

# Introducción a la Computación Móvil

## Firestore

Pontificia Universidad Javeriana  
Departamento de Ingeniería de Sistemas  
Profesor: Carlos Andrés Parra  
E-mail: [ca.parraa@javeriana.edu.co](mailto:ca.parraa@javeriana.edu.co)



# Resumen de Android

✓ Layouts (Linear, Frame, Relative, Constraint)

✓ Activities, LifeCycle, Intents, bundles,

✓ Permissions

✓ Hardware Access

✓ Camera, Gallery, Contacts

✓ Sensors

✓ Location

✓ Maps

▪ Data base?

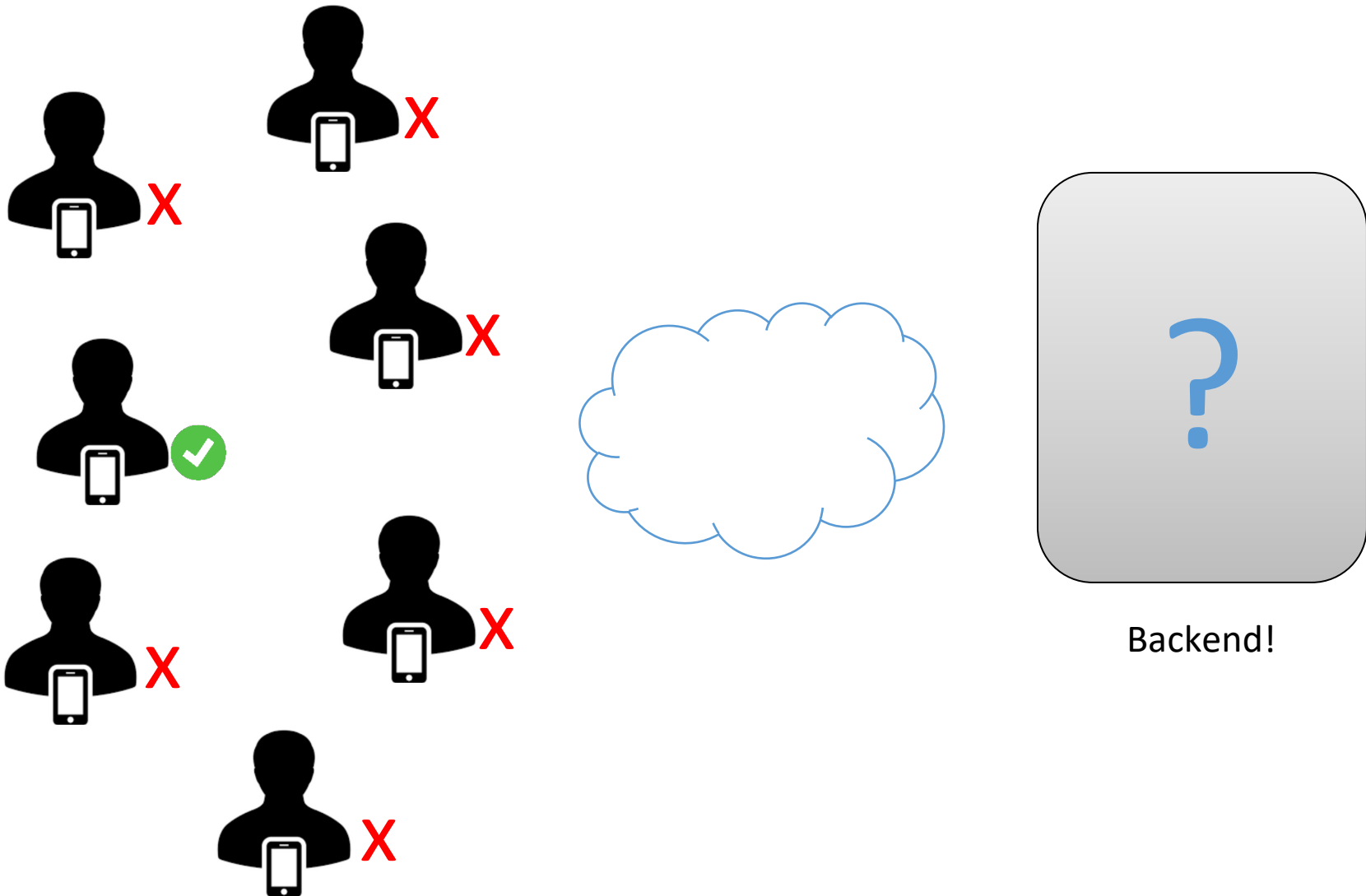
▪ Multi-User?

▪ Shared Information?

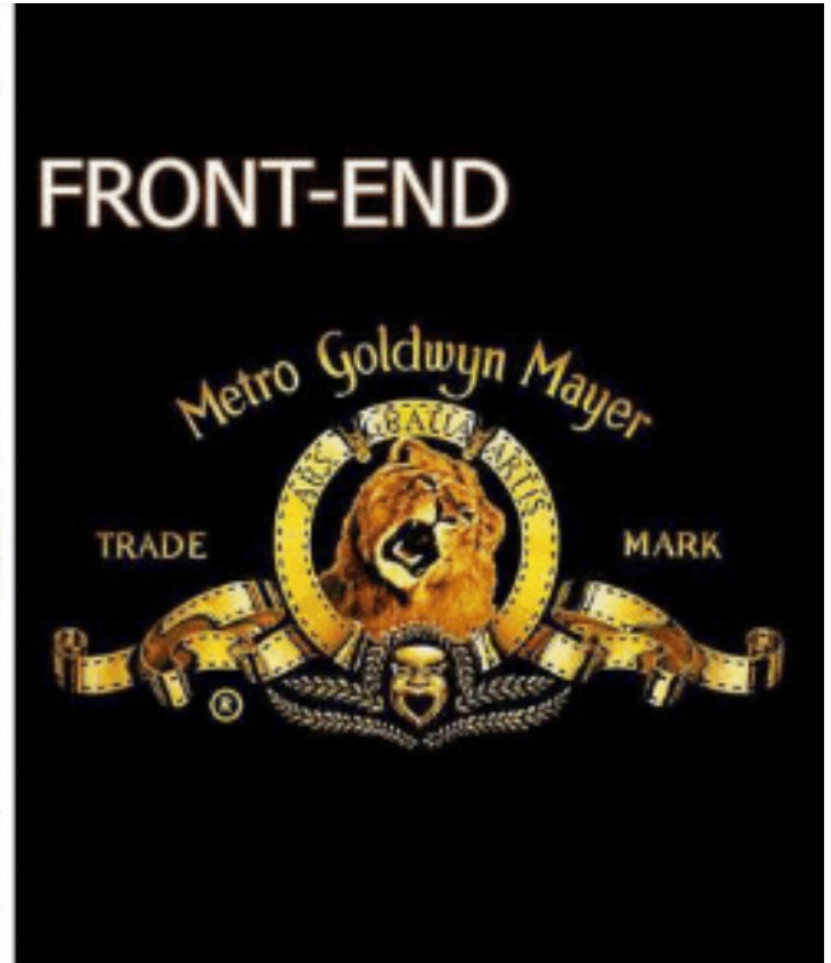
▪ REST Web Services?

▪ Notifications?

# Multi-User

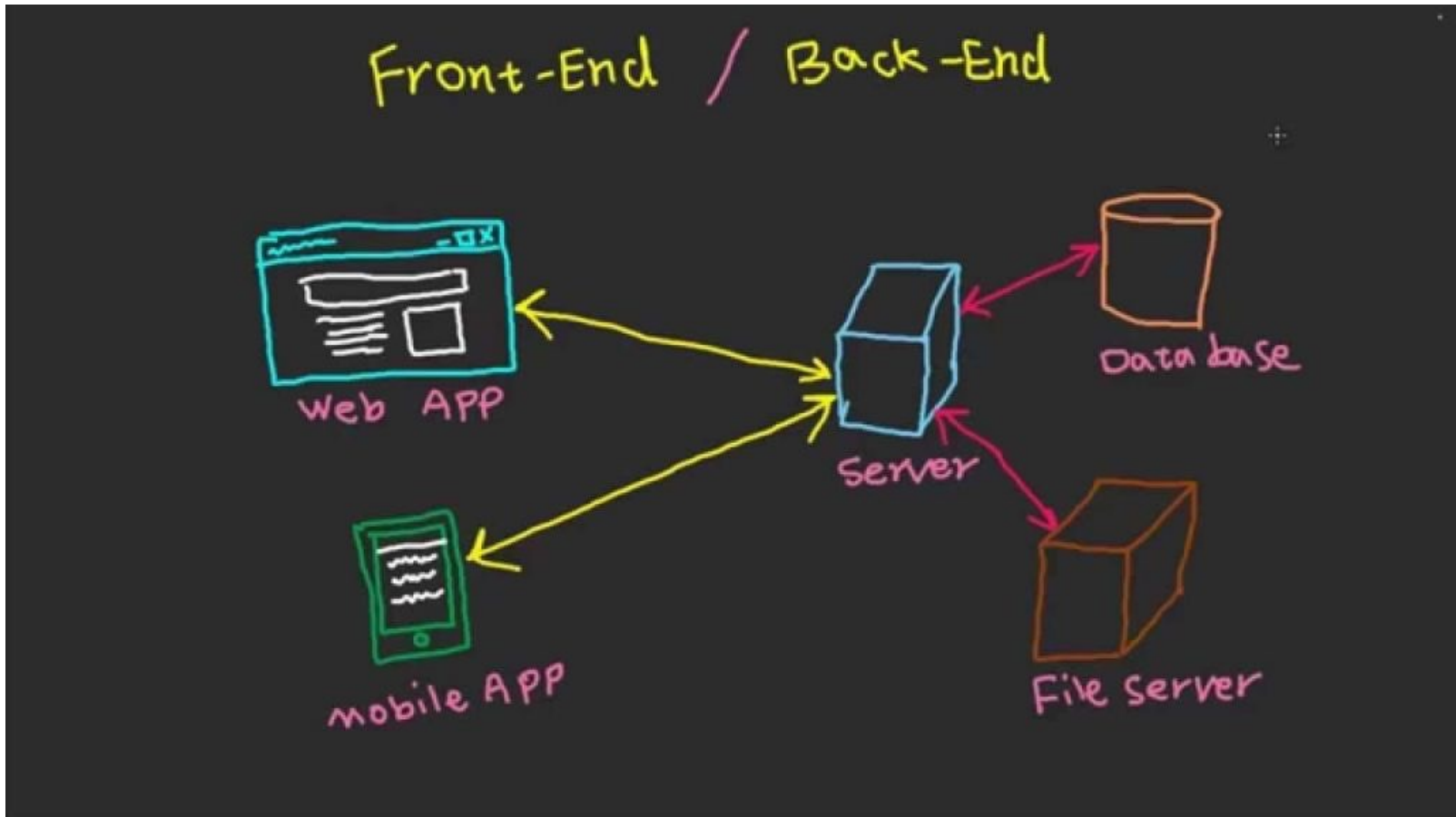


# Backend



## Frontend Vs. Backend

# Backend



[http://icreativelabs.com/blog/wp-content/uploads/2019/01/45102766\\_10156936732174238\\_1065802795996676096\\_o-1024x576.jpg](http://icreativelabs.com/blog/wp-content/uploads/2019/01/45102766_10156936732174238_1065802795996676096_o-1024x576.jpg)

