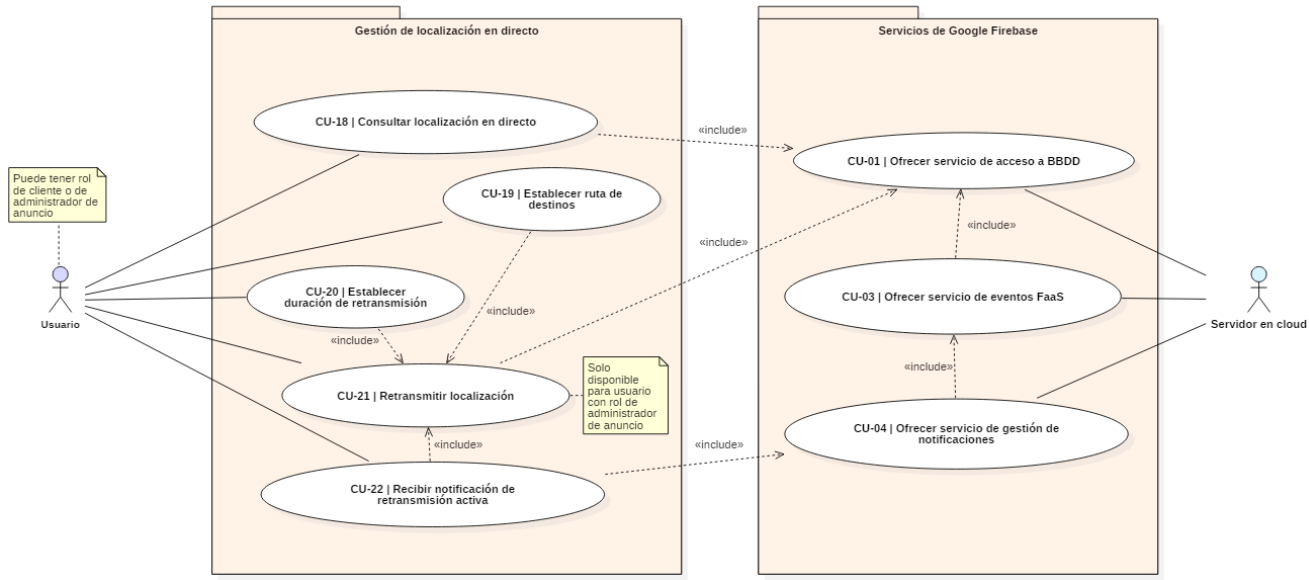


Ilustración 19: Diagrama de casos de uso de subsistema de Gestión de localización

CU-18	**Establecer configuración de retransmisión**
Precondición	El usuario debe estar logueado en la aplicación y poseer rol de administrador de un chat de anuncio.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión • CU-09: Crear anuncio
Descripción	La aplicación deberá permitir al usuario con rol de administrador de chat de anuncio establecer una serie de destinos de ruta, además de un tiempo máximo de retransmisión.
Postcondición	No procede.

Tabla 29: Establecer configuración de retransmisión

CU-19	**Retransmitir localización**
Precondición	El usuario debe estar logueado en la aplicación y poseer rol de administrador de un chat de anuncio.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión • CU-09: Crear anuncio • CU-18: Establecer configuración de retransmisión
Descripción	La aplicación deberá permitir al usuario con rol de administrador de chat de anuncio

	iniciar el servicio de localización como fuente de la información. Su localización será actualizada cada 6 segundos y enviada a la BBDD.
Postcondición	No procede.

Tabla 30: Retransmitir localización

CU-20	**Consultar localización en directo**
Precondición	El usuario debe estar logueado en la aplicación y poseer rol de cliente de un chat de anuncio.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión • CU-12: Unirse a chat
Descripción	La aplicación deberá permitir al usuario con rol de cliente de chat de anuncio iniciar el servicio de localización como receptor de la información. La localización del administrador será visualizada en un mapa dinámicamente, además de la ruta de destinos que ha establecido.
Postcondición	No procede.

Tabla 31: Consultar localización en directo

CU-21	**Recibir notificación de retransmisión activa**
Precondición	El usuario debe estar logueado en la aplicación y formar parte de un chat de anuncio.
Dependencias	• CU-05: Registrar usuario • CU-06: Iniciar sesión • CU-09: Crear anuncio • CU-12: Unirse a chat • CU-18: Establecer configuración de retransmisión
Descripción	La aplicación deberá lanzar notificaciones nativas de Android para indicar que existe una retransmisión activa en el caso de rol de cliente. Deberá lanzar una notificación fija en caso de ser administrador y estar retransmitiendo.
Postcondición	No procede.

Tabla 32: Recibir notificación de retransmisión activa

3.4 Requisitos funcionales

A continuación se encontrarán los requisitos funcionales y no funcionales que se han identificado a partir de los requisitos generales y de los casos de uso.

3.4.1 Requisitos funcionales de información

Los requisitos de información describen la estructura de datos que debe almacenar el sistema para poder ofrecer la funcionalidad descrita en los apartados anteriores.

RFI-01	**Estructura de base de datos en cloud**
Versión	1.0
Descripción	La base de datos de la que se extrae la información para el funcionamiento de la app. Podemos diferenciar entre las dos instancias de las que hacemos uso a través de Firebase: • BBDD de Realtime Database: fuente principal de donde se extrae la información de la que hará uso la aplicación. • BBDD de Storage: almacena los archivos de imagen que serán usados como avatar por los usuarios.
Datos específicos	La base de datos de Realtime Database poseen nodos a modo de tablas donde se recogerá información relativa a: • Usuarios • Anuncios • Chats de anuncios • Datos de localización en directo • Notificaciones
Prioridad	Esencial

Tabla 33: Estructura de base de datos en cloud

RFI-02	**Información de usuario**
Versión	1.0
Descripción	Contiene información sobre los usuarios registrados en la aplicación.
Datos específicos	• ID de usuario (cadena) • Nombre (cadena) • URL de avatar (cadena) • ID de anuncio (cadena) • Lista de chats (mapa de cadena-entero)

Prioridad	Esencial

Tabla 34: Información de usuario

RFI-03	**Información de anuncio**
Versión	1.0
Descripción	Contiene la información sobre los anuncios que han sido publicados por usuarios.
Datos específicos	• ID de usuario administrador (cadena) • Ciudad (cadena) • Zona (cadena) • Grado (cadena) • Horario (lista de objetos)
Prioridad	Esencial

Tabla 35: Información de anuncio

RFI-04	**Información de chat**
Versión	1.0
Descripción	Contiene la información sobre los anuncios que han sido publicados por usuarios.
Datos específicos	• ID de usuario administrador (cadena) • Clientes (mapa de cadena-entero) • Mensajes (lista de objetos) • Flag de existencia de clientes (booleano) • Flag de existencia de mensajes (booleano)
Prioridad	Esencial

Tabla 36: Información de chat

RFI-05	**Información de localización**
Versión	1.0
Descripción	Contiene la información sobre la retransmisión de posición y de ruta por parte de un usuario con rol de administrador.
Datos específicos	• ID de usuario administrador (cadena) • Latitud (entero con signo)

	• Longitud (entero con signo) • Precisión (entero) • Tiempo restante de retransmisión (entero) • Hora actual (entero) • Ruta (lista de objetos)
Prioridad	Esencial

Tabla 37: Información de localización

RFI-06	**Información de notificaciones**
Versión	1.0
Descripción	Contiene la información sobre un nuevo evento que debe ser notificado: un nuevo mensaje de chat o una nueva retransmisión de localización.

Datos específicos	Podemos dividirlo en:
	• Localización
	  o ID de usuario administrador de chat (cadena)
	  o Nombre de usuario (cadena)
	• Mensaje
	  o ID de usuario administrador de chat (cadena)
	  o Nombre de autor del mensaje (cadena)
	  o URL de avatar de autor del mensaje (cadena)
	  o Contenido del mensaje (cadena)

Prioridad	Alta

Tabla 38: Información de notificaciones

3.4.2 Requisitos de reglas de negocio

Los requisitos de reglas de negocio especifican las reglas de negocio que nunca debe incumplir el sistema en su funcionamiento.

RFRN-01	**Diferentes usuarios no podrán tener sesiones activas en el mismo dispositivo**
Versión	1.0
Descripción	Dos usuarios nunca podrán autenticarse y tener sesión activa en el mismo dispositivo al mismo tiempo. Uno debe cerrar sesión antes de que el siguiente lo haga.

Prioridad	Esencial

Tabla 39: Diferentes usuarios no podrán tener sesiones activas en el mismo dispositivo

RFRN-02	**Múltiples sesiones de mismo usuario simultáneas**
Versión	1.0
Descripción	Un mismo usuario podrá tener estar autenticado y tener su sesión activa en cuantos dispositivos requiera.
Prioridad	Esencial

Tabla 40: Múltiples sesiones de mismo usuario simultáneas

RFRN-03	**El usuario no debe recargar los datos**
Versión	1.0
Descripción	El sistema está diseñado de forma que los datos de las diferentes vistas nunca se deben recargar manualmente, pues se muestran dinámicamente a medida que se actualizan.
Prioridad	Esencial

Tabla 41: El usuario no debe recargar los datos

RFRN-04	**Anuncio único por usuario**
Versión	1.0
Descripción	Un mismo usuario solo podrá tener publicado un único anuncio. Para cualquier cambio debe recurrir a la edición del anuncio.
Prioridad	Esencial

Tabla 42: Anuncio único por usuario

RFRN-05	**El servicio de localización permitirá retrasmitir localización al menos a creadores de anuncio**
Versión	1.0
Descripción	El sistema estará diseñado de forma que al menos los usuarios con rol de administrador pueden iniciar la retrasmisión de su localización en directo.
Prioridad	Esencial

Tabla 43: El servicio de localización permitirá retrasmitir localización al menos a creadores de anuncio

3.4.3 Requisitos funcionales de conducta

En los requisitos de este apartado se describen los comportamientos del sistema que no han sido especificados en los casos de uso de apartados anteriores.

RFC-01	**Mostrar error de conexión**
Versión	1.0
Descripción	Cuando entremos en una nueva vista, en caso de no tener conexión activa a internet se mostrará un diálogo informativo.
Prioridad	Alta

Tabla 44: Mostrar error de conexión

RFC-02	**Mostrar animaciones de carga de datos**
Versión	1.0
Descripción	El sistema mostrará animaciones circulares de carga, tanto en las listas de datos tales como anuncios o mensajes de chat, así como en las imágenes de avatar de usuarios.
Prioridad	Media

Tabla 45: Mostrar animaciones de carga de datos

RFC-03	**Mostrar lista de chats ordenada según las consultas más recientes**
Versión	1.0
Descripción	En la pantalla principal, los chats de los que formamos parte aparecerán ordenados de más recientes a menos recientes. Al salir de un chat volveremos a esta pantalla y se actualizará la lista de chats para que esté en primera posición el último que se consultó.
Prioridad	Alta

Tabla 46: Mostrar lista de chats ordenada según las consultas más recientes

RFC-04	**Mostrar pantalla correspondiente al pulsar en la burbuja de notificación**
Versión	1.0
Descripción	El usuario podrá pulsar sobre las notificaciones nativas de Android que se implementan para los nuevos mensajes o nuevas retransmisiones de localización. Al hacerlo, la aplicación debe abrir la correspondiente pantalla de chat o de localización en directo, respectivamente.

Prioridad	Esencial

Tabla 47: Mostrar pantalla correspondiente al pulsar en la burbuja de notificación

RFC-05	**Mostrar imagen con iniciales de usuario cuando no existe avatar**
Versión	1.0
Descripción	La aplicación mostrará una imagen con color de fondo y con las dos iniciales del nombre del usuario en el caso de que no haya establecido un archivo de imagen como avatar.
Prioridad	Media

Tabla 48: Mostrar imagen con iniciales de usuario cuando no existe avatar

RFC-06	**Seguimiento dinámico del autor de retransmisión**
Versión	1.0
Descripción	La aplicación, en el caso de consulta de localización en directo, realizará un seguimiento del autor de la retransmisión que permitirá tenerlo en el centro de la pantalla sin tocarla.
Prioridad	Medio

Tabla 49: Seguimiento dinámico del autor de retransmisión

RFC-07	**Mostrar contador de mensajes de chat no leídos**
Versión	1.0
Descripción	La aplicación debe mostrar al usuario en la pantalla principal, dentro de la lista de chats recientes, un indicador por cada icono de chat con el número de mensajes que no ha leído desde la última vez que salió de ese chat.
Prioridad	Alta

Tabla 50: Mostrar contador de mensajes de chat no leídos

3.5 Requisitos no funcionales

3.5.1 Requisitos no funcionales de fiabilidad

Estos requisitos indican el nivel de tolerancia a fallos y robustez que el sistema debe cumplir como mínimo.

En nuestro caso, los requisitos de fiabilidad derivan directamente de los de las plataformas Google Firebase y Google Maps Platform, pues el sistema depende íntegramente de la conexión a sus servicios.

RNFF-01	Tiempo máximo de recuperación del sistema de 10 minutos
Versión	1.0
Descripción	El sistema deberá tardar como máximo 10 minutos tras un fallo externo o interno que provoque su caída.
Prioridad	Esencial

Tabla 51: Tiempo máximo de recuperación del sistema de 10 minutos

RNFF-02	Mínimo de accesibilidad del 99.9% del tiempo
Versión	1.0
Descripción	El sistema deberá permanecer accesible en todas las instancias y servicios Firebase y Maps Platform que se utilizan, en al menos el 99.9% del tiempo total de un mes de uso. Esto representa un tiempo máximo de inaccesibilidad de 43 minutos al mes.
Prioridad	Esencial

Tabla 52: Mínimo de accesibilidad del 99.9% del tiempo

3.5.2 Requisitos no funcionales de usabilidad

Los requisitos de usabilidad establecen varios aspectos relacionados con la experiencia de usuario y lo intuitivo de la interfaz del sistema a desarrollar.

RNFU-01	Máximo de 5 clics para acceder a información en el 75% de las veces
Versión	1.0
Descripción	Todas las funciones del sistema deben estar disponibles para el usuario a un máximo de 5 clics partiendo desde la pantalla principal.
Prioridad	Esencial

Tabla 53: Máximo de 5 clics para acceder a información en el 75% de las veces

RNFU-02	Interfaz que responda a toda interacción del usuario de forma visual
Versión	1.0
Descripción	El sistema debe estar diseñado de tal forma que las acciones en cualquier elemento con el que se pueda interaccionar, provoquen una respuesta visual que conforme una experiencia intuitiva de cara al usuario. Por ejemplo, todos los elementos clicables deben mostrar al menos un sombreado o "*ripple effect*" al ser pulsados que indique que lo son, o cualquier otro tipo de efecto

	visual.
Prioridad	Esencial

Tabla 54: Interfaz que responda a toda interacción del usuario de forma visual

RNFU-03	**La aplicación estará disponible al menos en español**
Versión	1.0
Descripción	El idioma disponible en el sistema será, al menos, el español de España.
Prioridad	Esencial

Tabla 55: La aplicación estará disponible al menos en español

3.5.3 Requisitos no funcionales de eficiencia

Los requisitos de eficiencia establecen niveles de efectividad en el uso de la aplicación.

RNFE-01	**Tiempo de respuesta inferior a 5 segundos**
Versión	1.0
Descripción	El tiempo que pasa entre que se clica una función y se muestra la siguiente con los datos cargados, deberá ser como máximo de 5 segundos.
Prioridad	Esencial

Tabla 56: Tiempo de respuesta inferior a 5 segundos

3.5.4 Requisitos no funcionales de mantenibilidad

Los requisitos de mantenibilidad establecen los niveles que debe cumplir el sistema a desarrollar en aspectos como facilidad de mantenimiento, análisis, cambio o pruebas.

RNFM-01	**El código que se implemente deberá cumplir las recomendaciones de la IEEE**
Versión	1.0
Descripción	El código fuente implementado durante el desarrollo deberá adaptarse a los estándares con el fin de facilitar su comprensión a los encargados de mantener el código.
Prioridad	Esencial

Tabla 57: El código que se implemente deberá cumplir las recomendaciones de la IEEE

RNFM-02	Gestión de servicios independiente del cliente
Versión	1.0
Descripción	La gestión de los diferentes servicios de los que hace uso la aplicación serán totalmente independientes de esta, evitando así la necesidad de actualizaciones cuando se requieran cambios en el lado del servidor.
Prioridad	Esencial

Tabla 58: Gestión de servicios independiente del cliente

3.5.5 Requisitos no funcionales de seguridad

Los requisitos de seguridad aluden a cuestiones de privacidad, protección de datos de usuario o robustez frente a ataques.

RNFS-01	Los datos almacenados en BBDD no deben ser accesibles sin estar autenticado en el sistema
Versión	1.0
Descripción	Los datos de los usuarios que se almacenan en la nube de Google por medio de Firebase no pueden ser accesibles ni en lectura ni en escritura sin un registro y posterior autenticación en la aplicación.
Prioridad	Esencial

Tabla 59: Los datos almacenados en BBDD no deben ser accesibles sin estar autenticado en el sistema

RNFS-02	Los servicios no deben ser accesibles sin estar autenticado en el sistema
Versión	1.0
Descripción	Los servicios que se utilizan en la aplicación por medio de la plataforma Firebase no deben ser utilizados por usuarios no autenticados en el sistema.
Prioridad	Esencial

Tabla 60: Los servicios no deben ser accesibles sin estar autenticado en el sistema

RNFS-03	Un usuario no deberá acceder a datos que no le corresponde ver
Versión	1.0
Descripción	Un usuario registrado y autenticado en la aplicación podrá acceder a sus datos personales y a los que le corresponda ver, pero nunca tendrá manera de acceder a los datos personales de otros usuarios o datos que no le corresponde ver.

Prioridad	Esencial

Tabla 61: Un usuario no deberá acceder a datos que no le corresponde ver

3.5.6 Requisitos no funcionales de portabilidad

Los requisitos de portabilidad establecen los niveles que debe cumplir el sistema a desarrollar en aspectos relacionados con la escalabilidad.

RNFP-01	**Evitar uso de software propietario**
Versión	1.0
Descripción	En el caso de requerirse, el sistema deberá implementar librerías y código fuente de medios “open source” para evitar sobrecostes.
Prioridad	Esencial

Tabla 62: Evitar uso de software propietario

3.6 Prototipos de la aplicación

En este apartado se adjuntan las maquetas (*mockups*) o prototipos de diseño de la aplicación Android.

Los diseños fueron realizados en la plataforma InVision previo al comienzo del desarrollo, por lo que han sufrido cambios, bien por mejorar la idea primigenia o por no adecuarse del todo al sistema en la etapa de implementación.

Ilustración 20: Maqueta de pantalla de inicio de sesion

Ilustración 21: Maqueta de pantalla de registro

Si hablamos de la pantalla de inicio de sesión, en las primeras etapas del diseño se optó por variedad de

opciones de inicio de sesión: autenticación por correo y contraseña, por cuenta de Google o por cuenta de Facebook. Finalmente, se desechó la opción de autenticación con cuenta de Facebook porque el sistema ya se consideró suficientemente completo.

La pantalla de registro también ha sufrido cambios. Uno de los más importantes es el selector de avatar, que en el diseño final muestra la imagen que se ha seleccionado.

Ilustración 22: Maqueta de pantalla de anuncios

El mockup de pantalla de anuncios (o pantalla principal) inicialmente se diseñó como en la ilustración anterior. El diseño final incluye una lista de chats reciente que se ubica en la parte superior, además de un botón inferior extra que permite filtrar anuncios.

Adicionalmente, se planteó la siguiente vista diseñada en Adobe Photoshop para representar los anuncios:

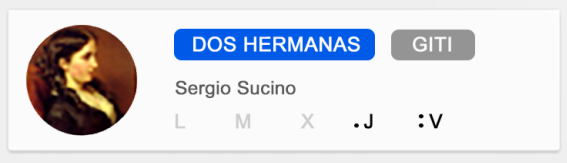


Ilustración 23: Prototipo de diseño de anuncio

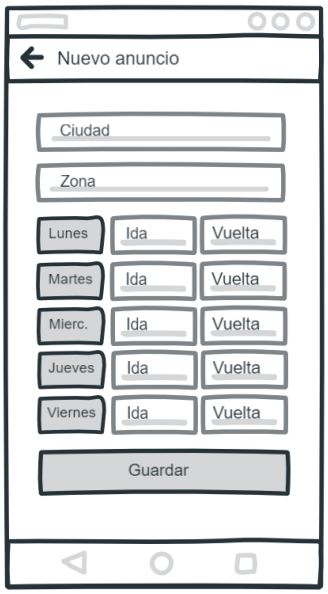


Ilustración 24: Maqueta de pantalla de nuevo anuncio

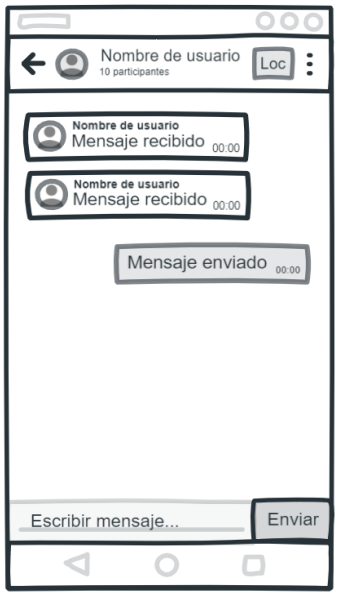


Ilustración 25: Maqueta de pantalla de chat

En el chat también se han realizado algunos cambios en el diseño de los mensajes y la parte inferior de introducción de mensajes.

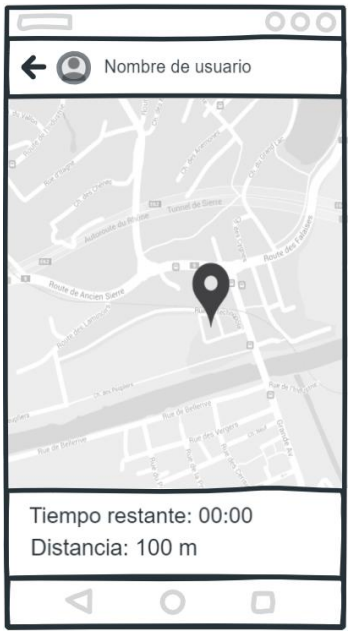


Ilustración 26: Maqueta de pantalla de localización

La pantalla de localización ha sufrido un rediseño con líneas más modernas, pero el esquema sigue siendo el mismo.

4 TECNOLOGÍAS

En este capítulo se describirá las tecnologías en la que se basa el proyecto. Además, se aportan diagramas de la arquitectura y fragmentos de código útiles para entender lo que supone trabajar con estas plataformas, además de para comprender mejor el funcionamiento del sistema.

4.1 La plataforma Firebase

Ilustración 27: Logo de Firebase

Firebase es una plataforma de de desarrollo de aplicaciones móviles y web creada por Firebase Inc. en 2011 y más tarde adquirida por Google en 2014. Su primer producto fue Firebase Real-time Database, aunque actualmente posee 18 productos que son usados por 1.5 millones de aplicaciones.

La plataforma se encuentra integrada en la nube en Google Cloud Platform, lo que aporta una gran cantidad de ventajas a los desarrolladores que la utilizan:

- Está integrada en la **infraestuctura de Google**, permitiendo escalar automáticamente cualquier tipo de tamaño de aplicación.
- **Sincronización sencilla** de los datos de los proyectos sin necesidad de administrar conexiones o escribir lógica compleja para sincronización. Esto puede verse en la Ilustración 28.
- Usa un conjunto de **herramientas multiplataforma** que se integran fácilmente tanto en web como en aplicaciones móviles. Es compatible con grandes plataformas como Android, iOS, aplicaciones web, Unity y C++.
- Como servicio BaaS, permite crear proyectos **sin necesidad de servidor**, por lo que no es necesario un esfuerzo extra de administración de un backend tradicional. Las herramientas se incluyen en las librerías, que se comunican automáticamente con los servidores necesarios.

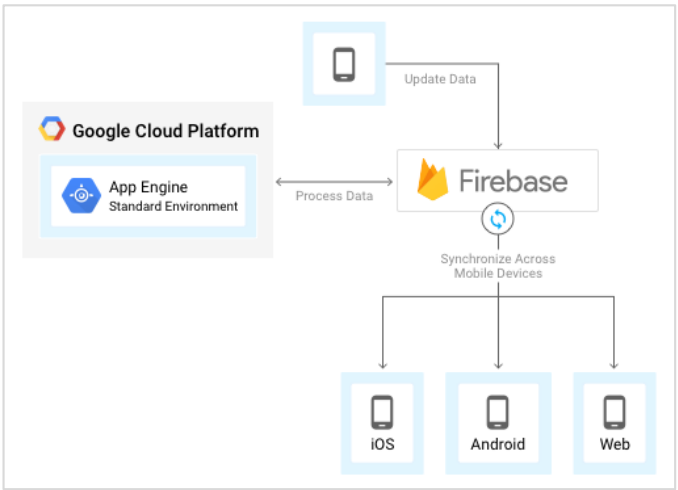


Ilustración 28: Ejemplo de arquitectura de Firebase en sincronización de datos

Firebase cuenta con una gran cantidad de documentación gratuita y disponible para los usuarios, gran actividad en GitHub y StackOverflow. También tienen su propia conferencia anual de desarrolladores, donde ahondan en los productos de la plataforma y suben el contenido a Youtube.

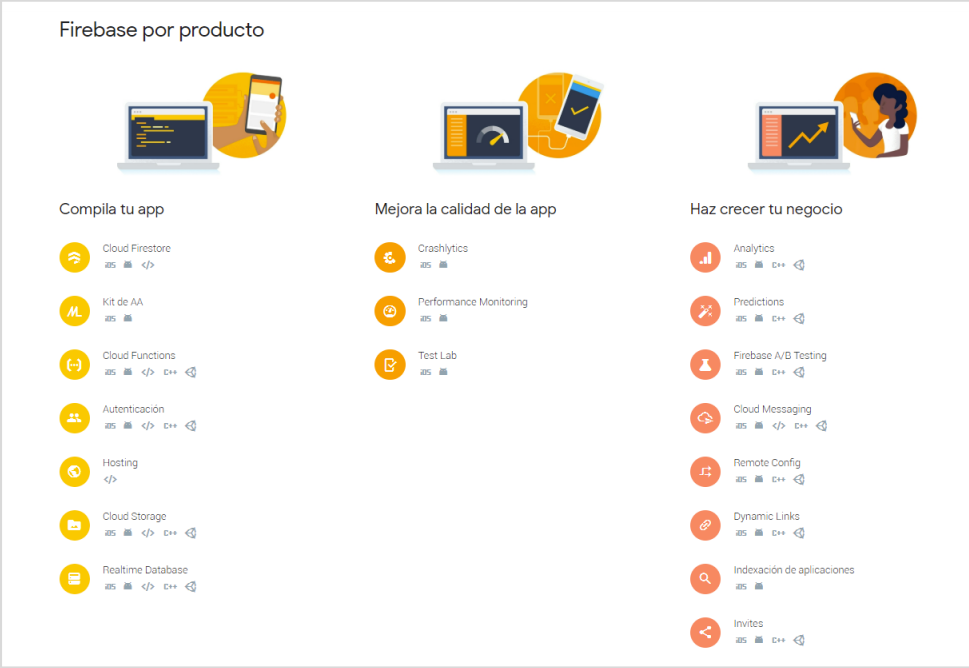


Ilustración 29: Productos disponibles en Firebase

Como se ve en la imagen anterior, la plataforma dispone de una gran cantidad de productos para diferentes plataformas, pero nosotros entraremos en profundidad solo en los que competen a nuestra aplicación.

4.1.1 Firebase Realtime Database

A través de esta herramienta, Firebase nos proporciona acceso a una base de datos en tiempo real en la nube y a un *backend as a service*. Se trata de una base de datos NoSQL con formato de datos JSON.

Realtime Database se sincroniza en tiempo real con cada cliente conectado que acceda por medio de la biblioteca proporcionada. También es accesible por medio de REST API y puede integrarse con gran cantidad de plataformas.

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. Esto está representado en la Ilustración siguiente.

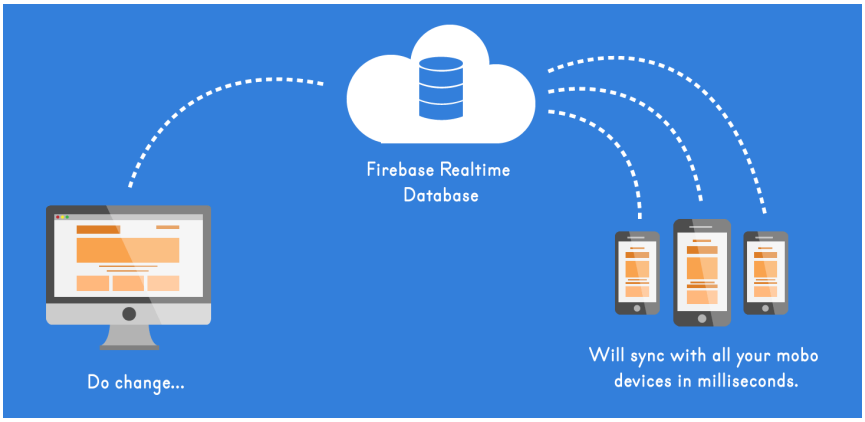


Ilustración 30: Esquema de ejemplo de Realtime Database

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

4.1.2 Firebase Authentication

Firebase Authentication (también llamado 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, así como cuotas de registro para los usuarios. Todo esto gestionado mediante los servidores de la plataforma.

