



Encuentra
a tu canguro
más cercan@



Jesús González
David Arribas



INDICE

INDICE.....	1
1. Idea del proyecto	3
2. Objetivos.....	3
3. Arquitectura de la app.....	4
4. Tecnologías empleadas	6
4.1 IDE y lenguaje de programación	6
4.2 Base de datos	7
4.3 Diseño y prototipo.....	7
4.4 Control de versiones Git y GitHub	8
4.5 Google Maps Platform.....	8
5. Fases del proyecto.....	9
6. Base de datos (Firebase).....	10
6.1 Implementación Firebase en app Android.....	10
6.2 Servicios de Firebase utilizados.....	17
6.2.1 Firebase Authentication.....	17
6.2.2 Firebase Storage	19
6.2.3 Firebase Cloud Firestore	21
6.2.4 Consultas en la base de datos Firebase Firestore.....	24
6.2.5 Modelo de datos.....	25
7. Casos de Uso	31
8. Diseño de la app.....	35
8.1. Carpeta res	35
8.2 Material Design	39
8.2.1 ¿Qué es Material Design?	39
8.2.2 Implementación de Material Design en nuestra aplicación.....	39
8.2.3 Componentes de Material Design utilizados en BBsitter	40
8.3 Animaciones Lottie	42
8.3.1 ¿Qué es Lottie?	42
8.3.2 ¿Cómo trabaja Lottie?.....	42
8.3.3 Banco de recursos lottiefiles.com.....	42
8.3.4 Implementación de recursos Lottie	43



8.4 Identidad visual de BBsitter.....	45
8.4.1 Logo.....	45
8.4.2 Tipografía	45
8.4.3 Selección de color	46
8.4.4 Imágenes, iconos.....	46
9. Google Maps Platform.....	47
9.1 Credencial clave API de Google Maps Platform	47
9.2 Obtener clave API para nuestro proyecto	48
9.3 Implementar Google Maps y Google Places en nuestro proyecto.....	49
10. GitHub.....	52
11. Usabilidad de nuestra app	56
12. Mejoras de la aplicación	57
13. Bibliografía.....	58



1. Idea del proyecto

Nuestra aplicación, basada en Android, intenta hacer la vida más fácil a los padres que necesitan un cuidador/a para sus hijos. Hay muchos padres que, debido sobre todo al trabajo, no disponen de todo el tiempo que quisieran para poder cuidar a sus hijos y es una tarea muy complicada poder encontrar a alguien de confianza para que los cuide.

De ahí nace nuestra aplicación, donde los padres pueden ver los anuncios de los cuidadores/as y disponer de toda la transparencia y confianza ya que implementamos valoraciones y chat para que los padres estén seguros de elegir el mejor candidato y los niños estén en las mejores manos.

2. Objetivos

Uno de los objetivos primordiales de nuestra aplicación es la confianza. Por ello implementamos valoraciones para que las familias puedan calificar a los cuidadores y así sea más fácil poder elegir.

Otro objetivo es la comunicación segura y eficaz entre familias y cuidadores/as, por ello implementamos un chat para que la comunicación sea fluida y rápida. (Mejora)

Por último, un objetivo muy importante es hacer la vida más fácil a nuestros clientes. Encontrar a un cuidador/a de confianza puede ser muy difícil y por ello creamos esta plataforma para que padres y cuidadores/as puedan comunicarse de manera sencilla y transparente.

El usuario se registra en nuestra aplicación mediante un correo electrónico el cual verificamos desde Firebase. Una vez verificado el usuario tiene que crearse un perfil de familia o de canguro llenando todos sus datos. Cuando entramos en la aplicación en el caso de ser una familia el usuario podrá ver todos los canguros que hay cerca de ella, los más valorados y los más baratos aparte de poder consultar en el mapa los canguros que tiene por la zona en la que esté en ese momento ya que accedemos a su ubicación.

En el usuario familia también puede crear un anuncio si lo desea, para que los canguros puedan contactar con ella mediante el menú desplegable introduciendo todos los datos para que el canguro sepa que necesidades tiene la familia.

El objetivo es que tanto las familias como los canguros tengan una experiencia de uso plena y gratificante, que puedan comunicarse entre familias y canguros con la mayor transparencia posible.



3. Arquitectura de la app

La arquitectura de aplicación describe los patrones y las técnicas que se utilizan para diseñar y desarrollar una aplicación. La arquitectura le proporciona un plan y las prácticas recomendadas que debe seguir al momento de diseñar una aplicación, de modo que obtenga una aplicación bien estructurada.

En una arquitectura de aplicaciones habrá servicios de frontend y de backend. El desarrollo de frontend se refiere a la experiencia del usuario con la aplicación, mientras que el desarrollo de backend implica proporcionar acceso a los datos, los servicios y otros sistemas actuales que permiten el funcionamiento de la aplicación.

La arquitectura de BBsitter comienza con los **lenguajes** empleado que vamos a utilizar que será java, como hemos empleado durante todo el curso, XML (utilizado en el frontend de nuestra app) y Json en caso de escribir en la base de datos de Firebase.

Estos lenguajes los utilizaremos usando un **IDE** para la creación de la App, el cuál será Android Studio. Ya que estamos familiarizados con este IDE desde el comienzo de nuestro 2º curso de Desarrollo de Aplicaciones Multiplataforma, debíamos utilizarlo para desarrollar nuestro proyecto.

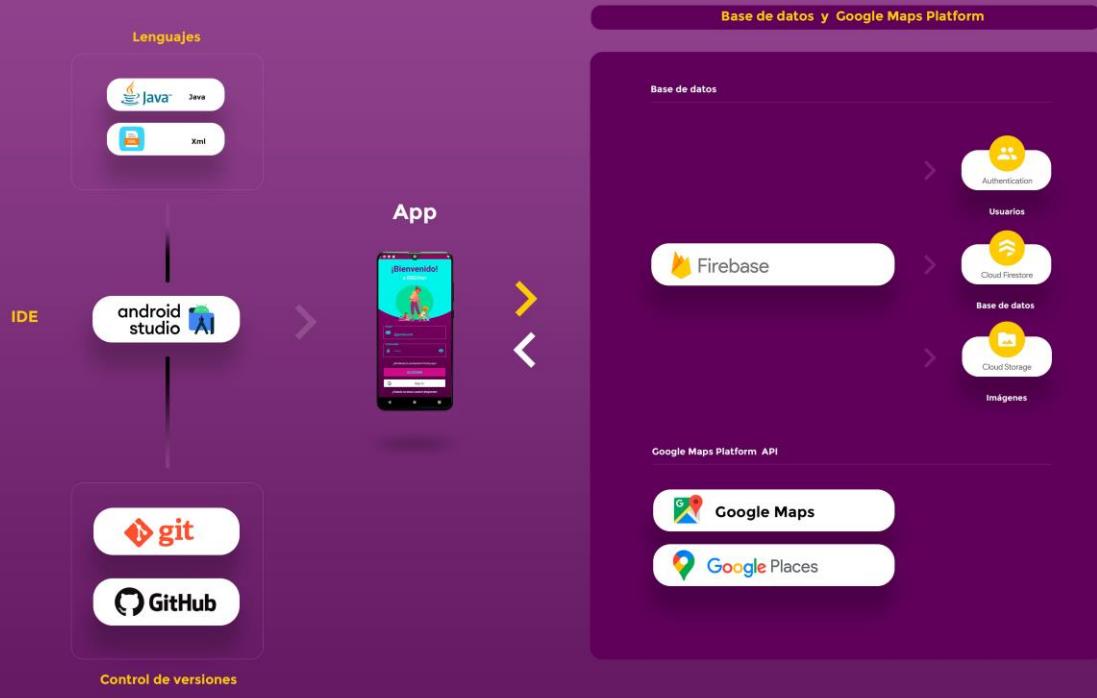
Para poder ir guardando todo el proyecto y no perder nada de él, hemos elegido la manera más extendida y profesional que existe en estos momentos para el **control de versiones** y ese es Git.

Git es un sistema de control de versiones. Un sistema de control de versiones nos va a servir para trabajar en equipo de una manera mucho más simple y óptima cuando estamos desarrollando software. Cuando acabamos de desarrollar nuestro código, utilizamos Git para mezclar los cambios con los otros compañeros.

Unido a Git, usaremos **GitHub**. Un repositorio en la nube donde guardar nuestros proyectos y una a la vez una red social del mundo de la programación.

Una arquitectura de aplicación no podría existir si no existiese un Backend, donde poder guardar los datos y acceder a ellos. En nuestro caso usaremos Firebase, desarrollado por Google. El cual nos facilitará el posible quebradero de cabeza de configurar un servidor, ya que esto lo hará **Firebase** por nosotros. Firebase tiene herramientas que nos facilitará el registro de usuarios (Firebase Authentication), el grabar y acceder a datos con Cloud Firestore y el almacenamiento de imágenes con Firebase Storage.

Dentro de una arquitectura de aplicación también podemos tener otros servicios de terceros que nos ayudarán a implementar algunas funcionalidades, como en nuestro caso puede ser Google Maps Platform y Google Places para poder implementar un mapa donde colocar los marcadores de los canguros.





4. Tecnologías empleadas

4.1 IDE y lenguaje de programación

Android Studio

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android. Fue anunciado el 16 de mayo de 2013 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.

Desde el 7 de mayo de 2019, Kotlin es el lenguaje preferido de Google para el desarrollo de aplicaciones de Android. Aun así, Android Studio admite otros lenguajes de programación, como Java y C++.

Java

Java es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán, probablemente, a menos que tengan Java instalado y cada día se crean más.

Java es rápido, seguro y fiable. Desde ordenadores portátiles hasta centros de datos, desde consolas para juegos hasta computadoras avanzadas, desde teléfonos móviles hasta Internet, Java está en todas partes, si es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos diez millones de usuarios reportados.

JSON

JSON (acrónimo de JavaScript Object Notation) es un formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de JavaScript, aunque, debido a su amplia adopción como alternativa a XML, se considera un formato independiente del lenguaje.



4.2 Base de datos

Firebase

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

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

Firebase Firestore

Cloud Firestore es la base de datos más reciente de Firebase para el desarrollo de apps para dispositivos móviles. Es una solución eficiente y de baja latencia destinada a las apps para dispositivos móviles que necesitan estados sincronizados entre los clientes en tiempo real.

Firebase Authentication

Firebase Authentication nos permite implementar fácilmente la autenticación en cualquier proyecto sea web o app. Firebase ofrece la posibilidad de autenticar al usuario a través de diferentes proveedores más allá de email o password.

Firebase Storage

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

4.3 Diseño y prototipo

Adobe XD

Adobe XD es un editor de gráficos vectoriales desarrollado y publicado por Adobe Inc para diseñar y crear un prototipo de la experiencia del usuario para páginas web y aplicaciones móviles.

Adobe Photoshop

Adobe Photoshop es un editor de fotografías desarrollado por Adobe Systems Incorporated. Usado principalmente para el retoque de fotografías y gráficos.

Lo hemos utilizado para la creación de logos, imágenes de Login, de bienvenida y el menú desplegable



Material Desing

Material Design es una normativa de diseño enfocado en la visualización del sistema operativo Android, además en la web y en cualquier plataforma. Fue desarrollado por Google.

Lottie

Lottie es una biblioteca para Android, iOS, Web y Windows que analiza animaciones de Adobe After Effects exportadas como json con Bodymovin y las representa de forma nativa en el móvil y en la web

Más tarde lo explicamos con más detalle.

AirDroid

AirDroid es una aplicación que hace de necesario puente entre el PC y tu móvil Android para que puedas gestionarlo desde la comodidad del PC: ver tus imágenes, leer tus mensajes o incluso manejar tu teléfono de forma remota.

La utilizamos para poder controlar el móvil sin necesidad de cables ni máquinas virtuales

4.4 Control de versiones Git y GitHub

Git

Git es un sistema de control de versiones libre y gratuito. Un sistema de control de versiones nos va a servir para trabajar en equipo de una manera mucho más simple y optima cuando estamos desarrollando software. Cuando acabamos de desarrollar nuestro código, utilizamos Git para mezclar los cambios con los otros compañeros.

GitHub

Unido a Git, usaremos **GitHub**. Un repositorio en la nube donde guardar nuestros proyectos y a la vez una red social del mundo de la programación. Donde poder colaborar con otros proyectos y compartir los tuyos para que alguien contribuya en su realización.

