



Sitter

Encuentra
a tu canguro
más cercan@



Jesús González
David Arribas

INDICE

1. Idea del proyecto	5
2. Objetivos.....	5
3. Arquitectura de la app.....	6
4. Tecnologías empleadas	8
4.1 IDE y lenguaje de programación	8
4.2 Base de datos	9
4.3 Diseño y prototipo.....	9
4.4 Control de versiones Git y GitHub.....	10
4.5 Google Maps Platform.....	10
5. Base de datos (Firebase).....	11
5.1 Servicios de Firebase utilizados.....	11
5.1.1 Firebase Authentication.....	11
5.1.2 Firebase Storage	14
5.1.3 Firebase Cloud Firestore	16
5.1.4 Consultas en la base de datos Firebase Firestore.....	18
5.1.5 Modelo de datos.....	19
6. Casos de Uso.....	25
7. Diseño de la app.....	29
7.1 Material Design	29
7.1.1 ¿Qué es Material Design?	29
7.1.2 Componentes de Material Design utilizados en BBSitter	29
7.2 Animaciones Lottie	32
7.2.1 ¿Qué es Lottie?.....	32
7.2.2 ¿Cómo trabaja Lottie?.....	32
7.2.3 Banco de recursos lottiefiles.com.....	32
7.3 Identidad visual de BBSitter	33
7.3.1 Logo.....	33
7.3.2 Tipografía	33
7.3.3 Selección de color.....	34
7.3.4 Imágenes, iconos.....	34
8. GitHub	35
9. Usabilidad de nuestra app.....	39
10. Mejoras de la aplicación	40

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 rellenando 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.

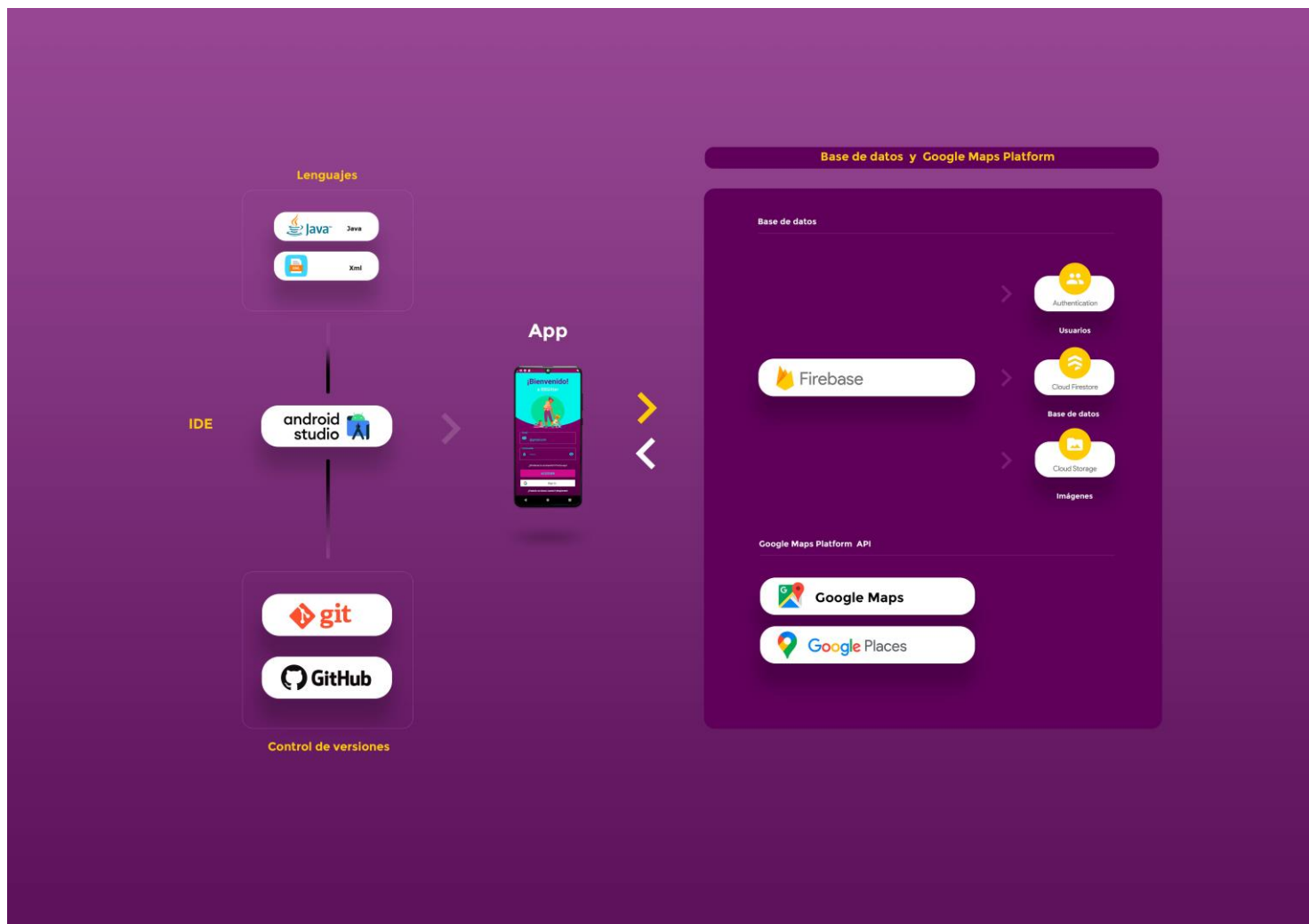
Para poder ir guardando todo el proyecto y no perder nada de el, hemos elegido la manera más extendida y profesional que existe en estos momentos para el **control de versiones** y ese es Git.