4.1.3 Cloud Storage

Firebase Cloud Storage es una solución similar a Realtime Database pero enfocada en el almacenamiento de archivos y contenido generado por los usuarios, tales como imágenes o vídeo. Esto es representado en la siguiente imagen.

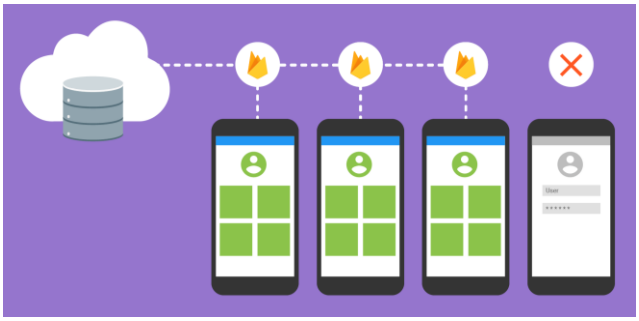


Ilustración 31: Esquema de Firebase Storage

Proporciona cargas y descargas seguras de archivos para aplicaciones Firebase, sin importar la calidad de la red. Se basa en el almacenamiento de Google Cloud Storage, por lo que posee gran potencial de escalabilidad. Está diseñado para escalar a exabytes si se diera el caso de que una aplicación se vuelve viral o sufre un pico de transferencias por cualquier otro motivo.

4.1.4 Firebase Cloud Messaging

Firebase Cloud Messaging (FCM), también conocida por Google Cloud Messaging (GCM) por su dependencia con la arquitectua de la que hace uso, es una solución multiplataforma en la nube para la gestión de notificaciones y mensajes. Está disponible para Android, iOS y aplicaciones web.

FCM puede lanzar notificaciones push a una app cliente o un grupo de clientes en tiempo real tan solo implementando sus bibliotecas.

También permite enviar notificaciones globales personalizables a grupos de usuarios desde la propia plataforma. Está enfocado, por ejemplo, a promociones y eventos donde se requiere una notificación del administrador de la aplicación hacia todos los usuarios.

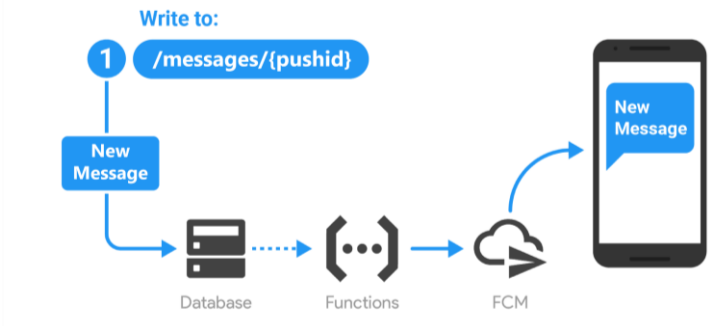


Ilustración 32: Ejemplo de uso de FCM

Como representa la ilustración anterior, mediante la interacción entre diferentes productos de Firebase podemos lograr reacciones en cadena que finalicen en la notificación de clientes. Esto aporta gran valor al software de cara al usuario con un mínimo impacto en el desarrollo, comparado con la solución tradicional de construcción de un servidor de notificaciones que implemente XMPP.

4.1.5 Functions

Firebase Functions es una solución serverless que permite activar eventos automáticamente desde el backend. Estos eventos son activados por otras funciones de Firebase o por peticiones HTTPS.

El código de los métodos publicados en Functions se almacena y ejecuta en un entorno seguro de la nube de Google. Firebase Functions realmente es una pasarela que permite el uso de Google Cloud Functions desde el backend de una aplicación.

Como vimos en la ilustración que acompañaba al apartado de Cloud Messaging, se puede integrar una cadena de acciones de productos de Firebase activada por Functions.

En la página de documentación oficial pueden encontrarse múltiples ejemplos útiles, como emplear Functions en tareas con un procesado intensivo y así aliviar al dispositivo [6]. Uno de ellos es la imagen que se adjunta a continuación.

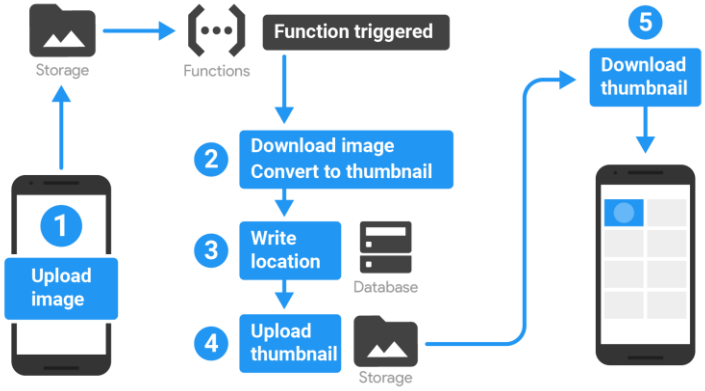


Ilustración 33: Ejemplo de uso de Functions

4.2 Google Maps Platform

La plataforma de Google Maps es el conjunto de herramientas y soluciones enfocadas a dar a los desarrolladores servicios de mapas interactivos similares a la propia Google Maps. Su tecnología es fácilmente aplicable a cualquier aplicación o página web de sectores específicos. Gracias a ella se puede disponer de los datos que interesen siempre actualizados y en tiempo real.

Tiene tres grandes grupos de API [7]:

- **Maps**: contiene las librerías básicas necesarias para incorporar mapas a aplicaciones móviles o web y la función de Street View para navegación inmersiva en 360º. Es gratuito.
- **Routes**: permite facilitar a los usuarios la información para poder ir de un sitio a otro, determinar la ruta que recorre un vehículo y crear itinerarios con indicaciones precisas y actualizaciones de tráfico en tiempo real. Es de pago.
- **Places**: permite buscar lugares basados en datos de ubicación concretos como números de teléfono, coordenadas o direcciones en tiempo real, así como cualquier información disponible sobre esos lugares. Es de pago.

En el sistema que tratamos, se han utilizado las funciones de **Maps** (grupo Maps) y de **Directions (**grupo Routes).

5 APLICACIÓN ANDROID

En este apartado explicaremos con mayor profundidad todas las funcionalidades de la aplicación.

5.1 Modelo de datos

La base de datos en la nube a la que accedemos por medio de Firebase Realtime Database sigue un formato NoSQL en JSON. Los objetos se envían para escritura en base de datos y son serializados durante el proceso.

Dicho esto, el modelo de datos es el siguiente:

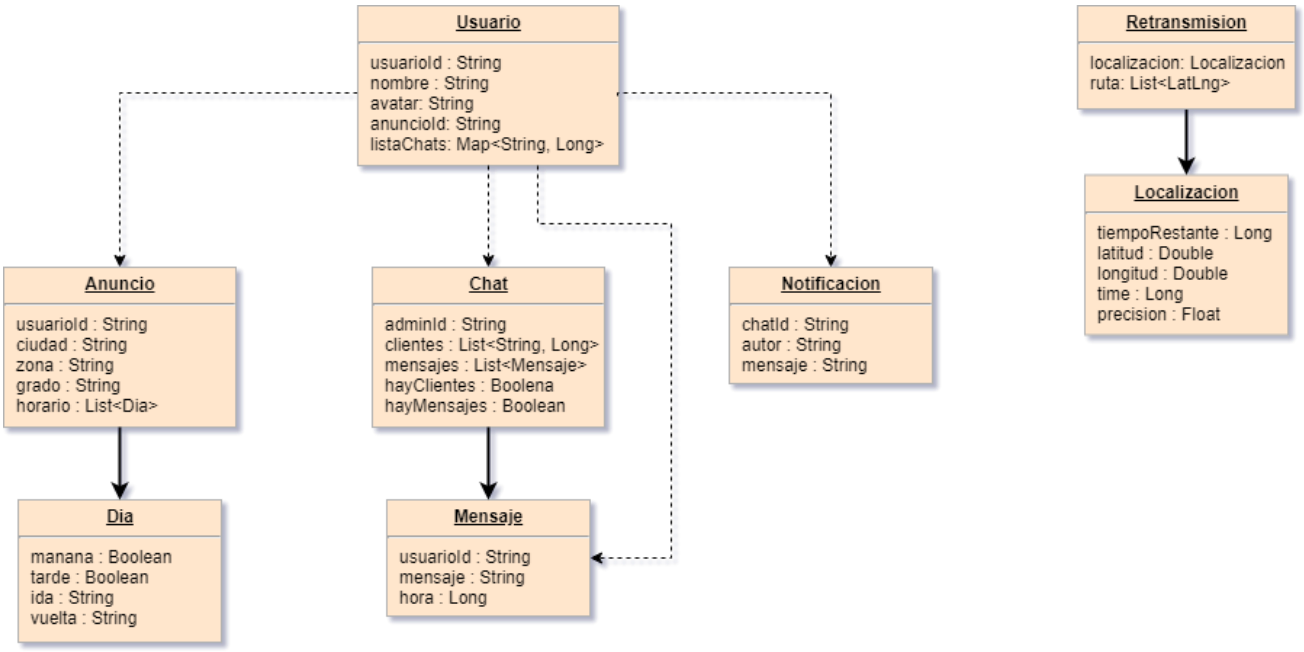


Ilustración 34: Modelo de datos de la aplicación

Hay que hacer un apunte sobre los datos del modelo: los atributos usuarioId, adminId y chatId **significan lo mismo**.

Es decir, un usuario con identificador *usuarioId*, crea un anuncio al que se le asocia un chat con identificador *chatId* igual al identificador de su creador. Este chat identifica a su usuario con rol de administrador con *adminId*, que a su vez es el mismo que el identificador de chat.

Esto, que a priori resulta redundante, facilita las tareas de petición de datos a Realtime Database.

5.1.1 Diagramas de actividad

En este apartado se adjuntarán los diagramas de actividad que reflejan todas las acciones que se pueden realizar por parte del usuario en la aplicación y el comportamiento a consecuencia de ello.

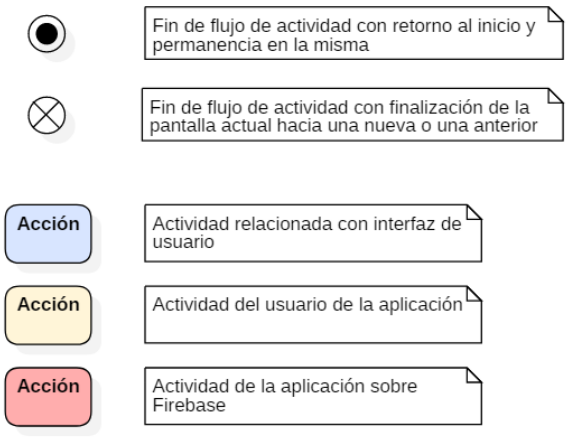


Ilustración 35: Leyenda de fin de flujos de actividad

En la primera imagen, se representa el subsistema referente al usuario. Contempla todas las acciones que se pueden realizar en la pantalla de inicio de sesión y en la pantalla de registro relacionadas con los datos de usuario.

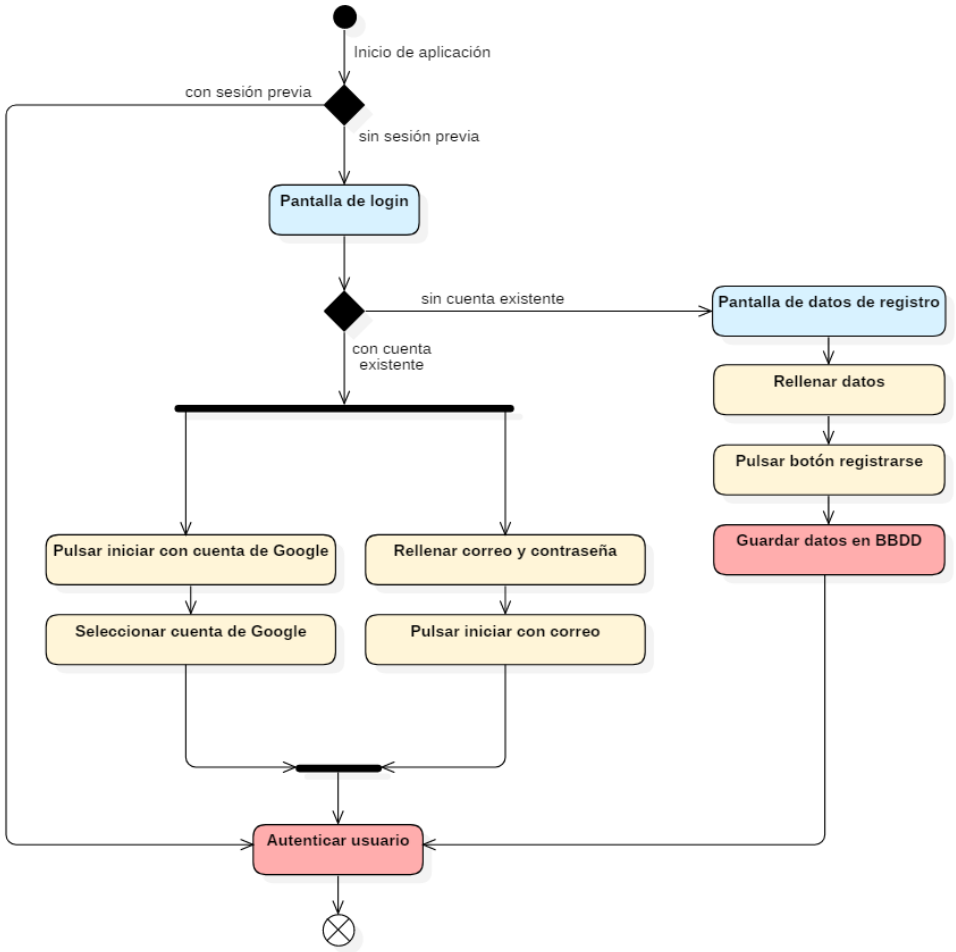


Ilustración 36: Diagrama de actividad de subsistema Usuarios

En segundo lugar, se representa el subsistema de anuncios. En el diagrama se pueden ver todas las posibles acciones a relaizar en la pantalla principal o de anuncio.

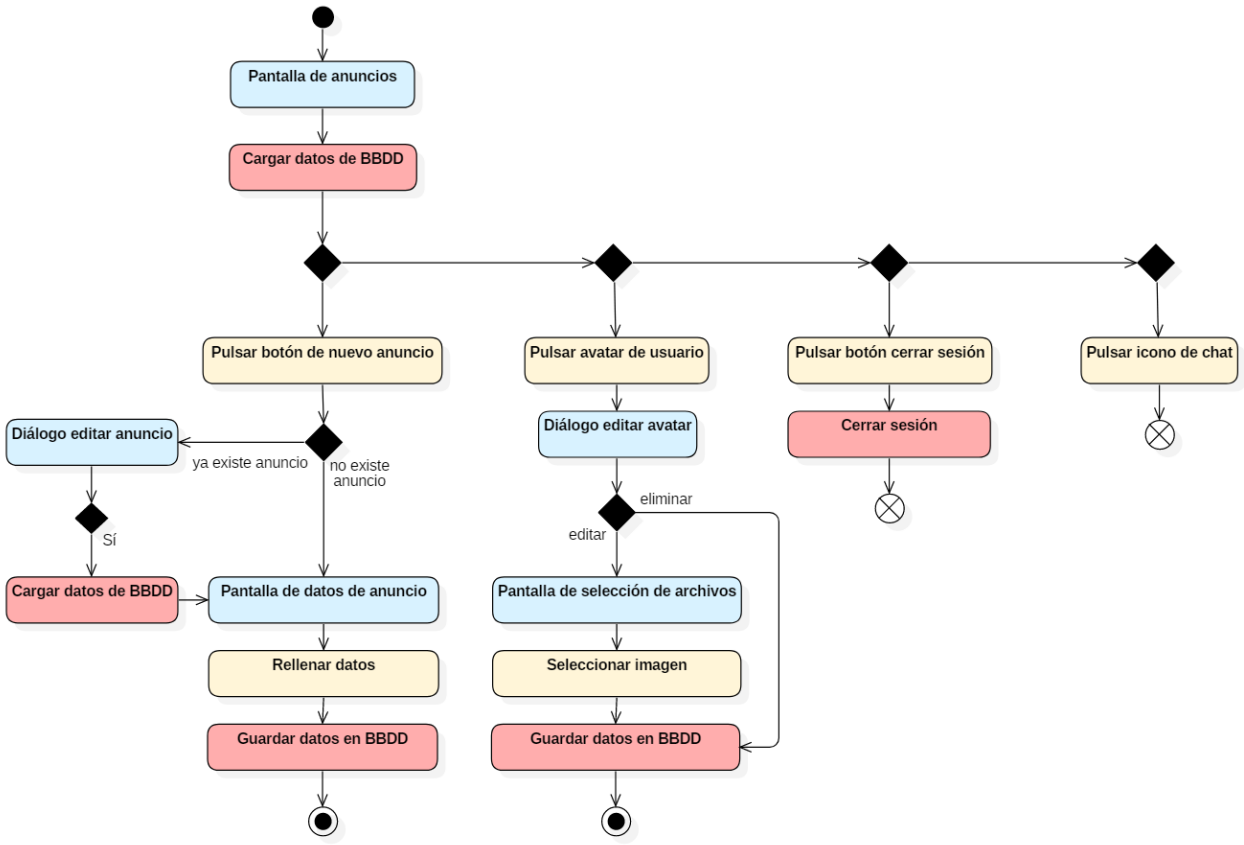


Ilustración 37: Diagrama de actividad de subsistema Anuncios

La siguiente representación hace referencia a la pantalla de chat y todas las actividades que se pueden realizar sobre ella.

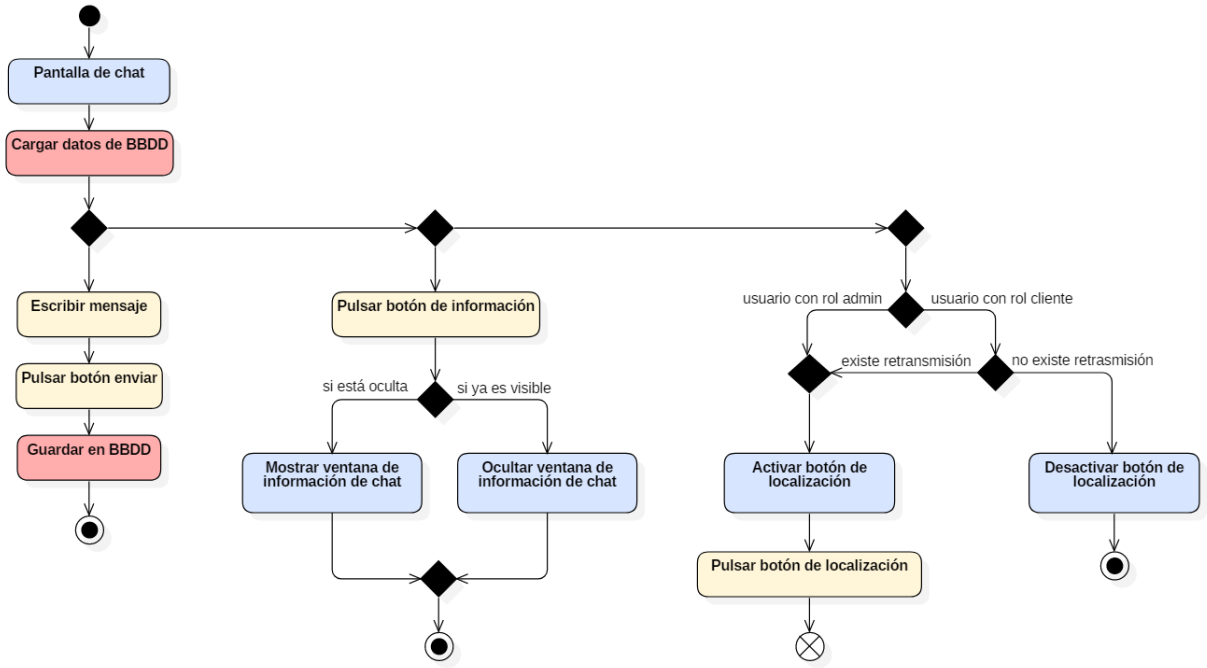


Ilustración 38: Diagrama de actividad de subsistema Chat

Finalmente, tenemos el subsistema de localización, con todas las posibilidades de acción que ofrece.

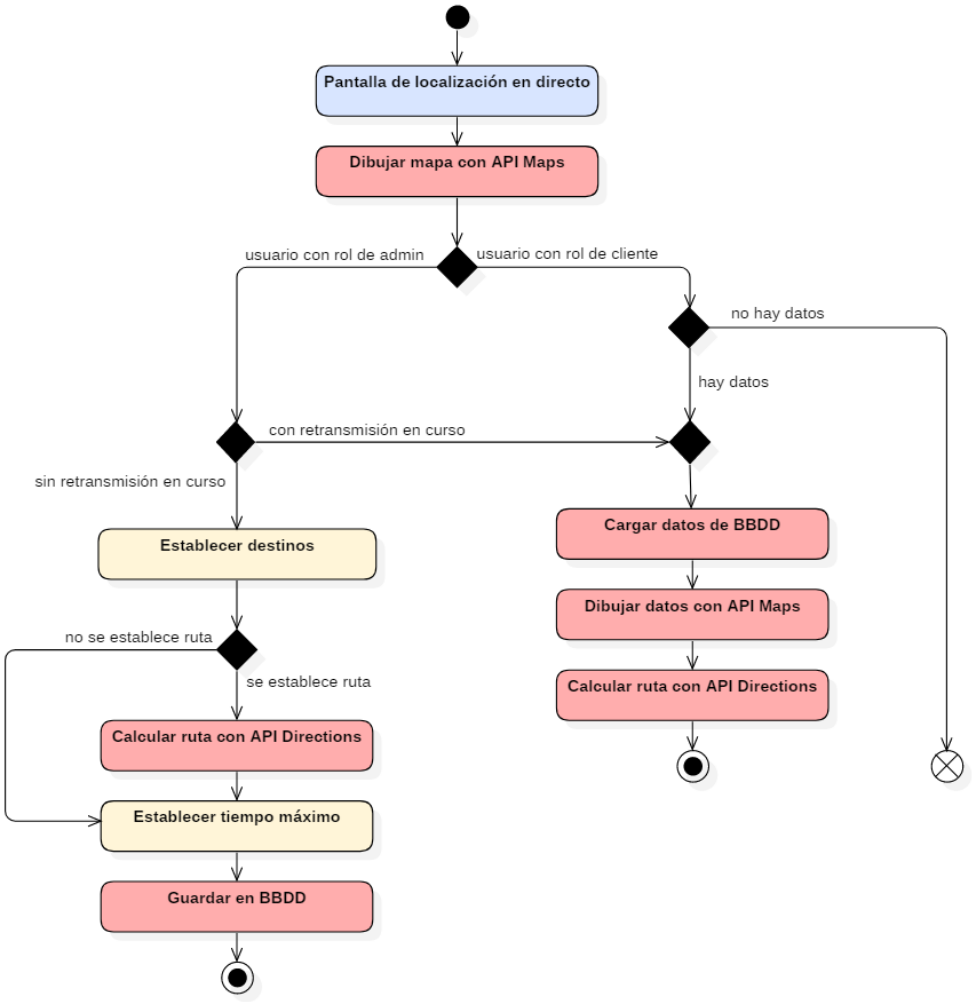


Ilustración 39: Diagrama de actividad de subsistema Localización

5.2 Interfaz gráfica

En este apartado se describirá con detalle el diseño que se ha implementado para la interfaz gráfica de la aplicación. A continuación, se adjunta un esquema con todas las pantallas entre las que se puede navegar y su relación.

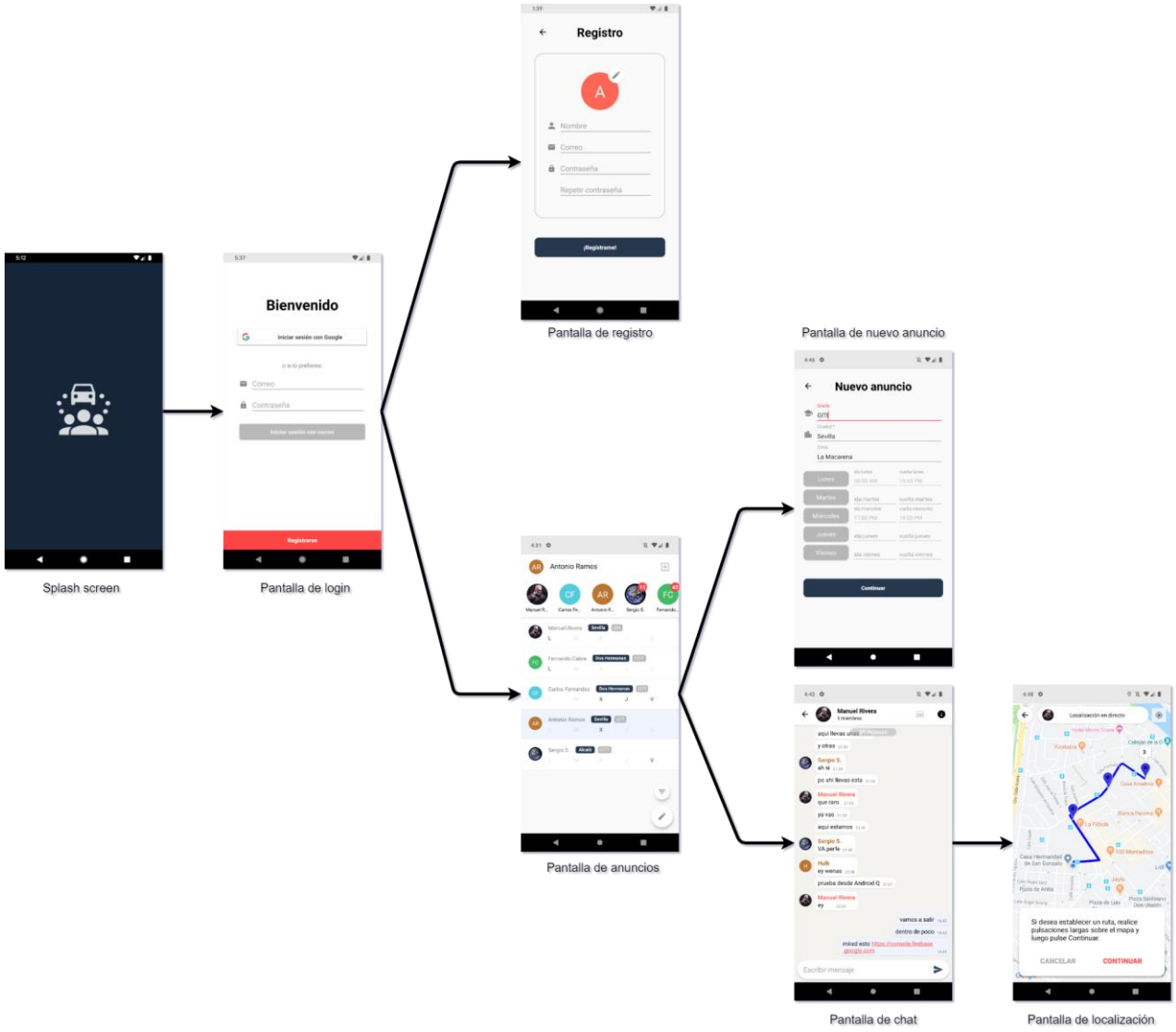


Ilustración 40: Flujo de navegación de pantallas

La **interfaz de usuario** de la aplicación que nos ocupa ha sido diseñada con la **experiencia de usuario** (UX) en el punto de mira. Los colores y las formas se han elegido de forma que resulte agradable a la vista y siguiendo las líneas de diseño más modernas del paradigma móvil.

Además, se ha establecido como requisito que toda interacción con elementos de la interfaz responda de alguna manera, resultando intuitiva y fácil de utilizar. Esto puede comprobarse en todos los elementos pulsables, que producen un efecto de sombra o *ripple effect*.

En definitiva, el apartado de diseño se ha acometido como si se tratara de un producto que se va a lanzar para consumidores reales.