4.5 Google Maps Platform

Google Maps Platform

Google Maps Platform es una plataforma de servicio gratuito de **Google** de mapas a través de la Web. Ofrece imágenes de mapas desplazables, así como fotos de satélite del mundo entero y de ciudades, e incluso la ruta entre diferentes ubicaciones con especificación del detalle del recorrido.

Gracias a las APIs que ofrecen podemos implementar más funcionalidades a nuestra aplicación, como en nuestro caso Google Places.



5. Fases del proyecto

- **Planificación de requisitos:** Antes de empezar con el desarrollo de la aplicación debíamos tener claro que requisitos mínimos tendría que tener nuestra App, y entre ellos fundamentalmente sería la base de datos e implementar un mapa. Más adelante nos irían saliendo más, pero estos teníamos que tenerlos claros para comenzar a desarrollar.
- **Diseño de la arquitectura de software:** Una vez teníamos claros que requisitos mínimos tendría que tener nuestra aplicación, debíamos investigar cómo llevarlos a cabo y qué herramientas existían para poder desarrollarlos. Para la base de datos elegimos la plataforma que ofrece Firebase sobre todo por la facilidad de integrarlo en nuestro proyecto y en nuestro IDE Android Studio, tanto como el no tener que configurar un servidor por nosotros mismos.
- **Diseño de la base de datos Firebase:** Una vez elegido Firebase como base de datos de nuestro proyecto, era la hora de buscar información y entender el manejo de Firebase.
- **Diseño de la aplicación Android:** Para el diseño de la App nos hemos apoyado mucho en la guía de diseño y componentes que nos ofrece Material Design.
- **Desarrollo de la aplicación Android:** Una vez tenido claro todo lo anterior, era hora de ponerse a desarrollar la App, luchando con todas las dificultades de que se nos pusieron por delante, sobre todo la espinita que se nos queda de no haber conseguido implementar un chat por la falta de tiempo que teníamos.
- **Lanzamiento y testeo de la aplicación:** Finalmente y después de tantas horas utilizadas para el desarrollo tocaba el testearla y ver qué fallos tenía nuestra app y cómo poder eliminarlos



6. Base de datos (Firebase)

Uno de los grandes retos que hemos tenido que superar ha sido la base de datos, ya que elegimos una base de datos noSQL, llamada Firebase.

Esta plataforma fue creada por Google en 2014 para el desarrollo de aplicaciones móviles y aplicaciones web. Es una plataforma ubicada en la nube, integrada con Google Cloud Platform, que usa un conjunto de herramientas para la creación y sincronización de proyectos que serán dotados de alta calidad, haciendo posible el crecimiento del número de usuarios y dando resultado también a la obtención de una mayor monetización.

Firebase dota a sus usuarios de una gran documentación para crear aplicaciones usando esta plataforma. Aparte de esto, ofrece soporte gratuito mediante correo electrónico para todos sus usuarios, y además sus desarrolladores participan en plataformas como GitHub, como más tarde explicaremos.

6.1 Implementación Firebase en app Android

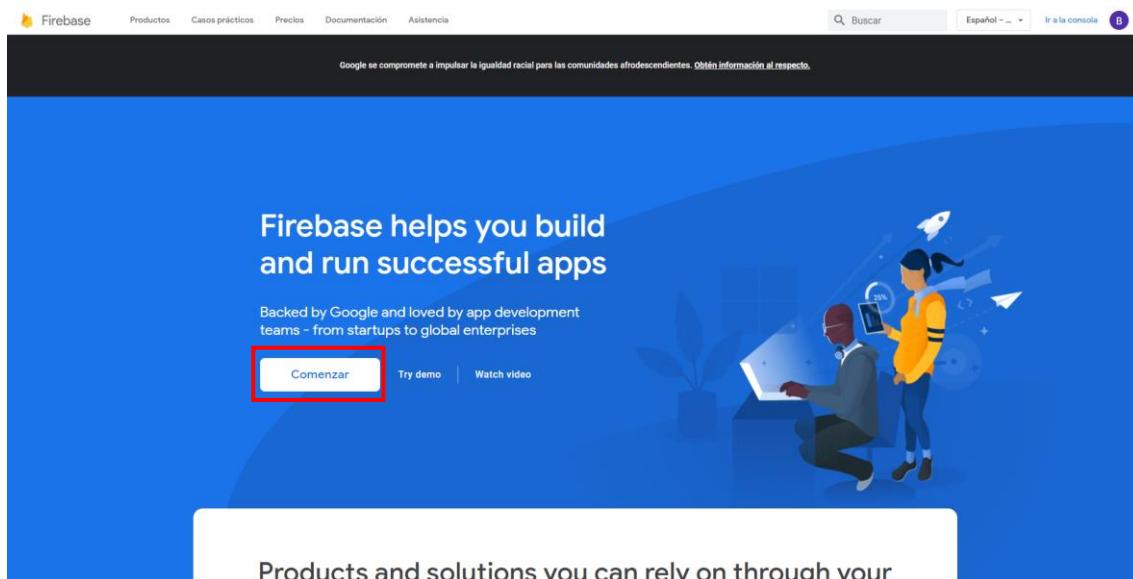
Para implementar Firebase en la aplicación hay que seguir unos pasos que los explicamos a continuación. Los requisitos previos son los siguientes:

- Instalación de Android Studio o actualizar a su versión más reciente.
- Se orienta al nivel de API 16 (Jelly Bean) o una versión posterior.
- Usa Gradle 4.1 o una versión posterior.
- Implementación de com.android.tools.build:gradle 3.2.1 o una versión posterior
- compileSdkVersion 28 o una versión posterior
- Configura un dispositivo físico o utiliza un emulador para ejecutar tu app. Los emuladores deben usar una imagen que cuente con Google Play.
- Accede a Firebase con tu Cuenta de Google.

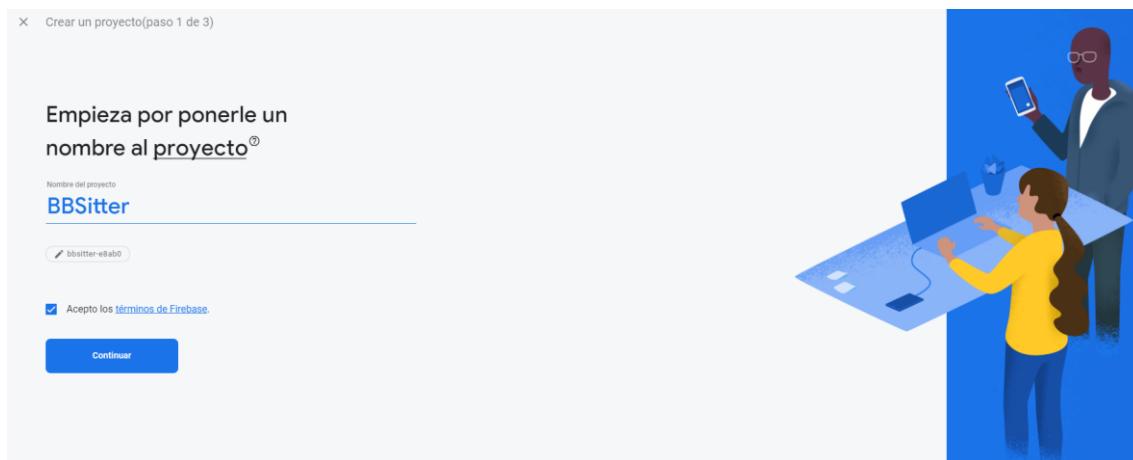


Una vez completados los requisitos previos debemos conectar la aplicación con nuestro proyecto de Firebase. Para ello lo primeros que debemos hacer es crear un proyecto de Firebase.

Nos iremos a [Firebase \(google.com\)](https://firebase.google.com), para crear un nuevo proyecto tenemos que presionar el botón Comenzar.



Debemos ponerle un nombre al proyecto y aceptar los términos de Firebase





Activamos Google Analytics para Firebase ya que proporciona informes ilimitados y gratuitos de hasta 500 eventos diferentes. El SDK captura de manera automática ciertos eventos clave y propiedades del usuario; además, puedes definir tus propios eventos personalizados a fin de medir factores particularmente importantes.

Crear un proyecto(paso 2 de 3)

Google Analytics para tu proyecto de Firebase

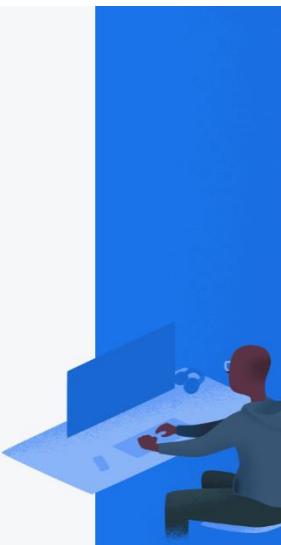
Google Analytics es una solución de analíticas gratuita e ilimitada que permite crear informes y realizar tareas de segmentación, entre otras acciones, en Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, Predictions y Cloud Functions.

Google Analytics permite usar:

- A/B Testing
- Segmentación de usuarios en productos de Firebase
- Predicción del comportamiento de usuarios
- Usuarios sin fallos
- Activadores de Cloud Functions basados en eventos
- Informes gratuitos ilimitados

Habilitar Google Analytics en este proyecto
Opción recomendada

[Anterior](#) [Continuar](#)





Y configuramos Google Analytics para nuestro país.

Crear un proyecto(paso 3 de 3)

Configurar Google Analytics

Ubicación de Analytics ⓘ

España

Términos de Google Analytics y configuración de uso compartido de datos

- Usar la configuración predeterminada para compartir datos de Google Analytics. [Learn more](#)
 - ✓ Comparte tus datos de Analytics con nosotros para ayudarnos a mejorar nuestros productos y servicios.
 - ✓ Comparte tus datos de Analytics con nosotros para habilitar las comparativas.
 - ✓ Comparte tus datos de Analytics con Google para habilitar el servicio de asistencia técnica.
 - ✓ Comparte tus datos de Analytics con los especialistas en cuentas de Google.
- Acepto los términos de protección de datos entre responsables del tratamiento de datos y responsables del tratamiento de datos de mediciones y reconozco que estoy sujeto a la [política de consentimiento de usuarios finales de la Unión Europea](#). Es obligatorio marcar esta casilla si vas a compartir tus datos de Analytics para mejorar los productos y servicios de Google. [Más información](#)
- Acepto los términos de Google Analytics.

Al crear el proyecto, también se creará una propiedad de Google Analytics que se asociará a tu proyecto de Firebase. De esta forma, se habilitará el flujo de datos entre los productos. Los datos que se exportan de tu propiedad de Google Analytics a Firebase están sujetos a los términos del servicio de Firebase, mientras que los datos que se importan de Firebase a Google Analytics se rigen por los términos del servicio de Google Analytics. [Más información](#)

Anterior

Crear proyecto

Una vez creado el proyecto nos llevará a la pantalla principal de Firebase donde podremos encontrar todos los servicios de la plataforma y donde podremos enlazar el proyecto con nuestra app en Android Studio haciendo clic en el ícono de Android situado en el centro de la pantalla.

Información general

Compilación

- Authentication
- Cloud Firestore
- Realtime Database
- Storage
- Hosting
- Functions
- Machine Learning

Lanzamiento y monitorización

Analytics

- Dashboard
- Events
- Conversions
- Audiences
- Funnels
- User Properties
- Latest Release
- Retention
- Extensivos

Spark Gratis 0 USD/mes

Actualizar

BBSitter Plan Spark

Para empezar, añade Firebase a tu aplicación

iOS Android < / >

Almacena y sincroniza datos de aplicaciones en cuestión de milisegundos

Authentication

Cloud Firestore

Para ello nos pedirán dos datos obligatorios: el nombre del paquete Android y un apodo.