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 optima 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. 3 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 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 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. 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.

5.1 Servicios de Firebase utilizados

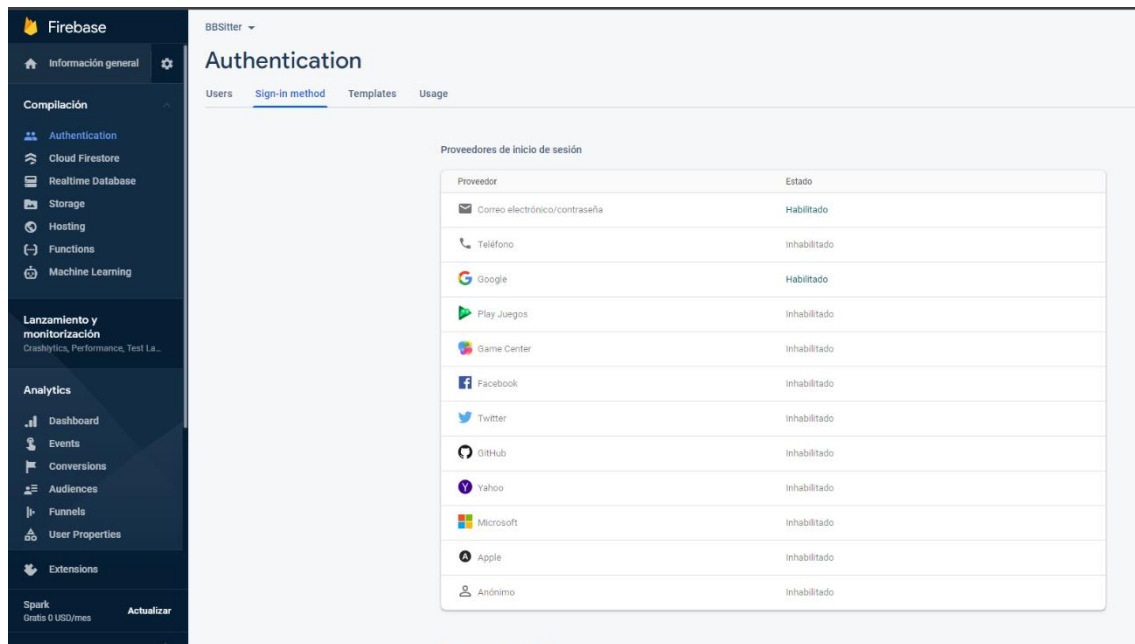
Firebase tiene muchos servicios disponibles, pero en nuestro caso hemos utilizado 3 servicios: Firebase Authentication, Firebase Storage y Firebase Cloud Firestore.