Las pantallas que podemos encontrar en la aplicación son las siguientes:

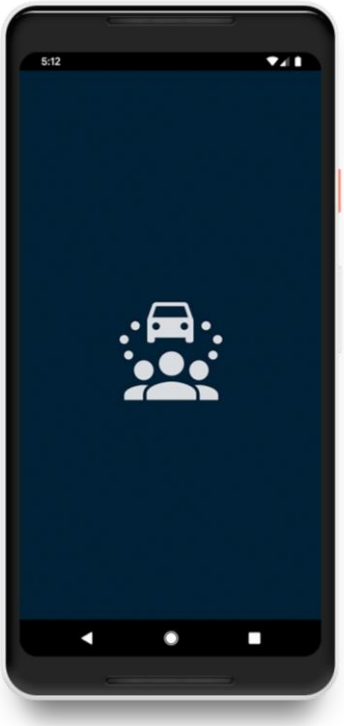
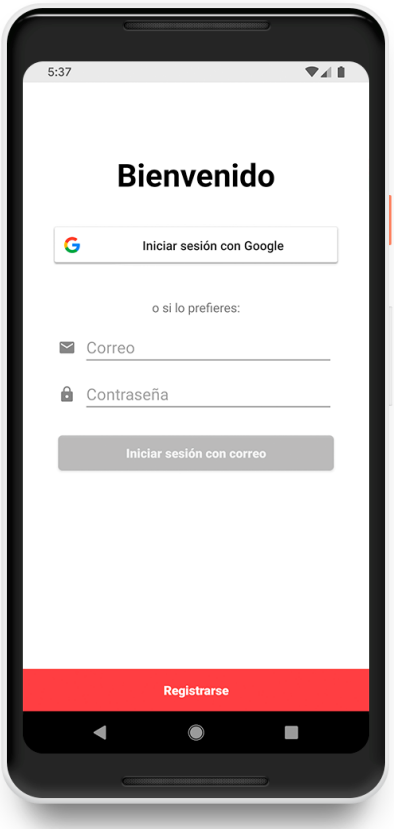


Ilustración 41: pantalla de carga inicial de la app

- Cuano se inicia la aplicación, aparece una **pantalla de carga inicial** o *splash screen* con un logo que se ha diseñado especialmente para la aplicación.



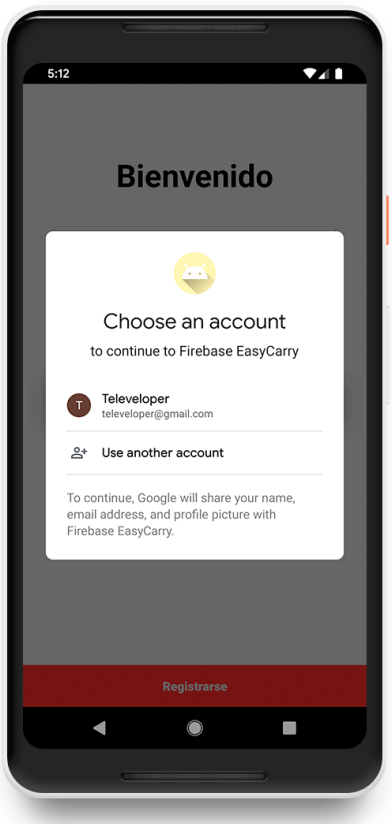


Ilustración 42: pantalla de inicio de sesión

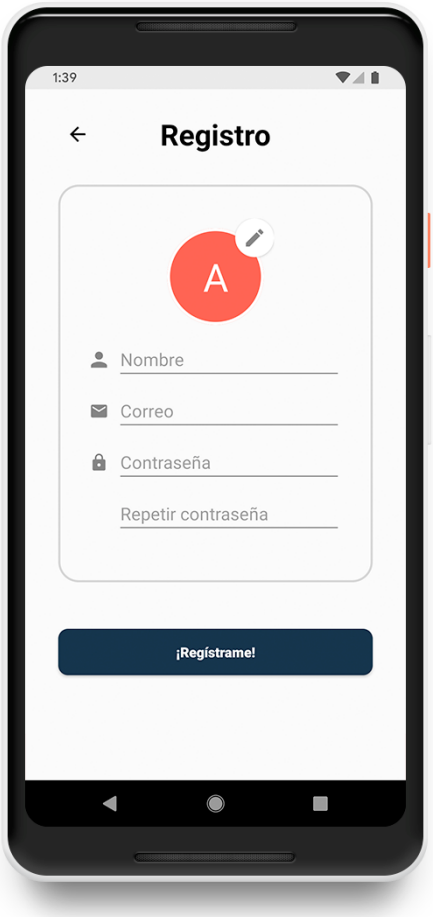
Ilustración 43: pantalla de inicio de sesión con Google

- A continuación se abre la **pantalla de login**. Este *Activity* aparecerá siempre que no hayamos dejado la sesión iniciada la última vez que cerramos la aplicación.

  Contiene las opciones de inicio con cuenta de Google o con correo y contraseña:

  - Cuando pulsamos iniciar con cuenta de Google, se lanza un *Intent* que abre el menú de selección de la cuenta, donde aparecerán las que existen en el dispositivo. Al elegir una se incia sesión automáticamente a la vez que se lanza una ventana (*ProgressDialog)* de espera.
  - Cuando escribimos el correo y la contraseña de nuestra cuenta y pulsamos en iniciar sesión, puede lanzar un error en forma de mensaje *Toast* si el correo o la contraseña son incorrectos. Cuando la introducción sea correcta, se incia sesión con diálogo de espera.

  Con el botón de registro inferior se lanza el *Intent* de la pantalla de registro.



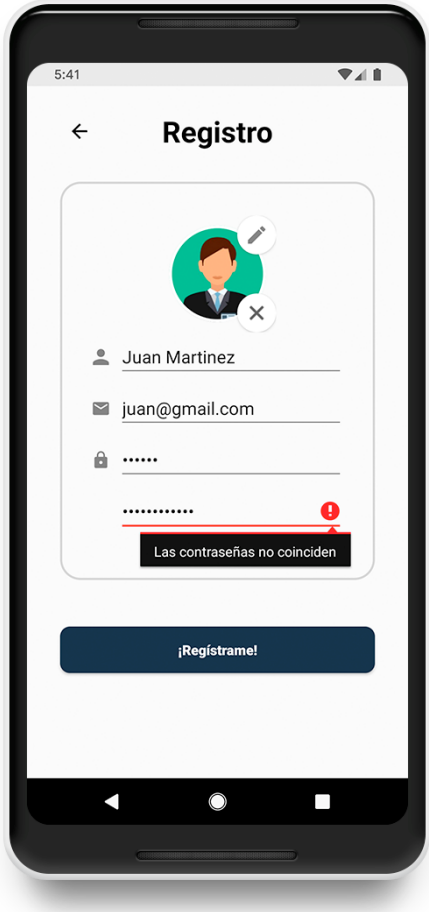


Ilustración 44: pantalla de registro

Ilustración 45: pantalla de registro con datos

- La **pantalla de registro** contiene un formulario necesario para crear un nuevo usuario que se va a autenticar con el método de correo y contraseña.
  - El selector de imagen de perfil lanza un *Intent* con el explorador de archivos de Android. Al seleccionar una imagen, nos devuelve al formulario de registro con el *AvatarImageView* establecido de forma provisional. Se puede eliminar la imagen o cambiarla por otra.

- Cuando hay algún error en los campos *EditText*, los cuales son todos obligatorios, lo muestra por pantalla como se ve en la Ilustracion 24.

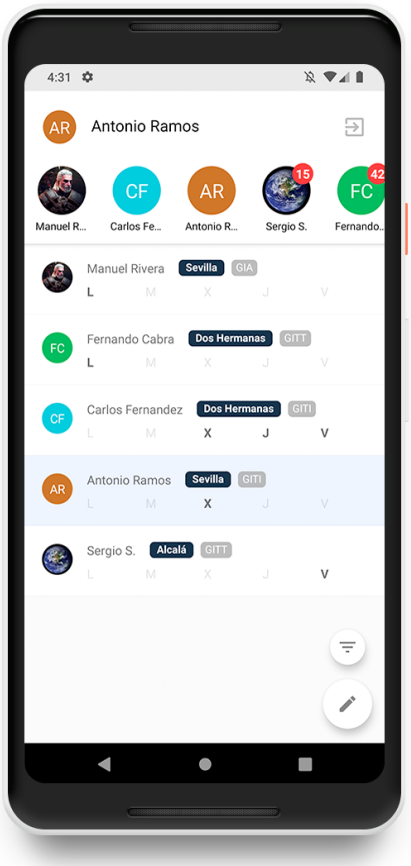


Ilustración 46: pantalla de anuncios

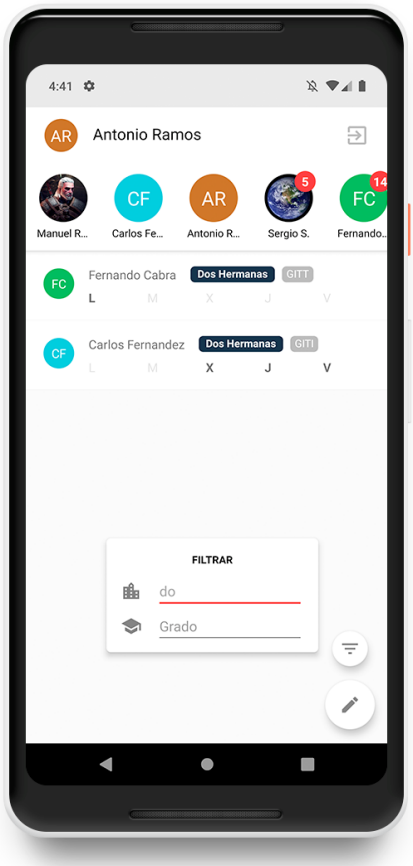


Ilustración 47: pantalla de anuncios con filtro

- Tras el inicio de sesión, pasamos a la **pantalla de anuncios**. Es la pantalla principal de la aplicación en cuanto a que es la fuente de las actividades principales de la misma.

  Como puede verse en la Ilustración 25, está compuesta por:

  - *Toolbar* superior, donde se encuentra el avatar y nombre del usuario. Al pulsar el avatar se mostratá un cuadro de diálogo para elegir si eliminarlo o cambiarlo por otra imagen.

    También tenemos a la derecha el botón para cerrar sesión, el cual nos llevará a la pantalla de login.

  - Lista de chats recientes. *RecyclerView* que contiene el avatar y nombre de los creadores de los chats a los que estamos suscritos. Cada uno tiene un indicador de mensajes no leídos si se diera el caso. Al pulsarlos se lanza una pantalla de chat que nos lleva al que hemos elegido.

  - Lista de anuncios. Todos los anuncios publicados por los usuarios se mostratán en este *RecyclerView*. Cada elemento tiene información resumida sobre el anuncio, y al pulsarlos nos suscribirá y mostrará el chat relacionado.

  - Botón de filtrado. *FloatingActionButton* que abre un cuadro de diálogo donde se filtran los anuncios a medida que escribimos según ciudad o grado.

  - Botón de creación de anuncio. *FloatingActionButton* que lleva a la pantalla de creación o edición de anuncio. Se explica a continuación

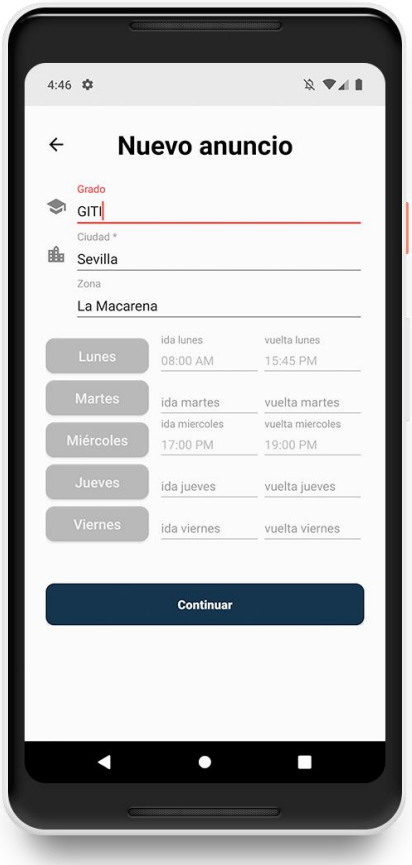


Ilustración 48: pantalla de nuevo anuncio

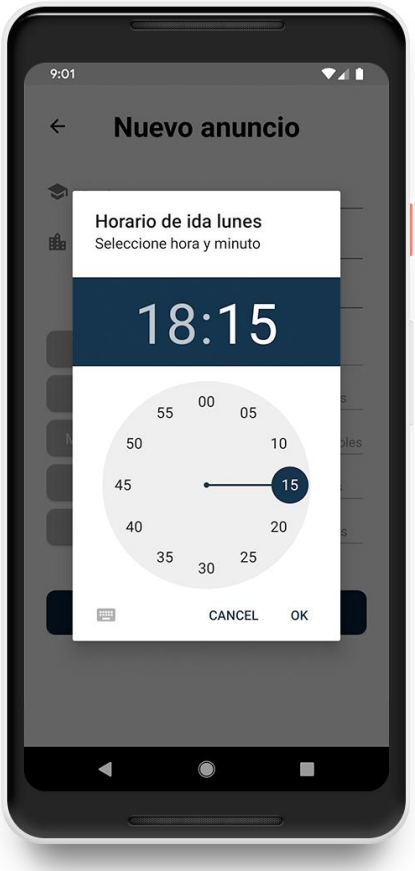


Ilustración 49: pantalla de nuevo anuncio con selector de hora

- La **pantalla de nuevo anuncio** es la que nos permite publicar o modificar un anuncio. Tiene campos de texto y campos horarios *TextInputLayout*.

  Los campos horarios son opcionales y permiten confeccionar el horario del creador del anuncio. Al pulsar el botón de un día de la semana, se muestra un selector de hora nativo o *TimePicker* que nos permite elegir cómodamente el horario. Puede verse en la Ilustración 28.

  Si el campo obligatorio es correcto, al pulsar el botón Continuar nos devuelve a la pantalla de anuncios con nuestro anuncio ya creado.

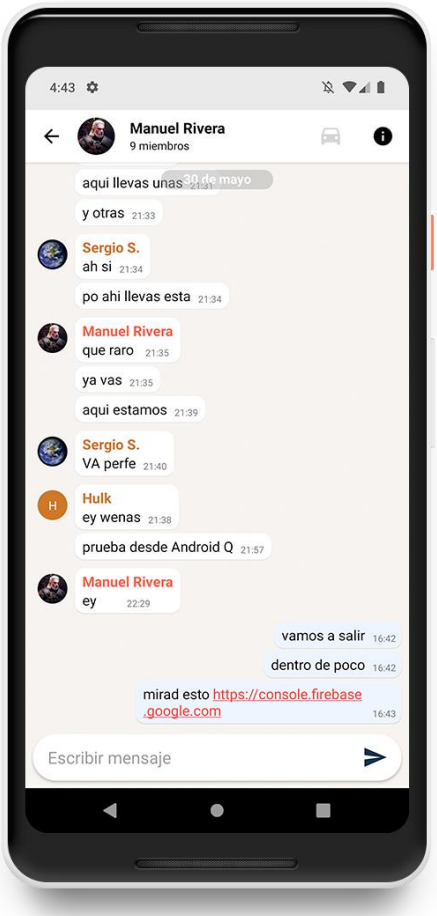


Ilustración 50: pantalla de chat

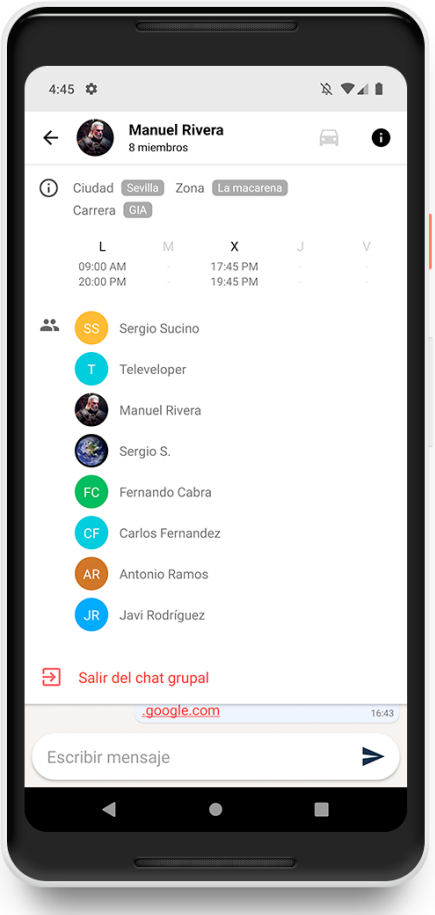


Ilustración 51: pantalla de chat con menú de información

- En cuanto a la **pantalla de chat**, tiene varias partes de su interfaz a destacar:
  - *Toolbar* superior. Nos muestra el avatar y nombre del creador del anuncio asociado al chat. Además, los botones de acceso a la función de localización y de despliegue del menú de información.
  - Menú de información. Este *Fragment* representa los datos del anuncio de forma detallada, así como la lista de personas suscritas al chat (*RecyclerView*) y el botón para dejar de formar parte de este. Al pulsarlo nos lleva a la pantalla de anuncios y elimina el chat de chats recientes.
  - Lista de mensajes. *RecyclerView* donde se encuentra la principal fuente de datos de esta pantalla: los mensajes intercambiados entre los usuarios. Solo carga los mensajes a partir del momento en el que nos unimos al grupo. Su diseño se ha inspirado en las principales aplicaciones de mensajería del mercado.
  - Introducción de texto de mensaje. Permite escribir texto y pulsar el botón para enviarlo a base de datos.

- La **pantalla de localización** es la que realiza la función de retransmisión de localización. Está compuesta por una barra superior (*ConstraitLayout*) que nos permite ir a la posición del conductor o a la nuestra, un mapa interactivo de la librería de Maps y una serie de cuadros de diálogo informativos.

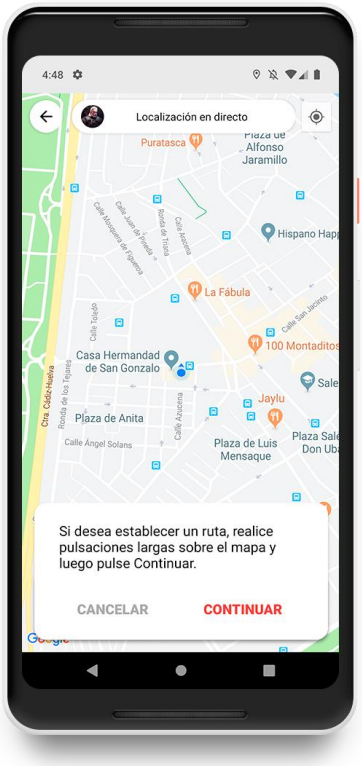


Ilustración 52: pantalla de localización de administrador

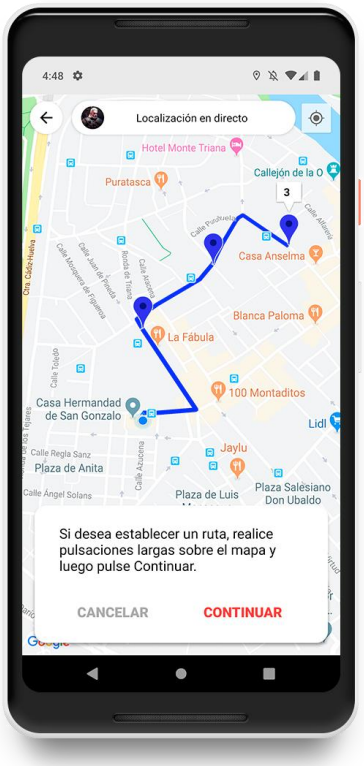


Ilustración 53: pantalla de localización con establecimiento de ruta

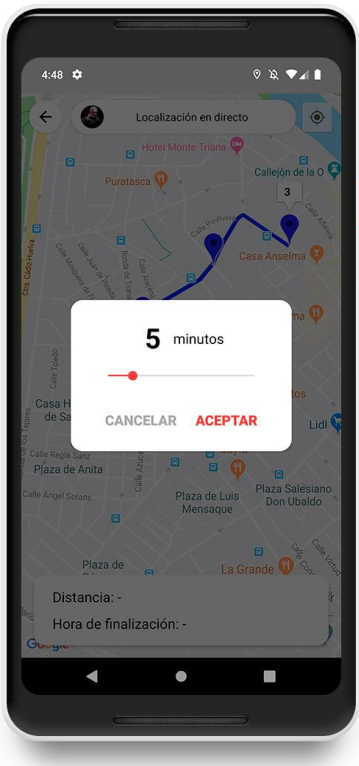


Ilustración 54: pantalla de chat con establecimiento de tiempo

Se puede dividir en dos ámbitos:

- Cuando el usuario que pulsa el botón de localización del chat es el **creador del chat** (rol de administrador):
  - Se abre el mapa junto a un cuadro de diálogo *ConstraitLayout* que nos propone establecer una ruta a seguir, como se puede ver en la Ilustracion 29. Esta ruta será compartida también hacia los clientes. Con pulsaciones largas se establecen los lugares que se quieren marcar.
  - Tras guardar la ruta, se muestra otro cuadro *AlertDialog* donde especificar el tiempo que durará la retransmisión de la localización. Puede verse en la Ilustración 31.

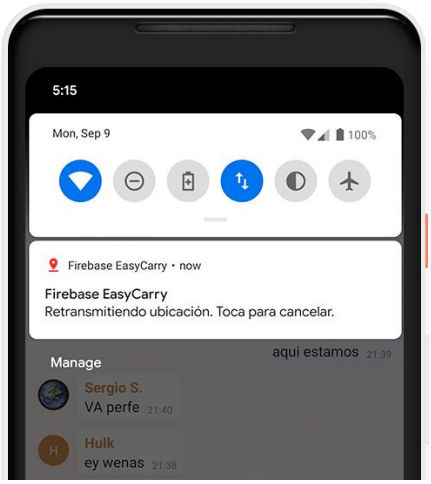


Ilustración 55: notificación de retransmisión desde dispositivo

- Cuando se acepta esto último, lleva al administrador de vuelta a su chat y comienza la retransmisión, a la vez que se genera una notificación permanente que avisa de que se está retransmitiendo. Si se pulsa la notificación, cancela el proceso.

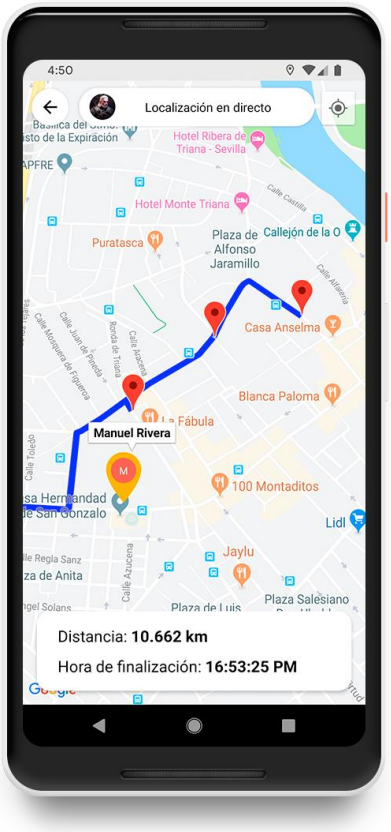


Ilustración 56: pantalla localización de cliente

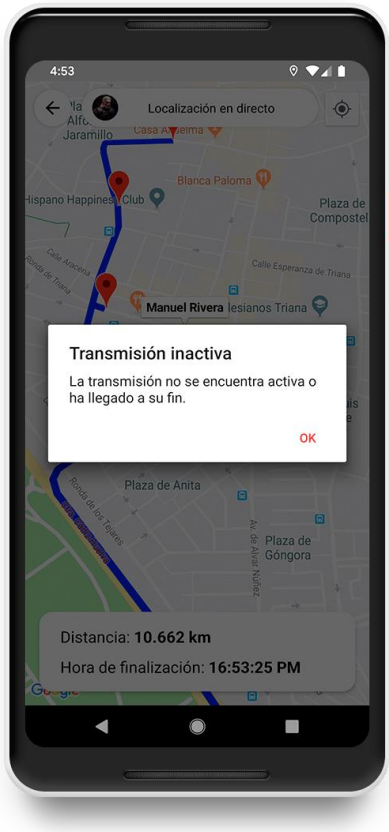


Ilustración 57: pantalla de localización de cliente finalizada

- Cuando el usuario que pulsa el botón de localización del chat es un **miembro del grupo** (rol de cliente):
    - Se cargan los datos de la retransmisión en el mapa. Se dibuja la ruta y se va actualizando la posición del *Marker* del administrador en tiempo real. El tiempo de actualización es aproximadamente 5 segundos.
    - En el cuadro informativo inferior (*ConstraitLayout*) se muestran la distancia a la que estamos del conductor y la hora a la que finalizará la retransmisión.
    - Cuando finaliza el tiempo o el creador de la retransmisión la cancela, se muestra una *AlertDialog* informativa que nos dice que está inactiva. Al confirmar, nos lleva de vuelta al chat asociado.

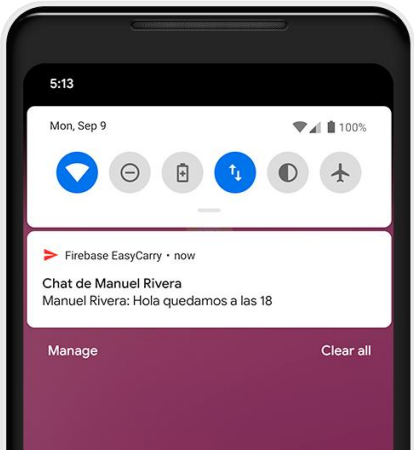


Ilustración 58: notificación de nuevo mensaje

Ilustración 59: notificación de nueva retransmisión

Mención especial merecen las **notificaciones** proporcionadas por Firebase Cloud Messaging. Estas se producen cuando nos suscribimos a un chat, momento en el que a su vez nos suscribimos al *topic* correspondiente al chat.

El aspecto de las notificaciones podemos verlo en las Ilustraciones 37 y 38. Al pulsar sobre ellas, se abre la aplicación lanzando un *Intent* con el chat correspondiente.

5.3 Implementación

Vamos a tratar aspectos relacionados con el desarrollo y programación del proyecto de Android.

La estructura del proyecto en Android Studio es la siguiente:

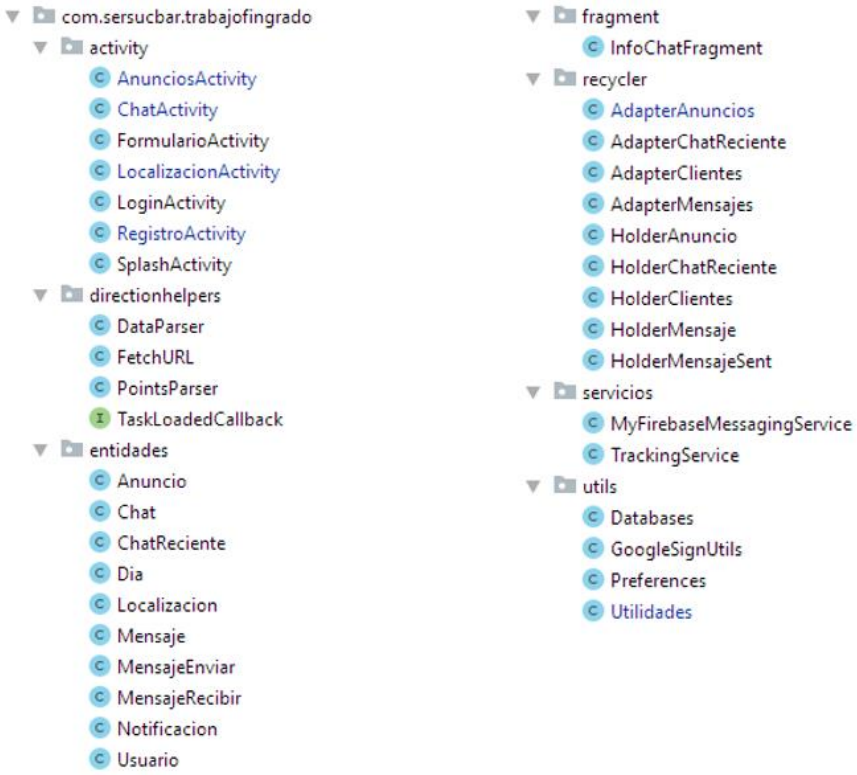


Ilustración 60: Estructura del proyecto de Android Studio

Los paquetes de clases reunen las siguientes funciones de la aplicación:

- *activity*: contiene las pantallas principales de la aplicación y todo lo que ocurre dentro de ellas.
- *directionhelpers*: contiene las clases necesarias para el cálculo y dibujado de la ruta en la pantalla de localización.
- *entidades*: contiene las clases que se corresponden con el modelo de base de datos.
- *fragment*: contiene la lógica de los elementos *Fragment* que se utilizan en la aplicación. En este caso solo el menú de información de chat.
- *recycler*: contiene las clases de tipo Adapter y Holder necesarias para implementar listas de items dinámicas.
- *servicios*: contiene los servicios que se ejecutan en segundo plano en la aplicación.
- *utils*: contiene clases con métodos comunes o útiles para el proyecto.