# Backend

- Para la creación de aplicaciones móviles existen dos formas de trabajar el backend
    - Utilizar un backend genérico provisto por un tercero, por ejemplo:
      - [Google Firebase](#)
      - [Amazon Amplify](#)
      - [Parse Platform](#) (Sin infraestructura!)
    - Construir un backend a la medida, típicamente exponer servicios REST a través de:
      - Java (JAX-RS)
      - Spring
      - Django
      - .Net
      - Etc!
- También proveer la infraestructura (on premise) ó contratar un proveedor de Cloud IaaS o PaaS!*

# Firebase

- Backend para aplicaciones móviles

## Develop & test your app



**Realtime Database**

iOS   C++ 



**Authentication**

iOS   C++ 



**Cloud Storage**

iOS   C++ 



**Test Lab**





**Crash Reporting**

iOS 



**Cloud Functions**

iOS   C++ 



**Hosting**





**Performance Monitoring**

iOS 

# Firebase

## Grow & engage your audience



**Analytics**

iOS  C++ 



**Dynamic Links**

iOS  C++ 



**Invites**

iOS  C++ 



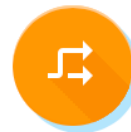
**AdWords**

iOS  C++ 



**Cloud Messaging**

iOS  `</>` C++ 



**Remote Config**

iOS  C++ 



**App Indexing**

iOS 



**AdMob**

iOS  C++ 



# Precio\$!

<https://firebase.google.com/pricing/>

# Authentication

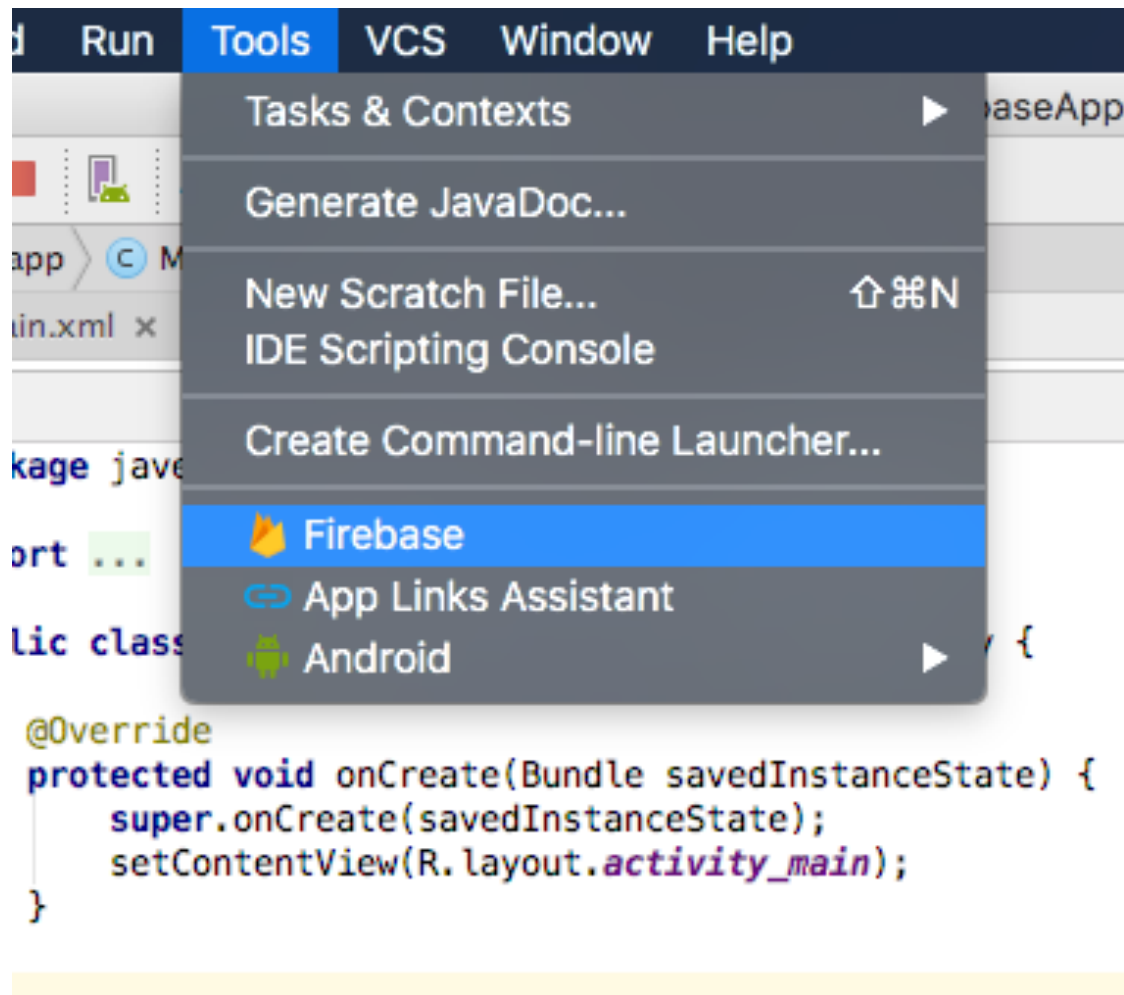
Firebase

# Conectar la aplicación con Firebase

## Pre-requisitos

- A device running Android 4.0 (Ice Cream Sandwich) or newer, and **Google Play services 11.4.0** or higher
- The Google Play services SDK from the **Google Repository**, available in the [Android SDK Manager](#)
- The latest version of [Android Studio](#), version **1.5** or higher


# Conectar la aplicación a firebase




**Nota.** Si el asistente no funciona o no esta disponible, ver la forma manual en: <https://firebase.google.com/docs/android/setup>

# Conectar la aplicación a firebase


Assistant

 **Firebase**


Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. [Learn more](#)

▶  **Analytics**

Measure user activity and engagement with free, easy, and unlimited analytics. [More info](#)


▶  **Cloud Messaging**

Deliver and receive messages and notifications reliably across cloud and device. [More info](#)


▼  **Authentication**

Sign in and manage users with ease, accepting emails, Google Sign-In, Facebook and other login providers. [More info](#)


▶ [Email and password authentication](#)

▶  **Realtime Database**


Store and sync data in realtime across all connected clients. [More info](#)

▶  **Storage**

Store and retrieve large files like images, audio, and video without writing server-side code. [More info](#)

▶  **Remote Config**

Customize and experiment with app behavior using cloud-based configuration parameters. [More info](#)

▶  **Test Lab**

Test your apps against a wide range of physical devices hosted in Google's cloud. [More info](#)

# Autenticación basada en correo y password

## Email and password authentication

You can use Firebase Authentication to let your users sign in with their email addresses and passwords, and to manage your app's password-based accounts. This tutorial helps you set up an email and password system and then access information about the user.

[Launch in browser](#)

---

### 1 Connect your app to Firebase

Connect to Firebase

### 2 Add Firebase Authentication to your app

Add Firebase Authentication to your app

To use an authentication provider, you need to enable it in the [Firebase console](#). Go to the Sign-in Method page in the Firebase Authentication section to enable Email/Password sign-in and any other identity providers you want for your app.

### 3 Listen for auth state

Declare the `FirebaseAuth` and `AuthStateListener` objects.


```
private FirebaseAuth mAuth;
```

```
private FirebaseAuth.AuthStateListener mAuthListener;
```

In the `onCreate()` method, initialize the `FirebaseAuth` instance and the `AuthStateListener` method so you can track whenever the user signs in or out.

# Autenticación basada en correo y password

Connect to Firebase

 **Firebase**

☒ Create new Firebase project [What's this?](#)

Signed in as **cmovilpuj@gmail.com** [Sign out](#)

☐ Choose an existing Firebase or Google project

Please select an existing project.

Country/region [What's this?](#)

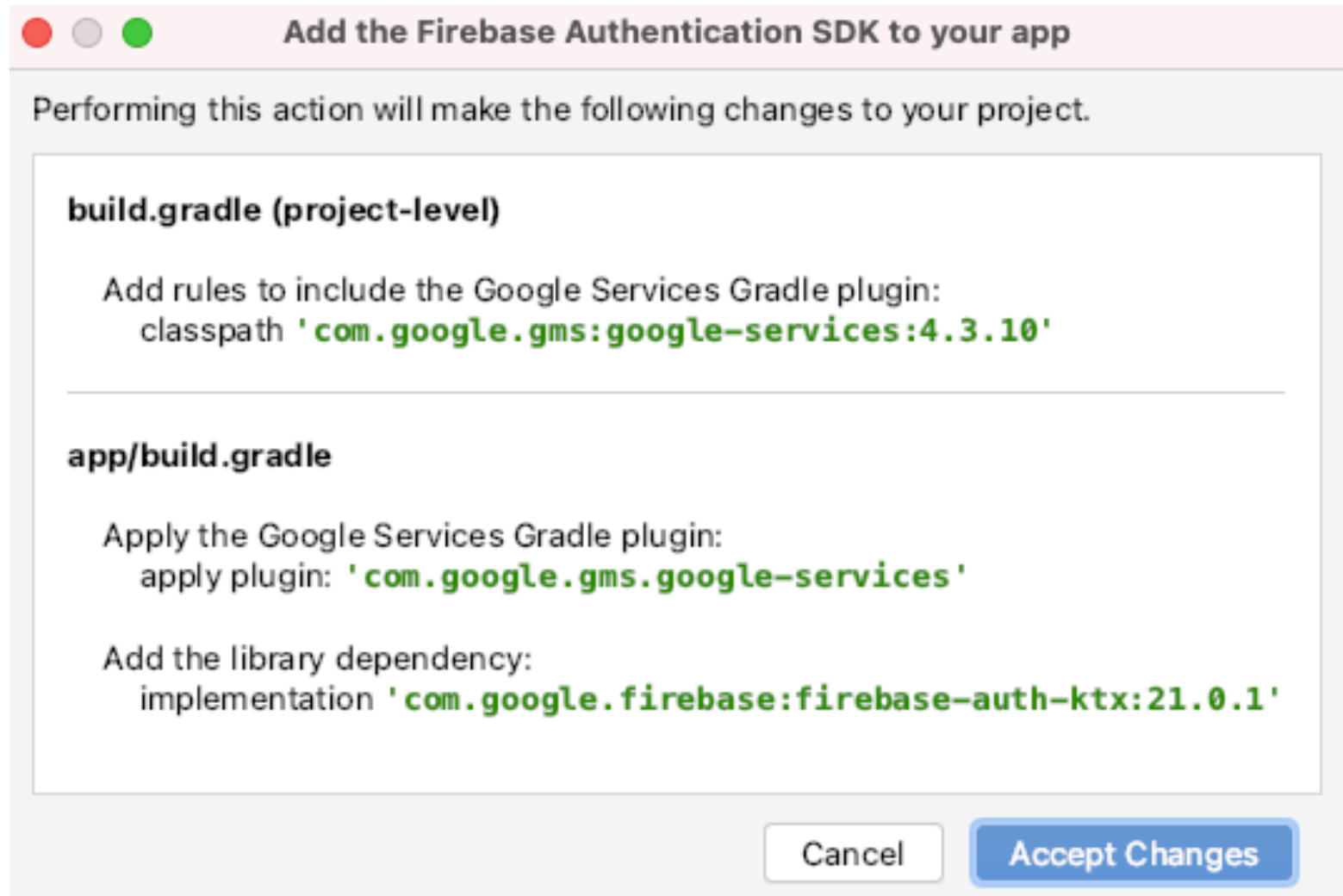
By default, your Firebase Analytics data will enhance other Firebase features and Google products. You can control how your Firebase Analytics data is shared in your settings at anytime. [Learn more](#)