5.1.1 *Firebase Authentication*

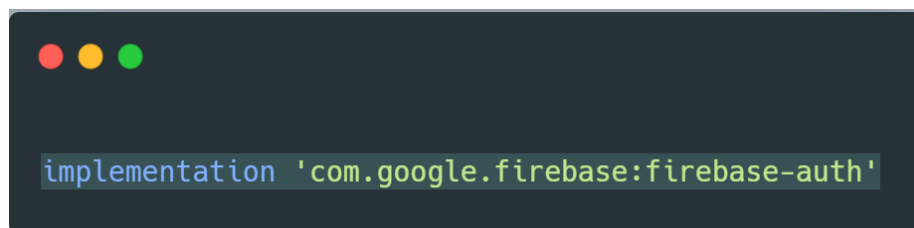
La mayoría de las apps necesitan identificar a los usuarios. Conocer la identidad de un usuario permite que una app guarde sus datos en la nube de forma segura y proporcione la misma experiencia personalizada en todos los dispositivos del usuario.

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 autenticación por correo en Firebase, para ello vamos a ir al servicio Authentication y habilitamos las opciones que vayamos a utilizar para registrarnos.



Una vez habilitados tendremos que implementar en nuestro build.gradle a nivel de proyecto la dependencia de 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();
```

5.1.2 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.firebase: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

gs://bbsitter-61bd3.appspot.com > img_familias

<input type="checkbox"/>	Nombre	Tamaño	Tipo	Última modificación
<input type="checkbox"/>	 0acBKPvY4VRF0g7TbB2JSL0dJjd2	468.81 KB	image/jpeg	2 dic 2020
<input type="checkbox"/>	 2jSSJeMHEfARhInJULX6C1zL022	141.42 KB	image/jpeg	10 dic 2020
<input type="checkbox"/>	 31wNUUWLS0e3ZrW0Vc7D7J59NoY2	495.21 KB	image/jpeg	10 dic 2020
<input type="checkbox"/>	 4tfuq8ZX1Ugy2qfn6gTjHREIMu1	5.37 MB	image/jpeg	5 dic 2020
<input type="checkbox"/>	 5GA2bqKqXjfnKotLfeR3j9mX2z2	532.53 KB	image/jpeg	3 dic 2020
<input type="checkbox"/>	 6XREIRkFRxMrdl4VDbQJOeJ6Dem1	3.82 MB	image/jpeg	11 dic 2020
<input type="checkbox"/>	 901NZ3NcqveUlhiv1AhTh8BGYSI2	502.19 KB	image/jpeg	2 dic 2020
<input type="checkbox"/>	 9TdmzVhbGSZYSHF5Jf0H11KdiO2	532.53 KB	image/jpeg	3 dic 2020
<input type="checkbox"/>	 BUBcNe7l6hdpqz1KYWwJCZ5nDF02	481.26 KB	image/jpeg	10 dic 2020
<input type="checkbox"/>	 DKLKuM2mPvSjehlaG4P3yk5sEsJ2	418.65 KB	image/jpeg	2 dic 2020
<input type="checkbox"/>	 FGHMRXTM6JeSu7XikidQAtqu323	467.15 KB	image/jpeg	10 dic 2020

Subir archivo



Nombre
[0acBKPvY4VRF0g7TbB2JSL0dJjd2](#)

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://bbsitter-61bd3.appspot.com/img_familias/0acBK...](#)

Token de acceso [generar](#)
6d4d9ad6-0db6-4682-b49f-9a51f18d0fb8

[Clear nuevo token de acceso](#)

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

5.1.3 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.firebase:firebase-firestore: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 {
            }
        }
    });
```

5.1.4 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.getId());
```

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").whereGreaterThan("rating", "3");
```

5.1.5 Modelo de datos

Colección Usuarios

```
{
  "usuarios": {
    "GFAsdafoiuejEDWfew": {
      "email": "david@gmail.com",
      "perfil": true,
      "tipo": "familia",
      "uid": "GFAsdafoiuejEDWfew"
    },
    "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": {
    "GFAsdafoiuejEDWfew": {
      "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": "GFAsdafoiuejEDWfew"
    },
    "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": {
    "GFAsdafoiuejEDWfew": {
      "nombre": "David",
      "idHijo": "GFAsdafoiuejEDWfew",
      "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