Los permisos necesarios en el archivo *Manifest.xml* para el correcto fucionamiento de la aplicación son los siguientes:

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

Código 6: Permisos de la aplicación en el manifiesto

5.3.1 Problemas encontrados

A lo largo del desarrollo del proyecto, se han sucedido varios retos o implementaciones problemáticas. Estas son algunas de las más importantes:

- **Prototipos de módulos principales**.

  Al comenzar el desarrollo, la primera dificultad fue habituarse a utilizar las herramientas de Firebase. Me ayudé de varios tutoriales donde planteaban prototipos de chat [11] [12] [13] [14], lo cual se adaptaba tanto a la pantalla de anuncios como a la pantalla de chat del proyecto.

  Tras un análisis de funcionamiento y muchas pruebas, alcancé un conocimiento suficiente para seguir adelante en el desarrollo.

- **Multiples llamadas al mismo *listener* de base de datos**.

  A medida que el proyecto fue tomando complejidad, en la pantalla de anuncios se iban añadiendo llamadas con espera de base de datos. Esto provocó que los anuncios aparecieran duplicados o triplicados, así como el número de mensajes no leídos.

  Tras investigar y realizar pruebas, se determinó que las escuchas de tipo *addChildEventListener* y *addValueEventListener* no se eliminaban al parar la aplicación y se replicaban al abrirla de nuevo.

  La solución fue reformular la manera en la que se traían los datos para utilizar más veces *addListenerForSingleValueEvent* y menos las otra dos opciones. Además, en el ciclo de vida de la actividad, los listener llamados en *onCreate* se eliminan en *onDestroy*, al igual que pasa con *onStart* y *onStop* [8].

- **Condiciones de carrera de listeners de base de datos**.

  Otro problema relacionado con la implementación de Realtime Database ha sido que el orden en el que se ejecutan las peticiones de consulta, no es el mismo que el que nosotros escribimos en el código.

  Esto quiere decir, por ejemplo, si esperábamos un dato de un *addValueEventListener* y en la siguiente línea queriamos rellenar un array con el dato dentro de un *addChildEventListener*, la orden no se ejecuta de esa manera necesariamente. Existe una prioridad interna en las peticiones a base de datos.

  Las excepciones de Java se lanzaban con frecuencia, debido a que se accedían a variables *null* o *arrays* vacíos, pues aún no habían llegado las respuestas de lectura de los *listener* con menos prioridad.

  La solución pasó por repensar el orden de carga de datos e intentar evitar que una variable dentro de un *listener* dependa de otra en otro *listener* paralelo. Además, por supuesto, de realizar comprobaciones de *null*.

- **Paso de datos precargados entre actividades**.

  En la primera etapa del desarrollo, se pensó que una forma de agilizar el funcionamiento de la aplicación y evitar descargar datos varias veces, sería precargar todos los datos de los chats del usuario en la pantalla de anuncios y serializarlos [8] para enviarlos a la de chat.

  Conforme se iba haciendo más grande el proyecto esta idea se desechó. Primero porque aportaba mucha complejidad al tener que comprobar si los datos se habían descargado correctamente y el procesado de estos mismos para mostrarlos en el chat. En segundo lugar, y más importante, porque cuando se implementaron las notificaciones y la funcionalidad de localización, el flujo de navegación cambió. Ahora podiamos lanzar un chat desde una notificación [9], o volver desde la pantalla de localización al chat. En estos casos no se ha pasado por la pantalla de anuncios y no se han precargado los datos.

  La solución fue optar por cargar los datos del chat cada vez que entremos en uno, de forma que solo se necesita su identificador de una actividad a otra (o desde la notificación hasta la actividad).

  Otra solución es persistir los datos precargados, como se explica en el apartado de Conclusiones.

- **Integración de notificaciones con FCM y Functions**.

Aparentemente, incluir notificaciones de nuevos mensajes en una aplicación es algo sencillo, pues todas las que usamos en el día a día tienen esa funcionalidad. Nada más lejos de la realidad.

Desde el momento de documentarse sobre las opciones que existen para notificar a los usuarios, se divisaba que para nada sería algo trivial [17] [18]. A base de tutoriales y blogs sobre las posibilidades de Cloud Messaging, llegué a la conclusión de que lo mejor era automatizar nuevas entradas en base de datos y enviarlas a los usuarios a través de scripts publicados en Functions.

Una vez planteada esa idea, llegaba la hora de implementarlo. Tampoco fue sencillo porque arrancar el SDK Admin de Firebase no es algo inmediato. Unido además a la dificultad de depuración del código que desplegamos en la nube.

Tras muchas pruebas, se llegó a una soltura y comprendimiento suficiente para dar salida a nuestros requisitos.

- **Fallos en el funcionamiento de servicios de Firebase o Maps**.

  Muchos de los errores bloqueantes del proyecto, fueron a causa de servicios de Firebase mal configurados, o falta de permisos de API en Google Cloud. Esto consumió mucho tiempo, ya que el desconocimiento de las posibilidades de error de la plataforma era total.

  Se solucionaron realizando tareas de depuración e investigación por internet [10].

- **Documentación escasa para la API Directions**.

  Uno de los últimos problemas que se han encontrado fue que Directions, al tratarse de una API de pago, tiene una documentación muy escueta y centrada solamente en cómo construir peticiones.

  En el proyecto que tratamos se necesitaba calcular la ruta entre dos puntos del mapa y dibujar una línea que marque la ruta, por lo que sin indicaciones de como interpretar la respuesta de la petición a la API, no se podía hacer nada que no implicara un gran impacto en el desarrollo.

  La solución se encontró gracias a un vídeo de Youtube donde un desarrollador explicaba el proceso y aportaba el código de GitHub [11]. Gracias a este código, se consiguió recibir una petición de Directions y representarla en el mapa fácilmente.

6 PLAN DE PRUEBAS

En este capítulo expondremos las pruebas que hemos llevado a cabo para considerar que el sistema funciona como se espera según los requisitos y casos de uso expuestos en apartados anteriores.

6.1 Pruebas manuales

Las pruebas manuales son las que se han realizado por el desarrollador haciendo de tester, para garantizar que el sistema hace lo que debe y que no tiene fallos.

PP-01	**Lanzamiento de aplicación sin sesión previa exitoso**
Descripción	Abrimos la aplicación desde cualquier parte que lo permita del sistema operativo estando está cerrada.
Resultado esperado	Debe mostrar un splash screen y a continuación lanzar correctamente la pantalla de login, que debe estar totalmente funcional.

Tabla 63: Lanzamiento de aplicación sin sesión previa exitoso

PP-02	**Lanzamiento de aplicación con sesión previa exitoso**
Descripción	Abrimos la aplicación desde cualquier parte que lo permita del sistema operativo estando está cerrada.
Resultado esperado	Debe mostrar un splash screen, cargar la última sesión con la que se autenticó el usuario y a continuación lanzar la pantalla principal de la aplicación, que comenzará a cargar la información de base de datos y mostrarla correctamente.

Tabla 64: Lanzamiento de aplicación con sesión previa exitoso

PP-03	**Inicio de sesión con cuenta de Google de forma correcta**
Descripción	Encontrándonos en la pantalla de login, pulsamos el botón de inicio con cuenta de Google e indicar la cuenta con la que deseamos autenticarnos.
Resultado esperado	Debe mostrar un cuadro de espera de carga y al finalizar llevarnos a la pantalla principal de la aplicación, que comenzará a cargar la información de base de datos y mostrarla correctamente.

Tabla 65: Inicio de sesión con cuenta de Google de forma correcta

PP-04	Registro de usuario de forma correcta
Descripción	Encontrándonos en la pantalla de login, pulsamos el botón de registro.
Resultado esperado	Debe mostrar la pantalla de registro, con un formulario de registro totalmente funcional. Al rellenar los campos obligatorios correctamente, deberá crear el nuevo usuario en base de datos e iniciar sesión automáticamente.

Tabla 66: Registro de usuario de forma correcta

PP-05	Carga de imagen de usuario de forma correcta
Descripción	Encontrándonos en la pantalla de registro, pulsamos el botón de editar imagen de perfil.
Resultado esperado	Debe mostrar la pantalla de selección de archivos de Android y tras elegir una imagen, volverá a la pantalla de registro y cargará dicha imagen en la vista de avatar.

Tabla 67: Carga de imagen de usuario de forma correcta

PP-06	Inicio de sesión con correo de forma correcta
Descripción	Encontrándonos en la pantalla de login, rellenamos la entrada de texto de correo y contraseña con la cuenta con la que deseamos autenticarnos y pulsamos el botón de inicio de sesión con correo.
Resultado esperado	Debe mostrar un cuadro de espera de carga y al finalizar llevarnos a la pantalla principal de la aplicación, que comenzará a cargar la información de base de datos y mostrarla correctamente.

Tabla 68: Inicio de sesión con correo de forma correcta

PP-07	Inicio de sesión con correo de forma errónea
Descripción	Encontrándonos en la pantalla de login, rellenamos la entrada de texto de correo y contraseña con valores erróneos: ▪ Correo inexistente con caracteres aleatorios ▪ Correo correcto y contraseña incorrecta ▪ Ambos campos vacíos A continuación pulsamos el botón de inicio de sesión con correo.
Resultado esperado	Debe mostrar un mensaje de error y permanecer en la pantalla de login.

Tabla 69: Inicio de sesión con correo de forma errónea

PP-08	**Cierre de sesión exitoso**
Descripción	Encontrándonos en la pantalla principal de la aplicación, pulsamos en el botón superior derecho de cierre de sesión.
Resultado esperado	Debe cerrar la sesión del usuario, cerrar la pantalla principal y mostrarnos la pantalla de login.

Tabla 70: Cierre de sesión exitoso

PP-09	**Edición de datos de usuario exitoso**
Descripción	Encontrándonos en la pantalla de principal con los datos ya cargados, pulsamos en la imagen de avatar del usuario.
Resultado esperado	Debe mostrarnos la pantalla de registro de usuario pero con nuestros datos precargados. Al rellenar o modificar cualquier campo, nos pedirá confirmación y volverá a la pantalla principal de la aplicación con los nuevos datos cargados.

Tabla 71: Edición de datos de usuario exitoso

PP-10	**Edición de datos de usuario exitoso**
Descripción	Encontrándonos en la pantalla de principal con los datos ya cargados, pulsamos en la imagen de avatar del usuario.
Resultado esperado	Debe mostrarnos la pantalla de registro de usuario pero con nuestros datos precargados. Al rellenar o modificar cualquier campo, nos pedirá confirmación y volverá a la pantalla principal de la aplicación con los nuevos datos cargados.

Tabla 72: Edición de datos de usuario exitoso

PP-11	**Entrar a un nuevo chat de anuncio correctamente**
Descripción	Encontrándonos en la pantalla de principal con los datos ya cargados, pulsamos en un anuncio de la lista.
Resultado esperado	Debe mostrarnos un cuadro de confirmación y a continuación llevarnos a la pantalla de chat relativa a ese anuncio, que comenzará a cargar sus datos y finalmente los mostrará.

Tabla 73: Entrar a un nuevo chat de anuncio correctamente

PP-12	**Apertura de chat de anuncio desde chats recientes de forma exitosa**
Descripción	Encontrándonos en la pantalla de principal con los datos ya cargados, se debe mostrar una lista horizontal con los chats de anuncio de los que

	formamos parte. Pulsamos uno de ellos.
Resultado esperado	Debe mostrarnos llevarnos a la pantalla de chat relativa a ese anuncio, que comenzará a cargar sus datos y finalmente los mostrará.

Tabla 74: Apertura de chat de anuncio desde chats recientes de forma exitosa

PP-13	**Creación de anuncio exitoso**
Descripción	Encontrándonos en la pantalla de principal con los datos ya cargados, pulsamos en el botón inferior derecho de creación/edición de anuncio.
Resultado esperado	Debe mostrar la pantalla de registro de anuncio, con un formulario totalmente funcional. Al rellenar los campos obligatorios correctamente, deberá crear el nuevo anuncio en base de datos y volver a la pantalla principal.

Tabla 75: Creación de anuncio exitoso

PP-14	**Edición de anuncio exitoso**
Descripción	Encontrándonos en la pantalla de principal con los datos ya cargados y habiendo creado un anuncio, pulsamos en el botón inferior derecho de creación/edición de anuncio.
Resultado esperado	Debe mostrar la pantalla de registro de anuncio, con el formulario con los datos del anuncio precargados. Al rellenar los campos obligatorios correctamente, deberá guardar el anuncio editado en base de datos y volver a la pantalla principal.

Tabla 76: Edición de anuncio exitoso

PP-15	**Intercambio correcto de mensajes de chat**
Descripción	Encontrándonos en la pantalla de chat de anuncio con los datos ya cargados, escribimos en la entrada de texto de mensajes y pulsamos el botón de envío.
Resultado esperado	Debe enviar la información de los mensajes a base de datos y a continuación actualizar la lista de mensajes de chat y mostrarlo. Cuando envía un mensaje otro usuario debe tener el mismo comportamiento.

Tabla 77: Intercambio correcto de mensajes de chat

PP-16	**Recepción correcta de notificaciones**
Descripción	Encontrándonos fuera de un chat de anuncio, cuando otro usuario haya enviado un mensaje o cuando el administrador retransmita su localización, debe lanzarse una notificación nativa de Android.

Resultado esperado	Debe mostrarse la información relativa al tipo de mensaje en la notificación, así como el autor y el contenido del nuevo dato. Al pulsar la notificación, debe enviarnos a la pantalla de chat de anuncio correspondiente y comenzar a cargar los datos del mismo.

Tabla 78: Recepción correcta de notificaciones

PP-17	**Apertura correcta de funcionalidad de seguimiento de localización**
Descripción	Encontrándonos en un chat de anuncio y siendo el creador del anuncio o bien siendo miembro del chat con retrasmisión activa, debemos poder pulsar el botón de la función de localización.
Resultado esperado	Debe llevarnos a la pantalla de localización.

Tabla 79: Apertura correcta de funcionalidad de seguimiento de localización

PP-18	**Establecimiento correcto de seguimiento de localización con rol de administrador**
Descripción	Encontrándonos en la pantalla de la función de localización, podemos de forma opcional establecer la ruta que se va a seguir con pulsaciones largas en la vista de mapa. A continuación debemos indicar el tiempo máximo de transmisión con un cuadro de dialogo.
Resultado esperado	Debe guardar la información en base de datos y comenzar a retransmitir nuestra ubicación sobrescribiendo la información en tiempo real mediante el servicio de localización de Android.

Tabla 80: Establecimiento correcto de seguimiento de localización con rol de administrador

PP-19	**Recepción correcta de seguimiento de localización con rol de cliente**
Descripción	Nos encontramos en la pantalla de la función de localización con una retransmisión activa.
Resultado esperado	Debe actualizarse la posición del administrador del chat en tiempo real en el mapa, que debe permitir interactuar con él. También se debe mostrar la ruta establecida y el tiempo restando para concluir la retransmisión.

Tabla 81: Recepción correcta de seguimiento de localización con rol de cliente

6.2 Pruebas de aceptación

PP-20	**Validación de la aplicación por parte del tutor**
Descripción	Se entregará la versión definitiva al tutor del proyecto para que valide que todas las funcionalidades acordadas se llevan a cabo de manera exitosa y con la calidad esperada.
Resultado esperado	

Tabla 82: Validación de la aplicación por parte del tutor

7 CONCLUSIONES

7.1 Conclusiones

El proyecto que se ha acometido ha supuesto un hito importante a nivel de formación, ya que, como se ha comentado en el documento, las tecnologías utilizadas son de las más modernas y demandadas en la actualidad. Incluso puede dar pie a desarrollos personales por mi parte que utilicen estas herramientas.

La realización del proyecto ha sido dura, porque se han encontrado muchos problemas en el desarrollo con soluciones dificilmente accesibles en documentación o en búsquedas en internet. Sin embargo, ver completada una aplicación que cumple los requisitos planteados y que en general tiene un buen nivel de calidad es enorgullecedor.

Los conocimientos que se han adquirido en el grado, han ayudado en la parte de programación y de comprendimiento de la arquitectura, pero otra fracción de conocimientos puestos en práctica vienen del autoaprendizaje. Este aprendizaje no es más que el fruto de los dos últimos años, en los que me he dedicado a profundizar en el desarrollo de aplicaciones Android como afición personal.

La tarea más complicada desde mi punto de vista, ha sido la configuración, puesta en marcha e implementación de todo lo relacionado con Firebase. La gran cantidad de servicios y funciones que posee, adaptado a la naturaleza de red social de la aplicación, ha supuesto un quebradero de cabeza en algunos apartados. Tras el estudio de código libre aportado por otros usuarios de internet, documentación oficial y pruebas de concepto, finalmente se ha conseguido siempre avanzar con éxito en cada tarea.

En definitiva, creo que he realizado un proyecto final de carrera acorde a mis gustos y que me ha hecho esforzarme para completarlo con solvencia.

7.2 Líneas de desarrollo futuro

El proyecto tiene muchas posibilidades de mejora y adición de funcionalidades. Al tratarse de una red social, tiene infinitas formas de tornarse más completa de cara a los usuarios. Varias de las mejoras pueden ser:

- **Opciones de autenticación**. Aunque la oferta actual parece razonable, en un mercado real debería considerarse que los usuarios puedan iniciar sesión con otras pataformas como Facebook o Twitter.
- **Búsqueda de horarios compatibles**. Actualmente en el sistema el usuario busca un conductor de forma que hasta que no entra en el chat no sabe si coincide del todo con su horario. Permitir que el usuario pueda configurar su propio horario y filtrar a partir de él, conseguiría resultados más personalizados y eficaces.
- **Grupos de chat con varios conductores**. El sistema actualmente crea un chat por cada anuncio publicado. Sería interesante la opción de permitir chats en los que varios usuarios sean marcados como conductor y se pueda confeccionar un horario con la información todos ellos junta. Con esto se conseguiría que varios conductores de una misma ciudad tengan centralizados a sus pasajeros.
- **Funcionalidades de chat**. El chat está enfocado para comunicar lo mínimo necesario en el proceso de compartir coche. Sin embargo, pueden añadirse funciones como expulsar miembros, adjuntar imágenes, o enviar audios, que pueden resultar útiles.
- **Persistencia de datos de usuario**. Las funciones de la aplicación siempre dependen de la conexión a internet para descargar los datos y representarlos. Una buena opción para optimizar el consumo de datos y batería sería descargar y guardar los datos localmente. Una base de datos local como SQLite o Realm serían las mejores opciones. También existe la opción de generar un archivo JSON encriptado, aunque no es lo mejor porque es dificilmente escalable.

  La velocidad de carga de la aplicación sería mucho más rápida, pero la implementación no sería trivial, pues hay que tener muchas condiciones en cuenta a la hora de actualizar los datos locales.

- **Aplicación iOS**. Para ofrecer un servicio completo, debería desarrollarse también una versión de la aplicación para la plataforma de Apple.
- **Traducción a inglés**. Toda aplicación seria desarrollada desde España debería tener al menos disponible el inglés como idioma adicional.
- **Personalizar ámbito de funcionamiento**. Aunque este proyecto está enfocado a la Escuela Superior de Ingenieros, sería interesante ofrecer la opción de elegir la facultad o Universidad en la que se estudia para solo mostrar resultados de conductores que se dirigen la misma a diario.
- **Explorar más posibilidades de Firebase**. El catálogo de Firebase ofrece muchas funciones interesantes que no se han explorado. Por ejemplo, informes de Crashlytics para monitorizar errores. También se puede sacar más provecho de los productos ya utilizados, como por ejemplo emplear Functions para comprimir las imágenes que se suban a Storage y generar un tipo de archivo para cada resolución necesaria en la aplicación.

7.3 Dedicación total temporal

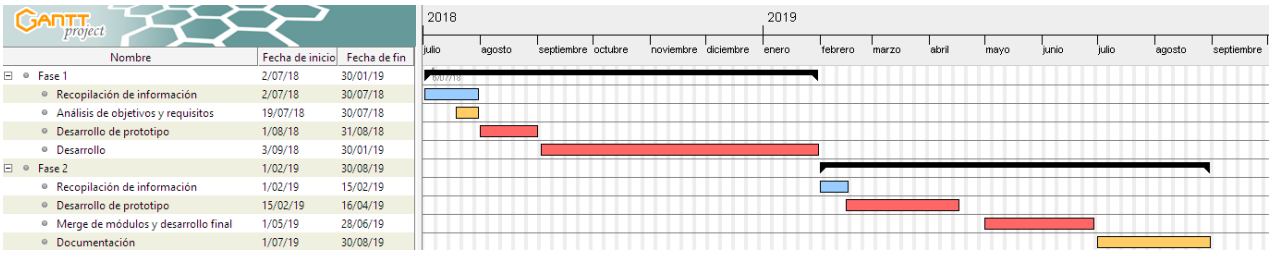
Antes de comenzar el desarrollo, se llevó a cabo una tarea de análisis y estimación de los esfuerzos en el tiempo.

Se dividió el desarrollo en dos partes: por un lado la fase de análisis de objetivos y requisitos junto con el desarrollo de los módulos básicos de la aplicación (anuncios, chat), y por otro lado, el desarrollo de módulos de funcionalidades extra (localización, notificaciones) y la finalización mediante la documentación del proyecto en el presente texto.

Sin embargo, durante el desarrollo del proyecto, la estimación se desvió mucho por diversos motivos: cambios en el diseño y los requisitos, pausas en el proyecto por motivos externos o un aumento en la complejidad de la implementación de la aplicación final que no se tuvo en cuenta en un principio.

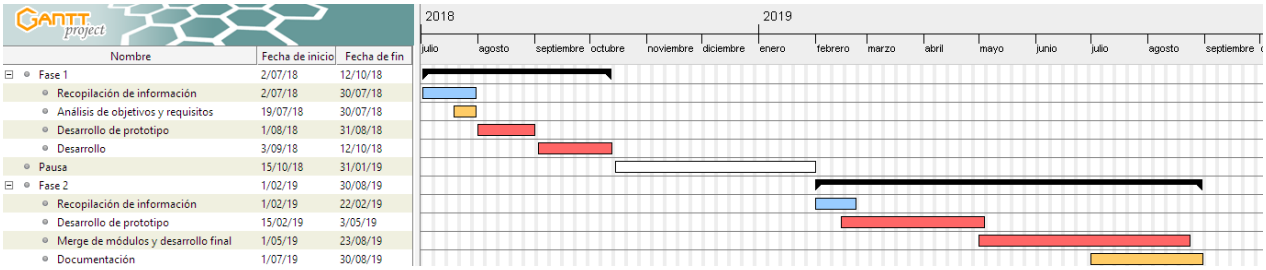
A continuación se adjunta la estimación temporal estimada mediante un digrama de Gantt y a continuación desglosaremos el tiempo real en el que fue llevado a cabo el proyecto.

7.3.1 Planificación temporal estimada



Nombre	Fecha de inicio	Fecha de fin
Fase 1	2/07/18	30/01/19
Recopilación de información	2/07/18	30/07/18
Análisis de objetivos y requisitos	19/07/18	30/07/18
Desarrollo de prototipo	1/08/18	31/08/18
Desarrollo	3/09/18	30/01/19
Fase 2	1/02/19	30/08/19
Recopilación de información	1/02/19	15/02/19
Desarrollo de prototipo	15/02/19	16/04/19
Merge de módulos y desarrollo final	1/05/19	28/06/19
Documentación	1/07/19	30/08/19

7.3.2 Planificación temporal real



Nombre	Fecha de inicio	Fecha de fin
Fase 1	2/07/18	12/10/18
Recopilación de información	2/07/18	30/07/18
Análisis de objetivos y requisitos	19/07/18	30/07/18
Desarrollo de prototipo	1/08/18	31/08/18
Desarrollo	3/09/18	12/10/18
Pausa	15/10/18	31/01/19
Fase 2	1/02/19	30/08/19
Recopilación de información	1/02/19	22/02/19
Desarrollo de prototipo	15/02/19	3/05/19
Merge de módulos y desarrollo final	1/05/19	23/08/19
Documentación	1/07/19	30/08/19

FASE 1	248 horas

Recopilación de información	**30 horas**
Recopilación de información sobre tecnologías a emplear	30 horas
Análisis de objetivos y requisitos	**8 horas**
Análisis del problema y objetivos	3 horas
Obtención de requisitos	5 horas
Desarrollo de prototipo	**90 horas**
Diseño de módulos de la aplicación	10 horas
Configuración de plataforma Firebase y puesta en marcha	10 horas
Implementación de módulos de la aplicación	60 horas
Pruebas y corrección de errores	10 horas
Desarrollo	**120 horas**
Configuración de plataforma Firebase y puesta en marcha	5 horas
Integración de prototipos en aplicación	20 horas
Implementación de la aplicación	75 horas
Pruebas y corrección de errores	20 horas
FASE 2	**355 horas**
Recopilación de información	**10 horas**
Recopilación de información sobre tecnologías a emplear	10 horas
Desarrollo de propotipo	**115 horas**
Diseño de módulos de la aplicación	10 horas
Configuración de plataforma Firebase y Maps y puesta en marcha	20 horas
Implementación de módulos de la aplicación	45 horas
Pruebas y corrección de errores	15 horas
Merge de módulos y desarrollo final	**185 horas**
Configuración de plataforma Firebase y Maps y puesta en marcha	5 horas
Integración de prototipos en aplicación	20 horas
Implementación de la aplicación final	130 horas
Pruebas y corrección de errores	30 horas

Documentación	**65 horas**
Redacción de memoria de proyecto	65 horas

TOTAL DEDICADO	***603 HORAS***

ANEXO 1: GUÍA DE DESPLIEGUE

En este apartado, se explicará con detalle como configurar la plataforma Firebase y Maps, así como todos sus productos.

1. Firebase

Para configurar desde cero un proyecto en la plataforma Firebase:

1. Crear un nuevo proyecto: desde la consola de Firebase, pulsamos en Nuevo proyecto. Luego escribimos un nombre, pulsamos siguiente, y elegimos si queremos los componentes opcionales de Firebase. Confirmamos y empezará a crear nuestro proyecto.
2. Configuración del proyecto: desde el icono de ajustes del menú lateral, pulsamos en configuración del proyecto. A continuación elegimos añadir una aplicación de Android, como se muestra en la captura siguiente.