By proceeding and clicking the **Connect** button below, you agree that you are using [Firebase services](#) in your app and agree to the updated [terms](#)

Cancel

Connect to Firebase

# Autenticación basada en correo y password

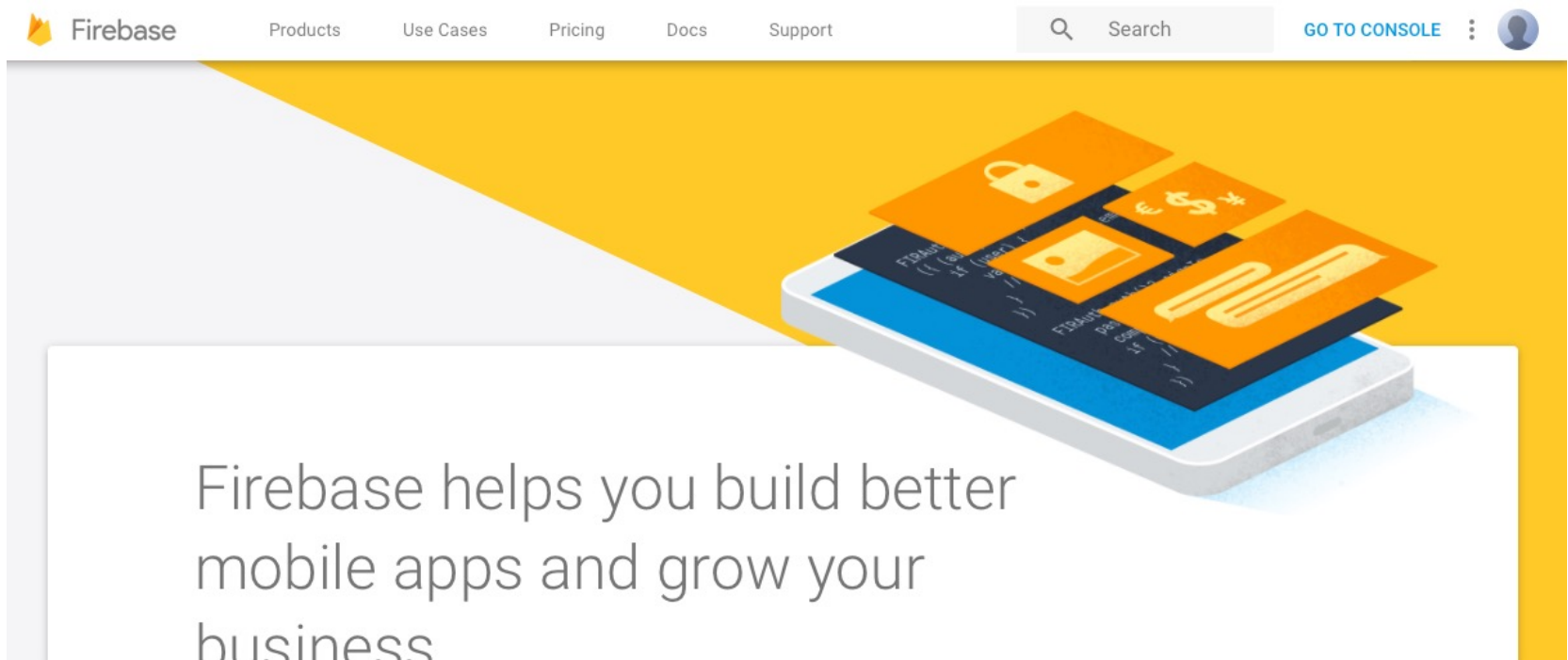




# Desde el navegador

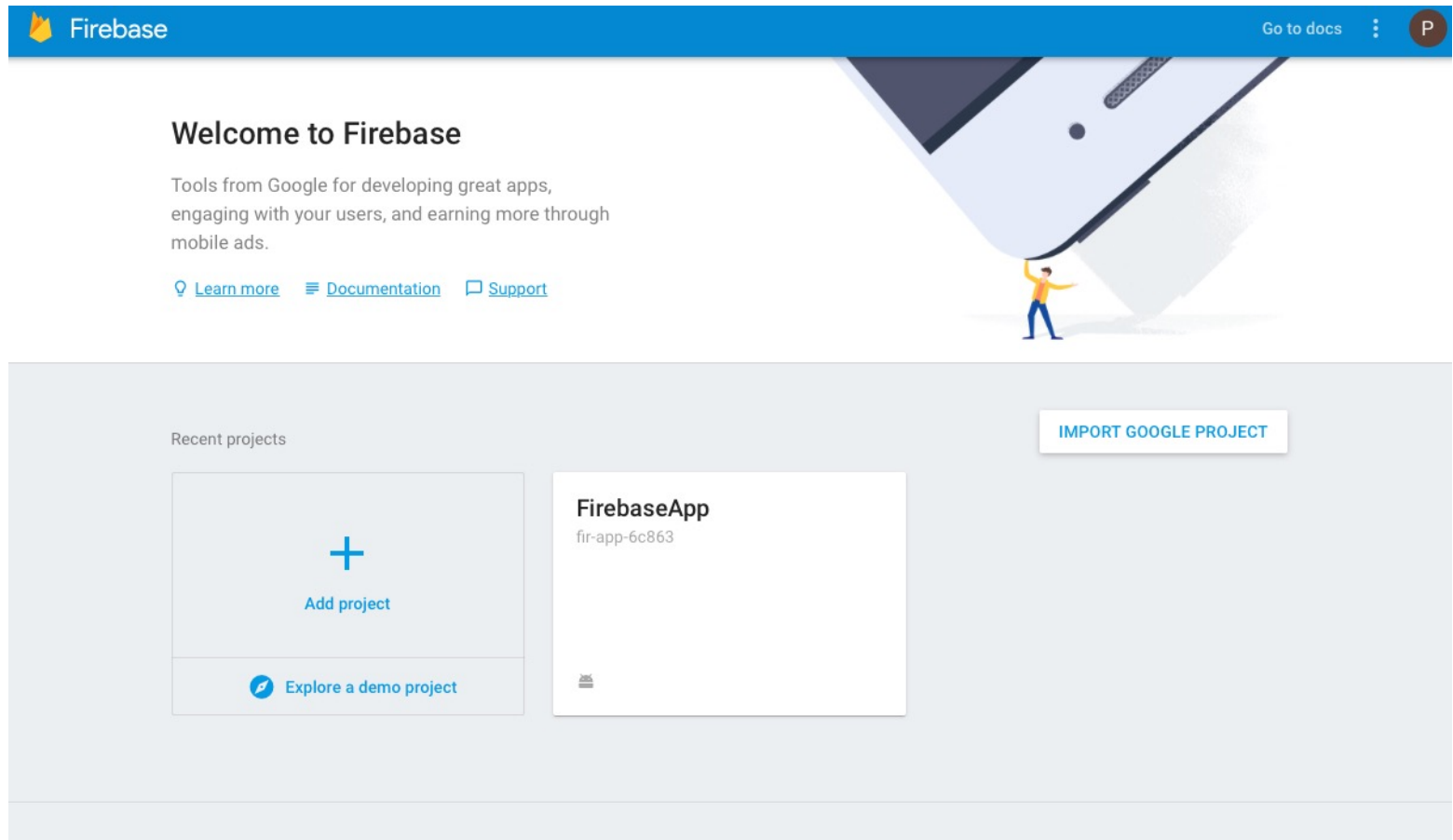
- Entrar a firebase con su usuario de gmail.