Para saber el nombre del paquete de Android tenemos que acceder al build.gradle de nuestra app y cogeremos el dato “applicationId” para utilizarlo en Firebase.

```
defaultConfig {  
    applicationId "com.bbsitter.bbsitter"  
    minSdkVersion 16  
    targetSdkVersion 29  
    versionCode 1  
    versionName "1.0"  
    multiDexEnabled true  
  
    testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
}
```

Y ponemos los datos que hemos recogido en Firebase, hacemos clic en Registrar aplicación

The screenshot shows the first step of the Firebase setup wizard for an Android app. It has a title bar with a close button and three colored dots. The main area is titled "Añadir Firebase a una aplicación de Android". Step 1, "Registrar la aplicación", is selected. It contains fields for "Nombre del paquete de Android" (set to "com.bbsitter.bbsitter") and "Apodo de la aplicación (opcional)" (set to "BBSitter"). Below these is a field for "Certificado de firma de depuración SHA-1 (opcional)", which contains a placeholder string of zeros. A note below the SHA-1 field states: "Obligatoria para Dynamic Links, Firebase Invites y la asistencia por teléfono o el inicio de sesión de Google en Auth. Edita los SHA-1 en la configuración." At the bottom is a blue "Registrar aplicación" button. To the right, a vertical sidebar lists steps 2 through 4: "Descargar el archivo de configuración", "Añadir el SDK de Firebase", and "Pasos siguientes".



Nos pedirá que descarguemos el archivo google-services.json y que lo implementemos en la carpeta app de nuestro proyecto

The screenshot shows two windows side-by-side. On the left, the 'Add Firebase to your application' wizard is displayed, specifically step 2: 'Descargar el archivo de configuración'. It shows a download button for 'google-services.json' and instructions to move the downloaded file to the 'app' directory of the Android project. On the right, the Android Studio file structure for a project named 'BBSitter' is shown. The 'app' directory is expanded, revealing its contents: build, src, .gitignore, build.gradle, google-services.json, and proguard-rules.pro. A blue arrow points from the 'google-services.json' file in the wizard to the 'google-services.json' file in the project's file tree.

El siguiente paso es verificar que nuestra aplicación tiene implementados los servicios de Google. Para ello vamos otra vez a build.gradle a nivel de proyecto de nuestra aplicación y tiene que tener estas líneas

```

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

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

```



Y en nuestro build.gradle a nivel de aplicación debemos poner estas líneas

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

dependencies {
    // Import the Firebase BoM
    implementation platform('com.google.firebase:firebase-bom:26.1.1')

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

Por último tenemos que sincronizar en build.gradle a nivel de proyecto para que ejecute los cambios

Gradle files have changed sir [Sync now](#)



6.2 Servicios de Firebase utilizados

Firebase tiene muchos servicios disponibles, pero en nuestro caso hemos utilizado 3 servicios: Firebase Authentication, Firebase Storage y Firebase Cloud Firestore.

6.2.1 Firebase Authentication

La mayoría de las apps necesitan identificar a los usuarios. Conocer la identidad de un usuario permite que una app guarde sus datos en la nube de forma segura y proporcione la misma experiencia personalizada en todos los dispositivos del usuario.

Firebase Authentication proporciona servicios de backend, SDK fáciles de usar y bibliotecas de IU ya elaboradas (en GitHub) para autenticar a los usuarios en tu app. Admite la autenticación mediante contraseñas, números de teléfono, proveedores de identidad federada populares, como Google, Facebook y Twitter, y mucho más, pero en nuestro caso implementamos únicamente registro mediante email y contraseña y Google.

Para empezar, tenemos que habilitar la opción de autentificación por correo en Firebase, para ello vamos a ir al servicio Authentication y habilitamos las opciones que vayamos a utilizar para registrarnos.

The screenshot shows the Firebase console's Authentication section. On the left, there's a sidebar with various services: Compilación (Authentication, Cloud Firestore, Realtime Database, Storage, Hosting, Functions, Machine Learning), Lanzamiento y monitorización (Crashlytics, Performance, Test Lab), and Analytics (Dashboard, Events, Conversions, Audiences, Funnels, User Properties). At the bottom, it says "Spark Gastos 0 USD/mes" and has an "Actualizar" button. The main area is titled "Authentication" and has tabs for "Users", "Sign-in method", "Templates", and "Usage". Under "Sign-in method", it says "Proveedores de inicio de sesión" and lists the following providers with their status:

Proveedor	Estado
Correo electrónico/contraseña	Habilitado
Teléfono	Inhabilitado
Google	Habilitado
Play Juegos	Inhabilitado
Game Center	Inhabilitado
Facebook	Inhabilitado
Twitter	Inhabilitado
Github	Inhabilitado
Yahoo	Inhabilitado
Microsoft	Inhabilitado
Apple	Inhabilitado
Anónimo	Inhabilitado



Una vez habilitados tendremos que implementar en nuestro build.gradle a nivel de proyecto la dependencia de Firebase Auth.

```
implementation 'com.google.firebaseio:firebase-auth'
```

Existen múltiples métodos para usar el servicio Authentication, los más usados sirven para crear usuarios, registrar usuarios o para cerrar sesión.

Crear usuario

```
FirebaseAuth mAuth;  
  
mAuth = FirebaseAuth.getInstance();  
  
mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(this, new  
OnCompleteListener<AuthResult>() {  
    @Override  
    public void onComplete(@NonNull Task<AuthResult> task) {  
  
        //si es verdad que la tarea ha sido satisfactoria...  
        if(task.isSuccessful())  
        {  
  
        }  
        else{  
        }  
    }  
});
```

Registrar usuario

```
FirebaseAuth mAuth;  
  
mAuth = FirebaseAuth.getInstance();  
  
mAuth.signInWithEmailAndPassword(email, password)  
.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {  
    @Override  
    public void onComplete(@NonNull Task<AuthResult> task) {  
        if (task.isSuccessful()) {  
        } else {  
        }  
    }  
});
```



Cerrar sesión

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

6.2.2 Firebase Storage

Cloud Storage para Firebase es un servicio de almacenamiento de objetos potente, simple y rentable construido para la escala de Google. Los SDK de Firebase para Cloud Storage agregan la seguridad de Google a las operaciones de carga y descarga de archivos para tus apps de Firebase, sin importar la calidad de la red. Puedes usar nuestros SDK para almacenar imágenes, audio, video y otros tipos de contenido generado por el usuario. En el servidor, puedes usar Google Cloud Storage para acceder a los mismos archivos.

En este servicio guardamos imágenes de logos de BBsitter y las imágenes de nuestros usuarios para que sea mucho más intuitiva y eficaz la recogida de datos al estar todo dentro de Firebase.

Para comenzar a cargar imágenes en Firebase Storage tenemos de implementar su dependencia.

```
● ● ●  
implementation 'com.google.firebaseio:firebase-storage'
```



En nuestra aplicación la utilizamos con el método `putFile()` para que suba la imagen a nuestro Storage. Lo hacemos de esta forma:

```
//Declaramos la clase FirebaseStorage
FirebaseStorage storageRef;
storageRef = FirebaseStorage.getInstance();

//Creamos una carpeta dentro de Storage y añadimos una foto con el nombre de la ID del usuario
storageRef.child("img_familias").child(uid);

//Subimos la imagen a FirebaseStorage
storageRef.putFile(uri).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>()
{
    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot)
    {
        storageRef.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>()
        {
            public void onSuccess(Uri uri) {}

        }).addOnFailureListener(new OnFailureListener()
        {
            @Override
            public void onFailure(@NonNull Exception exception){}
        });
    }
});
```

Al guarda la imagen de esta forma se crea una referencia y una URL de esa imagen

Storage

Files Rules Usage

Nombre	Tamaño	Tipo	Última modificación
0acBKPvY4VRF0g7TbB2JSL0dJjd2	468.81 KB	image/jpeg	2 dic 2020
2iSSjeMHEfArRHnJILX6C1zL022	141.42 KB	image/jpeg	10 dic 2020
31wNUuWLsOe2Z/W0Vc7D7JS9n6Y2	495.21 KB	image/jpeg	10 dic 2020
4tfuq8Zx1UgY2qfn6gTihIREMu1	5.37 MB	image/jpeg	5 dic 2020
5GAb2bqkqXfnKxtlf6R39rnX2z2	532.53 KB	image/jpeg	3 dic 2020
6xREIkfFRxMrdl4vDbQJ0eJ6Dem1	3.82 MB	image/jpeg	11 dic 2020
901NZ3NcqveUlhiy1AhT8BGY5i2	502.19 KB	image/jpeg	2 dic 2020
9tDmzVhb65ZYShF5Jf0Hi11KdiO2	532.53 KB	image/jpeg	3 dic 2020
BUBCNe7i6hdpqz1KYWsJCZ5nP02	481.26 KB	image/jpeg	10 dic 2020
DKLKuM2mPvSjehlaG4P3yk5sEsJ2	418.65 KB	image/jpeg	2 dic 2020
FQHMRXTM6JeSu7XlkidQAtqyu323	467.15 KB	image/jpeg	10 dic 2020

Nombre: [OacBKPvY4VRF0g7TbB2JSL0dJjd2](#)

Tamaño: 480.059 bytes

Tipo: image/jpeg

Fecha y hora de creación: 2 dic 2020 20:47:34

Actualizada: 2 dic 2020 20:47:34

Ubicación del archivo:

Ubicación de almacenamiento: gs://bbssitter-61bd3.appspot.com/img_familias/OacBKP...

TOKEN DE ACCESO: [6d4d9abd-0db6-4682-b49f-9a51f18d0fb8](#)

[Crear nuevo token de acceso](#)

Con Firebase Cloud Firestore accederemos a esa URL para poder utilizarla dentro de la aplicación.



6.2.3 Firebase Cloud Firestore

Cloud Firestore es una base de datos flexible y escalable para la programación en servidores, dispositivos móviles y la Web desde Firebase y Google Cloud Platform.

Al igual que Firebase Realtime Database (versión anterior de base de datos de Firestore), mantiene tus datos sincronizados entre apps cliente a través de agentes de escucha en tiempo real y ofrece asistencia sin conexión para dispositivos móviles y la Web, por lo que puedes compilar apps con capacidad de respuesta que funcionan sin importar la latencia de la red ni la conectividad a Internet.

Las funciones clave más importantes de Firebase Cloud Firestore son las siguientes:

- **Flexibilidad**

El modelo de datos de Cloud Firestore admite estructuras de datos flexibles y jerárquicas. Almacena tus datos en documentos, organizados en colecciones. Los documentos pueden contener objetos anidados complejos, además de subcolecciones.

- **Consultas expresivas**

En Cloud Firestore, puedes usar consultas para recuperar documentos individuales específicos o para recuperar todos los documentos de una colección que coinciden con los parámetros de la consulta.

Las consultas pueden incluir varios filtros en cadena y combinar los filtros con criterios de orden. También se indexan de forma predeterminada, por lo que el rendimiento de las consultas es proporcional al tamaño de tu conjunto de resultados, no del conjunto de datos.

- **Actualizaciones en tiempo real**

Al igual que Realtime Database, Cloud Firestore usa la sincronización de datos para actualizar los datos de cualquier dispositivo conectado.

Sin embargo, también está diseñado para ejecutar consultas de recuperación únicas y sencillas de manera más eficiente que en Realtime Database.

- **Diseñado para ajustarse a la escala de la aplicación**

Cloud Firestore te ofrece lo mejor de la poderosa infraestructura de Google Cloud Platform: replicación automática de datos multirregión, garantías de coherencia sólida, operaciones atómicas por lotes y asistencia real sobre transacciones.

Cloud Firestore sirve para controlar las cargas de trabajo de las bases de datos más complejas de las apps más grandes del mundo. Que no es nuestro caso porque nuestra aplicación es pequeña, pero podría ser más grande en un futuro.



Para poder empezar a utilizar esta potente base de datos NoSQL debemos implementar su dependencia en el build.gradle de nuestro proyecto

```
implementation 'com.google.firebaseio:firebase-firebase:21.2.1'
```

Una vez implementada la dependencia podemos escribir datos en Firestore desde nuestra aplicación creando un Mapa de datos y utilizando el método .add() de esta manera:

```
Map<String, Object> user = new HashMap<>();
user.put("nombre", "Antonio");
user.put("perfil", "false");
user.put("tipo", "familia");
user.put("uid", "EWRWRdfgdsSDFsa32");

db.collection("usuarios")
    .add(user)
    .addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
        @Override
        public void onSuccess(DocumentReference documentReference) {
        }
    })
    .addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
        }
    });
});
```



También podemos leer datos de Firestore desde nuestra aplicación. Para ello utilizaremos el método `.get()`

```
db.collection("usuarios")
    .get()
    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : task.getResult()) {
                    document.getData();
                }
            } else {
            }
        }
    });
}
```



6.2.4 Consultas en la base de datos Firebase Firestore

Cloud Firestore proporciona una potente función de consulta para especificar qué documentos deseas recuperar de una colección o de un grupo de colecciones. Estas consultas también se pueden usar con get() o addSnapshotListener().