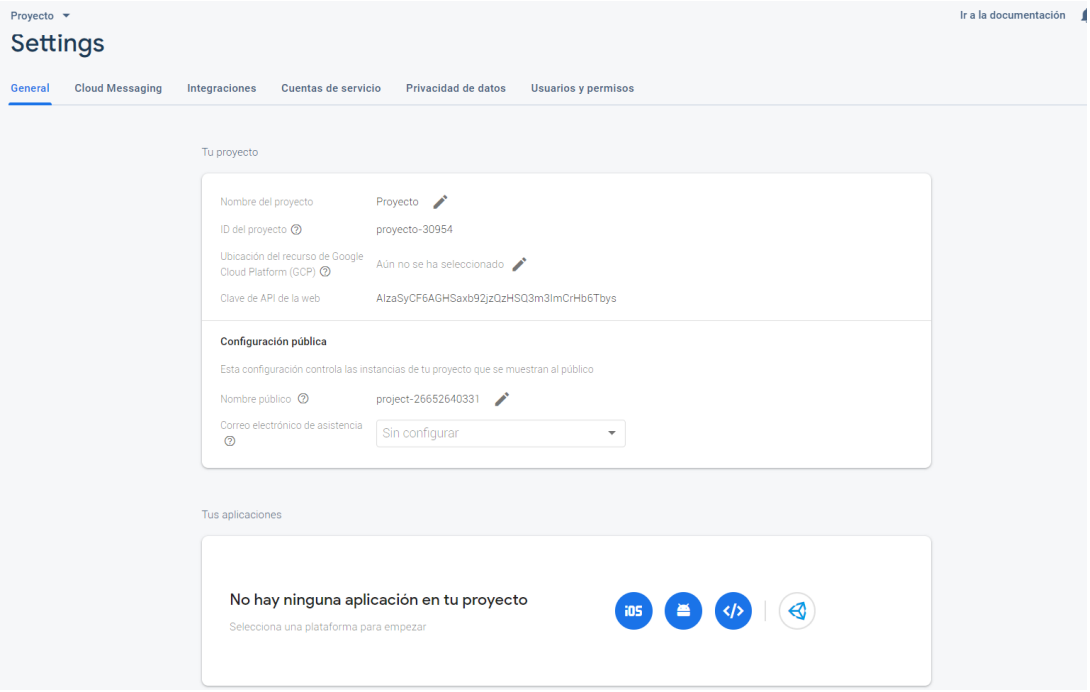


Ilustración 61: Configuración Firebase

3. Rellenamos la información de la aplicación que nos piden en la Ilustraciñon siguiente y pulsamos en Registrar:

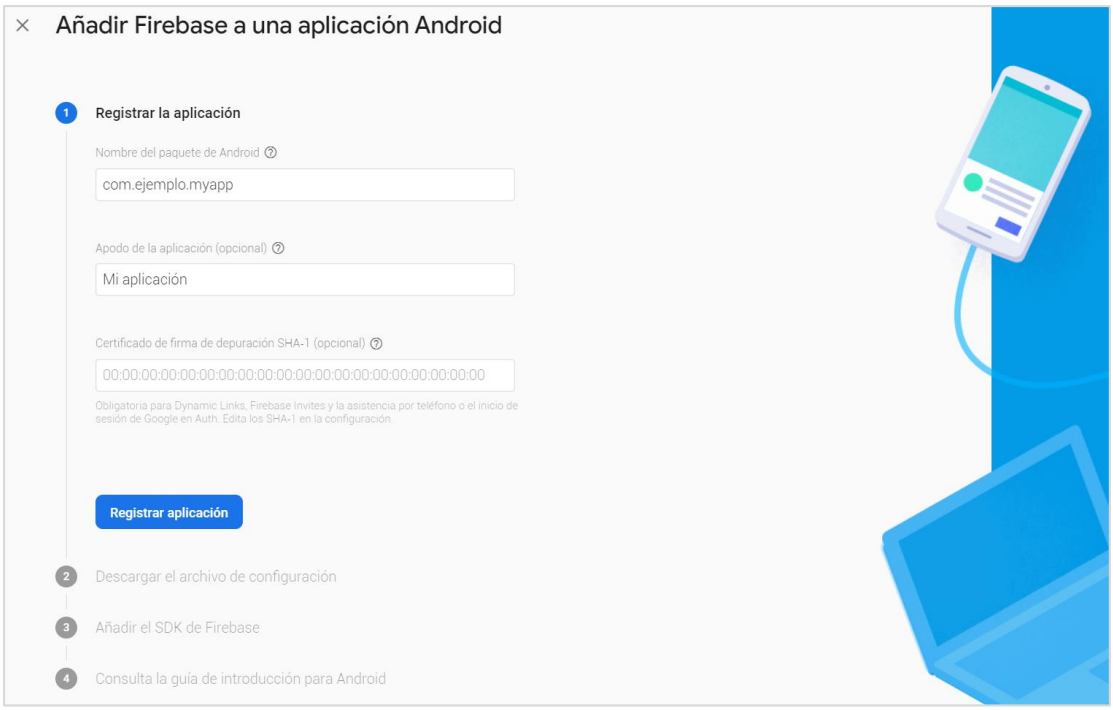


Ilustración 62: Añadir aplicación a proyecto Firebase

4. Descargamos el archivo JSON de servicios que nos facilitan y lo ubicamos dentro de la carpeta /app del proyecto, como nos indican en la siguiente Ilustración:

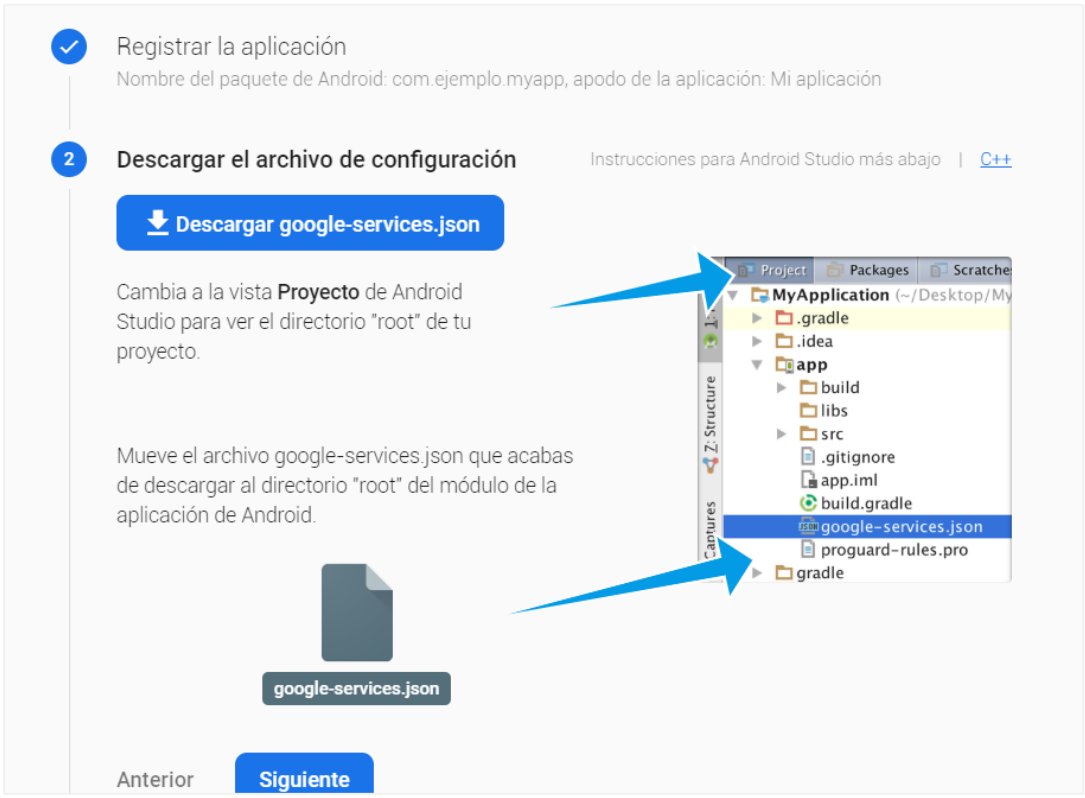


Ilustración 63: Archivo configuración Firebase

5. Se añaden las dependencias que nos indican en la captura siguiente:

Ilustración 64: Dependencias Firebase

Tras estos pasos, debería poder conectarse al proyecto de la plataforma principal que hemos creado.

2. Firebase Realtime Database

Configuración en la plataforma

Para configurar Realtime Database desde la plataforma no es necesario realizar ninguna acción, aunque es siempre recomendable establecer reglas para que la información no sea accesible por terceros que no estén autenticados por medio de Firebase Auth. Esto se hace en la pestaña de “Reglas”:

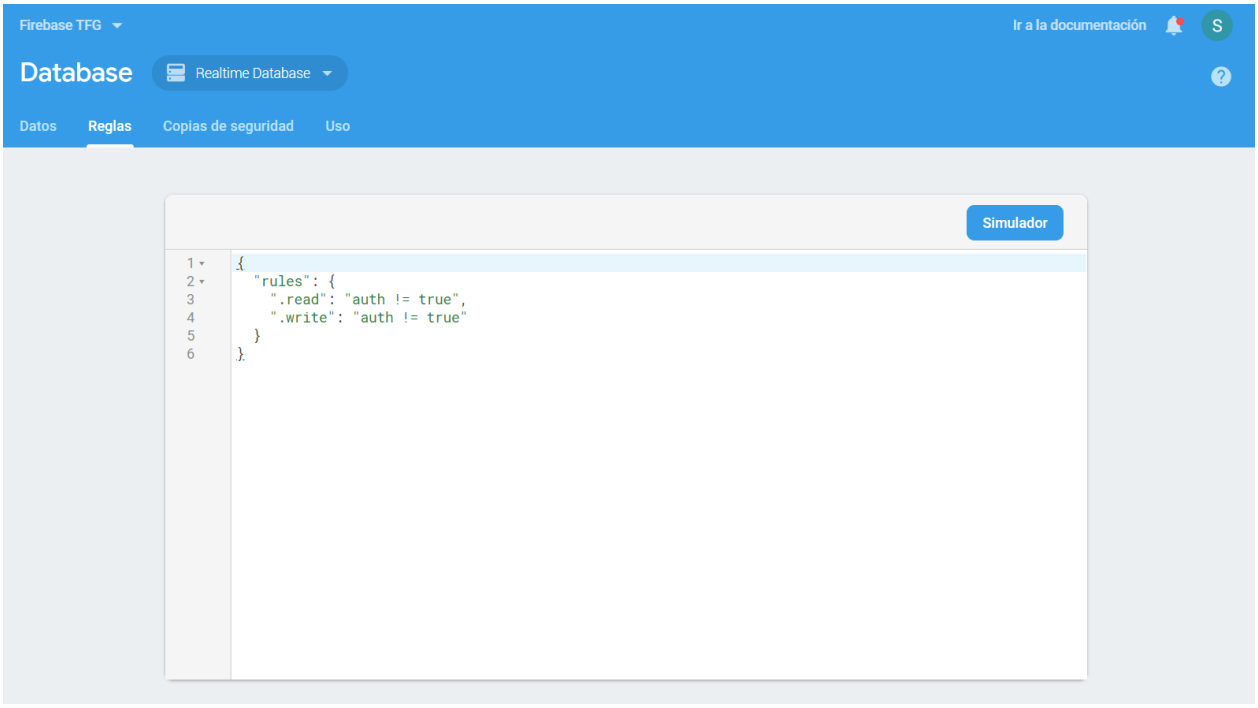


Ilustración 65: Reglas en consola de Firebase Realtime Database

La opción de configuración de Copias de seguridad sólo está disponible para planes de pago de la plataforma.

También podemos monitorizar desde este mismo menú el uso de la Realtime Database:

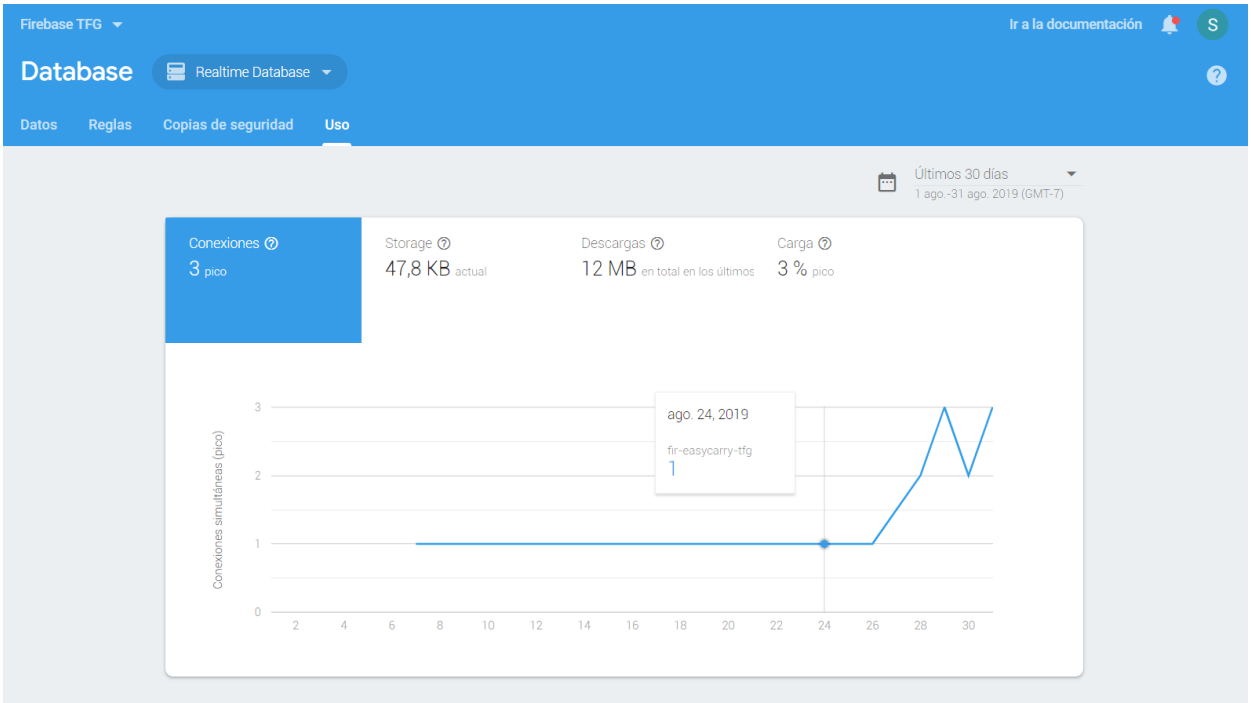


Ilustración 66: Uso en consola de Firebase Realtime Database

Cabe mencionar que dentro de la pestaña Datos podemos realizar varias operaciones de gestión de la base de datos:

- Modificación de datos: podemos cambiar valores de los nodos almacenados (pero no el identificador del objeto).

Ilustración 67: Aspecto de base de datos en Realtime Database

- Exportar o importar datos: podemos descargar en un archivo de formato JSON toda la información contenida en base de datos. Del mismo modo podemos subir un archivo con texto formateado en JSON para sobreescribir los datos actuales.

  Esto es muy útil por ejemplo en tareas de refactorización o migración de bases de datos, pudiendo procesar la información en texto plano con herramientas externas.

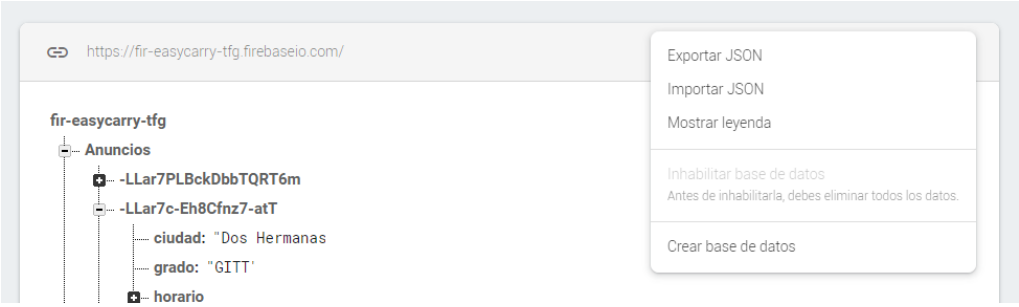


Ilustración 68: Opciones de base de datos en Realtime Database

- Crear base de datos: también se puede proceder a crear una nueva instancia de Realtime Database. Solo está permitido en planes de pago de la plataforma Firebase.

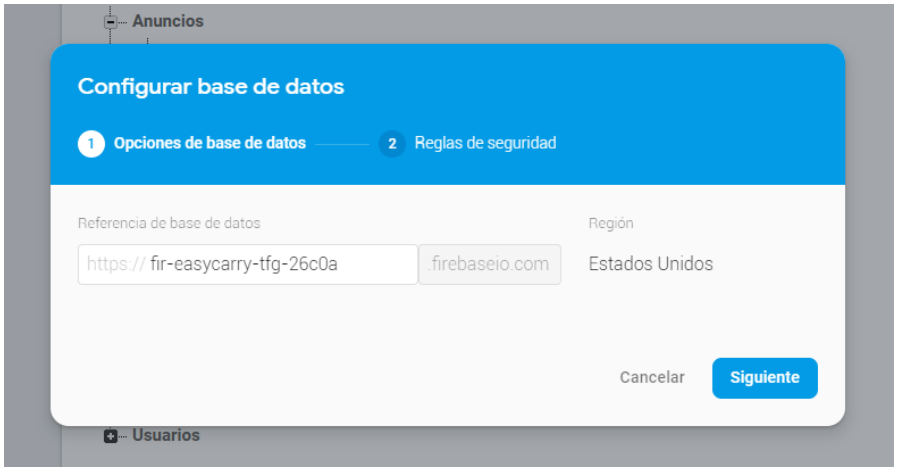


Ilustración 69: Configuración de nueva base de datos

Implementación en el proyecto

En primer lugar, es necesario añadir las dependencias en el archivo *build.gradle*:

```
implementation 'com.google.firebase:firebase-database:19.0.0'
```

Código 7: Adición de dependencias de Firebase Realtime Database

A continuación solo hay que empezar a desarrollar creando instancias de Database mediante referencias a las jerarquías JSON de la base de datos que se pretenda crear.

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference("message");

myRef.setValue("Hello, World!");
```

Código 8: Ejemplo de escritura en Realtime Database

Este es un ejemplo simple de **escritura en base de datos**, pero pueden realizarse peticiones más complejas, como por ejemplo estas que se extraen del proyecto:

```
private void enviaMensaje(String txt) {
    String autor = usuario.getNombre();
    String uid = usuario.getUsuarioId();

    if (!txt.trim().isEmpty()) {
        // Envía un objeto Mensaje a la lista de mensajes de chat
        dbChat.child("mensajes").push().setValue(
                new MensajeEnviar(uid, txt, ServerValue.TIMESTAMP));
        // Activa el flag que indica que el chat no está vacío
        dbChat.child("hayMensajes").setValue(true);
        // Envía un objeto Notificacion a la lista de notificaciones
        dbNotif.child("Mensajes").push().setValue(
                new Notificacion(chatId, autor, txt));

        inputMensajes.setText("");
    }
}
```

Código 9: Ejemplo de escritura en Realtime Database

En estos ejemplos podemos destacar dos formas de escritura en base de datos:

- Mediante referencia-push-valor: se crea un nuevo elemento con ID padre autogenerado.
- Mediante referencia-valor: se crea un nuevo elemento con ID igual al nombre de la referencia.

También podemos añadir un *listener* a la acción de escritura, en la que el servidor nos indica a la vuelta el éxito o no de la operación a traves de un *callback*.

```
//Usuarios > {uid} > avatar = IMAGEN
dbUsuarios.child(usuarioId).child("avatar").setValue(fotoPerfil)
        .addOnSuccessListener(new OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void aVoid) {
        utils.cargaAvatar(fotoPerfil, fotoPerfilView);
    }
});
```

Código 10: Ejemplo de escucha de escritura en Realtime Database

En cuando a la **lectura de base de datos**, hay múltiples opciones:

- **Lectura de valores hijos**: quedamos a la escucha de nuevos hijos añadidos, eliminados o modificados. El listener permanece hasta que se elimina manualmente o se destruye el Activity.

```
// Descarga de los id de todos los clientes de un chat
dbChat.child("clientes").addChildEventListener(new ChildEventListener() {
    @Override public void onChildChanged(@NonNull DataSnapshot dataSnapshot, String s) {}
    @Override public void onChildRemoved(@NonNull DataSnapshot dataSnapshot) {}
    @Override public void onChildMoved(@NonNull DataSnapshot dataSnapshot, String s) {}
    @Override public void onCancelled(@NonNull DatabaseError databaseError) {}

    @Override public void onChildAdded(@NonNull DataSnapshot data, @Nullable String s) {
            String clienteId = data.getKey();

            // Con el id de un cliente, pasa a descargar sus datos
            if (clienteId != null) {
                cargaUsuarioCliente(clienteId);
            }
        }
    });
}
```

Código 11: Ejemplo de escucha de lectura de valores hijos en Realtime Database

- **Lectura de valor**: quedamos a la escucha de una referencia para extraer un valor concreto que se vuelve a lanzar cada vez que se modifica. El listener permanece hasta que se elimina manualmente o se destruye el Activity.

```
dbChat.child("hayMensajes").addValueEventListener(new ValueEventListener() {
    @Override public void onCancelled(@NonNull DatabaseError databaseError) { }
    @Override public void onDataChange(@NonNull DataSnapshot data) {
        Boolean hayMensajes = data.getValue(Boolean.class);

        if (hayMensajes != null && hayMensajes) {
            txtNoHayMensajes.setVisibility(View.GONE);
        } else {
            txtNoHayMensajes.setVisibility(View.VISIBLE);
        }
    }
});
```

Código 12: Ejemplo de escucha de lectura de valor en Realtime Database

- **Lectura de valor único**: quedamos a la escucha de una referencia de base de datos para extraer un valor concreto que se ejecuta una única vez. Una vez nos devuelve los valores, el *listener* se elimina.

```
private void cargaUsuarioCliente(String clienteId) {

    dbUsuarios.child(clienteId).addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) { }
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            // Datos del cliente con id proporcionado
            Usuario clienteObj = dataSnapshot.getValue(Usuario.class);

            // Se añade a la lista y se muestra en la vista informativa
            clientesList.add(clienteObj);
            miembros.setText(String.format("%s miembros", clientesList.size()));
        }
    });
}
```

Código 13: Ejemplo de escucha de lectura de valor único en Realtime Database

Además podemos realizar consultas complejas, por ejemplo ordenando los valores hijos, estableciendo límites de valor, entre otras opciones.

```
// Empieza a cargar los mensajes de chat a partir de nuestra fecha de unión a este
dbChat.child("mensajes").orderByChild("hora").startAt(horaOrigenCarga)
          .addChildEventListener(new ChildEventListener() {

    @Override public void onChildChanged(@NonNull DataSnapshot dataSnapshot, String s) {}
    @Override public void onChildRemoved(@NonNull DataSnapshot dataSnapshot) {}
    @Override public void onChildMoved(@NonNull DataSnapshot dataSnapshot, String s) {}
    @Override public void onCancelled(@NonNull DatabaseError databaseError) {}

    @Override public void onChildAdded(@NonNull DataSnapshot dataSnapshot, String s) {
        MensajeRecibir m = dataSnapshot.getValue(MensajeRecibir.class);