<https://firebase.google.com>



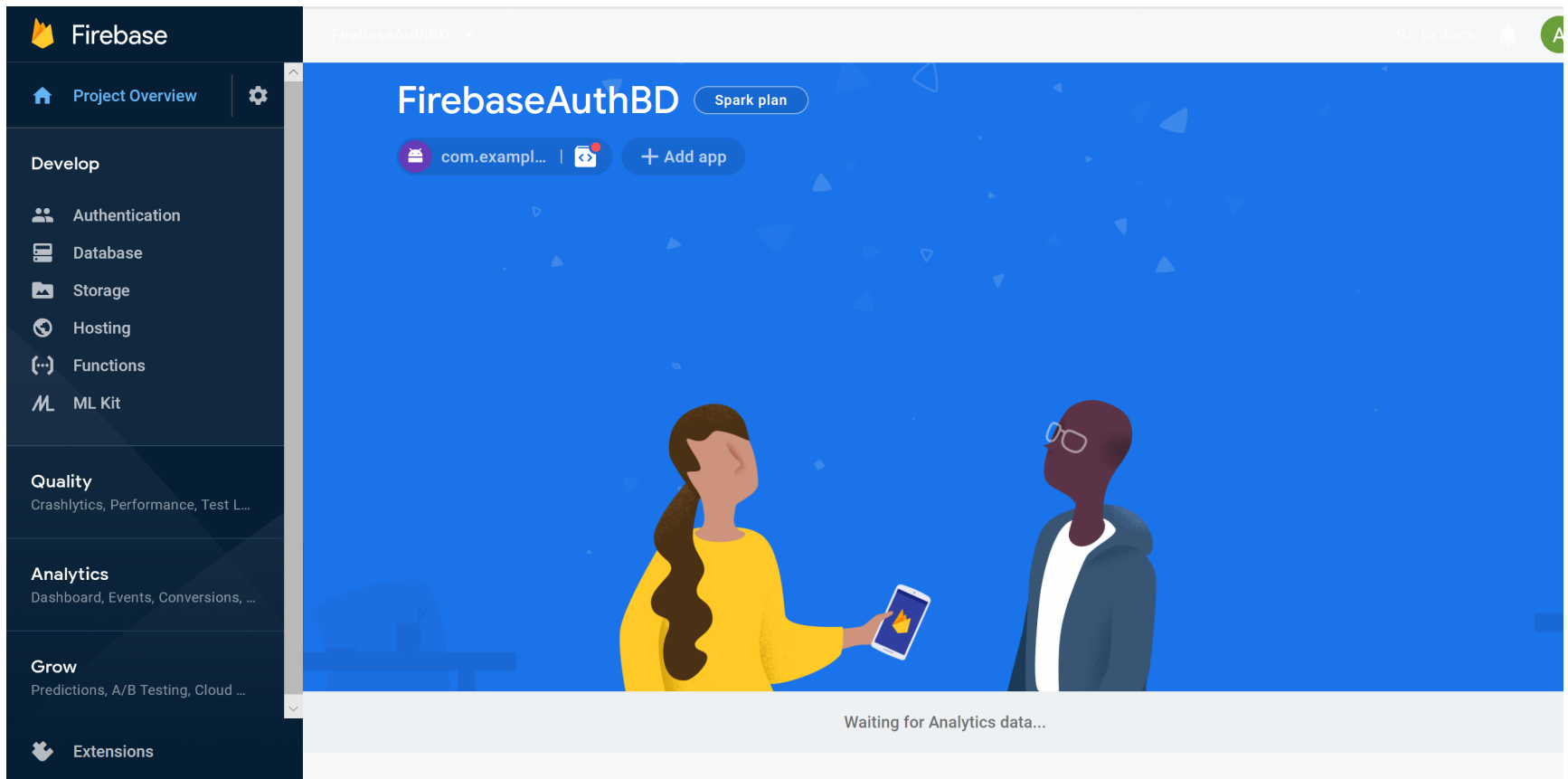
# Desde el navegador

- En la consola seleccionar su aplicación



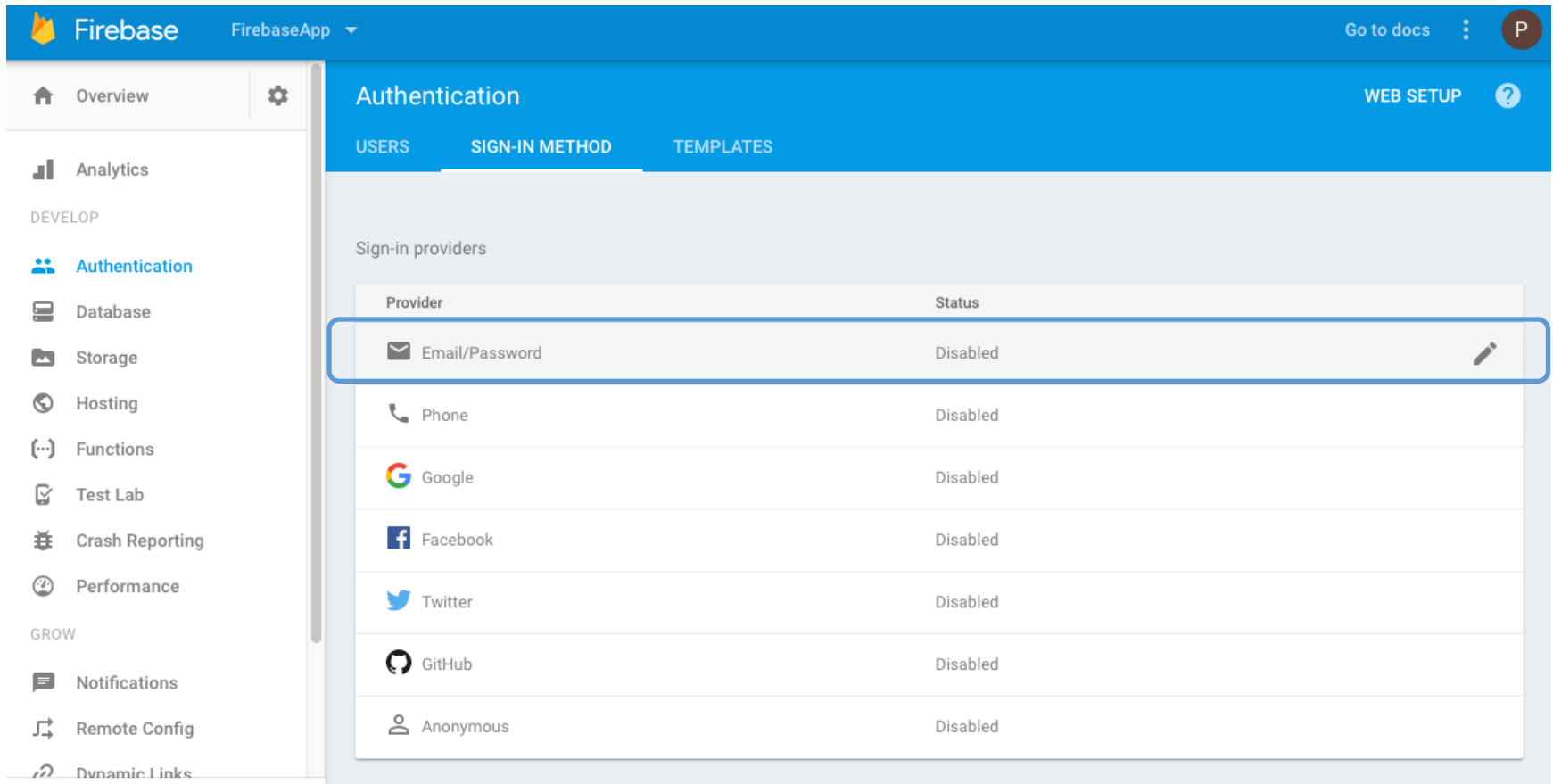
# Desde el navegador

- Página de la consola web de firebase



# Activar autenticación

- Ir a la opción Email y Password y activarla



The screenshot shows the Firebase Authentication console. The left sidebar contains navigation links for Overview, Analytics, and various development tools. The main content area is titled 'Authentication' and has tabs for USERS, SIGN-IN METHOD, and TEMPLATES. The 'SIGN-IN METHOD' tab is active, displaying a table of 'Sign-in providers'. The 'Email/Password' provider is highlighted with a blue box, indicating it is the one to be activated.

Provider	Status
Email/Password	Disabled
Phone	Disabled
Google	Disabled
Facebook	Disabled
Twitter	Disabled
GitHub	Disabled
Anonymous	Disabled

# Manejo de usuarios

- Crear usuario de prueba

The screenshot displays the Firebase Authentication console interface. On the left, a sidebar lists various Firebase services: Overview, Analytics, Authentication (highlighted), Database, Storage, Hosting, Functions, Test Lab, Crash Reporting, Performance, Notifications, and Remote Config. The main content area is titled 'Authentication' and includes tabs for 'USERS', 'SIGN-IN METHOD', and 'TEMPLATES'. A search bar at the top of the 'USERS' tab allows searching by email address, phone number, or user UID. Below the search bar is a table with columns: Identifier, Providers, Created, Signed In, and User UID. A modal dialog titled 'Add an Email/Password user' is open, featuring input fields for 'Email' and 'Password', and buttons for 'CANCEL' and 'ADD USER'. The background of the console shows a message: 'No users for this project yet'.

# Manejo de autenticación

- Definir los atributos de la actividad necesarios para la autenticación

```
public class MainActivity extends AppCompatActivity {  
  
    //firebase authentication  
    private FirebaseAuth mAuth;  
    //...  
}
```

# Manejo de autenticación

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    mAuth = FirebaseAuth.getInstance();  
}
```

# Manejo de autenticación

- En onStart se verifica el estado actual de la autenticación del usuario, en este caso, si el usuario esta autenticado, se lanza la siguiente actividad

```
@Override
protected void onStart() {
    super.onStart();
    FirebaseUser currentUser = mAuth.getCurrentUser();
    updateUI(currentUser);
}

private void updateUI(FirebaseUser currentUser) {
    if(currentUser!=null) {
        Intent intent = new Intent(getBaseContext(), HomeActivity.class);
        intent.putExtra("user", currentUser.getEmail());
        startActivity(intent);
    } else {
        email.setText("");
        password.setText("");
    }
}
```



# Sign In with Firebase

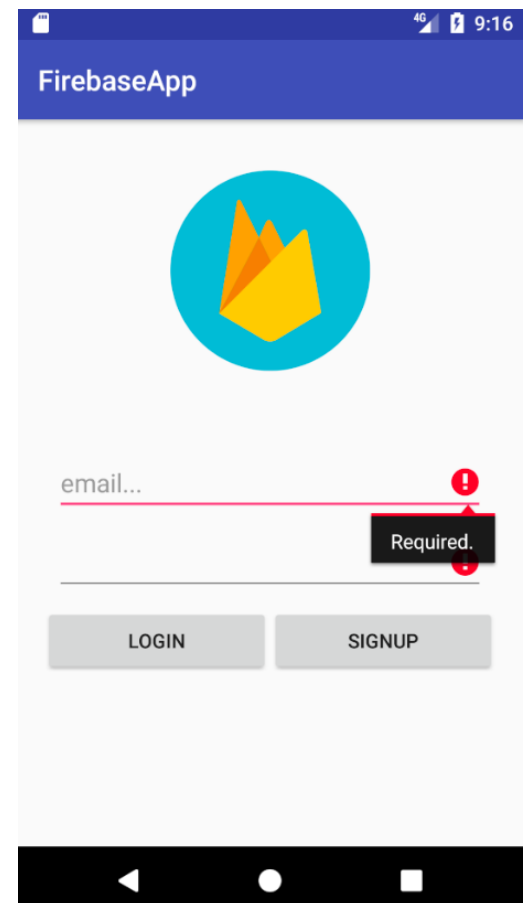
```
mAuth.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            Log.d(TAG, "signInWithEmail:complete:" + task.isSuccessful());

            // If sign in fails, display a message to the user. If sign in succeeds
            // the auth state listener will be notified and logic to handle the
            // signed in user can be handled in the listener.
            if (!task.isSuccessful()) {
                Log.w(TAG, "signInWithEmail:failed", task.getException());
                Toast.makeText(MainActivity.this, R.string.auth_failed,
                    Toast.LENGTH_SHORT).show();
                mUser.setText("");
                mPassword.setText("");
            }
        }
    });
```

# Validación de campos

- Antes de enviar la petición a Firebase, es importante validar los campos ingresados por el usuario

```
private boolean validateForm() {  
    boolean valid = true;  
    String email = mUser.getText().toString();  
    if (TextUtils.isEmpty(email)) {  
        mUser.setError("Required.");  
        valid = false;  
    } else {  
        mUser.setError(null);  
    }  
    String password = mPassword.getText().toString();  
    if (TextUtils.isEmpty(password)) {  
        mPassword.setError("Required.");  
        valid = false;  
    } else {  
        mPassword.setError(null);  
    }  
    return valid;  
}
```



# Validación para cada campo

- Verificación básica de correo

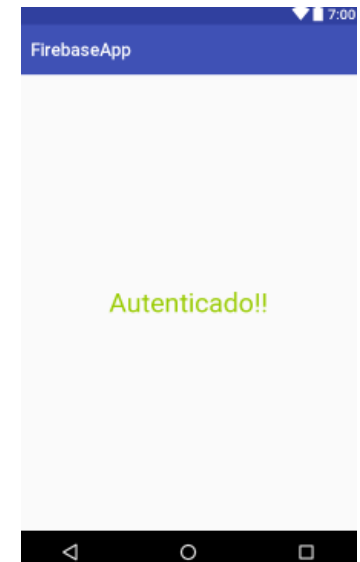
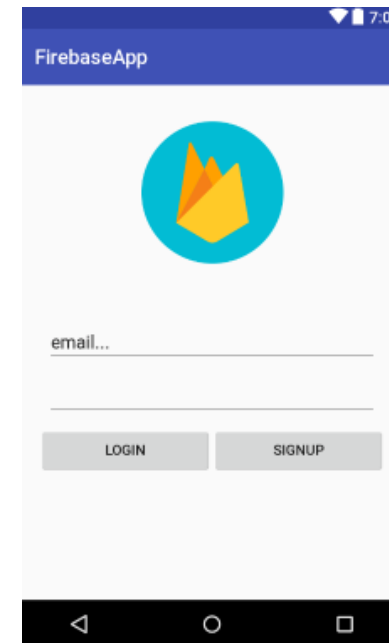
```
private boolean isEmailValid(String email) {  
    if (!email.contains("@") ||  
        !email.contains(".") ||  
        email.length() < 5)  
        return false;  
    return true;  
}
```

# Sign In with Firebase (Validado)