Para empezar, tenemos que llamar a la clase CollectionReference.

```
CollectionReference users = db.collection("usuarios");
```

Esta consulta es lo más simple que podemos hacer, accede a la colección de la base de datos llamada "usuarios" y nos saca todos los documentos que tenga dentro.

Podemos hacer consultas más complejas como estas:

```
//Consulta para sacar los anuncios más nuevos
Query anunciosNovedades = db.collection("anuncios").whereEqualTo("fechaPublicacion", Query.Direction.ASCENDING);

//Consulta Canguros más baratos
Query canguroMasBaratos = db.collection("canguro").whereEqualTo("precioHora", Query.Direction.ASCENDING);

//Consulta el para sacar os datos de mi propio usuario (para cargar mis datos en Mi perfil, por ejemplo)
FirebaseAuth mAuth;
mAuth = FirebaseAuth.getInstance();

Query user = db.collection("usuario").whereEqualTo("uid", mAuth.getCurrentUser().getUid());
```

También tenemos la opción de crear consultas compuestas, como si de un AND se tratara.

```
//Consulta para sacar los canguros con el precio de hora de 7€ y que su rating sea mayor que 3
Query filtroCanguro = db.collection("canguro");
filtroCanguro.whereEqualTo("precioHora", "7").whereGreaterThanOrEqualTo("rating", "3");
```



6.2.5 Modelo de datos

Colección Usuarios

```
{  
  "usuarios": {  
  
    "GFAasdafouejEDWfew": {  
      "email": "david@gmail.com",  
      "perfil": true,  
      "tipo": "familia",  
      "uid": "GFAasdafouejEDWfew"  
    },  
  
    "AWE45w3WESFDA3443wqFWSED": {  
  
      "email": "jesus@gmail.com",  
      "perfil": true,  
      "tipo": "canguro",  
      "uid": "AWE45w3WESFDA3443wqFWSED"  
    },  
  
    "DASd34e43wqaedSAds": {  
  
      "email": "prueba@gmail.com",  
      "perfil": true,  
      "tipo": "familia",  
      "uid": "DASd34e43wqaedSAds"  
    }  
  }  
}
```



Colección Familias

```

{
  "familias": {
    "GFAAsdafoiuejEDWfew": {
      "descripcion": "Hola, somos una familia muy amable",
      "direccion": "Calle Chile, Torrejón de Ardoz, Madrid, España",
      "email": "david@gmail.com",
      "img": "https://firebasestorage.googleapis.com/99549-b586-4306-8e58-c0b35455561e",
      "latitud": 40.434,
      "longitud": -3.324,
      "nombre": "Arribas",
      "uid": "GFAAsdafoiuejEDWfew"
    },
    "AWE45w3WESFDA3443wqFWSED": {
      "descripcion": "Hola, somos una familia genial",
      "direccion": "Calle Vitoria, Alcalá de Henares, Madrid, España",
      "email": "jesus@gmail.com",
      "img": "https://firebasestorage.googleapis.com/99549-b586-4306-8e58-c0b35455561e",
      "latitud": 40.342,
      "longitud": -3.543,
      "nombre": "González",
      "uid": "AWE45w3WESFDA3443wqFWSED"
    }
  }
}

```

Colección hijos

```

{
  "hijos": {
    "GFAAsdafoiuejEDWfew": {
      "nombre": "David",
      "idHijo": "GFAAsdafoiuejEDWfew",
      "edad": 3,
      "otrosDatos": "Tiene alergia a los cacahuetes",
      "uid": "EDSwrfqwedfQAQew"
    },
    "AWE45w3WESFDA3443wqFWSED": {
      "nombre": "Jesús",
      "idHijo": "AWE45w3WESFDA3443wqFWSED",
      "edad": 1,
      "otrosDatos": "Le encantan las matemáticas",
      "uid": "DSADWIHBDweqadf2343EWQe"
    }
  }
}

```



Colección Anuncios

```
{\n    "anuncios": {\n\n        "ADSdoiuawshjdnasijd231": {\n            "nombre": "Familia Arribas",\n            "titulo": "Necesito canguro este fin de semana",\n            "descripcion": "Hola, necesito un canguro este fin de semana de 17 a 19. Gracias",\n            "direccion": "Calle Chile, Torrejón de Ardoz, Madrid, España",\n            "email": "david@gmail.com",\n            "casa": "Casa de la familia",\n            "fechaPublicacion": "12 dic. 2020",\n            "idAnuncio": "ADSdoiuawshjdnasijd231",\n            "idiomas": {\n                "Aleman": true,\n                "Español": true\n            },\n            "img": "https://firebasestorage.googleapis.com/99549-b586-4306-8e58-c0b35455561e",\n            "pluses": {\n                "No fumador": true,\n                "Primeros auxilios": true,\n                "Puedo cocinar": true\n            },\n            "preferenciaEdades": {\n                "De 1 a 3 años": true,\n                "De 6 a 12 meses": true,\n                "Más de 6 años": true\n            },\n            "tiempo": "Fin de semana",\n            "uid": "GFAsdafoiuejEDWfew"\n        },\n\n        "AWE45w3WESFDa3443wqFWSED": {\n            "nombre": "Familia González",\n            "titulo": "Necesito canguro días de diario",\n            "descripcion": "Hola, necesito un canguro para los lunes y miércoles. Gracias",\n            "direccion": "Calle Vitoria, Alcalá de Henares, Madrid, España",\n            "email": "jesus@gmail.com",\n            "casa": "Casa de la familia",\n            "fechaPublicacion": "09 dic. 2020",\n            "idAnuncio": "AWE45w3WESFDa3443wqFWSED",\n            "idiomas": {\n                "Ingles": true,\n                "Español": true\n            },\n            "img": "https://firebasestorage.googleapis.com/99549-b586-4306-8e58-c0b35455561e",\n            "pluses": {\n                "No fumador": true,\n                "Primeros auxilios": true,\n                "Puedo cocinar": true\n            },\n            "preferenciaEdades": {\n                "De 1 a 3 años": true,\n                "De 6 a 12 meses": true,\n                "Más de 6 años": true\n            },\n            "tiempo": "Fin de semana",\n            "uid": "DASDasdsadSADEwqad"\n        }\n    }\n}
```



Colección Canguro

```

{
  "canguros": {
    "GFAAsdafoiuejEDWfew": {
      "nombre": "David",
      "apellidos": "Arribas",
      "descripcion": "Hola, soy un gran canguro",
      "direccion": "Calle Chile, Torrejón de Ardoz, Madrid, España",
      "email": "david@gmail.com",
      "edad": 36,
      "fechaCreacionPerfil": "12 dic. 2020",
      "fechaNacimiento": "11/06/1984",
      "idiomas": {
        "Alemán": true,
        "Español": true
      },
      "img": "https://firebasestorage.googleapis.com/99549-b586-4306-8e58-c0b35455561e",
      "latitud": 40.434,
      "longitud": -3.324,
      "pluses": {
        "No soy fumador": true,
        "Sé primeros auxilios": true,
        "Puedo cocinar": true
      },
      "precioHora": 7.5,
      "preferenciaEdades": {
        "De 1 a 3 años": true,
        "De 6 a 12 meses": true,
        "Más de 6 años": true
      },
      "rating": 2,
      "sexo": "Masculino",
      "telefono": "309874934",
      "uid": "GFAAsdafoiuejEDWfew"
    },
    "AWE45w3WESFDA3443wqFWSED": {
      "nombre": "Jesús",
      "apellidos": "González",
      "descripcion": "Hola, soy un gran canguro",
      "direccion": "Calle Victoria, Alcalá de Henares, Madrid, España",
      "email": "jesus@gmail.com",
      "edad": 26,
      "fechaCreacionPerfil": "09 dic. 2020",
      "fechaNacimiento": "25/01/1994",
      "idiomas": {
        "Inglés": true,
        "Español": true
      },
      "img": "https://firebasestorage.googleapis.com/99549-b586-4306-8e58-c0b35455561e",
      "latitud": 40.434,
      "longitud": -3.324,
      "pluses": {
        "No soy fumador": true,
        "Sé primeros auxilios": true,
        "Puedo cocinar": true
      },
      "precioHora": 7.5,
      "preferenciaEdades": {
        "De 1 a 3 años": true,
        "De 6 a 12 meses": true,
        "Más de 6 años": true
      },
      "rating": 2,
      "sexo": "Masculino",
      "telefono": "309874934",
      "uid": "AWE45w3WESFDA3443wqFWSED"
    }
  }
}

```



Como podemos observar el modelo de datos es no relacional. Cada colección es una clase en nuestro proyecto.

Clase Usuario

```
public class Usuario {  
  
    private boolean perfil;  
    private String email;  
    private String pass;  
    private StringUid;  
  
}
```

Clase Familia

```
public class Familia implements Serializable {  
  
    private String nombre;  
    private String descripcion;  
    private String direccion;  
    private double longitud;  
    private double latitud;  
    private String uid;  
    private String email;  
  
}
```

Clase Anuncio

```
public class Anuncio implements Serializable {  
  
    private String titulo;  
    private String descripcion;  
    private String fechaPublicacion;  
    private String tiempo;  
    private String casa;  
    private String img;  
    private String nombre;  
    private String direccion;  
    private String uid;  
    private String idAnuncio;  
  
}
```



Clase Canguro

```
public class Canguro implements Serializable {  
  
    //Atributos  
    private String img;  
  
    // Datos personales  
    private String nombre;  
    private String apellidos;  
    private String fechaNacimiento;  
    private int edad;  
    private String sexo;  
    private String email;  
    private String telefono;  
  
    // Direccion y fecha nacimiento  
    private String direccion; // Obtenida de autocompletar Direccion  
    private double longitud;  
    private double latitud;  
  
    // Demás atributos  
    private double precioHora; //Obtenida de slider precio/Hora  
    private String experiencia;  
    private String descripcion;  
  
    private int rating;  
  
    Map<String, Boolean> mapPluses; // Pluses  
    Map<String, Boolean> mapPrefenciaEdades; // PreferenciaEdades  
    Map<String, Boolean> mapIdiomas; // PreferenciaEdades  
  
    private String uid;  
    // Fecha creacion  
    private String fechaCreacionPerfil;  
}
```



7. Casos de Uso

Tenemos en cuenta unos requisitos claves para la utilización de nuestra aplicación. Queremos que el usuario pueda tener una experiencia lo más rápida y eficiente posible, para ello el usuario debería poder hacer estos pasos:

- **Usuario**

- Verificación de email
- Creación de perfil
- Cambiar contraseña

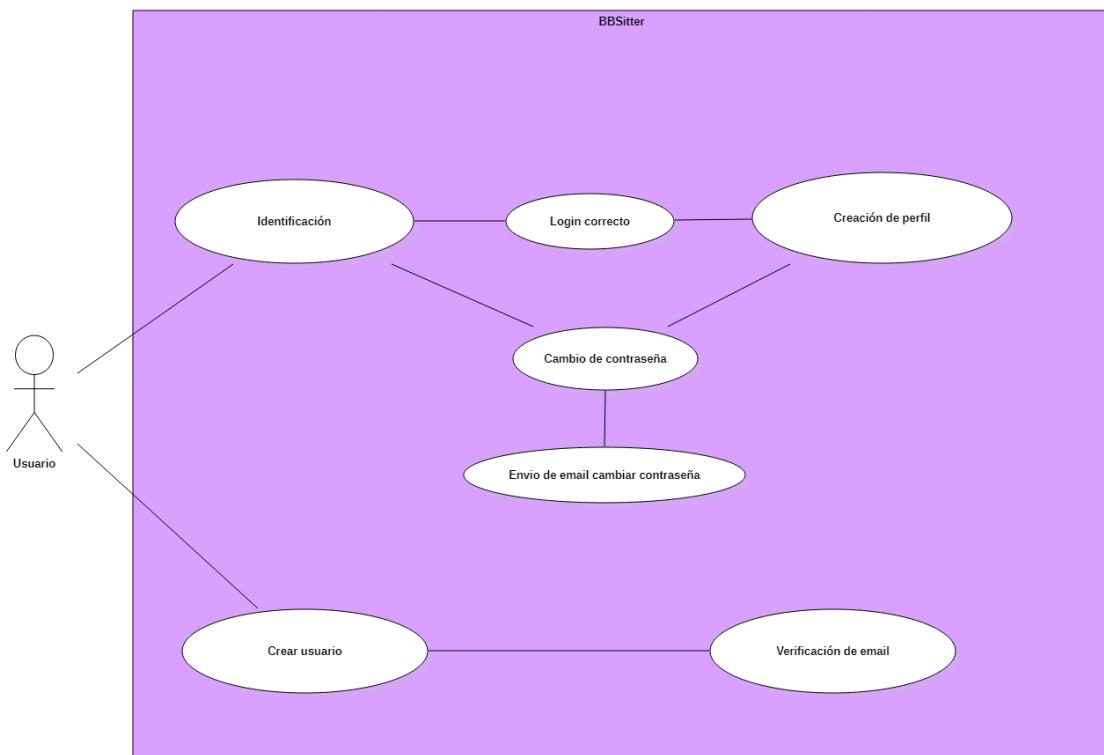
- **Familia**

- Crear un anuncio
- Visualizar los canguros disponibles filtrados
- Visualizar mapa con canguros
- Llamar y mandar un email a los canguros
- Añadir Hijos
- Contactar con Soporte
- Eliminar Anuncios
- Eliminar Hijos
- Eliminar su perfil
- Editar su perfil
- Cerrar sesión