        if(m != null) {
            adapterMsg.addMensaje(m);
            actualizaUltimaConexion();
        }
    }
});
```

Código 14: Ejemplo de consulta compleja de lectura en Realtime Database

El aspecto final de la base de datos tras las pruebas de la aplicación es la siguiente:

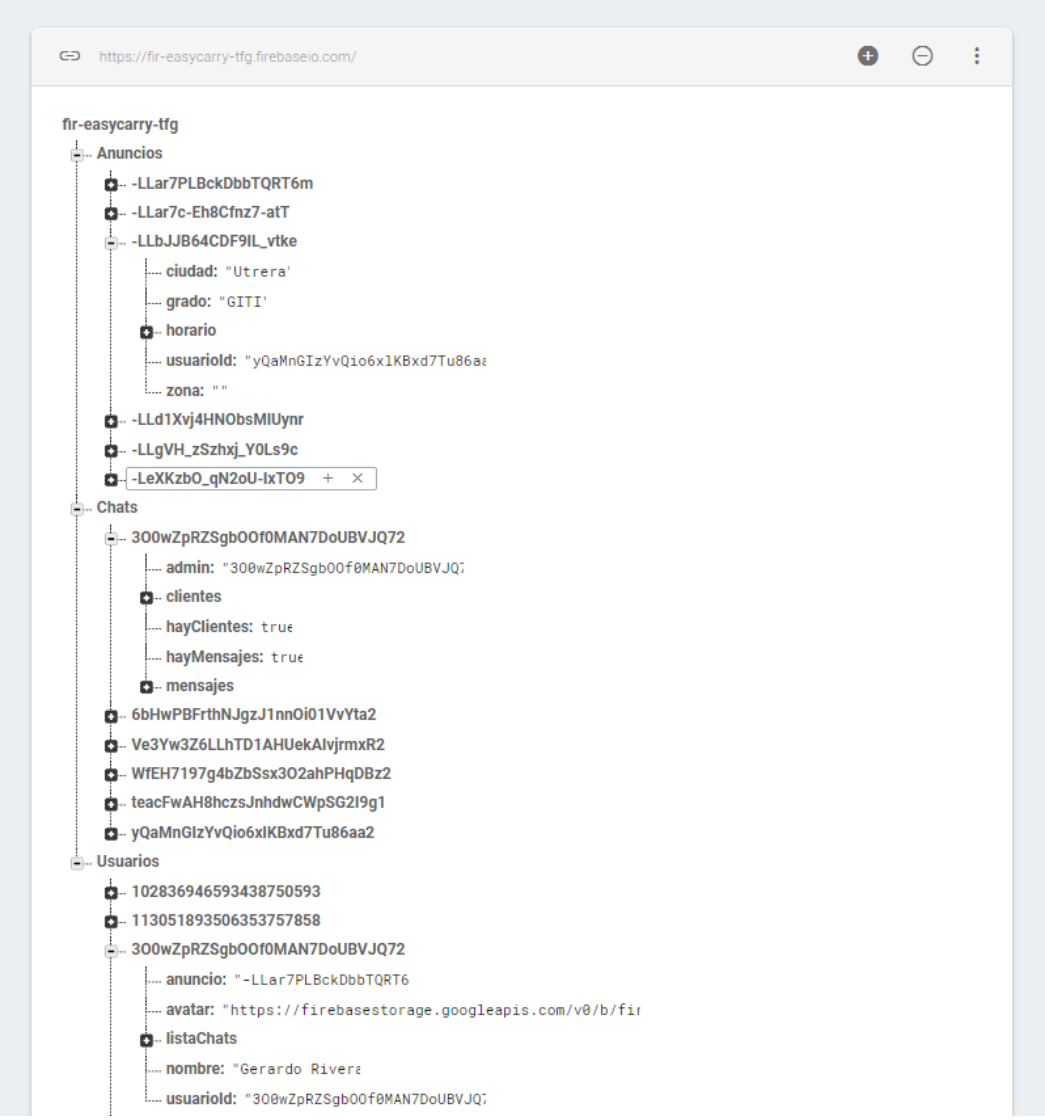


Ilustración 70: Captura de estado real de base de datos

3. Firebase Authentication

Configuración en la plataforma

La configuración previa consiste en habilitar los métodos de autenticación deseados. En nuestro caso, correo electrónico y cuenta de Google.

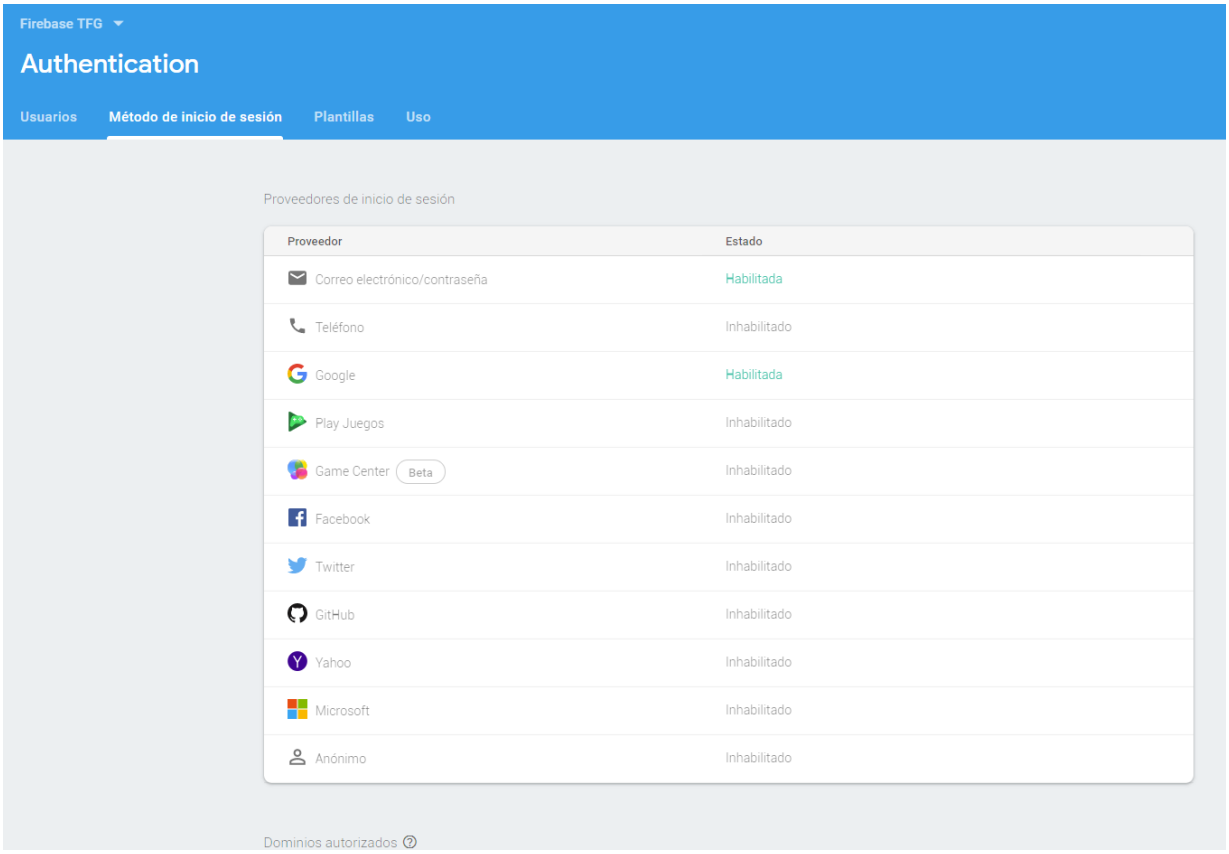


Ilustración 71: Métodos de autenticación en consola de Firebase Auth

1. Para habilitar el inicio con **correo** solo es necesario activarlo en el cuadro de diálogo.

Ilustración 72: Habilitar autenticación con correo y contraseña

2. Para configurar el acceso con **cuenta de Google**, también es necesario añadir la huella digital SHA-1 en la configuración del proyecto y descargar el fichero google-services.json que nos proporcionan para ubicarlo en nuestra aplicación. Estos pasos ya los realizamos anteriormente.

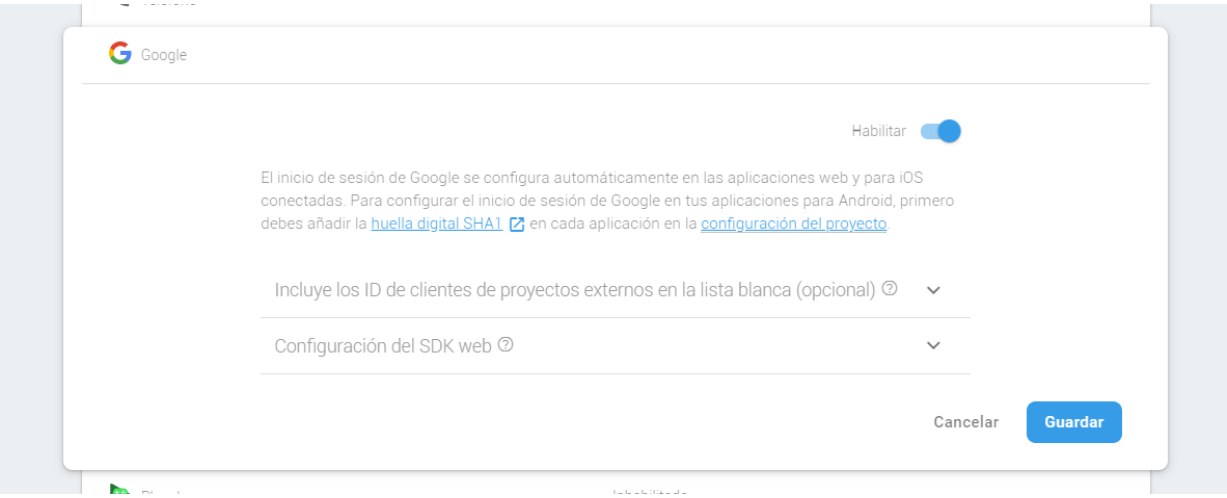


Ilustración 73: Habilitar autenticación con cuenta de Google

Los pasos siguientes son necesario para la autorización por parte de Google de sus métodos de autenticación por medio de las API y el acceso a cuentas de usuario:

3. Debemos añadir nuestro proyecto en la página de *Google Sign-In for Android*. En las siguientes imágenes vemos el portal donde debemos pulsar el boón de configurar y la ventana con la información generada.

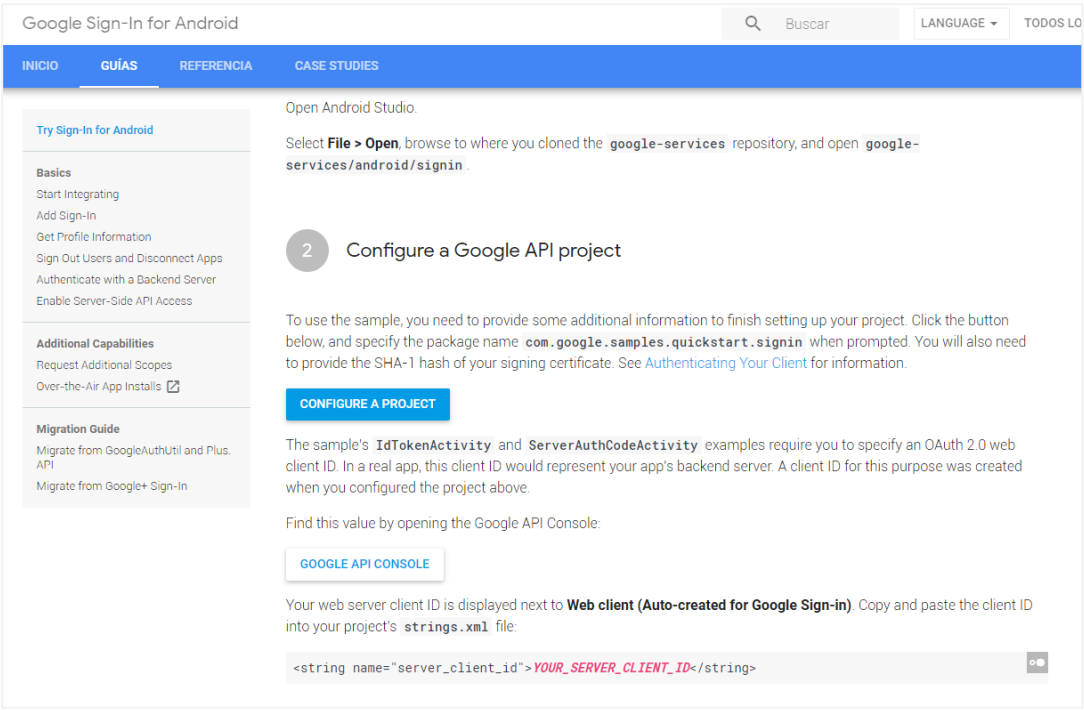


Ilustración 74: Configuración en Google Sign-In

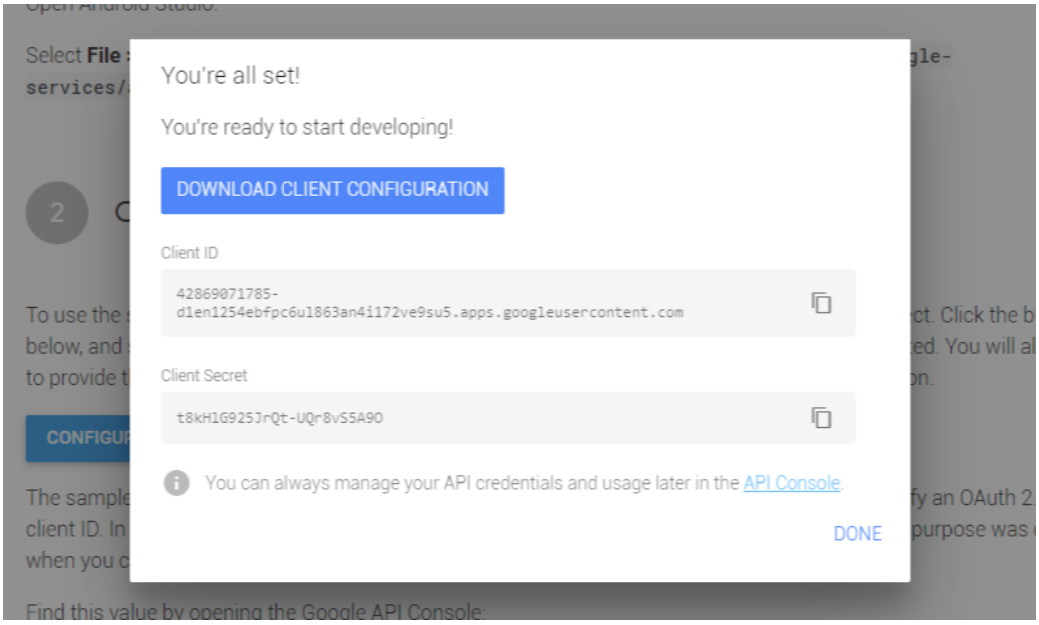


Ilustración 75: Configuración de Google Sign-In finalizada

4. Una vez añadido, vamos a la consola de Google APIs para activar el acceso a la API de autenticación de Google seleccionando nuestro proyecto. Pulsamos en un botón Guardar situado al final de la página.

Ilustración 76: Consentimiento de OAuth

Y finalmente, ya tendremos acceso a la autenticacón de Google.

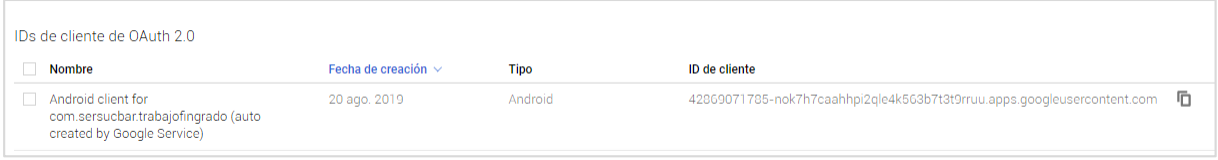


Ilustración 77: Credenciales de API OAUth generada

Firebase Auth tambien ofrece opciones de recuperación de cuentas de usuario y verificación de registro mediante correos electrónicos o SMS. En el proyecto que nos ocupa no haremos uso de ellas.

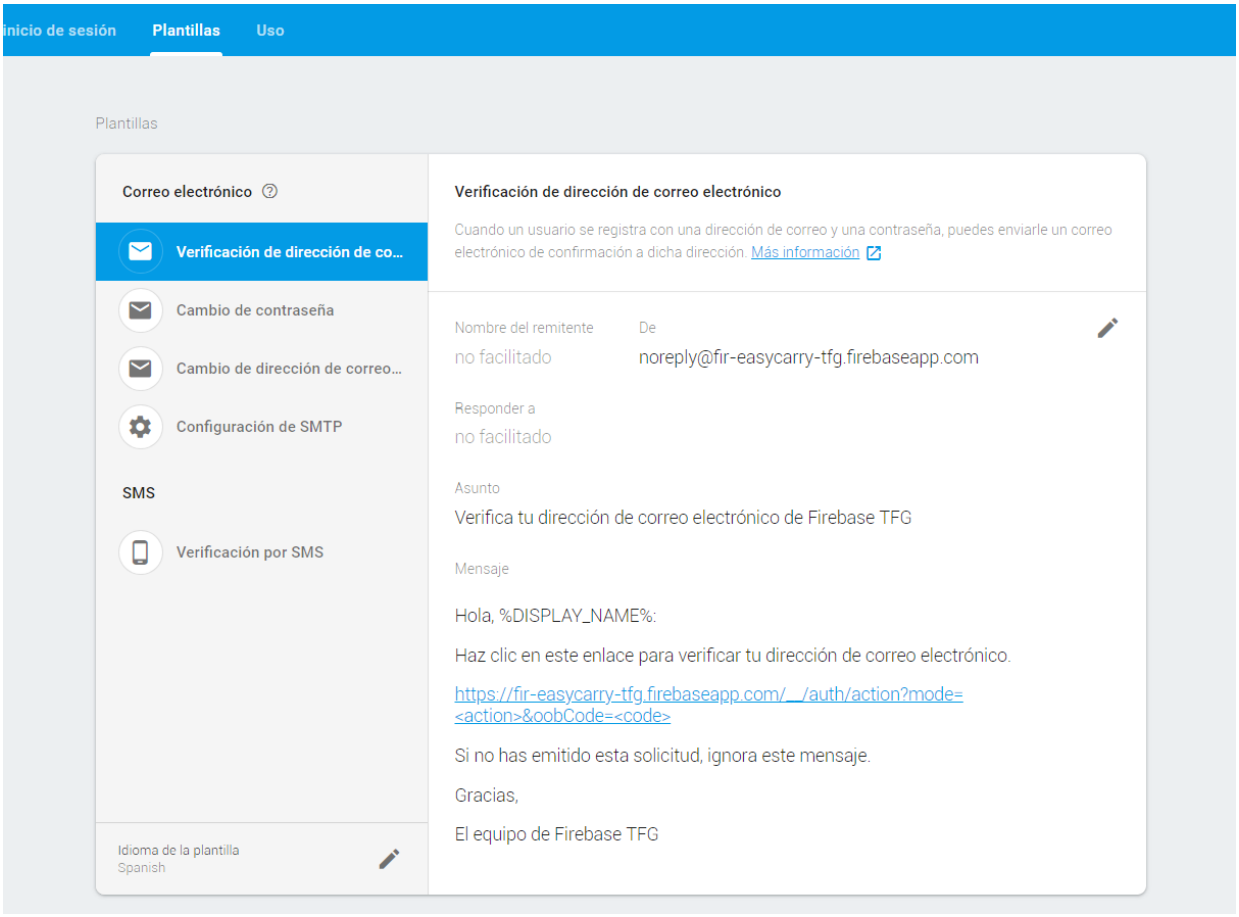


Ilustración 78: Plantilla de Firebase Auth

Implementación en el proyecto

En primer lugar, añadimos las dependencias:

```
implementation 'com.google.firebase:firebase-auth:19.0.0'
implementation 'com.firebaseui:firebase-ui-auth:4.3.1'
implementation 'com.google.android.gms:play-services-auth:17.0.0'
```

Código 15: Adición de dependencias de Firebase Authentication

Seguimos con la implementación del código para cada uno de los tipos de autenticación:

- El código necesario para crear una **cuenta de usuario con correo y contraseña** se requiere pasar los dos datos como parámetro de un método proporcionado por la librería:

```
// Se validan los datos y se procede a registrar
if (validar(correo, pass, passRep, nombre)) {
    loadDialog.show();

    mAuth.createUserWithEmailAndPassword(correo, pass)
    .addOnCompleteListener(RegistroActivity.this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {

        if (task.isSuccessful()) {
            // Registro correcto, se sube la foto y crea objeto en BBDD
            Usuario usuario = new Usuario(mAuth.getCurrentUser().getUid(), nombre);
            completaRegistro(usuario);
        } else
            muestraMensaje("No ha sido posible realizar el registro");
        }
    });
}
```

Código 16: Registro de usuario con correo y contraseña en Firebase Auth

- Para **inciar sesión con correo y contraseña** simplemente llamamos al método del objeto FirebaseAuth que realiza la autenticación:

```
mAuth.signInWithEmailAndPassword(correo, pass)
.addOnCompleteListener(LoginActivity.this, new OnCompleteListener<AuthResult>() {
    @Override public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()) {
            loadDialog.hide();
            nextActivity();
        } else {
            Toast.makeText(this, "Credenciales incorrectas.", Toast.LENGTH_SHORT).show();
        }
    }
});
```

Código 17: Autenticación de usuario con correo y contraseña en Firebase Auth

- Para **inicar sesión con cuenta de Google** se requieren más pasos:
    - Primero se inicializa el cliente con un objeto de opciones.

```
GoogleSignInOptions gso =
new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestEmail()
        .build();

mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
```

Código 18: Inicialización de cliente de Google SignIn

    - En segundo lugar se lanza un Intent con la pantalla de inicio de sesión de Google nativa de Android.

```
public void signIn(LoginActivity act) {
    Intent signInIntent = mGoogleSignInClient.getSignInIntent();
    act.startActivityForResult(signInIntent, REQUEST_GOOGLE_SIGN);
}
```

Código 19: Lanzamiento de pantalla de inicio de sesión con cuenta de Google

- Por último, tras seleccionar la cuenta deseada, se recupera la respuesta del Intent y se extrae la información de usuario que nos interesa.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    try {
        if (requestCode == REQUEST_GOOGLE_SIGN) {
            if (resultCode != RESULT_CANCELED) {
                loadDialog.show();

                Task<GoogleSignInAccount> task =
                    GoogleSignIn.getSignedInAccountFromIntent(data);
                GoogleSignInAccount account = task.getResult(ApiException.class);

                handleSignInResult(
                    account.getId(),
                    account.getDisplayName(),
                    account.getPhotoUrl()));
            }
        }

    } catch (ApiException | NullPointerException e) {
        Toast.makeText(this, "Ha ocurrido un error.", Toast.LENGTH_LONG).show();
    }
}
```

Código 20: Procesado de respuesta de inicio de sesión con cuenta de Google

- Para **cerrar sesión**, tanto el objeto FirebaseAuth como GoogleSignInClient poseen un método para ello:

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

GoogleSignInClient mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
mGoogleSignInClient.signOut();
```

Código 21: Cierre de sesión con los dos métodos de autenticación disponibles

Como hemos comprobado, implementar métodos de autenticación seguros y respaldados por los sistemas de Google Cloud es harto sencillo comparado con el desarrollo desde cero de una solución similar.

4. Firebase Cloud Storage

Configuración en la plataforma

Como ya sucedió con Realtime Database, desde la plataforma no es necesario realizar ninguna acción para configurar Firebase Storage, aunque es siempre recomendable establecer reglas para que la información no sea accesible por terceros. Esto se hace en la pestaña “Reglas”:

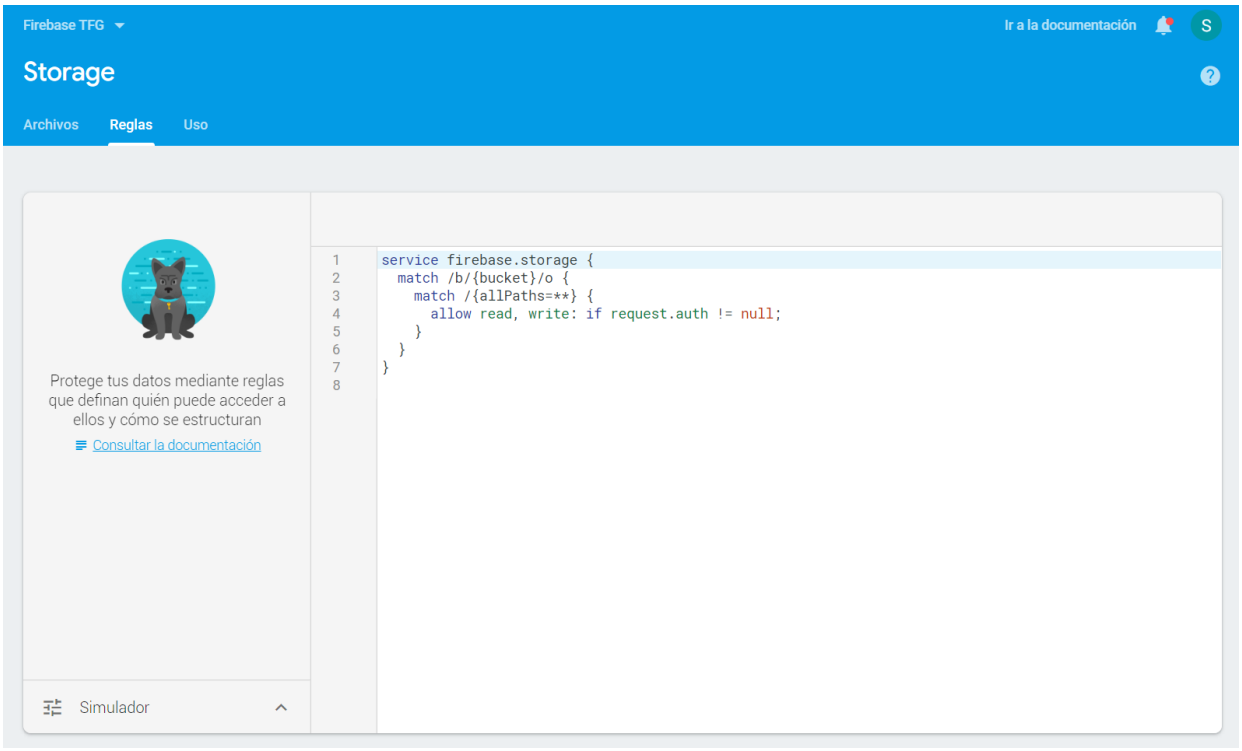


Ilustración 79: Reglas de seguridad de Storage

También, como en el caso de Database, podemos monitorizar el uso de la herramienta.

Cabe mencionar que una vez existe contenido Storage, podemos realizar acciones como eliminar archivos, ver las direcciones URL donde se encuentran alojadas y subir nuevos archivos desde la web.

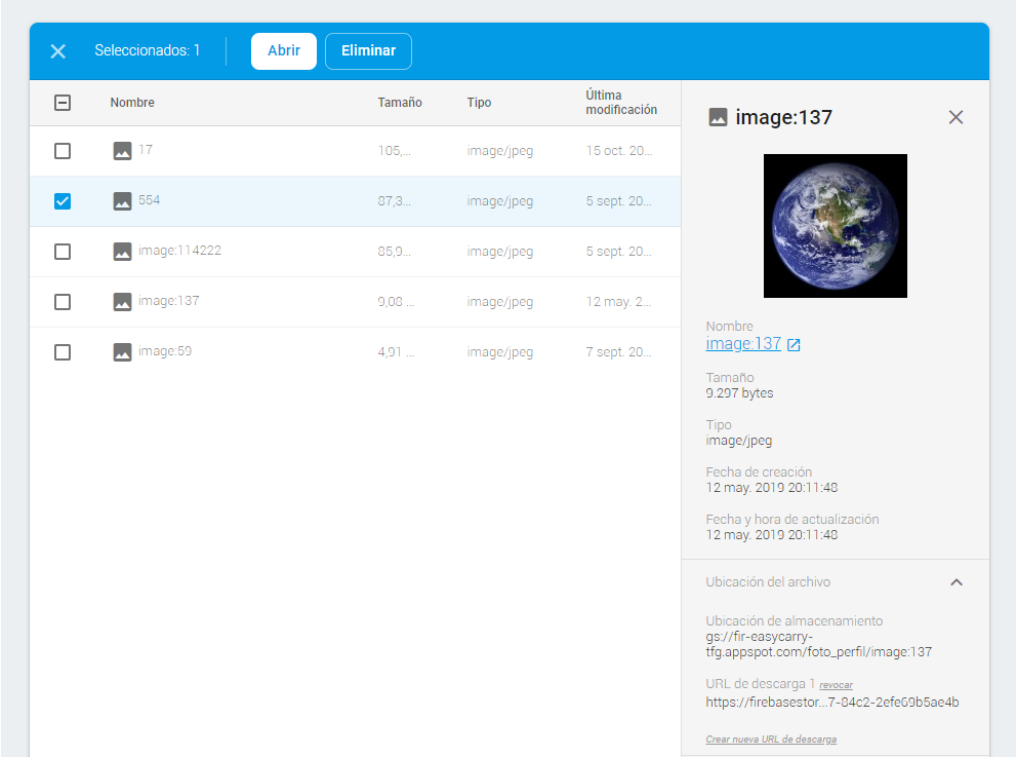


Ilustración 80: Datos en Storage

Implementación en el proyecto

En primer lugar, añadimos dependencias en *build.gradle*:

```
implementation 'com.google.firebase:firebase-storage:19.0.0'
```

Código 22: Adición de dependencias de Firebase Storage

Para la implementación de código, en nuestro caso tan solo hemos desarrollado la subida de archivos a Cloud Storage y hemos almacenado la URL resultante en BBDD para que tan solo haya que descargar desde ahí de forma tradicional:

```
// Ruta de Storage donde se subirá
final StorageReference fotoReferencia = storage.child(picture.getLastPathSegment());

// Listener que espera la subida de archivo
fotoReferencia.putFile(picture).addOnSuccessListener(this,
          new OnSuccessListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {

        // Listener que espera recibir la URL del archivo en Storage
        taskSnapshot.getMetadata().getReference().getDownloadUrl().addOnSuccessListener(
                  new OnSuccessListener<Uri>() {
            @Override
            public void onSuccess(Uri uri) {
                fotoPerfilCadena = uri.toString();
                nuevoUsuario.setAvatar(fotoPerfilCadena);
            }
        });
    }
});
```

Código 23: Ejemplo de subida de archivo a Firebase Storage

Una vez subido el archivo, podemos comprobar como la carpeta padre que se ha usado como referencia de Storage se ha creado correctamente:

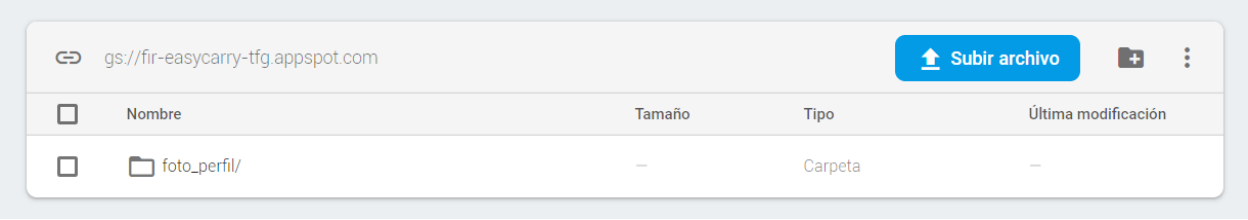


Ilustración 81: Carpeta en Storage

5. Firebase Cloud Messaging

Configuración en la plataforma

Desde la plataforma, no es necesario configurar Firebase Cloud Messaging si nuestro uso se limita a notificar desde dispositivos que implementen su biblioteca.

Sin embargo, la consola de Cloud Messaging ofrece posibilidades de creación de notificaciones interesantes.

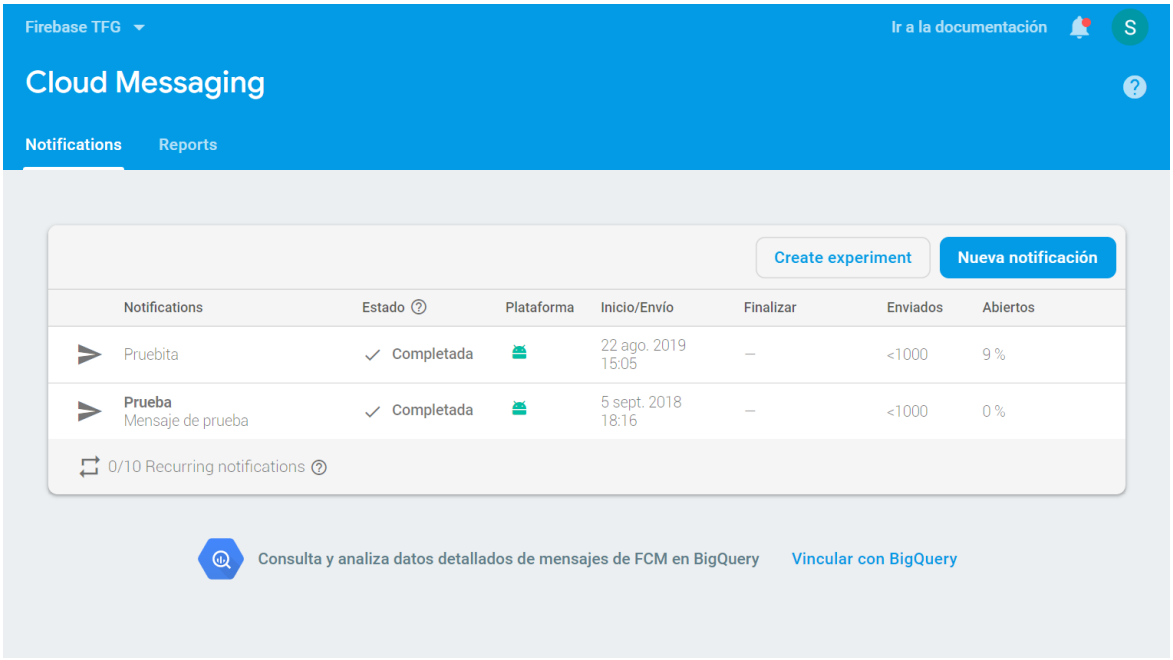


Ilustración 82: Sumario de FCM

En el menú principal, (ilustración anterior) pueden verse las notificaciones ya creadas con anterioridad y el impacto que han causado en los usuarios (ilustración siguiente).

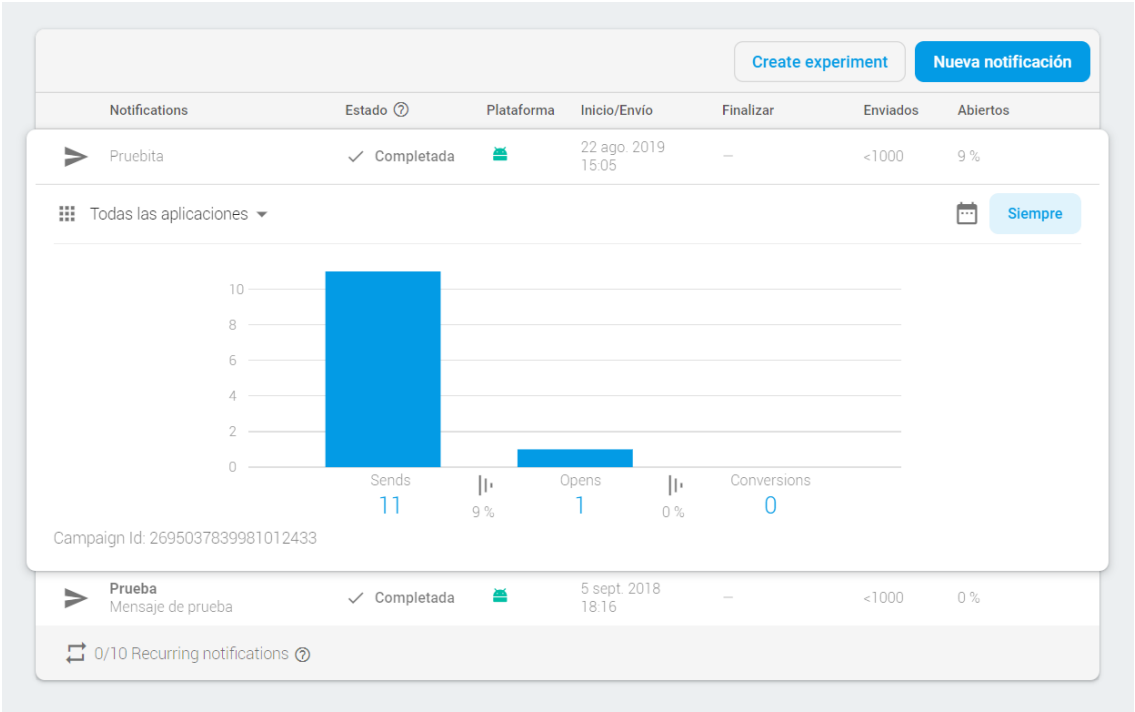


Ilustración 83: Detalle de notificación de FCM

Aunque no nos interesa para el proyecto, es bienvenido conocer que existe la posibilidad de monitorización de notificaciones globales, útil como modo de sondeo de campañas publicitarias o alertas de cualquier tipo.

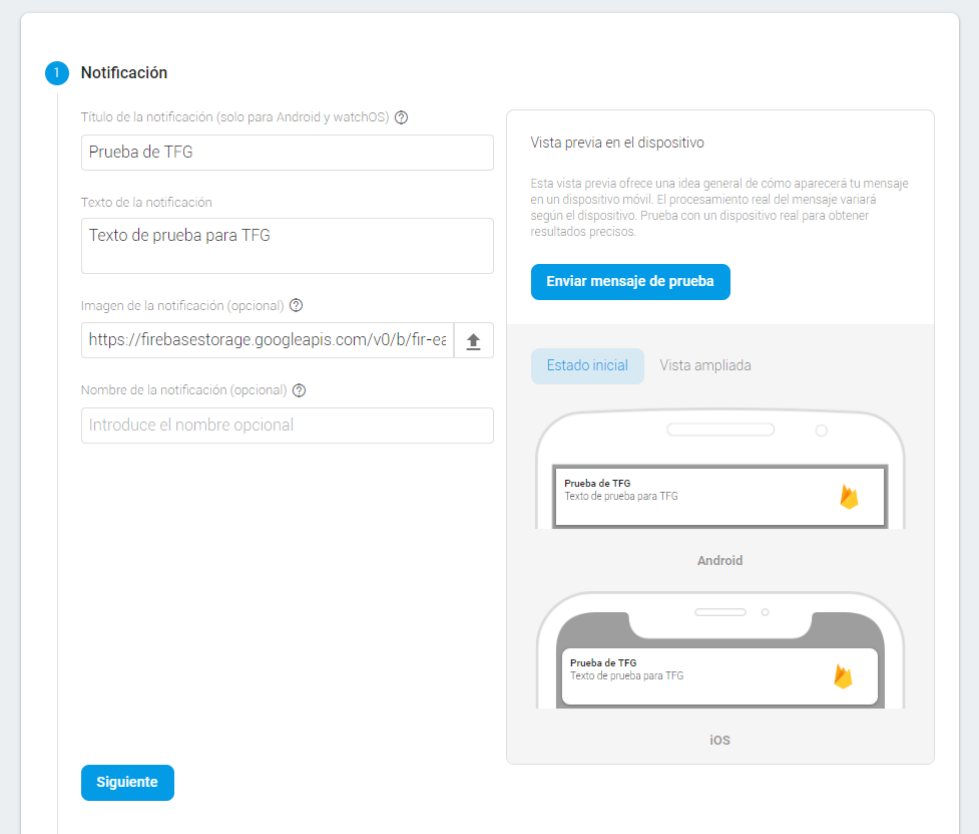


Ilustración 84: Creación de notificacion en FCM

En la imagen anteiror podemos ver como se configura una notificación global. Escribimos el contenido de la notificación y pulsamos Siguiente.

Tras rellenar los datos de segmentación y monitorización de la notificación, pulsamos en revisar y Enviar. Al instante llegará a los dispositivos afectados:

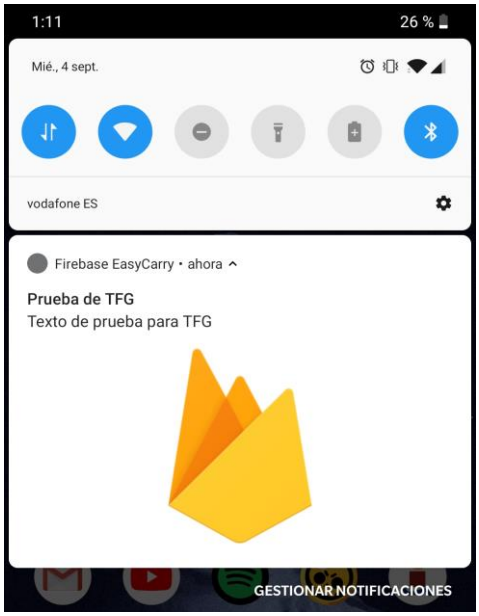


Ilustración 85: Recepción de notificación de FCM

Implementación en el proyecto

En primer lugar, añadimos dependencias en *build.gradle*:

```
implementation 'com.google.firebase:firebase-messaging:20.0.0
```

Código 24: Adición de dependencias de Firebase Cloud Messaging

A continuación, hay que configurar el archivo de manifiesto de la aplicación para añadir el servicio:

```
<service android:name=".servicios.MyFirebaseMessagingService">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>
```

Código 25: Adición de servicio en Manifest.xml

Si desearamos lanzar notificaciones a usuarios o dispositivos concretos, tendrías que generar un token en la primera ejecutción del servicio y luego reutilizarlo. No es este el caso en nuestra aplicación.

Ya solo nos quedaría implementar la clase del servicio de FCM:

```
public class MyFirebaseMessagingService extends FirebaseMessagingService {

    Preferences prefs;

    @Override
    public void onCreate() {
        super.onCreate();
        prefs = new Preferences(this);
    }

    @Override
    public void onMessageReceived(RemoteMessage msg) {
        // Comprueba si hay datos en el mensaje
        if (msg.getData().size() > 0) {
         // Código de procesado de mensaje
        }
    }
}
```

Código 26: Implementación del servicio de Firebase Cloud Messaging

Esta implementación solo comprende la parte del cliente de Cloud Messaging, es decir, el receptor de la notificacion. También es necesario configurar el tipo de notificación que se va a lanzar desde el servidor. En nuestro caso el servidor será Firebase Functions con los métodos *serverless* que implementaremos.

Adelantamos ya el formato JSON que deben tener los mensajes de notificacion desde el servidor [8]:

- **Mensaje de notificación**: es la que se muestra automáticamente al llegar al servicio del dispositivo cliente final. La notificación se muestra con los datos de pares clave-valor predefinidos. No pasa por el servicio implementado, se muestra de forma nativa.

```
{
  "message":{
    "token":"bk3RNwTe3H0:CI2k HHwgIpoDKCIZvvDMExUdFQ3P1...",
    "notification":{
      "title":"${dataSnap.autor} está retransmitiendo su localización",
      "body":"Pulsa para entrar en el chat"
    }
  }
}
```

Código 27: Ejemplo de formato de mensaje de tipo notificación de FCM

- **Mensaje de datos**: es la que solo posee pares clave-valor personalizados. Debe implementarse el procesado de los datos y la construcción de la notificación en el cliente. Pasa por el servicio implementado a través del listener onMessageReceived( ).

```
{
  "message":{
    "token":"bk3RNwTe3H0:CI2k HHwgIpoDKCIZvvDMExUdFQ3P1...",
    "data":{
      "chatId" : '${dataSnap.chatId}',
      "title" : '${dataSnap.autor} está retransmitiendo su localización',
      "Room" : "Pulsa para entrar en el chat"
    }
  }
}
```

Código 28: Ejemplo de formato de mensaje de tipo dato de FCM

- **Mensaje de notificación con carga de datos**: es un híbrido entre las dos anteriores. El comportamiento del servicio depende de si está en primer o segundo plano.
    - En segundo plano, recibe la carga de notificación y solo maneja la de datos cuando se pulsa en la notificación.
    - En primer plano, la app recibe un objeto notificación con ambas cargas.

En nuestro sistema se ha optado por el mensaje de datos. Esto permitía más flexibilidad a la hora de personalizar la notificación y lo más importante, lanzar un Intent pasándole datos extra al Activity.

Existen muchos parámetros de mensajes JSON para FCM y parámetros de configuración (prioridad, TTL, acciones de clic, etc.). Puede encontrarse en la documentación adjunta en bibliografía.

6. Firebase Functions

Configuración en la plataforma

La configuración comprende varios pasos, ya que Functions requiere la instalación de una consola local para realizar las tareas de administración de las funciones publicadas. También está entre los pasos el enlace con FCM, dado que en el caso que nos ocupa ambos trabajan juntos.

1. Primero hay que generar una clave privada del SDK de Firebase Admin desde el menú de configuración del proyecto:

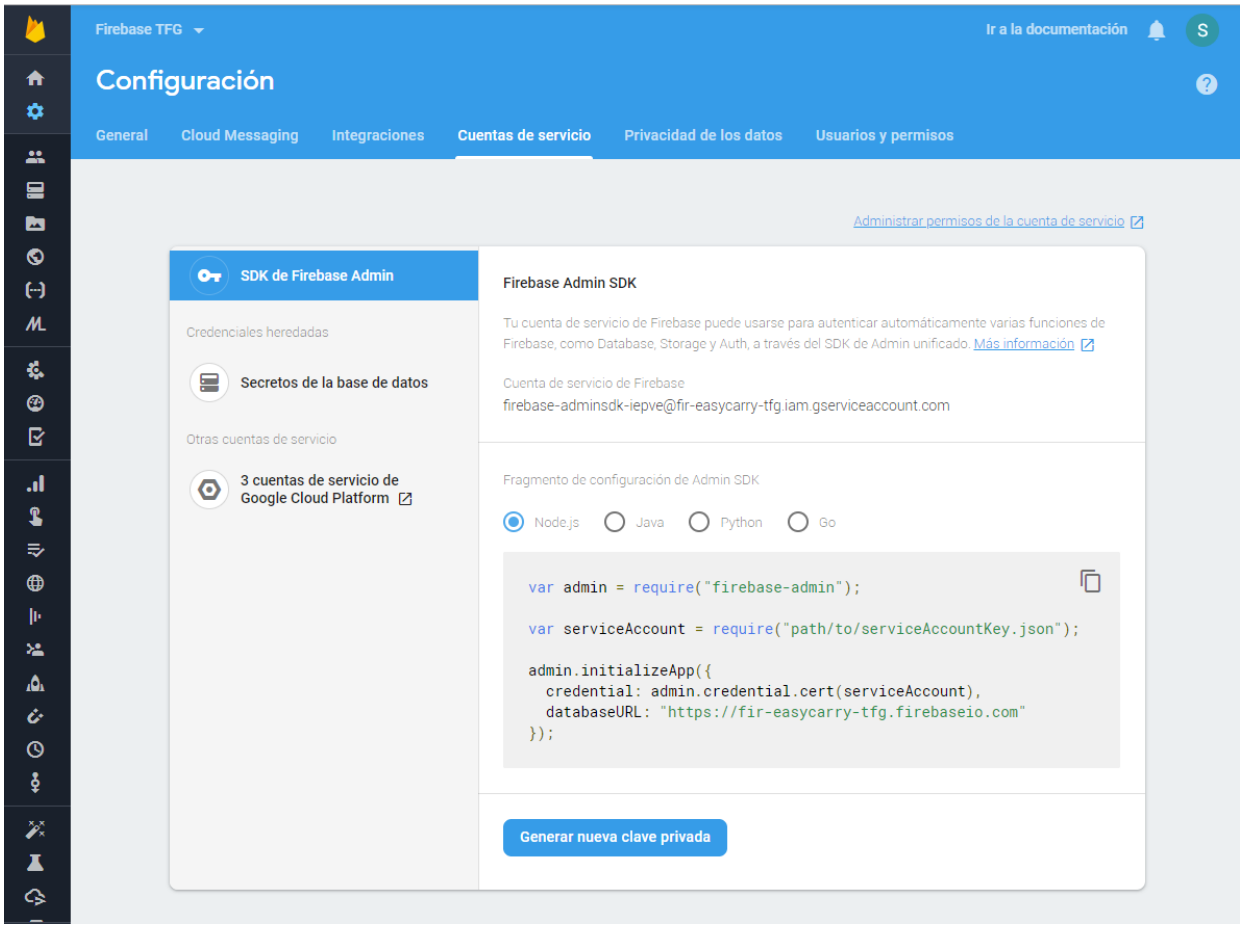


Ilustración 86: Generación de clave de Admin SDK Functions

2. Utilizamos esa clave para establecer Functions como servidor de Cloud Messaging:

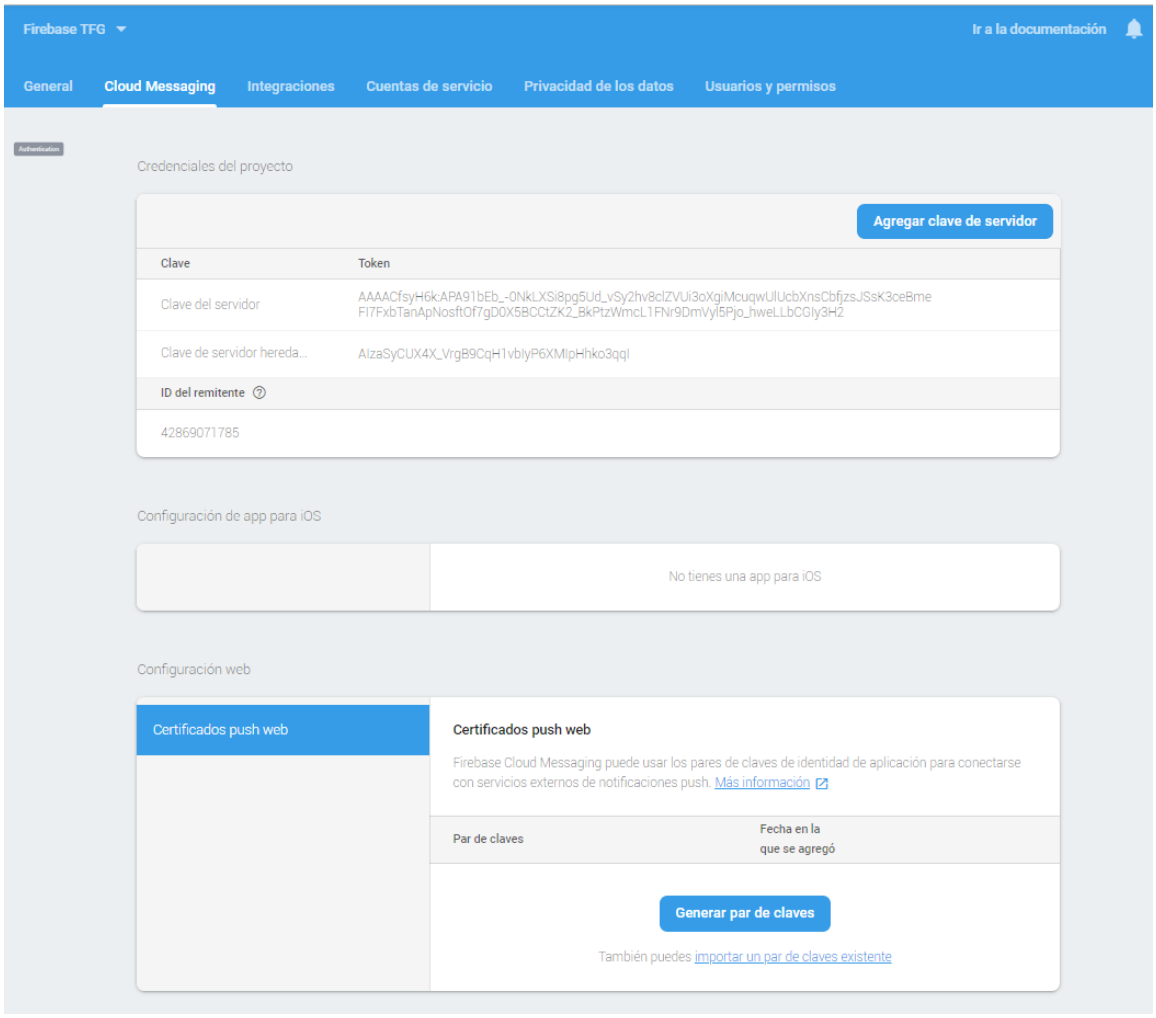


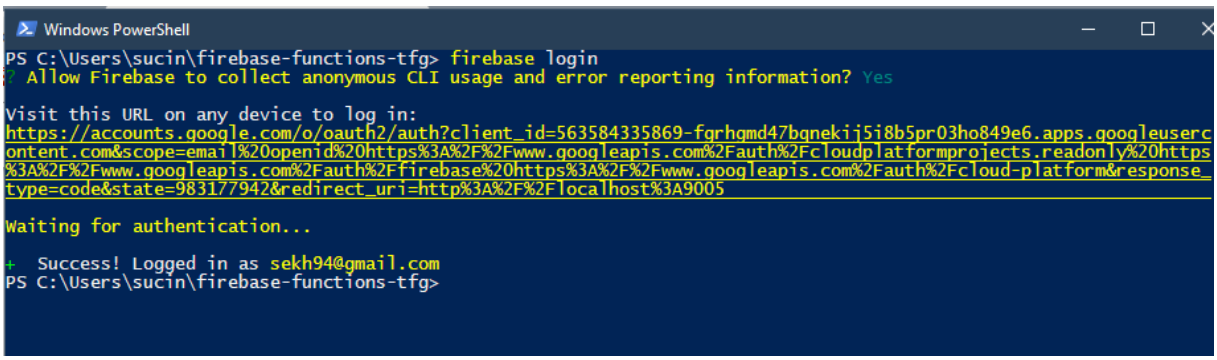
Ilustración 87: Agregación de clave de servidor de Functions a FCM

3. Se procede a instalar Firebase CLI. Para ello hemos tenido que instalar previamente Node.js y npm.

```
npm install -g firebase-tools
```

Código 29: instalación de Firebase CLI

4. Se inicia sesión en la consola:



Código 30: Login de Firebase CLI

Para inciar sesión también hay que pulsar en el enlace proporcionado y autenticarse en Google Cloud Platform. Si no se está registrado previamente, ahora sería el momento.

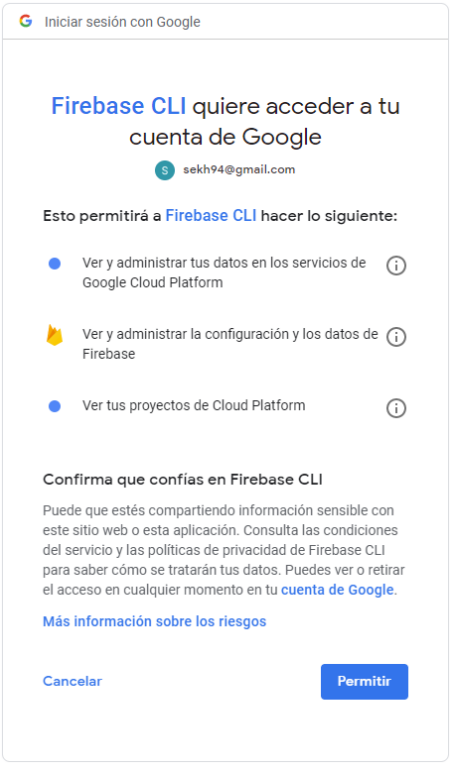


Ilustración 88: Permisos de cuenta para Firebase CLI

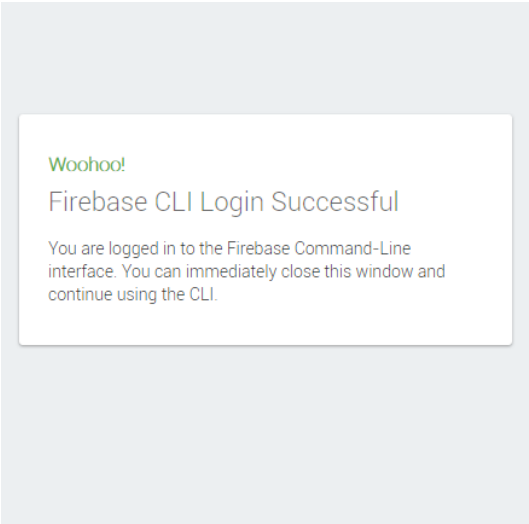


Ilustración 89: Login correcto de Firebase CLI

5. Se inicia el administrador de Firebase Functions en consola:

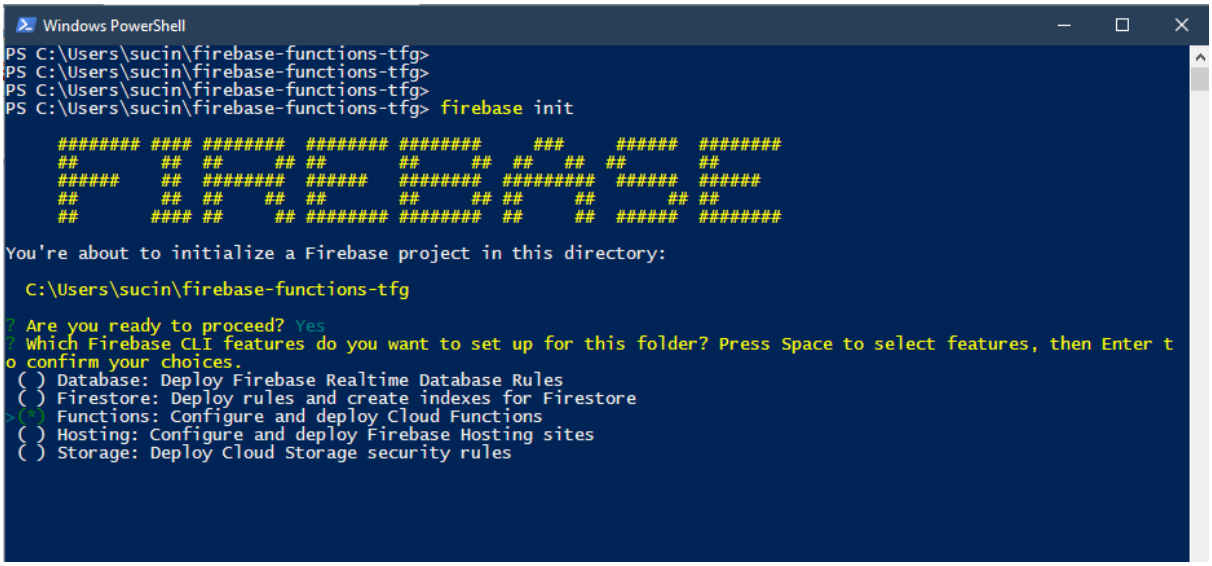


Ilustración 90: Arranque de Firebase CLI

6. Tras elegir las opciones de configuración para el proyecto deseado y empezará a descargar dependencias:

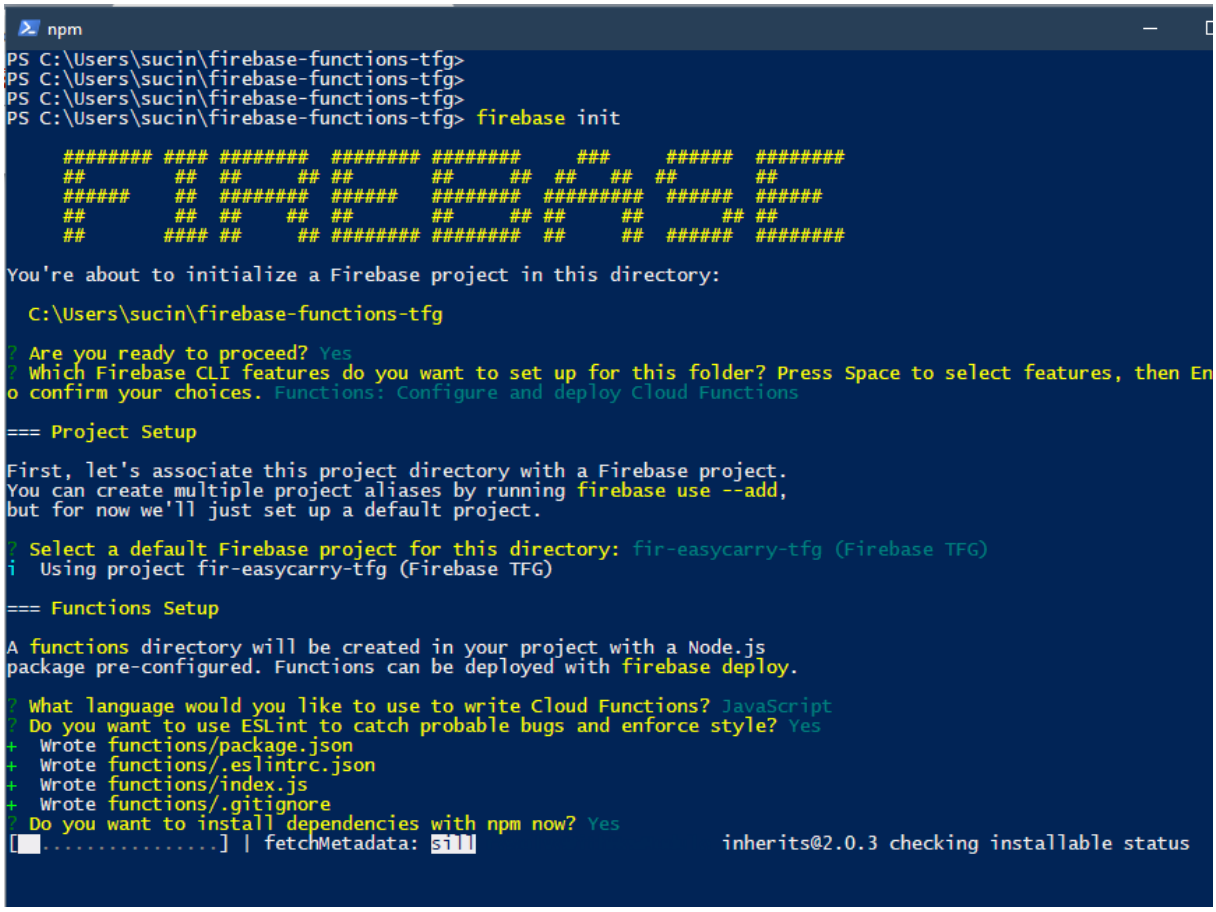


Ilustración 91: Descarga de dependencias de Firebase CLI

Al finalizar la descarga ya se puede comenzar a desarrollar para Firebase Functions.

Implementación en el proyecto

La implementación pasa por desarrollar los métodos de evento deseados en **Node.js**. Estos se ubicarán en la carpeta que se habrá creado tras la instalación de firebase-tools.

En nuestro caso el archivo *{functionsPath}/functions/index.js* es el que contiene el código.

Debe comenzar por una inicialización del SDK Admin:

```
var admin = require('firebase-admin');
var functions = require('firebase-functions')
var serviceAccount = require("./sdk-admin/fir-easycarry-tfg-firebase-adminsdk.json");

// Firebase Cloud Messaging Server API key
var API_KEY = "AAAACfsyH6k:APA91bEb_-
0NkLXSi8pg5Ud_vSy2hv8clZVUi3oXgiMcuqwUlUcbXnsCbfjzsJSsK3ceBmeFI7FxbTanApNosftOf7gD0X5BCCtZ
K2_BkPtzWmcL1FNr9DmVyl5Pjo_hweLLbCGIy3H2";

// Incializar app con cuenta del servicio con privilegios de administrador
admin.initializeApp({
        credential: admin.credential.cert(serviceAccount),
        databaseURL: "https://fir-easycarry-tfg.firebaseio.com"
});
```

Código 31: Inicialización del SDK Admin de Firebase Functions

A continuación, se ubicarán los métodos a exportar a Functions:

```
exports.notificaLocalizacion =
functions.database.ref('/Notificaciones/Localizacion/{notificationId}').onCreate(
        (dataSnapshot) => {