```
private void signInUser(String email, String password) {
    if (validateForm()) {
        mAuth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()) {
                        // Sign in success, update UI
                        Log.d(TAG, "signInWithEmail:success");
                        FirebaseUser user = mAuth.getCurrentUser();
                        updateUI(user);
                    } else {
                        // If sign in fails, display a message to the user.
                        Log.w(TAG, "signInWithEmail:failure", task.getException());
                        Toast.makeText(MainActivity.this, "Authentication failed.",
                                    Toast.LENGTH_SHORT).show();
                        updateUI(null);
                    }
                }
            });
    }
}
```

# Ejercicio 1

- Registrar la aplicación y el usuario con firebase.
- Usando la interfaz web, habilitar la autenticación y crear un usuario de prueba.
- Crear una aplicación con dos campos de texto para el login de un usuario como se muestra en la figura.
- Utilizar el API de Firebase para autenticar el usuario haciendo una validación previa de los campos.
- Si hay error en la autenticación mostrar un toast.
- Si la autenticación es exitosa, lanzar una actividad de bienvenida a la zona de usuarios autenticados.



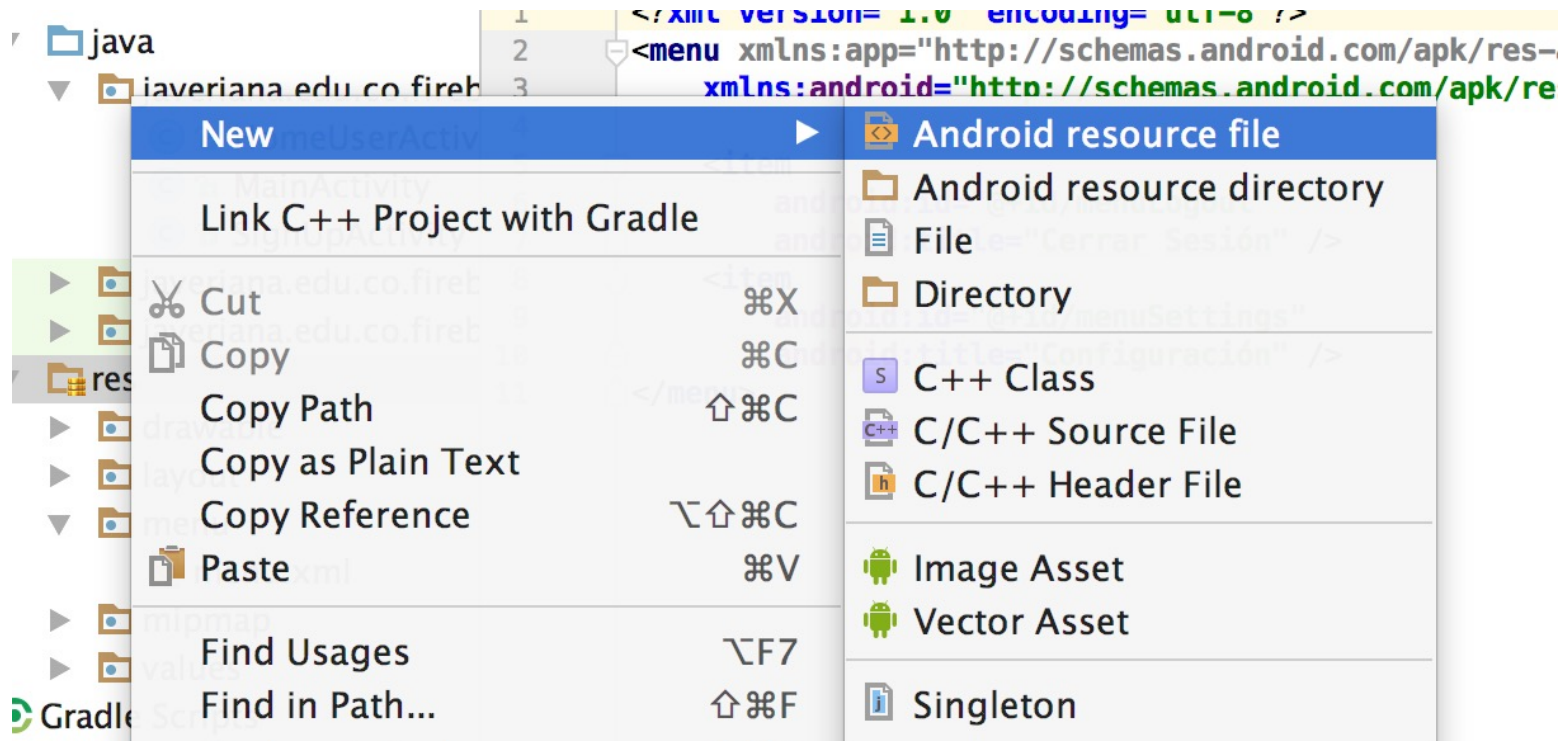
# Log out

- Qué pasa si se vuelve a correr la aplicación?
- Es necesario dar al usuario la opción de salir de la sesión autenticada.
- Una forma simple de hacerlo:

```
btSignOut.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        mAuth.signOut();  
    }  
});
```

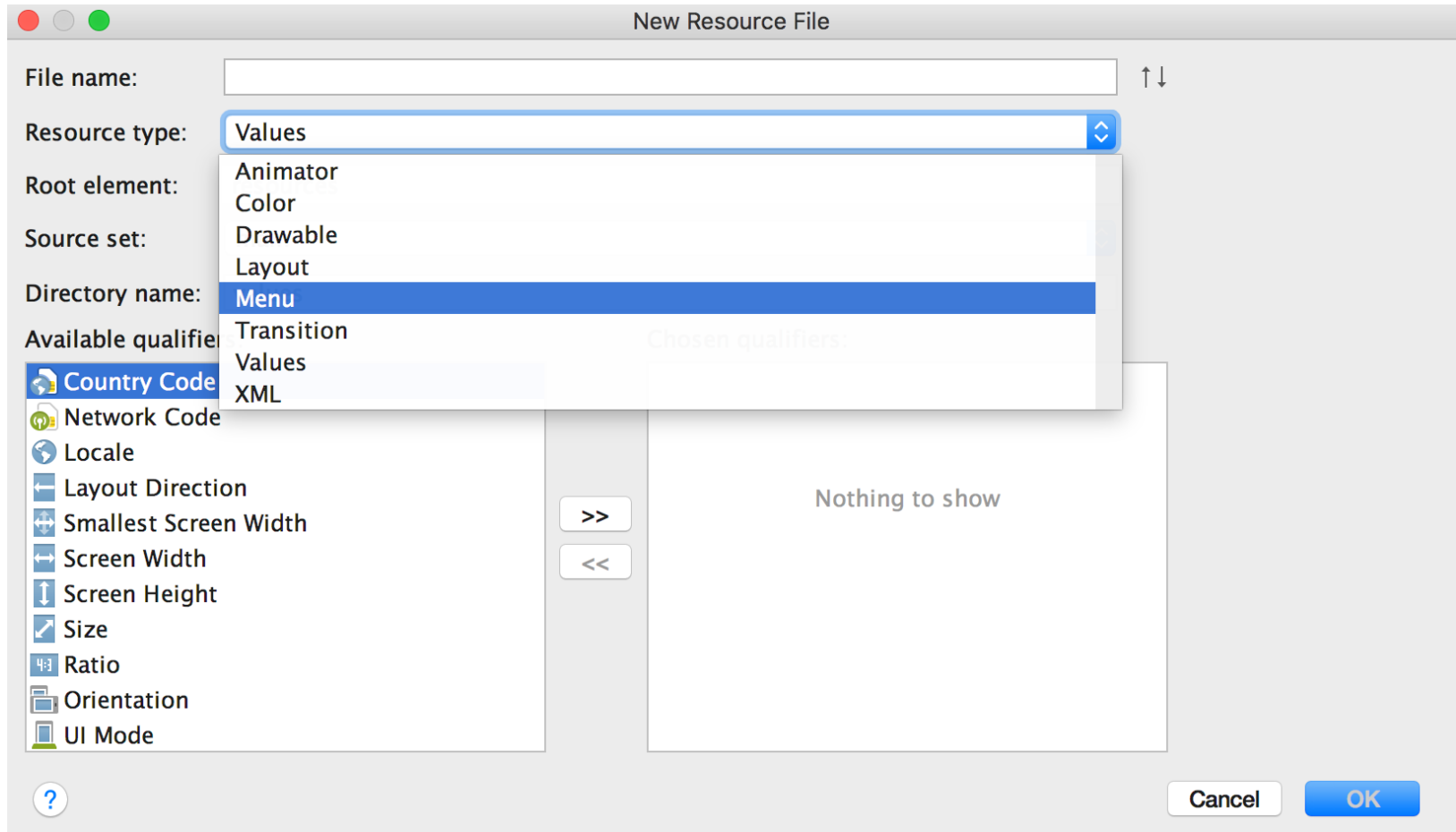
# Opciones de menú

- Se pueden definir opciones en la barra de título de la aplicación
- Para esto, hay que definir un archivo de menú en la carpeta de recursos



# Opciones de menú

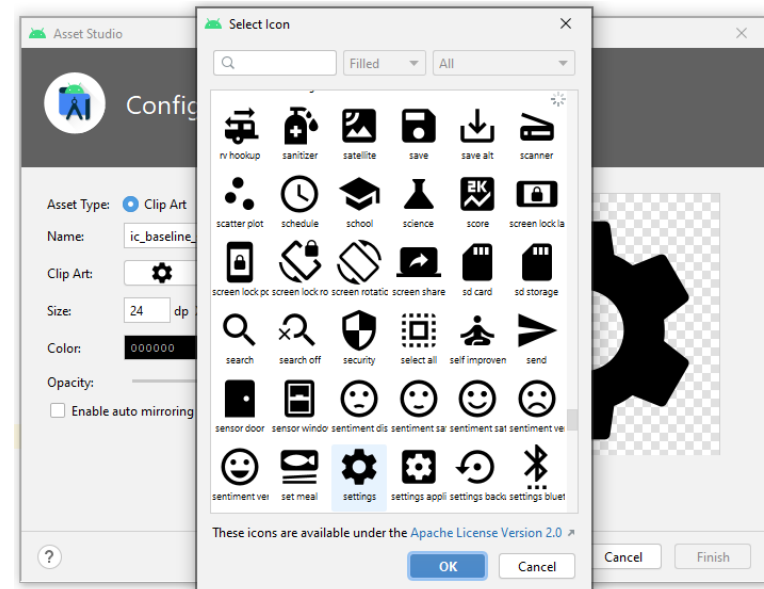
- Seleccionar la opción de menú:





# Opciones de Menú

- Agregar íconos disponibles en Android Studio
  - Sobre la carpeta res se hace click derecho -> *new* -> *vector asset*
  - En la opción clip art se abre una ventana de selección de iconos, después de seleccionarlos quedan disponibles en *drawable*

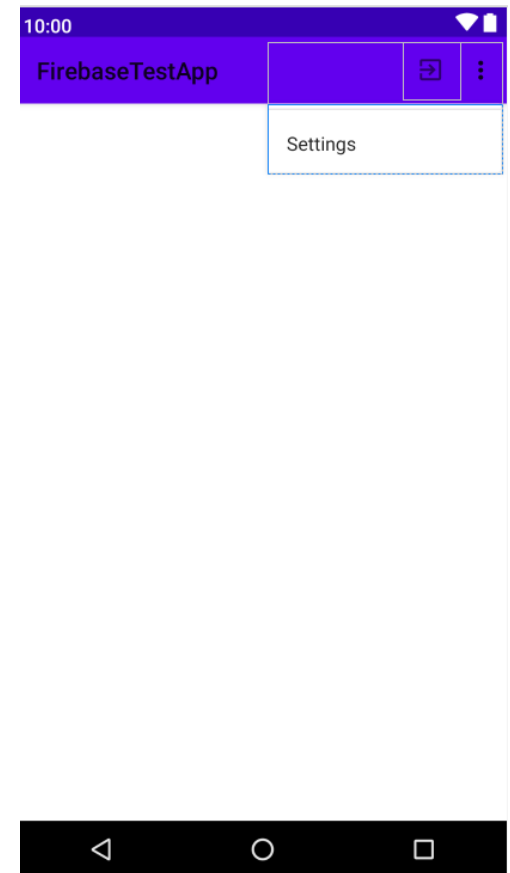


# Opciones de menú

- Al igual que un layout, el menú se puede editar en xml o con el diseñador

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:android="http://schemas.android.com/apk/res/android">

  <item
    android:id="@+id/menuLogout"
    android:icon="@drawable/ic_baseline_exit_to_app_24"
    android:title="@string/menulogout"
    app:showAsAction="ifRoom">
  />
  <item
    android:id="@+id/menuSettings"
    android:title="@string/menuConfiguracion" />
</menu>
```