- **Canguro**

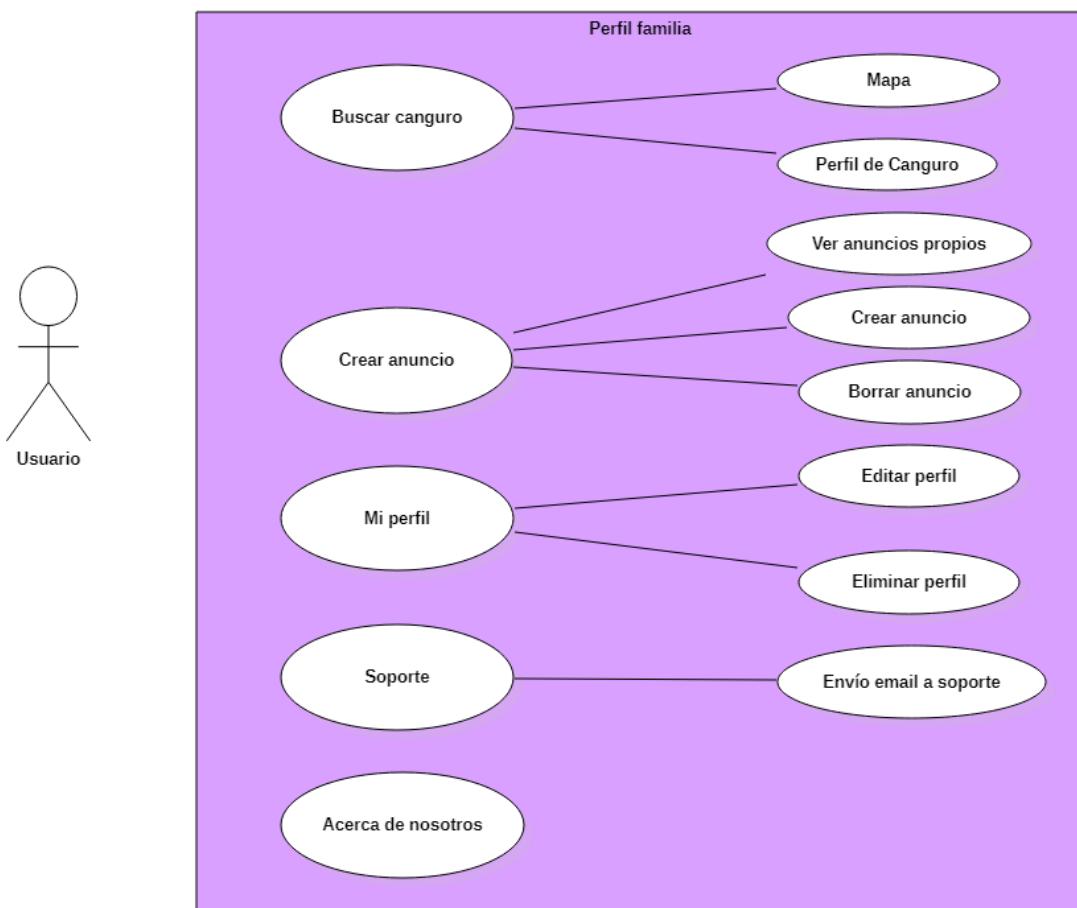
- Creación de perfil con sus datos
- Visualizar anuncios de las familias
- Editar su perfil
- Eliminar su perfil
- Contactar con Soporte
- Cerrar sesión

Una vez establecido los requisitos que queremos que tenga nuestra aplicación procedemos a hacer los casos de uso.



En esta pantalla podemos hacer tres cosas: Identificación, Registro y cambio de contraseña.

Cuando te identificas puede ser que te equivoques de contraseña, por eso tenemos un control de validación de campos, tanto para errores, para existencia de usuarios en Firebase como si el email esta verificado o no (Esto lo explicamos en el apartado de usabilidad).



Una vez creado el perfil de familia tenemos diversas opciones dentro de la aplicación.

Podemos ver a todos los canguros tanto en una lista como en un mapa con sus marcadores y dentro de ellos podemos ver su perfil y contactar con ellos tanto por email como por teléfono.

En mi perfil podemos editar nuestro perfil, añadir hijos y eliminar nuestro perfil

En Acerca de nosotros podrá ver nuestros datos y nuestro GitHub



En el perfil de canguro podemos ver los anuncios de las familias en forma de lista. Cuando nos metemos en el detalle del anuncio podemos contactar con la familia por email y podemos ver su perfil para ver sus datos y sus hijos

En Mi perfil podemos modificar nuestro perfil por si tenemos más pluses o más idiomas y podemos eliminar nuestro perfil

En soporte podemos contactar con el soporte de nuestra aplicación por si tuviera algún problema

En Acerca de nosotros podrá ver nuestros datos y nuestro GitHub



8. Diseño de la app

En Android para diseñar las interfaces de usuario de nuestras aplicaciones tenemos, por un lado, las propias actividades para crear cada una de las diferentes pantallas, y por otro lado las vistas para crear el contenido visual de cada actividad. Para este contenido visual utilizamos archivos **.xml** que serán los encargados de dar el aspecto visual de las pantallas y están conectados con los archivos **.java** que serán los encargados de hacer el trabajo lógico de la aplicación.

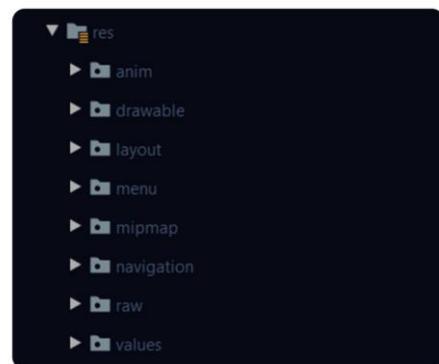
Casi todos los elementos gráficos de nuestro proyecto irán en la carpeta **res** del proyecto, desde los layout mencionados anteriormente hasta las imágenes, videos, animaciones, menús, colores... A continuación, os mostramos una imagen de la carpeta **res** de nuestro proyecto y más adelante detallaremos un poco cada carpeta que contiene y para qué sirve.

8.1. Carpeta res

Como hemos mencionado anteriormente es la carpeta que contiene los recursos usados por la aplicación. Las carpetas que contiene en su interior serán las que contengan los diferentes recursos en la mayoría referidos al tema gráfico.

Drawable

En esta carpeta se almacenan los ficheros de imágenes (JPG o PNG) y descriptores de imágenes en XML.



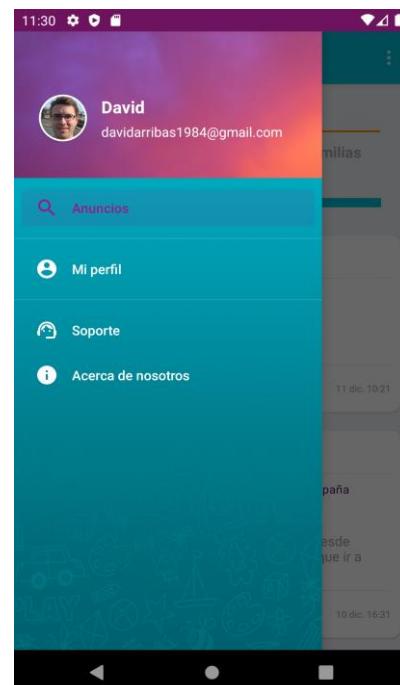
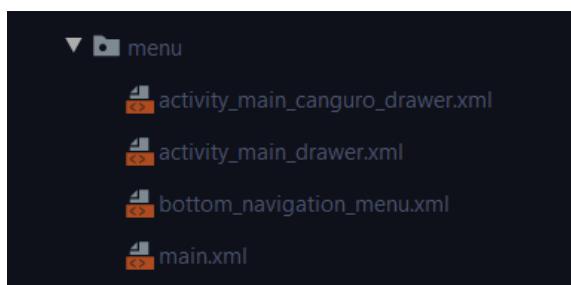
Mipmap

En una carpeta guardaremos el icono de la aplicación y también aquellas imágenes en las que queramos que la densidad gráfica sea la correcta dependiendo el dispositivo en el que sea utilizada la aplicación. En nuestro caso no hemos utilizado esta carpeta dejando nuestro logo guardado en la carpeta Drawable.

Menu

Ficheros XML con los menús de cada actividad. En esta carpeta tendremos los menús utilizados del Navigation Drawer de nuestra aplicación, tanto el de perfil Familia como el de perfil Canguro y que será el encargado de vincular el ítem del menú pulsado con el fragment correspondiente a dicho ítem.

Además, tenemos otro menú a la derecha de la toolbar que nos servirá para que el usuario puede cerrar sesión.





Values

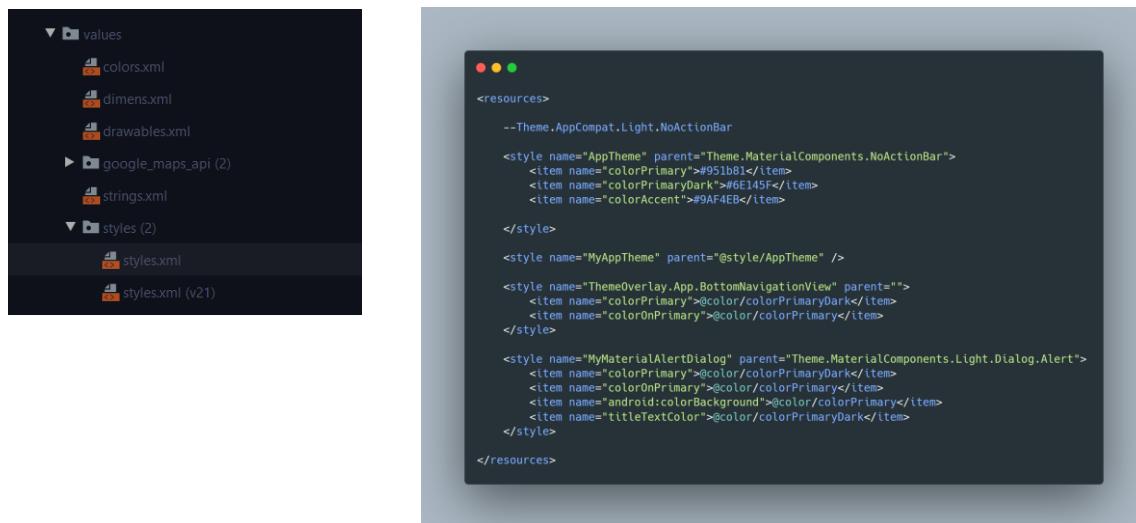
También utilizaremos ficheros XML para indicar valores usados en la aplicación, de esta manera podremos cambiarlos desde estos ficheros sin necesidad de ir al código fuente.

En **colors.xml** hemos definido los tres colores primarios de la aplicación, además de otros colores que hemos necesitado para ello. Esto facilita el tener que poner o saber el color que quieras utilizar en RGB.

En **dimens.xml**, se pueden definir dimensiones como el margen por defecto o el ancho de los botones.

En el fichero **strings.xml**, aquí se todas las cadenas de caracteres de la aplicación. Creando recursos alternativos resultará muy sencillo traducir una aplicación a otro idioma. Nosotros en esta carpeta no hemos usado strings para traducir simplemente Android Studio ha creado strings automáticos que ha ido necesitando cuando hemos introducido algún elemento a la app.