```
{
  "anuncios": {
    "ADSdoiawshjdnasijd231": {
      "nombre": "Familia Arribas",
      "titulo": "Necesito canguro este fin de semana",
      "descripcion": "Hola, necesito un canguro este fin de semana de 17 a 19. Gracias",
      "direccion": "Calle Chile, Torrejón de Ardoz, Madrid, España",
      "email": "david@gmail.com",
      "casa": "Casa de la familia",
      "fechaPublicacion": "12 dic. 2020",
      "idAnuncio": "ADSdoiawshjdnasijd231",
      "idiomas": {
        "Aleman": true,
        "Español": true
      },
      "img": "https://firebasestorage.googleapis.com/99549-b586-4306-8e58-c0b35455561e",
      "pluses": {
        "No fumador": true,
        "Primeros auxilios": true,
        "Puedo cocinar": true
      },
      "preferenciaEdades": {
        "De 1 a 3 años": true,
        "De 6 a 12 meses": true,
        "Más de 6 años": true
      },
      "tiempo": "Fin de semana",
      "uid": "GFASdafoiejEDWfew"
    },
    "AWE45w3WESFDa3443wqFWSED": {
      "nombre": "Familia González",
      "titulo": "Necesito canguro días de diario",
      "descripcion": "Hola, necesito un canguro para los lunes y miércoles. Gracias",
      "direccion": "Calle Vitoria, Alcalá de Henares, Madrid, España",
      "email": "jesus@gmail.com",
      "casa": "Casa de la familia",
      "fechaPublicacion": "09 dic. 2020",
      "idAnuncio": "AWE45w3WESFDa3443wqFWSED",
      "idiomas": {
        "Ingles": true,
        "Español": true
      },
      "img": "https://firebasestorage.googleapis.com/99549-b586-4306-8e58-c0b35455561e",
      "pluses": {
        "No fumador": true,
        "Primeros auxilios": true,
        "Puedo cocinar": true
      },
      "preferenciaEdades": {
        "De 1 a 3 años": true,
        "De 6 a 12 meses": true,
        "Más de 6 años": true
      },
      "tiempo": "Fin de semana",
      "uid": "DASDasdsaDSADewqad"
    }
  }
}
```

Colección Canguro

```
{
  "canguros": {
    "GFAsdafoiuejEDWfew": {
      "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": {
        "Aleman": 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": "GFAsdafoiuejEDWfew"
    },
    "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": {
        "Ingles": 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 String Uid;  
  
}
```

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;

}
```


6. Casos de Uso

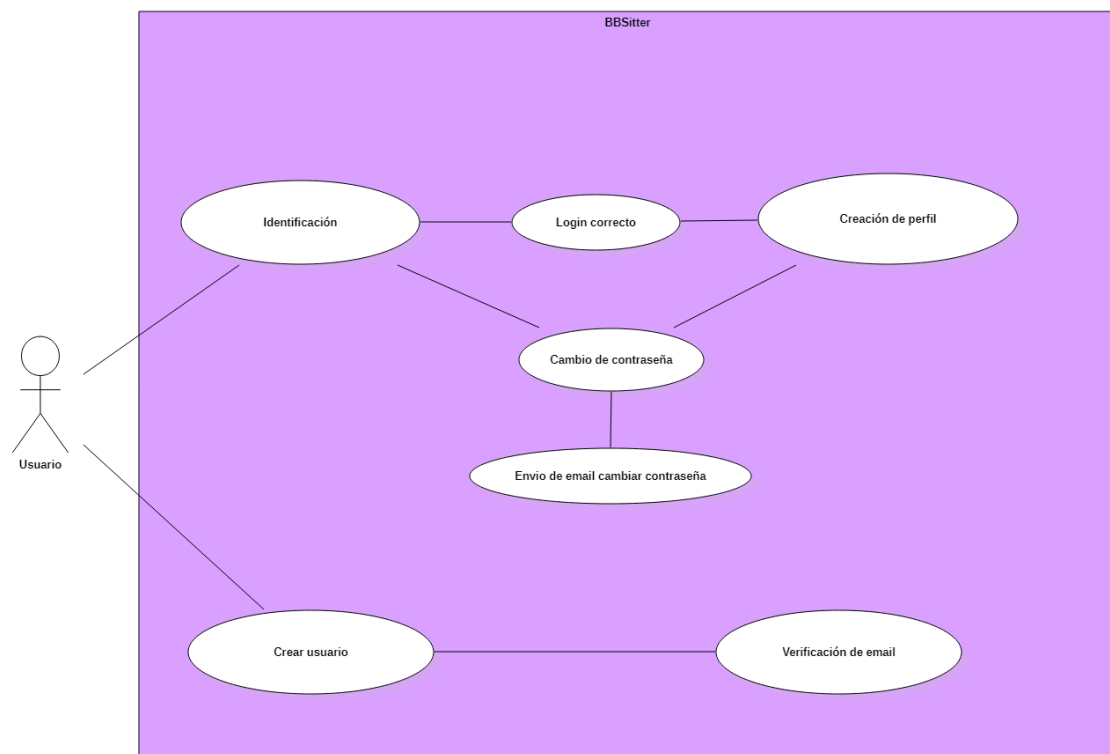
Tenemos en cuenta unos requisitos claves para la utilización de nuestra aplicación. Queremos que el usuario pueda tener una experiencia lo más rápida y eficiente posible, para ello el usuario debería poder hacer estos pasos:

- **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

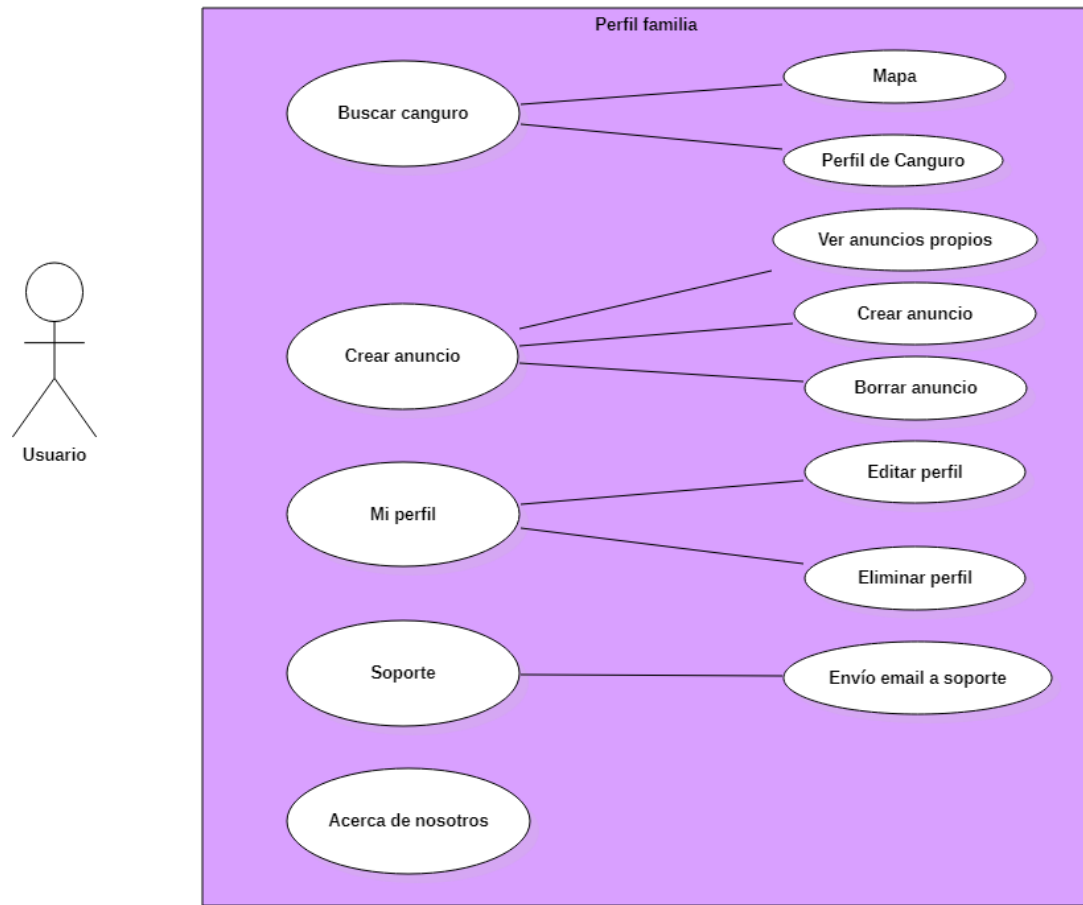
- **Canguro**
 - Creación de perfil con sus datos
 - Visualizar anuncios de las familias
 - Editar su perfil
 - Eliminar su perfil
 - Contactar con Soporte
 - Cerrar sesión

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 su tuviera algún problema

En Acerca de nosotros podrá ver nuestros datos y nuestro GitHub

7. 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.