# Opciones de Menú

- Desde el código es necesario hacer el inflate y luego reaccionar a los eventos de selección:

**@Override**

```
public boolean onCreateOptionsMenu(Menu menu){  
    getMenuInflater().inflate(R.menu.menu, menu);  
    return true;  
}
```

**@Override**

```
public boolean onOptionsItemSelected(MenuItem item){  
    int itemClicked = item.getItemId();  
    if(itemClicked == R.id.menuLogout){  
        mAuth.signOut();  
        Intent intent = new Intent(HomeUserActivity.this, MainActivity.class);  
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);  
        startActivity(intent);  
    }else if (itemClicked == R.id.menuSettings){  
        //Abrir actividad para configuración etc  
    }  
    return super.onOptionsItemSelected(item);  
}
```

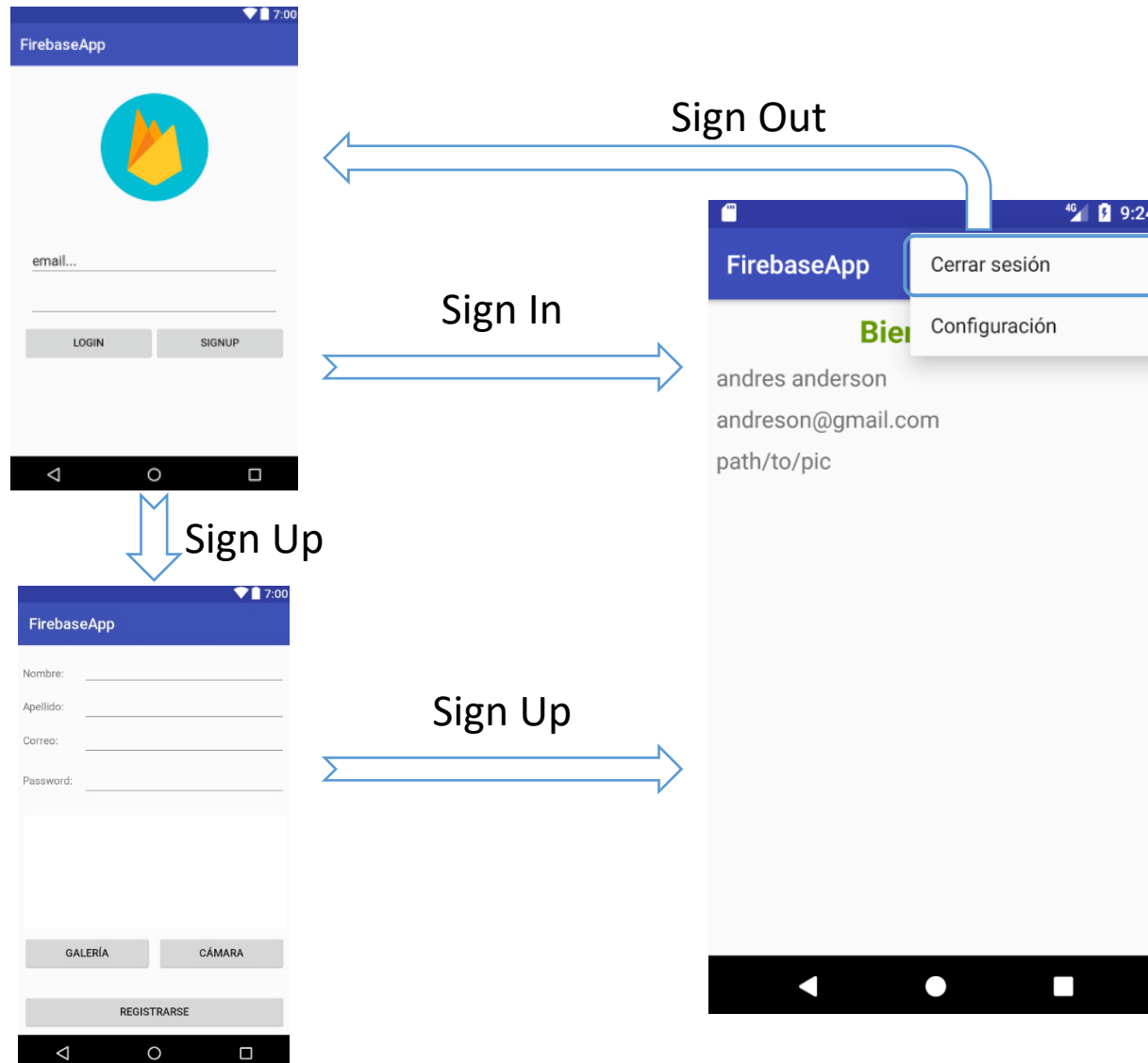
<https://developer.android.com/guide/topics/ui/menus.html>

# Registrar usuarios - Datos de perfil

- ¿Cómo se puede registrar un nuevo usuario desde la aplicación y no desde la consola web de firebase?

```
mAuth.createUserWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if(task.isSuccessful()){
                Log.d(TAG, "createUserWithEmail:onComplete:" + task.isSuccessful());
                FirebaseUser user = mAuth.getCurrentUser();
                if(user!=null){ //Update user Info
                    UserProfileChangeRequest.Builder upcrb = new UserProfileChangeRequest.Builder();
                    upcrb.setDisplayName(mUserName.getText().toString()+" "+mUserLastName.getText().toString());
                    upcrb.setPhotoUri(Uri.parse("path/to/pic")); //fake uri, use Firebase Storage
                    user.updateProfile(upcrb.build());
                    updateUI(user);
                }
            }
            if (!task.isSuccessful()) {
                Toast.makeText(SignUpActivity.this, R.string.auth_failed+ task.getException().toString(),
                    Toast.LENGTH_SHORT).show();
                Log.e(TAG, task.getException().getMessage());
            }
        }
    });
```

# Creación de usuarios



# Ejercicio 2

- Agregue una opción de menú a la actividad de usuario autenticado que le permita salir de la sesión.
- Desarrolle la funcionalidad para registrar nuevos usuarios en el botón sign up.
  - Para esto, lance una nueva actividad que recopile la información básica del usuario, nombre, correo, password y fotografía como se muestra en la figura.
  - Para agregar la imagen del usuario, programe la actividad de tal forma que permita seleccionar la imagen de una galería o tomar una fotografía utilizando la cámara del dispositivo. Por ahora sólo muestre la imagen en la actividad y envíe una URI de prueba en la actualización del perfil.
- Una vez recopilada la información registre el usuario en firebase y si el registro es exitoso lance la actividad de bienvenida, en donde se muestre la información del usuario.

FirebaseApp

Nombre: \_\_\_\_\_

Apellido: \_\_\_\_\_

Correo: \_\_\_\_\_

Password: \_\_\_\_\_

GALERÍA

CÁMARA

REGISTRARSE

# Realtime Database

Firebase

# Firestore Databases

- **Realtime Database** is Firebase's original database. It's an efficient, low-latency solution for mobile apps that require synced states across clients in realtime.
- **Cloud Firestore** is Firebase's new flagship database for mobile app development. It improves on the successes of the Realtime Database with a new, more intuitive data model. Cloud Firestore also features richer, faster queries and scales better than the Realtime Database.

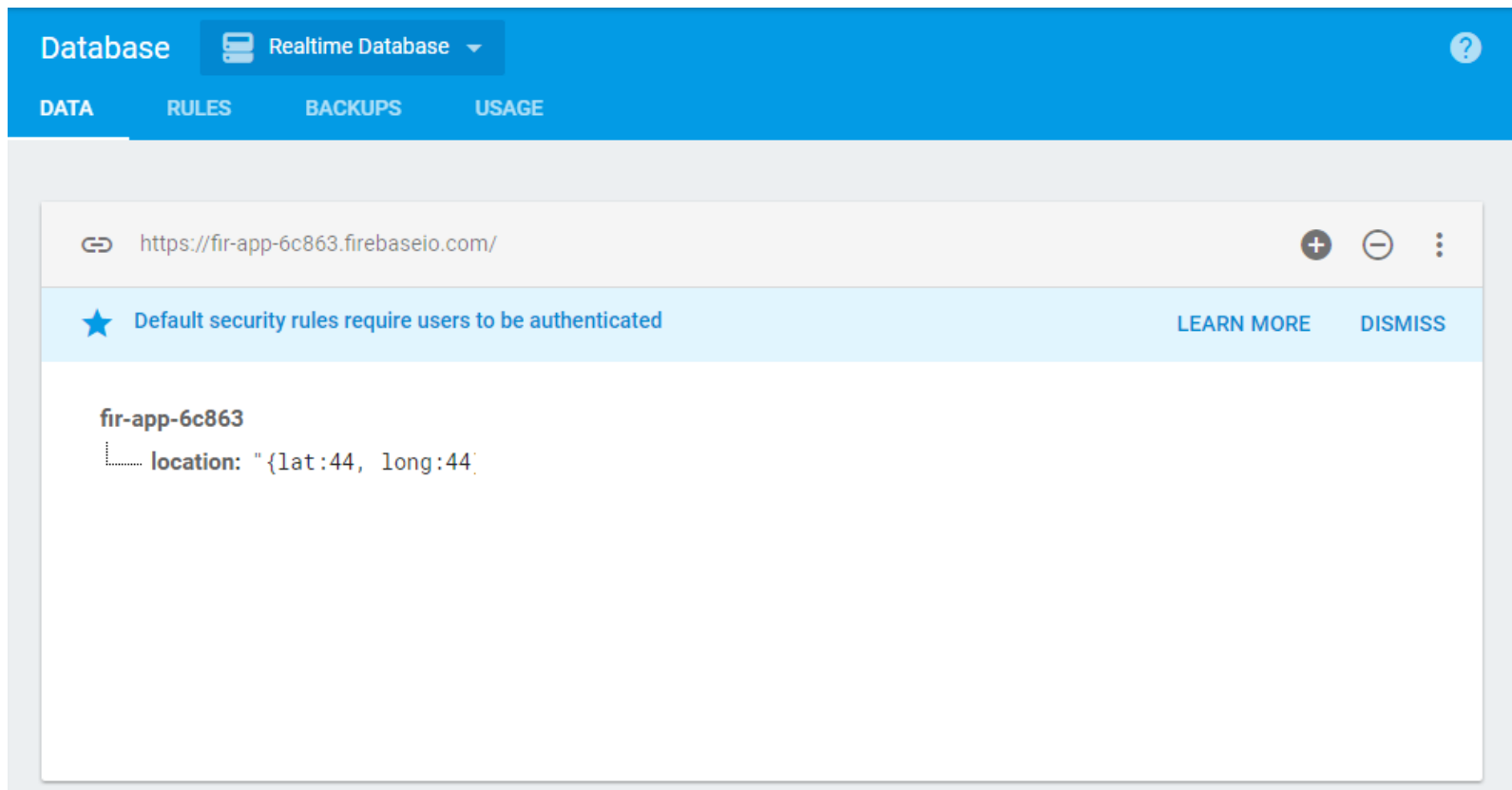
Realtime Database	Cloud Firestore
<p>Stores data as one large JSON tree.</p> <ul style="list-style-type: none"><li>• Simple data is very easy to store.</li><li>• Complex, hierarchical data is harder to organize at scale.</li></ul> <p>Learn more about the <a href="#">Realtime Database data model</a>.</p>	<p>Stores data in documents organized in collections.</p> <ul style="list-style-type: none"><li>• Simple data is easy to store in documents, which are very similar to JSON.</li><li>• Complex, hierarchical data is easier to organize at scale, using subcollections within documents.</li><li>• Requires less denormalization and data flattening.</li></ul> <p>Learn more about the <a href="#">Cloud Firestore data model</a>.</p>