Finalmente, en **styles.xml**, podrás definir los estilos y temas de tu aplicación. En nuestro caso hemos tenido que definir el theme necesario para poder usar elementos de Material Design, además de tener que modificar los colores en el alert Dialog de Material, ya que no mostraba colores oscuros que no queríamos tener.



Anim

Contiene ficheros XML con animaciones de vistas (Tween). En nuestro caso desarrollamos un par de animaciones, pero finalmente optamos por poner animaciones que tiene Android

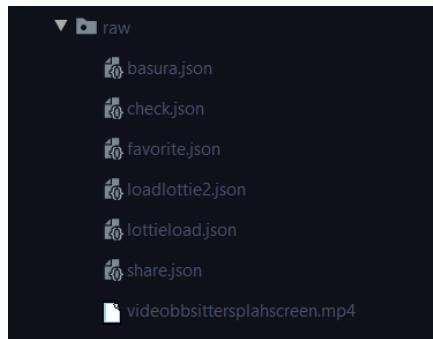




propiamente y que sólo con mencionarlas nos valdría y no tendríamos que desarrollarlas. Estas animaciones las usamos cuando una familia pincha en un ítem de la lista canguros y le aparece el detalle del perfil de canguro.

Raw

En esta carpeta metemos aquellos ficheros adicionales que no se encuentran en formato XML. Como son animaciones Lottie y videos, en nuestro caso.



Navigation

Esta carpeta la crea Android al crear una activity Navigation Drawer y es la encargada de alojar los archivos .xml, los cuales serán los encargados de alojar los diferentes fragmentos de los que se constituye un Navigation Drawer relacionado con el menu del propio Navigation Drawer. El código siguiente es el encargado del Navigation de nuestro perfil canguro:

```

<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mobile_navigation_canguro"
    app:startDestination="@+id/inicioCanguroFragment">

    <fragment
        android:id="@+id/inicioCanguroFragment"
        android:name="com.bbssitter.bbssitter.OpcionesMenuCanguro.Inicio.InicioCanguroFragment"
        android:label="Anuncios"
        tools:layout="@layout/inicio_canguro_fragment" />

    <fragment
        android:id="@+id/miPerfilCanguroFragment"
        android:name="com.bbssitter.bbssitter.OpcionesMenuCanguro.Perfil.MiPerfilCanguroFragment"
        android:label="Mi perfil"
        tools:layout="@layout/perfil_canguro_fragment" />

    <fragment
        android:id="@+id/soporteFragment"
        android:name="com.bbssitter.bbssitter.SoporteFragment"
        android:label="Soporte"
        tools:layout="@layout/soporte_fragment" />

    <fragment
        android:id="@+id/acercaNosotrosFragment"
        android:name="com.bbssitter.bbssitter.AcercaNosotrosFragment"
        android:label="Acerca de nosotros"
        tools:layout="@layout/acerca_nosotros_fragment" />

</navigation>

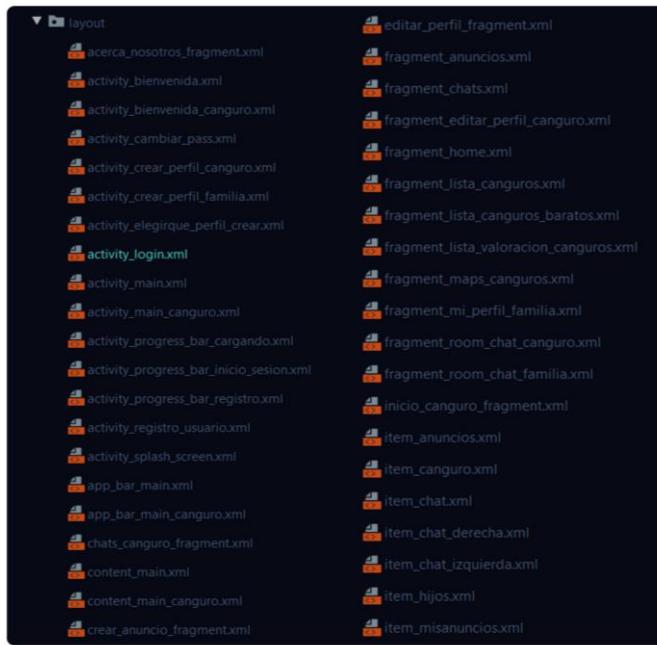
```



Layout

Por último, no nos podíamos olvidar de esta carpeta, ya que podría ser la carpeta más importante en cuanto a diseño, y la encargada de colocar los elementos visuales de nuestra aplicación.

Contiene ficheros XML con vistas de la aplicación. Las vistas nos permitirán configurar las diferentes pantallas que compondrán la interfaz de usuario de la aplicación. Se utiliza un formato similar al dom HTML usado para diseñar páginas web. En la imagen siguiente os mostramos algunos de los layout utilizados en BBsitter.



Un Layout es el contenedor principal que define el orden y secuencia en que se organizarán los widgets en nuestra actividad. Existen varios tipos de Layout (LinearLayout, RelativeLayout, ConstraintLayout, FrameLayout...)

Básicamente en nuestra aplicación utilizamos dos tipos de layout como son los LinearLayout (maquetación en línea de las vistas ya sean verticalmente u horizontalmente), ConstraintLayout y los RelativeLayout (similares a los LinearLayout pero nos permite más flexibilidad a la hora de colocar los objetos con la posibilidad de simular diferentes capas en las que están situadas las vistas).

Dentro de estos Layouts colocaremos todos aquellos componentes que queremos que aparezcan en nuestras activities o fragments. A estos componentes se les llama vistas (views) y existen muchísimos tipos de vistas como Button, TextView, EditText, ListView, ImageView... Todos estos componentes se pueden personalizar dándole el aspecto visual que queramos como tamaños, colores, etc y para ello el conocimiento en aspectos gráficos, visuales y maquetación son esenciales para darle un aspecto profesional a cualquier diseño de aplicación. Ante esto no todos los desarrolladores tienen tales conocimientos y puede que la aplicación pueda quedar un poco sobria o con aspecto desfasado a las tendencias de diseño que existen en el momento de desarrollo. Para esto apareció Material Design.



8.2 Material Design



8.2.1 ¿Qué es Material Design?

Históricamente Android ha tenido diseño **Holo**, a partir de ahora el nuevo estilo y la tendencia en Android es Material Design. **Material Design es un concepto, una filosofía, unas pautas enfocadas al diseño utilizado en Android**, pero también en la web y en cualquier plataforma.

Material Design es un **sistema de diseño creado por Google** con el fin de ser capaz de adaptarse a múltiples dispositivos y plataformas.

La idea de Google es la de incorporar este sistema de **coherencia estética y funcional** de forma progresiva a todos sus productos, incluyendo las aplicaciones web y móviles, con la finalidad de crear una experiencia similar en todas sus plataformas.

Los componentes de material son bloques de construcción interactivos para crear una interfaz de usuario e incluyen un sistema de estados integrado para comunicar los estados de enfoque, selección, activación, error, desplazamiento, presión, arrastre y deshabilitación. Todo esto no sólo favorece el tener un aspecto visual tremadamente profesional, sino que también facilita el controlar la usabilidad para que el usuario sepa que está ocurriendo en todo momento.

Material Design tiene bibliotecas de componentes que están disponibles tanto para Android, iOS, Flutter como para web. No todos los componentes de Material Design están disponibles para cada una de estas plataformas, nosotros obviamente hemos usado la documentación y componentes disponibles para Android.

8.2.2 Implementación de Material Design en nuestra aplicación.

Para usar Material Design en Android tendremos que añadir la dependencia de Material en el archivo **build.gradle** de nuestro proyecto:

```
// Implementación de material Material Desing  
implementation 'com.google.android.material:material:1.2.1.'
```



Para poder usar los componentes de Material tuvimos que cambiar el theme de nuestra app a un theme de Material:



8.2.3 Componentes de Material Design utilizados en BBSitter

Os vamos a mostrar algunos de los componentes de Material que hemos usado de BBSitter, obviamente existe una variedad muy grande de componentes que no hemos utilizado, pero los usados favorecen mucho el diseño de la app y también la usabilidad del usuario.

CardView

```
<com.google.android.material.card.MaterialCardView
    android:id="@+id/cardViewCanguro"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="4dp"
    app:cardBackgroundColor="@color/colorPrimaryDark"
    android:checkable="true"
    app:cardCornerRadius="10dp"
    android:elevation="2dp">
```



ChipGroup

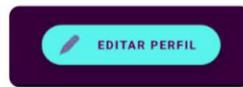
```
<com.google.android.material.chip.ChipGroup
    android:id="@+id/chingroupPlusesMiPerfilCanguro"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingHorizontal="5dp"
    style="@style/Widget.MaterialComponents.Chip.Entry"
    app:singleLine="false"
    app:singleSelection="false"
    app:chipSpacingHorizontal="15dp">
```





FloatingActionButton

```
<!-- BUTTON EDITAR PERFIL-->
<com.google.android.material.floatingactionbutton.ExtendedFloatingActionButton
    android:id="@+id/btnEditarPerfilCanguro"
    android:layout_width="wrap_content"
    android:layout_height="48dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="30dp"
    android:layout_marginRight="16dp"
    android:layout_marginBottom="20dp"
    android:contentDescription="chat"
    android:gravity="bottom|center|end"
    android:text="EDITAR PERFIL"
    android:textSize="12dp"
    android:textColor="@color/colorPrimaryDark"
    android:textStyle="bold"
    app:icon="@android:drawable/ic_menu_edit"
    app:iconTint="@color/colorPrimaryDark"
    app:backgroundTint="@color/colorPrimary"
/>
```



Material InputLayout / EditText

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/email_edit_text"
    android:layout_width="300dp"
    android:layout_height="80dp"
    android:layout_marginTop="10dp"
    android:hint="Email"
    android:background="@android:color/transparent"
    android:textColorHint="@color/colorPrimary"
    app:placeholderTextColor="@color/colorPrimary"
    app:errorIconTint="@android:color/holo_red_dark"
    app:boxStrokeErrorColor="@android:color/holo_red_dark"
    app:errorEnabled="true"
    app:errorIconDrawable="@android:drawable/stat_notify_error"
    app:startIconDrawable="@android:drawable/sym_action_email"
    app:startIconTint="@color/colorPrimary"
    app:boxBackgroundMode="outline"
    app:boxBackgroundColor="@android:color/transparent">

    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress"
        android:textColor="@color/colorPrimary"
        android:textColorHint="@android:color/white"
        android:maxLines="1"
        android:text="" />
</com.google.android.material.textfield.TextInputLayout>
```





8.3 Animaciones Lottie

No es ninguna sorpresa si os decimos que las micro interacciones son tendencia. Desde hace años las pequeñas animaciones copan las aplicaciones y las webs con el fin de mejorar su interacción, su usabilidad y, sobre todo, alegrarnos la vida haciendo mucho más agradable la experiencia de usuario.

Desde BBsitter valoramos mucho el introducir estos componentes ya que hacen nuestra aplicación mucho más llamativa e interesante para el usuario y gracias a su sencilla implementación nos permite tener componentes en nuestra app, muy interesantes.

8.3.1 ¿Qué es Lottie?

Lottie es una herramienta muy potente creada por **AirBnb** con la que implementar animaciones y microinteracciones para nuestras webs o apps.

Lottie es una librería capaz de reproducir animaciones en tiempo real y de forma nativa en web, Android e iOS. De esta forma, Lottie facilita el desarrollo e implementación de animaciones para diferentes plataformas, integrándose en Android e iOS permitiendo cargar animaciones como si fueran cualquier otro contenido estático. Además, Lottie las hace escalables y fáciles de adaptar a cualquier dispositivo y capaces de reaccionar a los eventos provocados por el usuario.

Lottie dispone incluso de su propia app en la que podrás comprobar cómo se reproducen tus animaciones sin necesidad de crear un proyecto desde cero.

8.3.2 ¿Cómo trabaja Lottie?

Lottie trabaja con animaciones de After Effects exportadas gracias al plugin Bodymovin, capaz de convertirlas a formato JSON. De esta forma, una animación creada en el potentísimo programa de Adobe puede presentarse de forma nativa en cualquier dispositivo Android e iOS.

Si eres capaz de desarrollar una animación solo hay que instalar el plugin Bodymovin en nuestro After Effects, crear la animación y exportarla en el formato adecuado según sea para web, Android o iOS. After Effects nos exportará un JSON con el que trabajaremos en el entorno en el que vayamos a implementar la animación.

8.3.3 Banco de recursos lottiefiles.com

Si no te ves capaz de crear una animación como hemos mencionado antes, lo más fácil es acceder a la página de recursos lottie **lottiefiles.com**. Donde podemos descargar numerosas animaciones de manera gratuita y por supuesto también de pago.



8.3.4 Implementación de recursos Lottie

En caso de que quieras incluir animaciones con Lottie en un proyecto Android, estos son los pasos que tendrás que tener en cuenta:

1. Agregar al build.gradle la **librería Lottie**:

```
// Implementacion de Lottie
implementation 'com.airbnb.android:lottie:3.4.4'
```

2. Introducir el archivo JSON de la animación en la **carpeta raw** del proyecto. Hay que tener en cuenta que tendrás que crear la carpeta raw si no la has creado, ya que no viene creada por defecto.

3. Introducimos el **componente en nuestro layout** de la siguiente forma:

```
// Animación lottie cuando se crea el perfil
<com.airbnb.lottie.LottieAnimationView
    android:id="@+id/lottieLoad"
    android:layout_width="300dp"
    android:layout_height="200dp"
    app:lottie_autoPlay="true"
    app:lottie_rawRes="@raw/check" />
```



Cómo veis, trabajamos con el componente **com.airbnb.lottie.LottieAnimationView** al que le añadimos:

1. En **lottie_fileName** el nombre del archivo JSON para Lottie.
2. En **lottie_loop** si habilitamos (true) o no (false) la animación en bucle, si no ponemos nada por defecto será false.
3. En **lottie_autoPlay** si habilitamos (true) o no (false) la reproducción automática de la animación.

En la documentación de Lottie podréis encontrar más atributos para controlar tu animación.

4. Por último, configuramos nuestro código en el activity o fragment dentro del onCreateView para poder **mostrar la animación** utilizando los diferentes métodos para ello:

```
// Inicializar la animación
private LottieAnimationView lottieFav;

// Instanciar la animación
lottieFav = view.findViewById(R.id.lottieFavorito);

// Al hacer click en la animación
lottieFav.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        lottieFav.playAnimation();

        /* En caso de haber desarrollado la lista de favoritos
           aquí desarrollaríamos esta parte */
    }
});
```

8.4 Identidad visual de BBsitter

Toda marca necesita una identidad y BBsitter no iba a ser menos. El conjunto de elementos gráficos que ayudan a la diferenciación, en general se compone de logotipo, tipografía, cromática... en definitiva, los aspectos visuales o expresión gráfica de una marca. La identidad visual es el conjunto de elementos que tangibilizan la propuesta de valor y la personalidad de una marca.

8.4.1 Logo

Sin duda, uno de los elementos más reconocibles en las grandes marcas es su logo, y adquiere vida propia conforme se posiciona en el público.



8.4.2 Tipografía

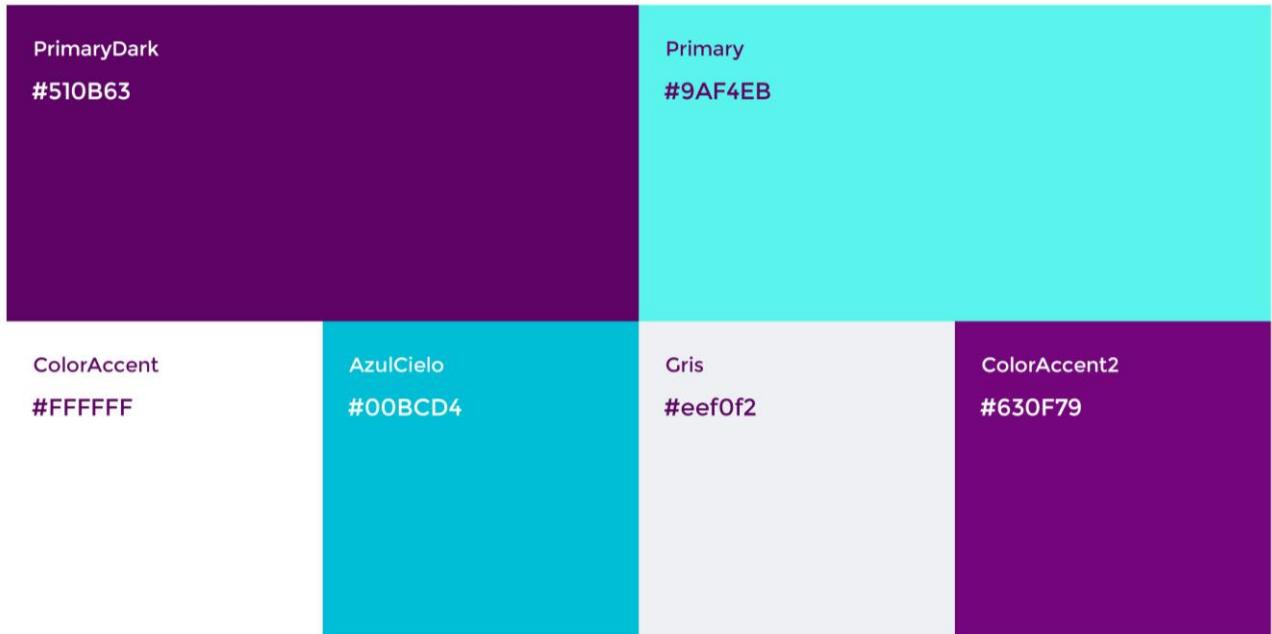
Elegir una fuente distintiva (o un grupo de fuentes) que realmente marquen la diferencia y que transmitan la personalidad de tu marca, es esencial para diferenciarse del resto. En nuestro caso optamos por la tipografía que nos usa Android por defecto, Robocop.





8.4.3 Selección de color

Los tonos que utilices en tu sitio web, en tu publicidad y hasta en tus empaques deben estar orientados a tu audiencia y a las emociones que quieras despertar en ella.



8.4.4 Imágenes, iconos....

Decidir cuál es el enfoque que prefieres en las imágenes que aparezcan en tu app o que estén relacionadas con tu marca hará que el usuario la reconozca y se sienta reflejado en ella. Para BBsitter no hemos utilizado grandes librerías de iconos, ni imágenes... simplemente hemos convertido a vector (ic_nombre.xml) los iconos que nos ofrece Android, y alguna descarga de icons icons, retocados con Photoshop.





9. Google Maps Platform



Desde un principio la idea principal de BBsitter era poder ver en un mapa todos aquellos canguros que estaban cercanos a una familia que estaba en búsqueda de canguro y que, de una manera rápida, pudiese ver a los más cercanos de un vistazo. Para después poder pinchar en aquel que eligiese y poder ver la información más esencial de dicho canguro, incluso pudiendo pinchar en él para ver sus detalles.

Para poder implementar esta funcionalidad, obviamente teníamos que echar mano de Google Maps y la mejor manera de utilizarlo para poder desarrollar algo así, era con las APIs que ofrece Google Maps Platform.

9.1 Credencial clave API de Google Maps Platform

Google actualizó **Google Maps** para implementarlo en sitios web de cara a mejorar su rendimiento en dispositivos móviles. Permitiendo una carga mucho más rápida y adaptando sus funcionalidades según las características de los smartphones actuales.

Para emplear esta nueva API se introdujeron cambios en el servicio para desarrolladores, ya que estas novedades traen consigo **limitaciones en su versión gratuita**. Una de las principales es la introducción de distintos tipos de clave necesarios en tu App para que la aplicación funcione correctamente.

Para poder disfrutar de las APIs que ofrece Google Maps Platform, tendremos que tener una clave API (es un tipo de autenticación para el uso de aplicaciones con **Google Maps JavaScript API**) que debes solicitar, vincular e insertar en tu proyecto de App.



9.2 Obtener clave API para nuestro proyecto

Desde una cuenta de Google, accederemos a la consola de Google Maps Platform y en el apartado de Credenciales, pincharemos en Nueva Clave API.

Tendremos o nos pedirá que nos creamos una cuenta gratuita y meter datos de una tarjeta de crédito para poder hacer futuros cargos en caso de que el proyecto siga adelante en un futuro, sino la cuenta gratuita durará 3 meses (a nosotros a día de hoy no nos han cobrado y queda todavía un mes de vigencia de la cuenta gratuita, ya veremos en un futuro, aunque con dar de baja al servicio ya valdría).

Crearemos una clave API que irá asociada a nuestro proyecto, la cual tendremos que incluir en nuestro proyecto de Android una vez creamos un Map Fragment.

Una vez tengamos nuestra clave generada tendremos que activar las APIs que necesitemos de aquellas que nos ofrece Google Platform. En nuestro caso nosotros tenemos dadas de alta las siguientes, pudiendo ver las métricas de uso de dichas APIs:

API	Solicitudes	Errores	Avg latency (ms)	Detalles
Geocoding API	0	0	-	Detalles
Geolocation API	0	0	-	Detalles
Maps SDK for Android	301	0	-	Detalles
Places API	3.870	1	40	Detalles



9.3 Implementar Google Maps y Google Places en nuestro proyecto

Una vez obtenida la clave y activadas las APIs necesarias, iremos a nuestro proyecto a implementar Google Maps y Google Places en nuestro **build.gradle**.

```
// Google Maps y Google Places

implementation 'com.google.android.gms:play-services-maps:17.0.0'
implementation 'com.google.android.gms:play-services-location:17.1.0'
implementation 'com.google.android.libraries.places:places:2.4.0'
```

Deberemos también poner esto en nuestro **manifest** para poder hacer uso de la geolocalización de nuestro dispositivo así como decirle cual es nuestra clave Api de Google Maps.

```
<!-- Permisos para utilizar ubicaciones -->

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-feature android:name="android.hardware.location.gps" />

<!-- KEY GOOGLE necesaria para trabajar con las APIs de Google Maps -->

<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="@string/google_maps_key" />
```



Para mostrar el mapa de Google Maps donde colocaremos los marcadores de los canguros tendremos que crear un Google Maps Fragment, una vez hecho esto nos creará un archivo xml con este nombre **google_maps_api.xml**, en el que nos explica brevemente como obtener la clave Api para poder usar Google Maps.

```

<resources>
    <!--
        TODO: Para correr la aplicacion necesitamos tener una Key paa usar Google Maps

        Para esto necesitamos tener una cuenta en Google Platform, podemos tener una cuenta gratuita
        durante 90 dias.

        Deberemos crear un proyecto y en CREDENCIALES, podremos crear nuestra KEY_GOOGLE.

        Posteriormente en nuestra Cuenta de Google Platform, deberemos habilitar las APIs de Google Maps
        que queremos usar, en nustro caso habilitaremos:

        - GOOGLE PLACES: Para autocomplementar las direcciones al introducir nuestra direccion en un
        campo de texto.
        - GEOCODING: Para convertir una localizacion (longitud y latitud) en un String.
        - ...