    { ... }

});
```

Código 32: Ejemplo de método exportado de Firebase Functions

El código completo se encuentra en el apartado de ANEXOS.

Para **publicar** en la plataforma la implementación:

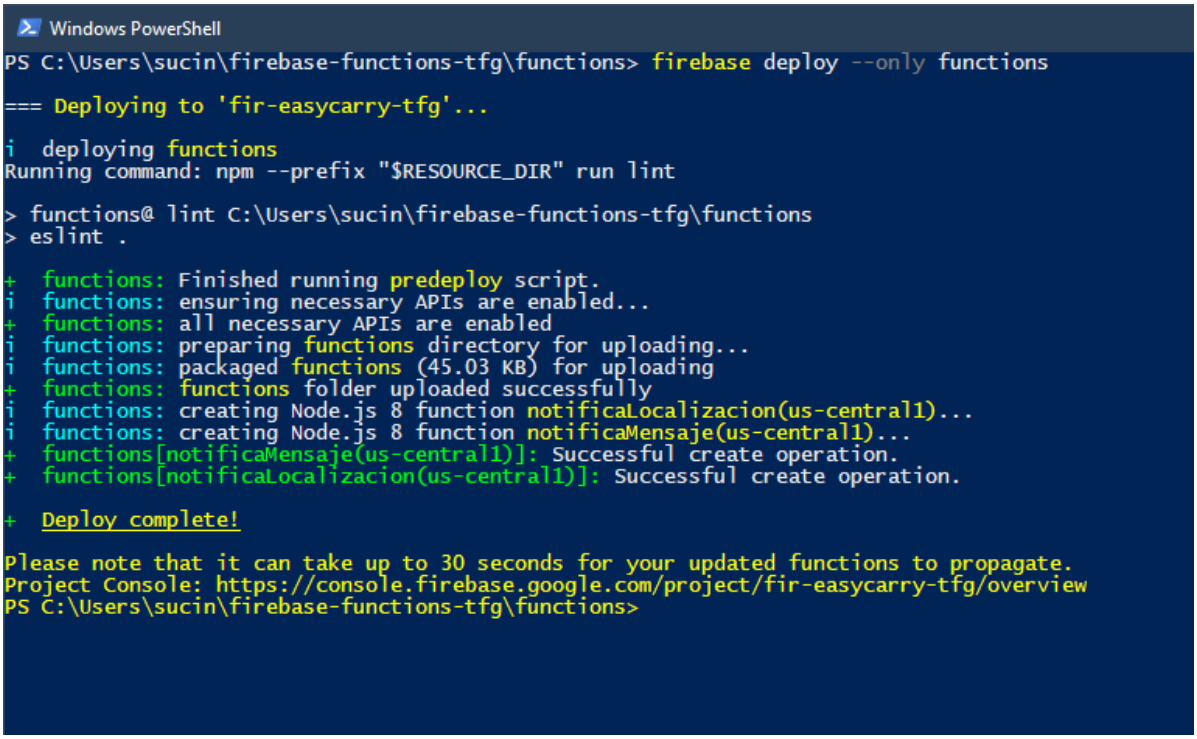


Ilustración 92: Despliegue de scripts en Functions

Con cualquier cambio en el archivo *index.js*, el comando provocará que se borre la versión anterior de la función y se publique la nueva.

Desde la consola de Firebase podemos ver las funciones publicadas:

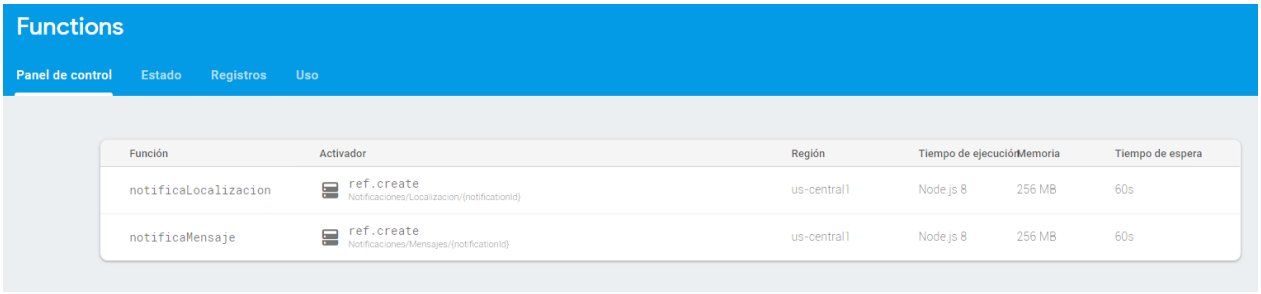


Ilustración 93: Funciones exportadas en Functions

Así como un log de ejecución de las funciones exportadas:

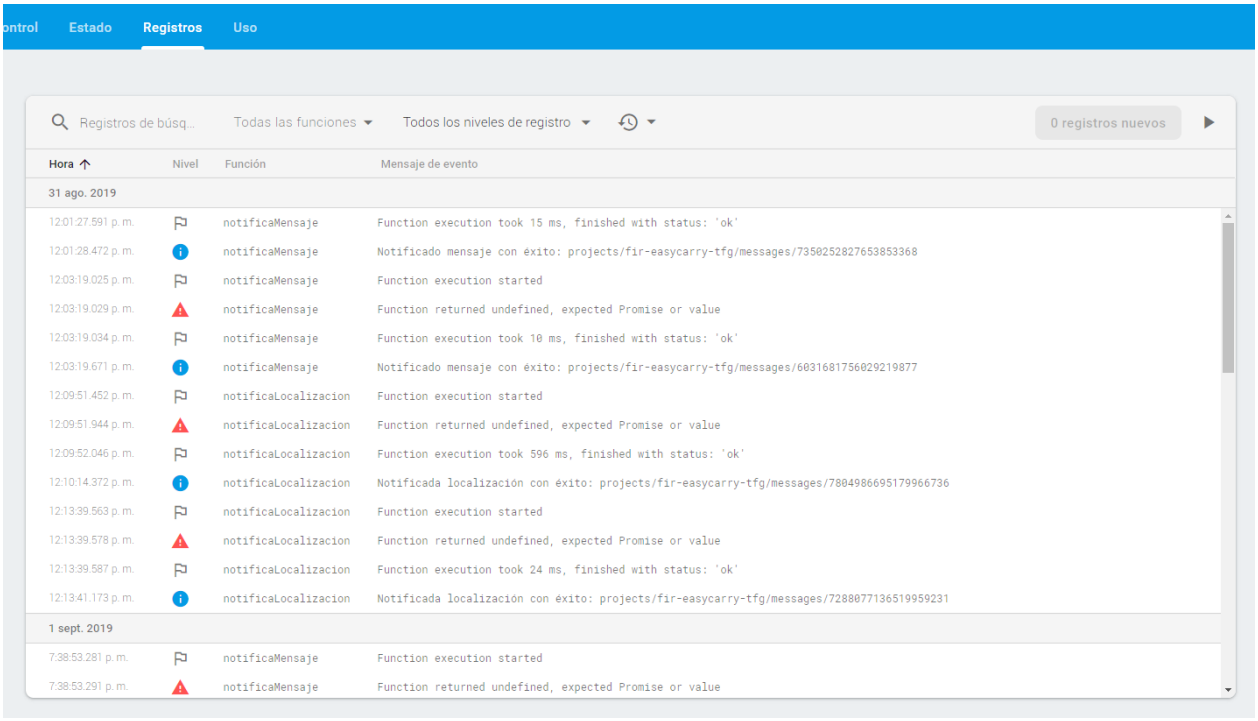


Ilustración 94: Log de ejecución de Functions

7. Google Maps Platform

Configuración de la plataforma

Para comenzar a utilizar la plataforma de Google Maps, son necesarios una serie de pasos:

1. Crear una cuenta de pago de Google Cloud: es necesaria porque se utilizará la API Directions, que no es gratuita. Pese a esto, tenemos un periodo de prueba de 12 meses con un crédito gratis de 300 USD. Prueba de ello es la imagen siguiente.

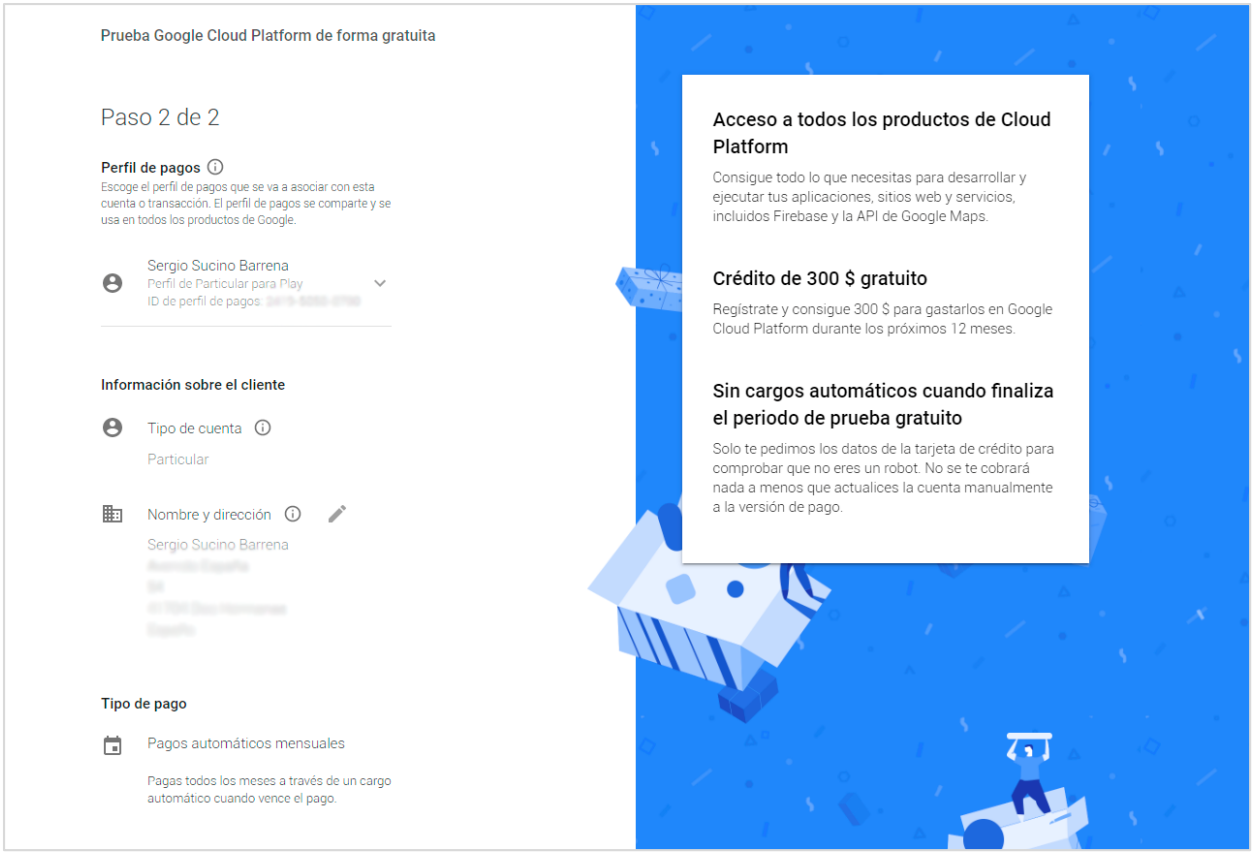


Ilustración 95: Creación de cuenta de pago en Google Cloud

2. Activar el acceso a *Maps SDK for Android* y *Directions API* desde el menú *APIs y servicios* > *Biblioteca* de la consola de Google Cloud Platform. Puede verse en las imágenes tras el texto:

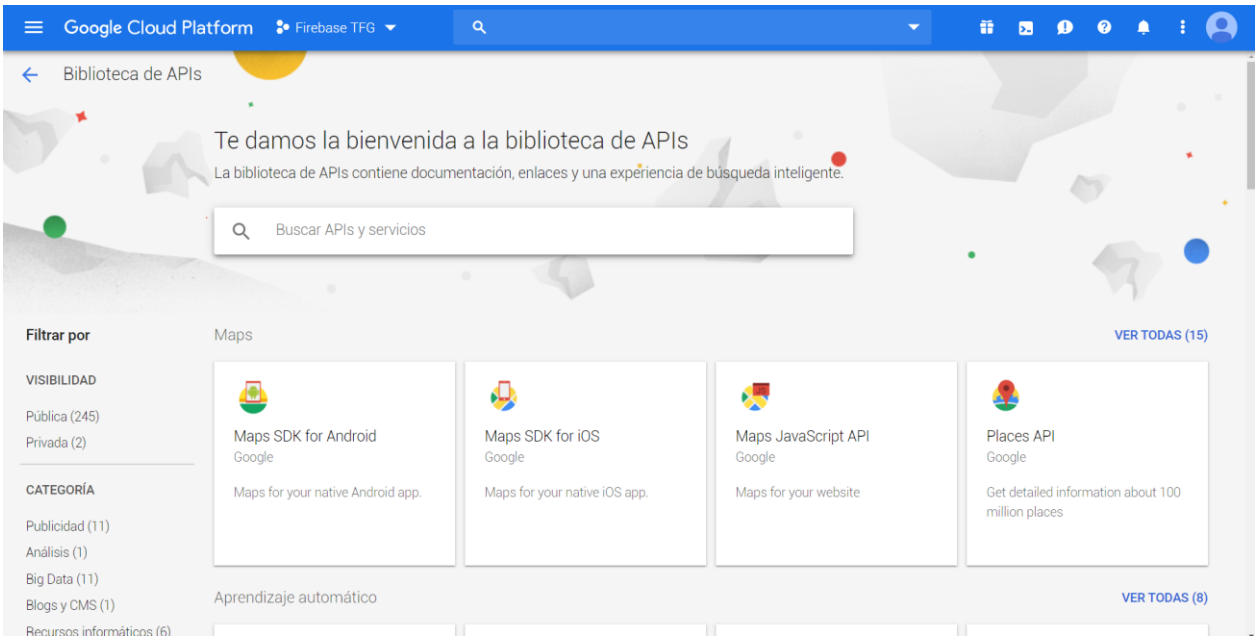


Ilustración 96: Menú bibliotecas en Google Cloud

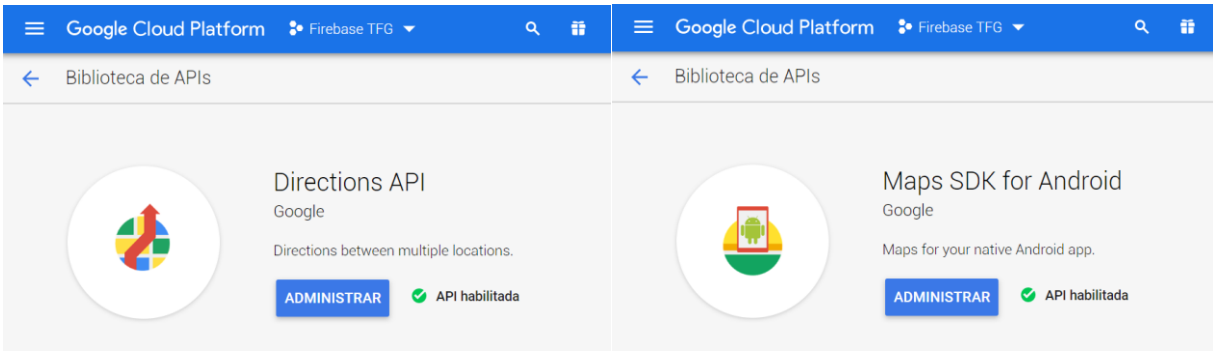


Ilustración 97: Activación de las API necesarias para Maps

3. Crear dos credenciales de API. Las necesitaremos para vincular cada API de Maps con la clave de API (también llamada API Key) que será generada.

   En la misma consola del punto anterior, vamos a la sección de *APIs y servicios* > *Credenciales*. A continuación generamos dos credenciales, como se ve en la imagen a continuación.

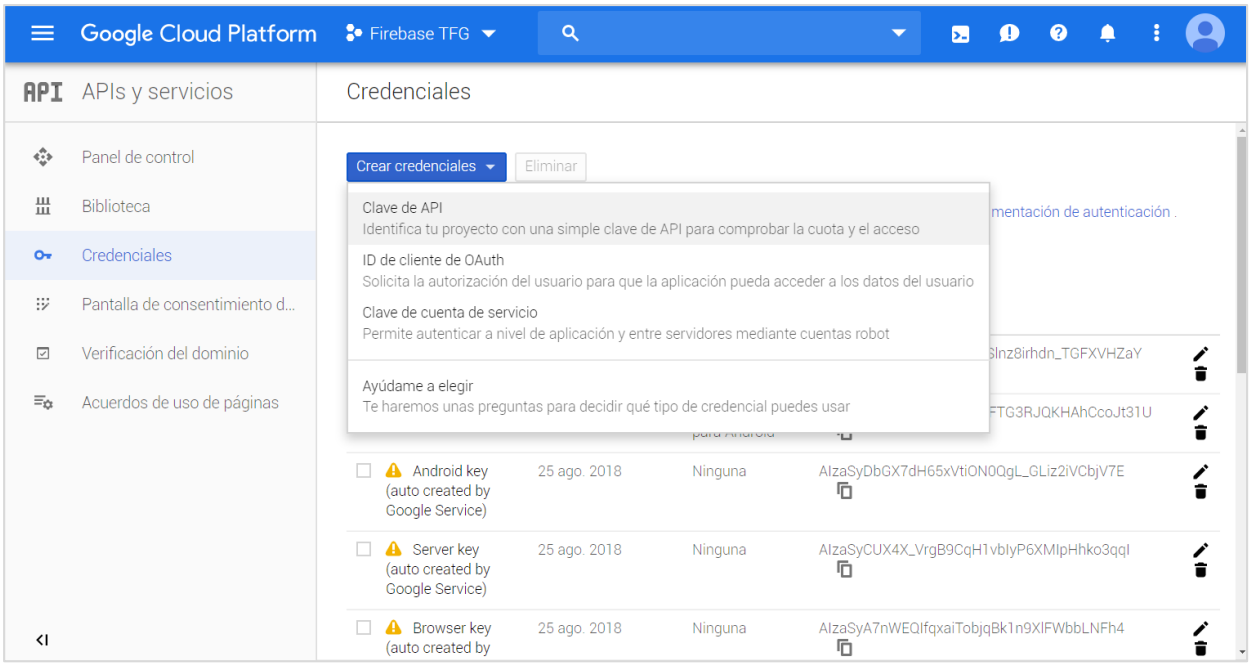


Ilustración 98: Creación de credencial de API para Maps

Tras estos pasos, ya se pueden implementar las funciones de Maps Platform en nuestra aplicación.

Implementación en el proyecto

Para implementar las API que hemos utilizado, en primer lugar hay incluir las claves de API de las credenciales:

```
<string name="google_maps_key">AIzaSyCRqHePTYS65WEyFTG3RJQKHAhCcoJt31U</string>
<string name="google_directions_key"AIzaSyCQwmqHGcoKG7aSlnz8irhdn_TGFXVHZaY</string>
```

Código 33: Almacenamiento de las claves de API en *google_maps_api.xml* en el proyecto

Importamos las dependencias en build.gradle:

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

Código 34: Adición de dependencias de Firebase Cloud Messaging

Para implementar una **vista de mapa** con la biblioteca de Maps solo habría que:

- Añadir el fragment correspondiente en el layout que utilicemos:

```
<fragment
    android:id="@+id/mapView"
    android:name="com.google.android.gms.maps.MapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activity.LocalizacionActivity" />
```

Código 35: Adición de vista de mapa en archivo layout

- Implementar en nuestra actividad las clases que requiramos para el control de flujo en el uso del mapa. Por ejemplo, OnMapReadyCallback contiene el método *onMapReady*, que se ejecuta cuando el mapa ha cargado.

Para implementar una **petición a la API Directions,** que en este caso se utiliza para calcular la ruta entre dos puntos, hay que construir una petición HTTP GET.

Se han utilizado clases aportadas por un desarrollador de GitHub para ayudarnos a ello [9].

```
private String getUrl(LatLng origin, LatLng dest) {
    // Origin of route
    String str_origin = String.format("origin=%s,%s", origin.latitude, origin.longitude);
    // Destination of route
    String str_dest = String.format("destination=%s,%s", dest.latitude, dest.longitude);
    // Mode
    String mode = "mode=driving";
    // Building the parameters to the web service
    String parameters = String.format("%s&%s&%s", str_origin, str_dest, mode);
    // Output format
    String output = "json";

    // Building the url to the web service
    return String.format("https://maps.googleapis.com/maps/api/directions/%s?%s&key=%s",
output, parameters, getString(R.string.google_directions_key));
}
```

Código 36: Método que contruye la petición a la API Directions

Tras la obtención de la URL, se realiza la petición HTTP GET. En la respuesta se encuentra la información para construir la ruta, pero no vamos a explicar ese procedimiento en este apartado.

ANEXO 2: POLÍTICA DE PRIVACIDAD

[10]

Sergio Sucino built the Firebase TFG app as a Free app. This SERVICE is provided by Sergio Sucino at no cost and is intended for use as is.

This page is used to inform visitors regarding my policies with the collection, use, and disclosure of Personal Information if anyone decided to use my Service.

If you choose to use my Service, then you agree to the collection and use of information in relation to this policy. The Personal Information that I collect is used for providing and improving the Service. I will not use or share your information with anyone except as described in this Privacy Policy.

The terms used in this Privacy Policy have the same meanings as in our Terms and Conditions, which is accessible at Firebase TFG unless otherwise defined in this Privacy Policy.

Information Collection and Use

For a better experience, while using our Service, I may require you to provide us with certain personally identifiable information, including but not limited to user data, location, app usage. The information that I request will be retained on your device and is not collected by me in any way.

The app does use third party services that may collect information used to identify you.

Link to privacy policy of third party service providers used by the app

- Google Play Services
- Firebase

Log Data

I want to inform you that whenever you use my Service, in a case of an error in the app I collect data and information (through third party products) on your phone called Log Data. This Log Data may include information such as your device Internet Protocol (“IP”) address, device name, operating system version, the configuration of the app when utilizing my Service, the time and date of your use of the Service, and other statistics.

Cookies

Cookies are files with a small amount of data that are commonly used as anonymous unique identifiers.

These are sent to your browser from the websites that you visit and are stored on your device's internal memory.

This Service does not use these “cookies” explicitly. However, the app may use third party code and libraries that use “cookies” to collect information and improve their services. You have the option to either accept or refuse these cookies and know when a cookie is being sent to your device. If you choose to refuse our cookies, you may not be able to use some portions of this Service.

Service Providers

I may employ third-party companies and individuals due to the following reasons:

- To facilitate our Service;
- To provide the Service on our behalf;
- To perform Service-related services; or
- To assist us in analyzing how our Service is used.

I want to inform users of this Service that these third parties have access to your Personal Information. The reason is to perform the tasks assigned to them on our behalf. However, they are obligated not to disclose or use the information for any other purpose.

Security

I value your trust in providing us your Personal Information, thus we are striving to use commercially acceptable means of protecting it. But remember that no method of transmission over the internet, or method of electronic storage is 100% secure and reliable, and I cannot guarantee its absolute security.

Links to Other Sites

This Service may contain links to other sites. If you click on a third-party link, you will be directed to that site. Note that these external sites are not operated by me. Therefore, I strongly advise you to review the Privacy Policy of these websites. I have no control over and assume no responsibility for the content, privacy policies, or practices of any third-party sites or services.

Children’s Privacy

These Services do not address anyone under the age of 13. I do not knowingly collect personally identifiable information from children under 13. In the case I discover that a child under 13 has provided me with personal information, I immediately delete this from our servers. If you are a parent or guardian and you are aware that your child has provided us with personal information, please contact me so that I will be able to do necessary actions.

Changes to This Privacy Policy

I may update our Privacy Policy from time to time. Thus, you are advised to review this page periodically for any changes. I will notify you of any changes by posting the new Privacy Policy on this page. These changes are effective immediately after they are posted on this page.

Contact Us

If you have any questions or suggestions about my Privacy Policy, do not hesitate to contact me at sekh94@gmail.com

GLOSARIO

API: Application Programming Interface (Interfaz de programación de aplicaciones).

GET: tipo de solicitud centrada en recuperar datos.

IDE: Integrated Development Environment (entorno de desarrollo integrado)

HTTP: Hypertext Transfer Protocol (Protocolo de transferencia de hipertexto).

REST: Representational State Transfer (Transferencia de estado representacional), arquitectura software.

SDK: Software Development Kit (Kit de desarrollo de software).

NoSQL: sistema de base de datos que no sigue los modelos clásicos.

JSON: JavaScript Object Notation (notación de objeto de JavaScript)

TI: tecnología de la información

UML: Unified Modeling Language (lenguaje unificado de modelado)

REFERENCIAS

[1] Wikipedia, «Cloud computing,» [En línea]. Available: https://en.wikipedia.org/wiki/Cloud_computing.

[2] Microsoft, «¿Qué es la informática en la nube?,» [En línea]. Available: https://azure.microsoft.com/es-es/overview/what-is-cloud-computing/.

[3] M. PRIETO, «Ni Microsoft ni Google, nadie puede con Amazon en la nube,» 27 febrero 2019. [En línea]. Available: https://www.expansion.com/economia-digital/companias/2019/02/27/5c6ef3b2ca4741474b8b45c5.html.

[4] R. Borillo, «Qué es serverless y por qué adoptarlo en el desarrollo de tu próxima aplicación,» [En línea]. Available: https://www.genbeta.com/desarrollo/que-serverless-que-adoptarlo-desarrollo-tu-proxima-aplicacion.

[5] Wikiepdia, «Wikipedia - Android Studio,» [En línea]. Available: https://es.wikipedia.org/wiki/Android_Studio.

[6] WangJianJun, «GitHub - AvatarImageView,» [En línea]. Available: https://github.com/Carbs0126/AvatarImageView.

[7] Bump Technologies, [En línea]. Available: https://github.com/bumptech/glide.

[8] J. Ericksen, «GitHub - Parceler,» [En línea]. Available: https://github.com/johncarl81/parceler.

[9] Google, «¿Qué puedo hacer con Cloud Functions?,» [En línea]. Available: https://firebase.google.com/docs/functions/use-cases?hl=es-419.

[10] Nubalia, «Google Maps Platform, soluciones para desarrolladores,» [En línea]. Available: https://nubalia.com/blog/google-maps-platform-soluciones-desarrolladores/.

[11] Google, «Docs Firebase - Recupera datos,» [En línea]. Available: https://firebase.google.com/docs/database/admin/retrieve-data?hl=es-419.

[12] Google, «Docs Firebase - Start an Activity from a Notification,» [En línea]. Available: https://developer.android.com/training/notify-user/navigation.

[13] Firebase, «GitHub - Firebase quickstart Android,» [En línea]. Available: https://github.com/firebase/quickstart-android.

[14] V. Shrestha, «GitHub - Vysh01/android-maps-directions,» [En línea]. Available: https://github.com/Vysh01/android-maps-directions.

[15] Google, «Acerca de los mensajes de FCM,» [En línea]. Available: https://firebase.google.com/docs/cloud-messaging/concept-options?hl=es-419.

[16] Nishant, «App Privacy Policty Generator,» [En línea]. Available: https://app-privacy-policy-

generator.firebaseapp.com/.

[17] A. Hathibelagal, «How to Create an Android Chat App Using Firebase,» 27 octubre 2016. [En línea]. Available: https://code.tutsplus.com/tutorials/how-to-create-an-android-chat-app-using-firebase--cms-27397.

[18] Google, «Documentation - Maps SDK for Android,» [En línea]. Available: https://developers.google.com/maps/documentation/android-sdk/.

[19] Firebase, «Firebase Android Codelab - Build Friendly Chat,» [En línea]. Available: https://codelabs.developers.google.com/codelabs/firebase-android/#0.

[20] K. P. F. Perez, «GitHub - kevin4dhd/FireBaseChat,» [En línea]. Available: https://github.com/kevin4dhd/FireBaseChat.

[21] T. -. S. Blog, «Android Chat Tutorial: Building a Messaging UI,» [En línea]. Available: https://blog.sendbird.com/android-chat-tutorial-building-a-messaging-ui.

[22] F. v. Puffelen, «The Firebase Blog - Sending notifications between Android devices with Firebase Database and Cloud Messaging,» [En línea]. Available: https://firebase.googleblog.com/2016/08/sending-notifications-between-android.html.

[23] Firebase, «GitHub - firebase/functions-samples,» [En línea]. Available: https://github.com/firebase/functions-samples/.

[24] J. Thornsby, «Android Authority - Create a GPS tracking application with Firebase Realtime Database,» 22 marzo 2018. [En línea]. Available: https://www.androidauthority.com/create-a-gps-tracking-application-with-firebase-realtime-databse-844343/.

[25] Google, «Docs Firebase - Agrega el SDK de Firebase Admin a tu servidor,» [En línea]. Available: https://firebase.google.com/docs/admin/setup.

[26] Google, «Docs Firebase - Cloud Messaging,» [En línea]. Available: https://firebase.google.com/docs/cloud-messaging/.

[27] Google, «Docs Firebase - Recibe mensajes en una app para Android,» [En línea]. Available: https://firebase.google.com/docs/cloud-messaging/android/receive?hl=es-419.

[28] S. Heinen, «StackOverflow - Converting a view to Bitmap without displaying it in Android,» [En línea]. Available: https://stackoverflow.com/a/3036736/12003746.

[29] D. SUNNEBO, «Ventas de Smartphones: Samsung contraataca en Europa,» [En línea]. Available: https://es.kantar.com/tech/m%C3%B3vil/2019/abril-2019-cuota-de-mercado-de-smartphones/.