<https://firebase.google.com/docs/database/rtdb-vs-firestore>



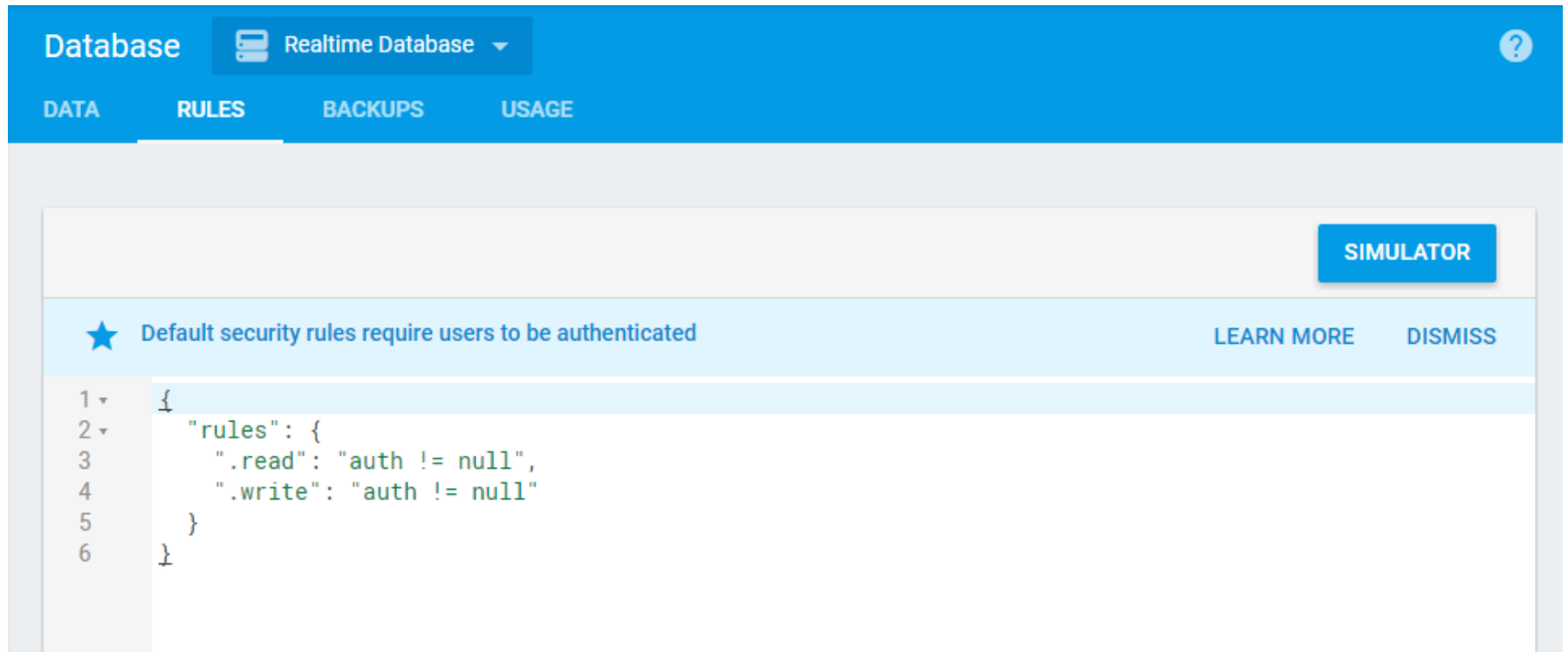
# Firestore Realtime Database

- Base de datos documental (basada en JSON) dinámica



# Firestore Realtime Database

- Por defecto, sólo usuarios autenticados pueden leer y escribir datos. Se pueden modificar las reglas para permitir hacer cambios a cualquier usuario, pero no se recomienda dejar sin protección los datos.

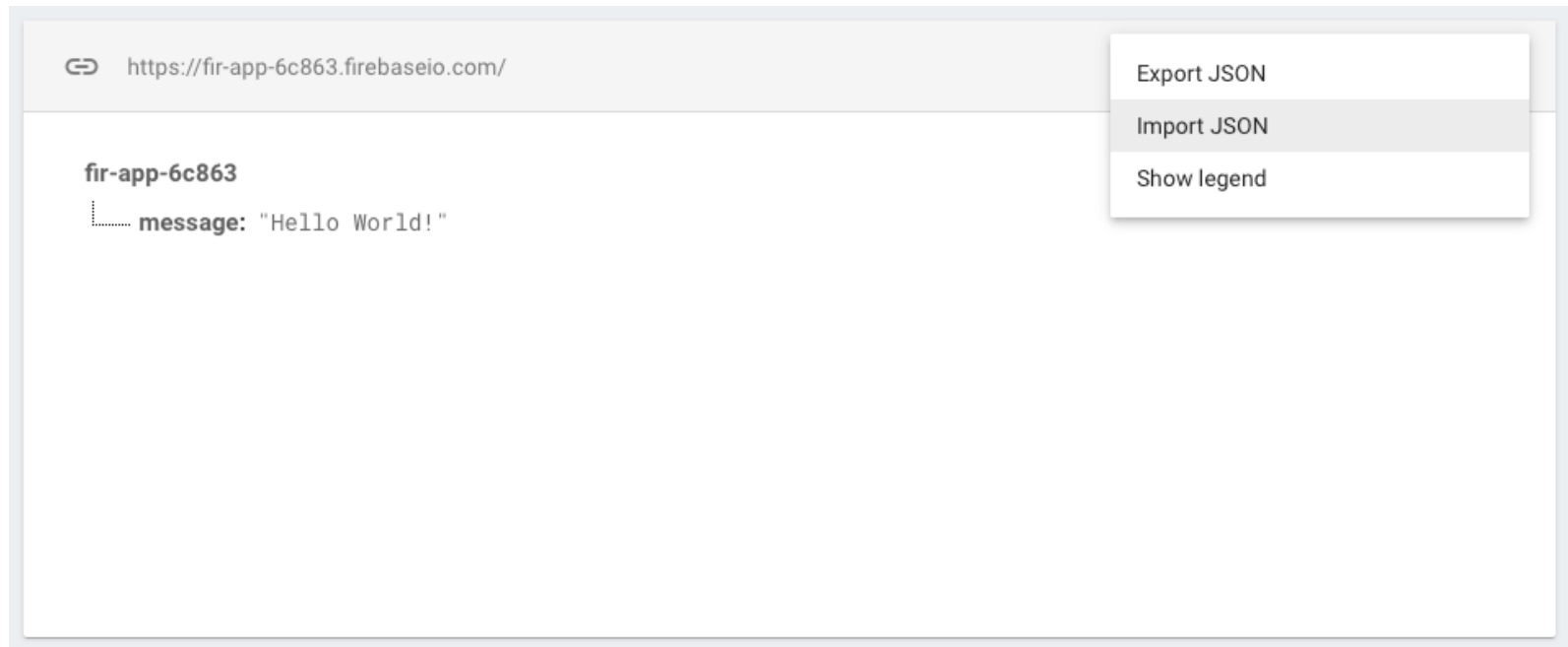


The screenshot shows the Firebase Realtime Database Rules editor. The top navigation bar is blue with the word "Database" and a dropdown menu currently set to "Realtime Database". Below this are four tabs: "DATA", "RULES" (which is selected), "BACKUPS", and "USAGE". On the right side of the top bar is a help icon (a question mark in a circle). Below the tabs, there is a "SIMULATOR" button. A light blue banner with a star icon on the left contains the text "Default security rules require users to be authenticated". To the right of this banner are two links: "LEARN MORE" and "DISMISS". Below the banner is a code editor with a line number margin on the left (lines 1 to 6). The code in the editor is as follows:

```
1 {  
2   "rules": {  
3     ".read": "auth != null",  
4     ".write": "auth != null"  
5   }  
6 }
```

# Inicializar la base de datos a partir de JSON

- Los datos en Firebase se pueden inicializar a partir de un documento JSON. También es posible exportar un documento con el contenido de la base de datos.



# Conectar la aplicación con firebase

- Seguir el procedimiento del asistente (ya debe estar conectado si se hizo el proceso con la autenticación). Luego agregar las dependencias. También se pueden agregar de forma manual a través de Gradle.

## Save and retrieve data

Our cloud database stays synced to all connected clients in realtime and remains available when your app goes offline. Data is stored in a JSON tree structure rather than a table, eliminating the need for complex SQL queries.

[Launch in browser](#)

### 1 Connect your app to Firebase

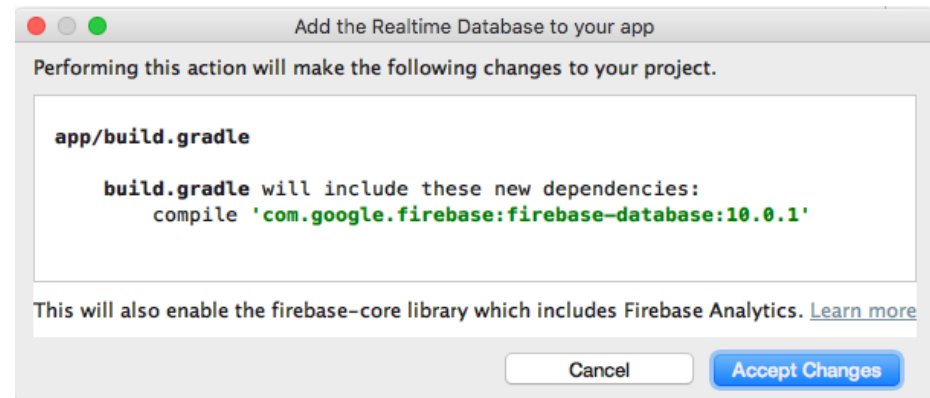
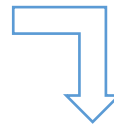
ON **Connected**

### 2 Add the Realtime Database to your app

Add the Realtime Database to your app

### 3 Configure Firebase Database Rules

The Realtime Database provides a declarative rules language that allows you to define how your data should be structured, how it should be indexed, and when your data can be read from and written to. By default, read and write access to your database is restricted so only authenticated users can read or write data. To get started without setting up [Authentication](#), you can [configure your rules for public access](#). This does make your database open to anyone, even people not using your app, so be sure to restrict your database again when you set up authentication.