7.1 Material Design



7.1.1 ¿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 tremendamente 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.

7.1.2 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/chipgroupPlusesMiPerfilCanguro"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingHorizontal="5dp"
    style="@style/Widget.MaterialComponents.Chip.Entry"
    app:singleLine="false"
    app:selection="false"
    app:chipSpacingHorizontal="15dp">
```

No soy fumador

Sé primeros auxilios

FloatingActionButton

```
<!-- BUTTON EDITAR PERFIL -->
<com.google.android.material.floatingactionbutton.ExtendedFloatingActionButton
    android:id="@+id/btnEditarPerfilCanguro"
    android:layout_width="wrap_content"
    android:layout_height="40dp"
    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"
/>
```

EDITAR PERFIL

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>
```



7.2 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.

7.2.1 ¿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.

7.2.2 ¿Cómo trabaja Lottie?

Lottie trabaja con animaciones de **After Effects** exportadas gracias al **plugging 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 plugging **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.

7.2.3 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.

7.3 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.

7.3.1 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.



7.3.2 Tipografía

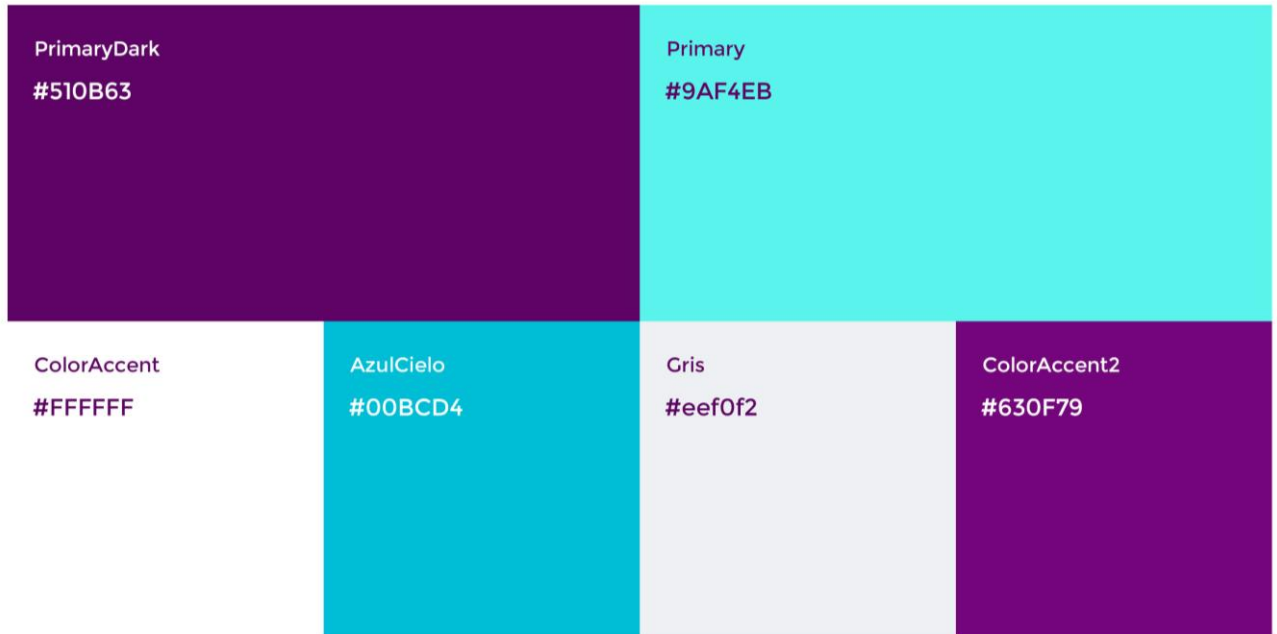
Elegir una fuente distintiva (o un grupo de fuentes) que realmente marquen la diferencia y que transmitan la personalidad de tu marca, es esencial para diferenciarse del resto. En nuestro caso optamos por la tipografía que nos usa Android por defecto, Robocop.