        Alternatively, follow the directions here:
        https://developers.google.com/maps/documentation/android/start#get-key

        Once you have your key (it starts with "AIza"), replace the "google_maps_key"
        string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve"
    translatable="false">AIzaSyCAq5pFIif49ezgqjq4x6ZEafMyuGXnCH0</string>
</resources>

```

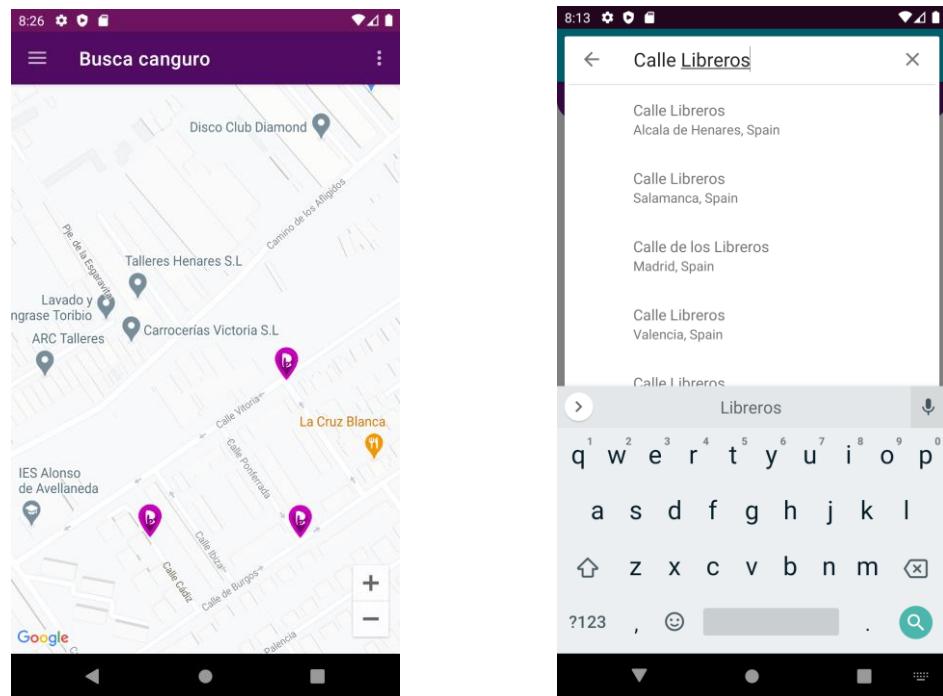
Google Maps Fragment

Android nos ofrece un fragmento Maps de Google en el cual desarrollaremos la lógica para obtener nuestra ubicación y colocar nuestro marcador. También haremos lo mismo recorriendo nuestra base de datos para obtener las ubicaciones de todos los canguros y poder colocar su marcador en el Mapa.

Google Places

Google Places es una API que debemos activar en nuestra cuenta de Google Platform y que nosotros hemos utilizado en nuestro proyecto para poder obtener un autocompletado de direcciones cuando el canguro o la familia escriben su dirección para poder registrarla.

Mientras el usuario va escribiendo su dirección, la actividad de autocompletado va generando direcciones acordes al texto escrito por el usuario, el cual podrá elegir aquella que necesita en cuanto salga en la lista de direcciones.





10. GitHub

GitHub es un sistema de gestión de proyectos y control de versiones de código, así como una plataforma de red social diseñada para desarrolladores que permite trabajar en colaboración con otras personas de todo el mundo, planificar proyectos y realizar un seguimiento del trabajo.

GitHub es también uno de los repositorios online más grandes de trabajo colaborativo en todo el mundo.

¿Qué es un control de versiones? Cuando los desarrolladores hacen un nuevo proyecto, siempre continúan haciéndole modificaciones al código. Incluso después de la puesta en marcha de los proyectos, todavía necesitan actualizar las versiones, corregir errores, agregar nuevas funciones, etc.

El sistema de control de versiones ayuda a registrar los cambios realizados al código. Aún más, registra quién realizó los cambios y puede restaurar el código borrado o modificado.

No hay códigos sobrescritos ya que Git guarda varias copias en el repositorio.

Esta parte del proyecto es la que más nos ha costado entender ya que esto no lo vimos en clase y además es un poco lioso. Finalmente entendimos el proceso de enviar datos a GitHub y de recoger datos después de vernos innumerables videos tutoriales.

Su funcionamiento es el siguiente:

Paso 1: Crear un repositorio en GitHub.

Para ello debemos registrarnos en GitHub.com, buscar la pestaña “Repositories” y hacer clic en “New”

The screenshot shows a GitHub user profile for 'Jesus GM' (jesusgm94). The profile picture is a circular logo with a purple and white geometric pattern. The user has 1 repository, 1 project, and 0 packages. The 'Repositories' tab is selected, showing a list with one item: 'Proyecto-BBSitter'. This repository is described as Java and was updated 21 minutes ago. There is a 'Star' button and a green line graph icon next to it. At the bottom of the page, there are links for Overview, Repositories, Projects, Packages, Edit profile, and a navigation bar with links for Terms, Privacy, Cookie Preferences, Security, Status, Help, Contact GitHub, Pricing, API, Training, Blog, and About.



Rellenamos el nombre de nuestro repositorio, le ponemos una descripción y le damos a “Create repository” y nos saldrá en nuestra pantalla principal un repositorio vacío donde debemos llenar con los datos de nuestra app.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *



jesusgm94 ▾

Repository name *

Proyecto BBSiter



Great repository names are

Your new repository will be created as Proyecto-BBSiter. It [miniature-enigma?](#)

Description (optional)

Una aplicación para la búsqueda de canguros o trabajo como canguro

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you’re importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license

A license tells others what they can and can’t do with your code. [Learn more.](#)

This will set main as the default branch. Change the default name in your [settings](#).

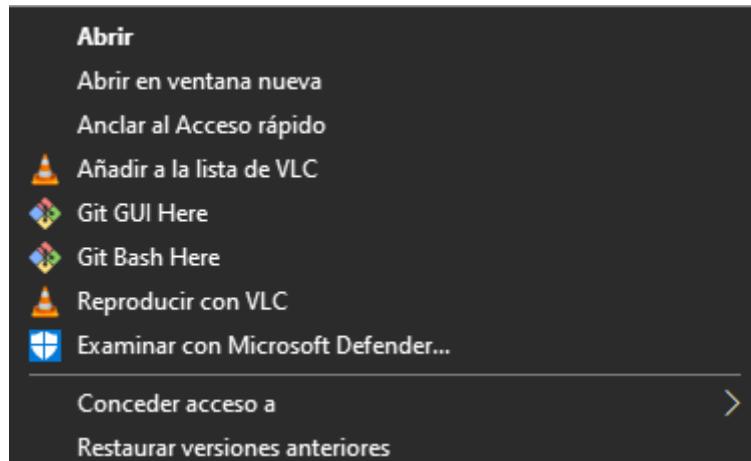
Create repository



Paso 2. Clonar repositorio en nuestro pc

En este paso es necesario descargarse [Git](#) mediante este enlace.

Una vez instalado en nuestro pc nos vamos a cualquier carpeta y presionamos el botón derecho para ver las opciones y seleccionamos “Git Bash Here”



Se nos abrirá la consola de Git y es el momento de clonar nuestro repositorio. Para ello necesitamos la URL de nuestro repositorio que lo encontraremos en nuestro repositorio de GitHub

The screenshot shows a GitHub repository page for 'Proyecto-'. At the top, there are three buttons: 'Go to file', 'Add file ▾', and a green 'Code ▾' button. A dropdown menu is open under 'Code ▾', showing the 'Clone' option. Below 'Clone', there are links for 'HTTPS', 'SSH', and 'GitHub CLI'. The 'HTTPS' link is underlined and shows the URL: <https://github.com/jesusgm94/Proyecto->. There is also a copy icon next to the URL. Below the clone options, there are other actions: 'Open with GitHub Desktop', 'Open with Visual Studio', and 'Download ZIP'. On the left side of the page, there is a sidebar with a list of commits:

- Aplicación completa
- Eliminar anuncios, hijos y perfiles (cons...
- Aplicación completa
- Comandos
- Initial commit
- Añadido video Splash Screen



En la consola de Git debemos poner el siguiente comando

```
git clone https://github.com/jesusgm94/Proyecto-BBSitter.git
```

Esto nos permitirá tener todo lo que haya en GitHub en nuestro pc y viceversa. Con lo cual el siguiente paso es crear nuestro proyecto de Android Studio dentro de esa carpeta para poder subirlo.

Una vez creado el proyecto de Android para poder subir el proyecto a GitHub tendrá que ser el siguiente comando

```
git commit -m "commit ejemplo"  
git push origin master
```

En cambio, si lo que queremos es recoger los datos de GitHub tendremos que hacer lo siguiente

```
git pull origin master
```



11. Usabilidad de nuestra app

La usabilidad de una App es un tema muy importante que hay que tratar con detalle para hacer de la experiencia del usuario algo cómodo, sencillo e intuitivo. Para ello hay numerosas maneras de hacer que un usuario no deje de usar los servicios de nuestra app solo porque no sabe que está pasando en algunos momentos cuando ha hecho clic en algún botón, cargar una imagen, etc.

Ahora os detallaremos algunas acciones que hemos implementado y creemos que son buenas para que el usuario tenga una buena usabilidad:

- **Menos pasos a la hora de registrarse y facilidad de cambiar contraseña**

La mayoría de las personas no quieren teclear demasiado en su celular. Sobre todo si existe la opción de llenar datos usando Facebook, Google + , Twitter u otra forma. También hay momentos que para resetear la contraseña se hace muy pesado el poder hacerlo de una manera sencilla.

- **Una sola activity, pocos campos y frases sencillas para crear perfil**

Entre menos pasos, páginas, botones y campos tenga que ir el usuario, más contento estará.

- **Mostrar Toast, Snackbars o comentarios en EditText para errores y acciones**

Mostraremos mensaje de diferentes tipos para avisar al usuario tanto para cuando se equivoca o hay un error, como cuando realiza una acción que conlleve algún cambio.

- **Mostrar CardView de canguro a la hora de hacer clic en un marcador del mapa**

Mostraremos mensaje de diferentes tipos para avisar al usuario tanto para cuando se equivoca o hay un error, como cuando realiza una acción que conlleve algún cambio.

- **PageView de bienvenida una vez creado el perfil**

Una vez creado el perfil, ya sea canguro o familia, hemos creado un pageView que el usuario podrá deslizar y que nos informará de lo que el usuario puede hacer con la app.

- **Animaciones de carga de datos, cambios o creación**

Cuando nosotros como usuarios pinchamos en algún botón de cualquier aplicación de las que usamos, esperamos saber que está ocurriendo o que ha pasado cuando hicimos clic en él. Para esto hemos creado tanto ProgressBar, como Toast, como animaciones que nos dirán que ha ocurrido.



12. Mejoras de la aplicación

Nuestra aplicación la pensamos con idea de que en un futuro pudiera ampliarse para que fuera más completa. Las mejoras que tenemos en mente son:

- Usuario
 - Inicio de sesión con otros proveedores (Google, Facebook,...)
 - Cambiar correo electrónico
- Perfiles
 - Crear chat
 - Calificaciones a los canguros
 - Comentarios a los canguros
 - Filtros personalizables
 - Filtro por distancia en perfil de familia
 - Calificación a las familias
 - Comentarios a las familias
- Base de datos
 - Mejorar la eficiencia de la base de datos sin repetir datos
 - Seguridad en las contraseñas, teléfonos, emails...
 - Compresión de las imágenes para cargas más rápidas



13. Bibliografía

[Android Studio](#)

[Firebase](#)

[Adobe Photoshop](#)

[Java \(lenguaje de programación\)](#)

[Documentación Firebase](#)

[Firebase Authentication](#)

[Autentica con Firebase mediante un vínculo de correo electrónico en Android](#)

[Autentica mediante el Acceso con Google en Android](#)

[APIs de geolocalización | Google Maps Platform | Google Cloud](#)

[Mapas personalizados | Google Maps Platform | Google Cloud](#)

[Places | Google Maps Platform | Google Cloud](#)

[Kotlin Programming Language \(kotlinlang.org\)](#)

[Cloud Firestore](#)

[Primeros pasos con Cloud Firestore](#)

[Modelo de datos de Cloud Firestore](#)

[Elige una base de datos: Cloud Firestore o Realtime Database](#)

[SDK y bibliotecas cliente](#)

[Elige una estructura de datos](#)

[Agrega datos a Cloud Firestore](#)

[Transacciones y escrituras en lotes](#)

[Borra datos de Cloud Firestore](#)

[Administra Cloud Firestore con Firebase console](#)

[Obtén datos con Cloud Firestore](#)

[Obtén actualizaciones en tiempo real con Cloud Firestore](#)

[Realiza consultas simples y compuestas en Cloud Firestore](#)

[Cloud Storage](#)

[Comienza a usar Cloud Storage en Android](#)

[Crea una referencia de Storage en Android](#)



[Sube archivos en Android](#)

[Descarga archivos en Android](#)

[Borra archivos en Android](#)

[Material Design](#)

[Design - Material Design](#)

[Components - Material Design](#)

[Develop - Material Design](#)

[LottieFiles](#)

[Iconos](#)

[Carbon | Código](#)

[Download Android Studio and SDK tools | Android Developers](#)

[Android Studio release notes | Android Developers](#)

[SDK Tools release notes | Android Developers](#)

[Emulator release notes | Android Developers](#)

[Java 8+ APIs available through desugaring | Android Developers](#)

[Stack Overflow en español](#)

[GitHub - jesusqm94/Proyecto-BBSitter](#)

[Git - gittutorial Documentation \(git-scm.com\)](#)

[Instant Chat Messenger with Cloud Firestore | Eric Decanini](#)

[This Person Does Not Exist](#)

[MoureDev by Brais Moure - YouTube](#)

[Firebase - YouTube](#)

[Fireship - YouTube](#)

[FalconMasters - YouTube](#)