Roboto

The July sun caused a fragment of black pine wax to ooze on the velvet quilt
The July sun caused a fragment of black pine wax to ooze on the velvet quilt
The July sun caused a fragment of black pine wax to ooze on the velvet quilt
The July sun caused a fragment of black pine wax to ooze on the velvet quilt
The July sun caused a fragment of black pine wax to ooze on the velvet quilt

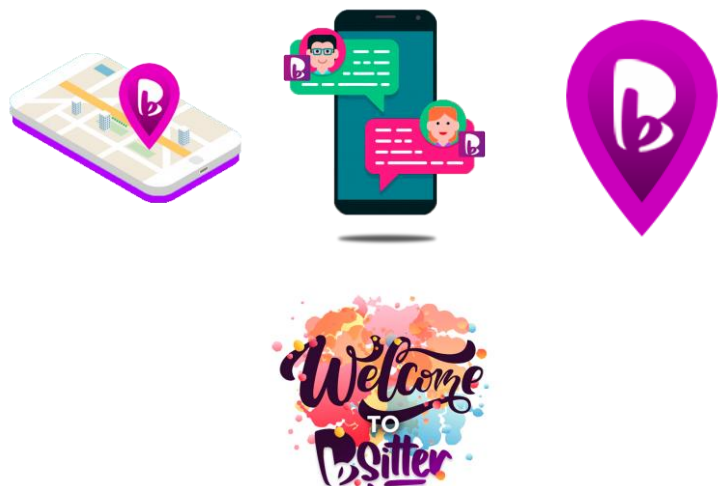
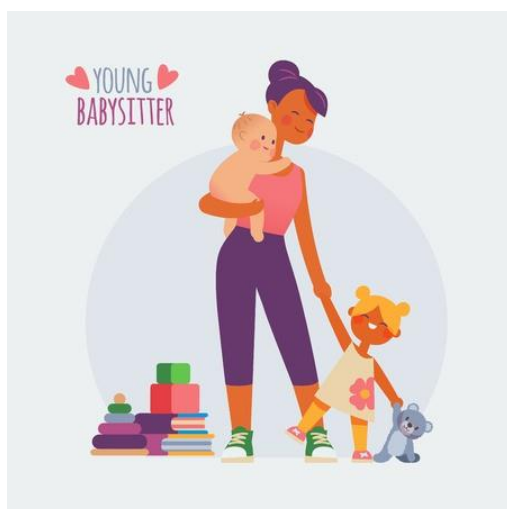
7.3.3 Selección de color

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 quieres despertar en ella.



7.3.4 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.



8. 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.

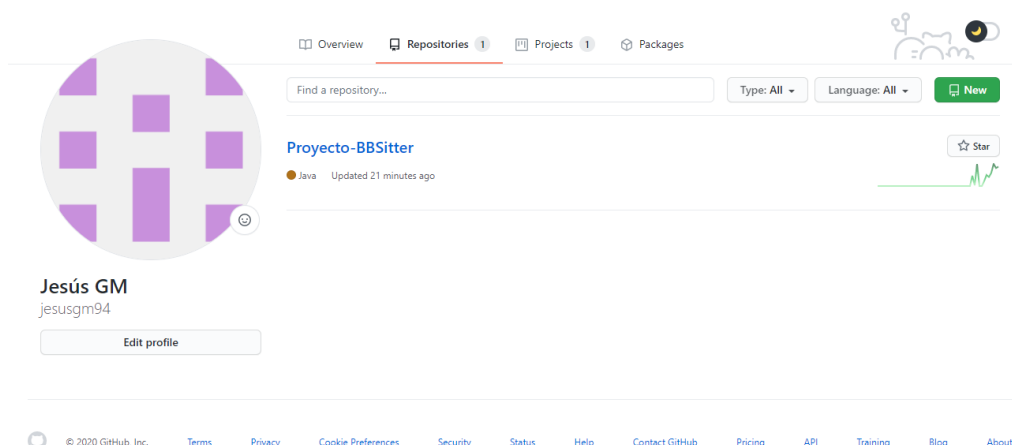
No hay códigos sobrescritos ya que Git guarda varias copias en el repositorio.

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](https://github.com), buscar la pestaña “Repositories” y hacer clic en “New”



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 rellenar 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.](#) [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.](#)

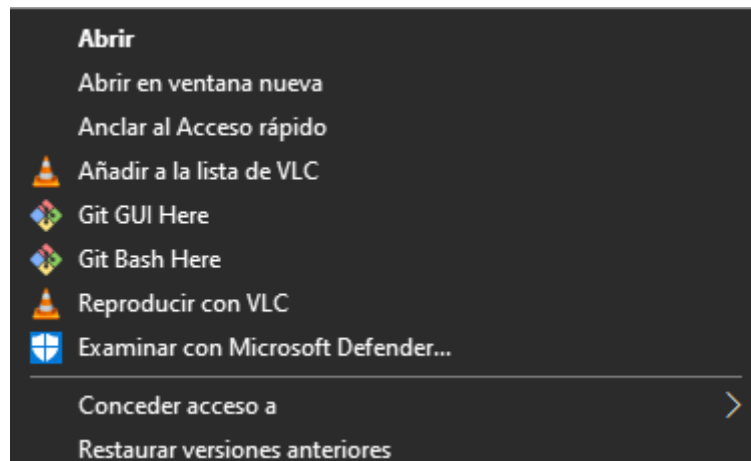
This will set  main as the default branch. Change the default name in your [settings](#).

Create repository

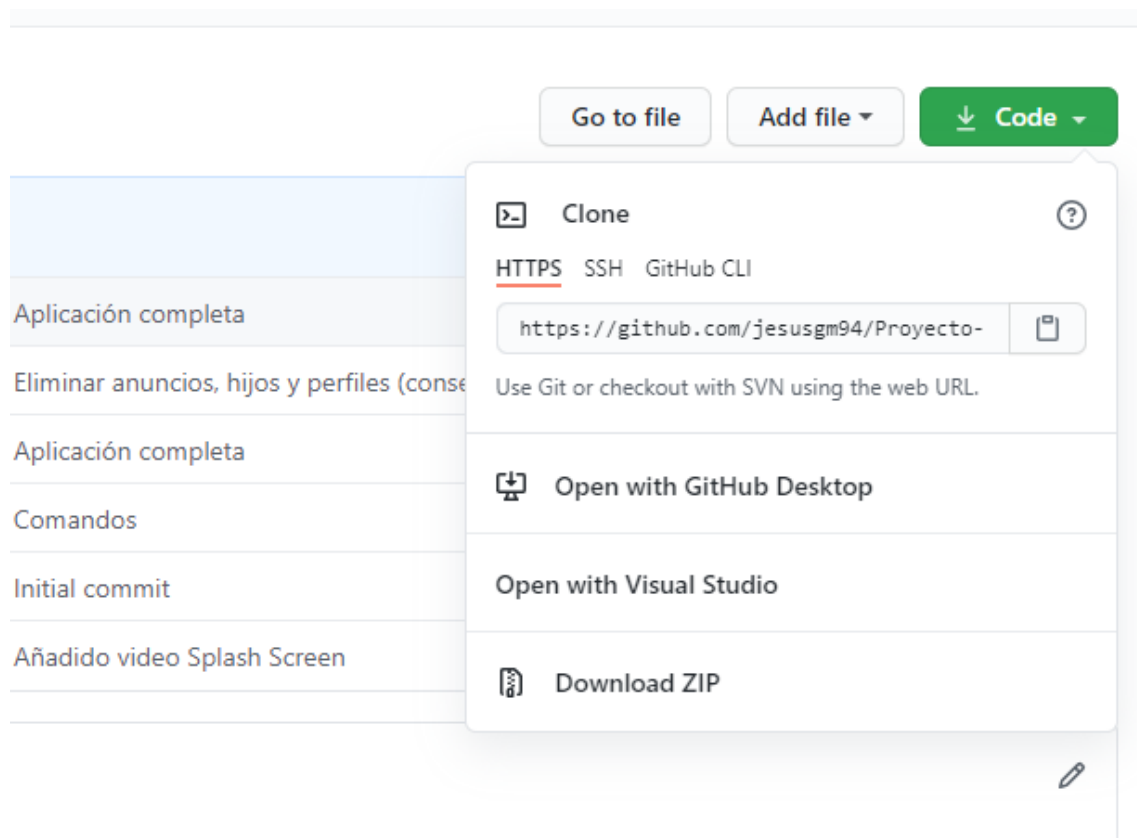
Paso 2. Clonar repositorio en nuestro pc

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



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
```

9. 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 teléfono móvil. 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**

Cuantos menos pasos, páginas, botones y campos tengan que hacer 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.

10. 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