# Escribir datos en Firebase

- Se definen dos atributos de Firebase Database y se inicializan:

*//Atributos*

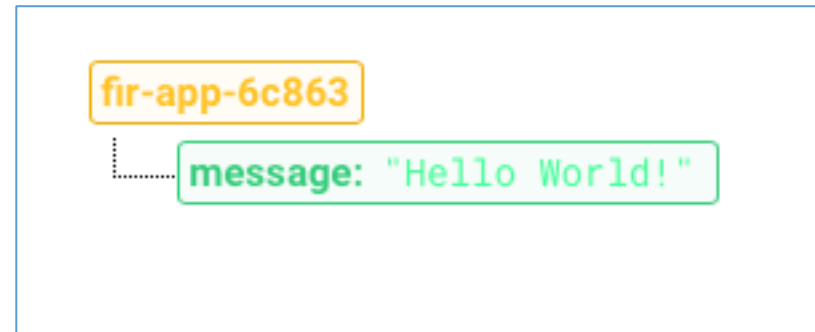
```
FirebaseDatabase database;  
DatabaseReference myRef;
```

*//Inicialización en onCreate()*

```
database = FirebaseDatabase.getInstance();
```

*//Escribir un dato cuando se pulse un botón*

```
boton.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        myRef = database.getReference("message");  
        myRef.setValue("Hello World!");  
    }  
});
```



*En la consola de Firebase*

# Escribir datos en Firebase

*You can save a range of data types to the database this way, including Java objects. When you save an object the responses from any getters will be saved as children of this location.*

# Escribir datos en Firebase

- Basic write operations
- For basic write operations, you can use `setValue()` to save data to a specified reference, replacing any existing data at that path. You can use this method to:
- Pass types that correspond to the available JSON types as follows:
  - String
  - Long
  - Double
  - Boolean
  - Map<String, Object>
  - List<Object>
- Pass a custom Java object, if the class that defines it has a default constructor that takes no arguments and has public getters for the properties to be assigned.

<https://firebase.google.com/docs/database/android/read-and-write>

# Escribir objetos en Firebase

- Definir un objeto POJO, con getters públicos, y un constructor sin argumentos.

```
public class MyUser {  
    String name; //...}
```

- Definir un path para todos los usuarios de la aplicación

*//Root path of every user in FB*

```
public static final String PATH_USERS="users/";
```

- Una vez los datos esten completos, persistir el objeto

```
MyUser myUser = new MyUser();  
myUser.setName("Andrés");  
myUser.setLastName("Acevedo");  
myUser.setAge(30);  
myUser.setHeight(1.80);  
myUser.setWeight(80);  
myRef=database.getReference(PATH_USERS+user.getId());  
myRef.setValue(myUser);
```



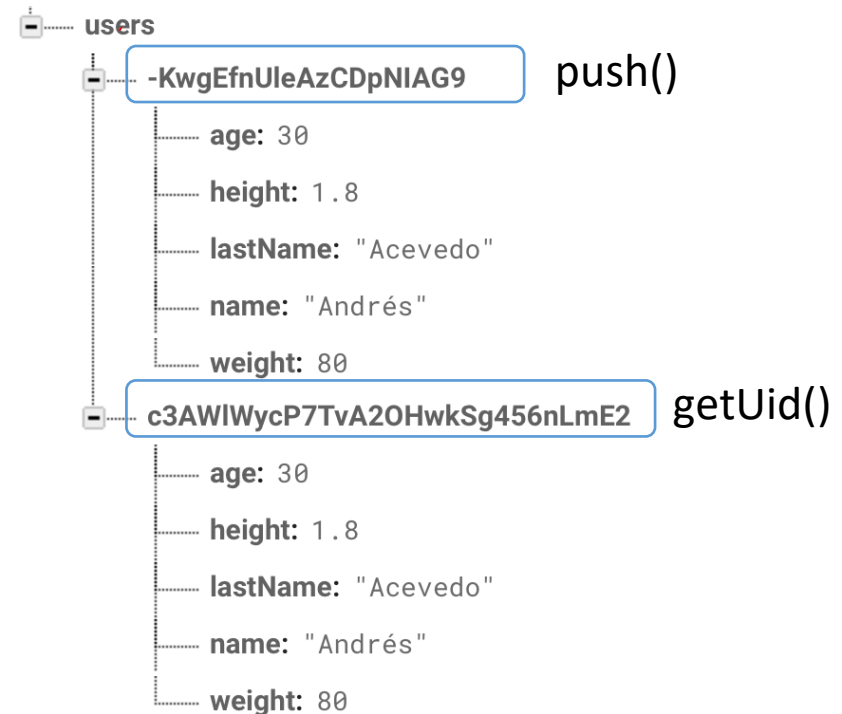


# Creación de índices

- Se puede dejar a Firebase la tarea de generar índices para objetos que se quieran persistir usando el método push:

```
String key = myRef.push().getKey();  
myRef=database.getReference(PATH_USERS+key);  
myRef.setValue(myUser);
```

Key contiene un nuevo identificador,  
luego se guarda el objeto dentro del  
path de usuarios con este índice



# Ejercicio 3

- Cree una aplicación que permita ingresar los datos un usuario como nombre, apellido, y edad. Para esto es necesario que:
  - Cree una (o varias) clases, para representar los datos de un usuario, para escribirlos como JSON en la base de datos. Recuerde que los datos a almacenar dependen de los getters de cada atributo
  - Defina un Path en su base de datos que almacenara todos los datos de los usuarios usando un valor constante estático.
  - Escriba los datos del usuario dentro del path del item anterior, y concatene el id generado por Firebase Authentication en el momento de la creación del usuario. Este ID está disponible en cualquier actividad del **usuario autenticado** usando: `user.getId();`
  - Pruebe la aplicación registrando varios usuarios

# Consulta vs suscripción

- El comportamiento típico de la base de datos dinámica de Firebase es suscribirse a todos los cambios que haya para algún dato (Path) en particular.
- A diferencia de SQL, se recibe un evento cuando algún dato al interior del path especificado cambia de valor usando el evento `addValueEventListener()`
- El evento se recibe en todos los dispositivos que corran la aplicación, y es multiplataforma (e.g., IOS, Ionic, etc.)
- Si sólo es necesario leer una vez los datos, es posible hacerlo con una consulta única a través del evento `addListenerForSingleValueEvent()`

# Leer datos (una vez)

```
public void loadUsers() {  
    myRef = database.getReference(PATH_USERS);  
    myRef.addListenerForSingleValueEvent(new ValueEventListener() {  
        @Override  
        public void onDataChange(DataSnapshot dataSnapshot) {  
            for (DataSnapshot singleSnapshot : dataSnapshot.getChildren()) {  
                MyUser myUser = singleSnapshot.getValue(MyUser.class);  
                Log.i(TAG, "Encontró usuario: " + myUser.getName());  
                String name = myUser.getName();  
                int age = myUser.getAge();  
                Toast.makeText(MapHomeActivity.this, name + ":" + age, Toast.LENGTH_SHORT).show();  
            }  
        }  
    });  
    @Override  
    public void onCancelled(DatabaseError databaseError) {  
        Log.w(TAG, "error en la consulta", databaseError.toException());  
    }  
};  
}
```

# Suscripción a cambios

```
public void loadUsers() {  
    myRef = database.getReference(PATH_USERS);  
    myRef.addValueEventListener(new ValueEventListener() {  
        @Override  
        public void onDataChange(DataSnapshot dataSnapshot) {  
            for (DataSnapshot singleSnapshot : dataSnapshot.getChildren()) {  
                MyUser myUser = singleSnapshot.getValue(MyUser.class);  
                Log.i(TAG, "Encontró usuario: " + myUser.getName());  
                String name = myUser.getName();  
                int age = myUser.getAge();  
                Toast.makeText(MapHomeActivity.this, name + ":" + age, Toast.LENGTH_SHORT).show();  
            }  
        }  
        @Override  
        public void onCancelled(DatabaseError databaseError) {  
            Log.w(TAG, "error en la consulta", databaseError.toException());  
        }  
    });  
}
```

# Remover suscripción

- Callbacks are removed by calling the *removeEventListener()* method on your Firebase database reference.
- If a listener has been added multiple times to a data location, it is called multiple times for each event, and you must detach it the same number of times to remove it completely.
- Calling *removeEventListener()* on a parent listener does not automatically remove listeners registered on its child nodes; *removeEventListener()* must also be called on any child listeners to remove the callback.

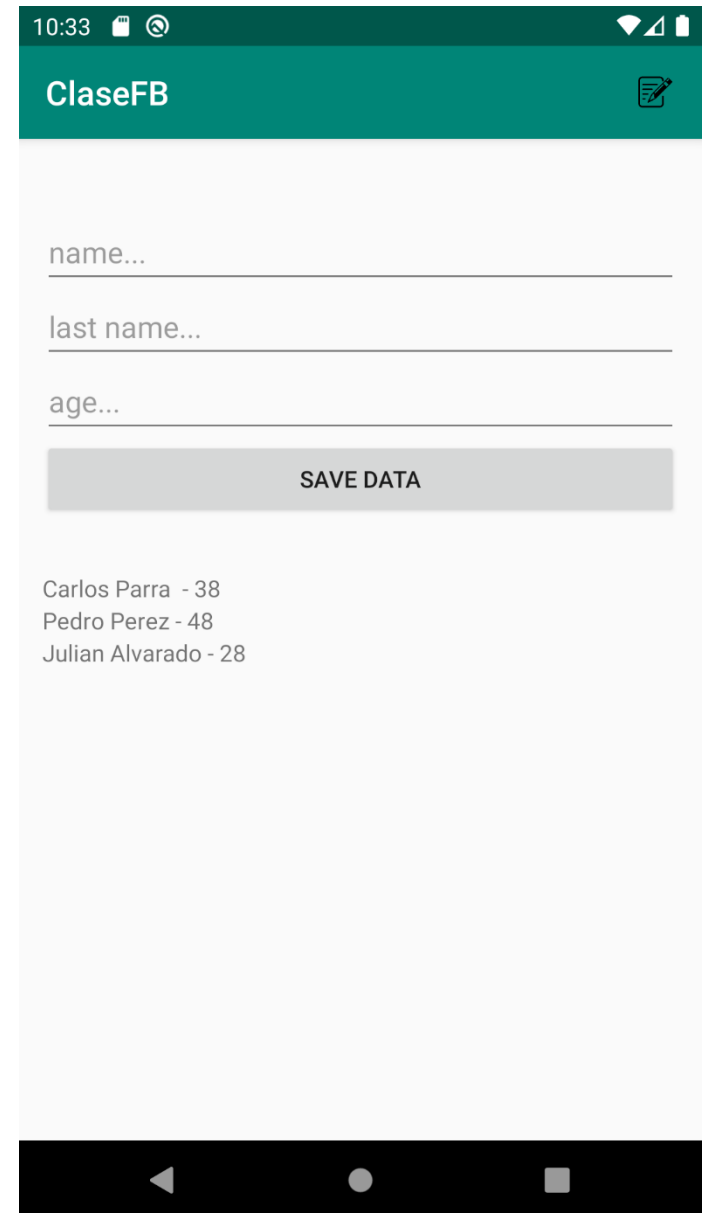
# Recomendaciones de almacenamiento

- La base de datos es un árbol en JSON, esto trae algunas implicaciones que requieren un diseño diferente al relacional:
  - Evitar anidamientos de datos, usar índices para identificar información relacionada.
  - Definir una estructura de datos tan plana como sea posible.
  - Duplicar datos para mejorar rendimiento en relaciones bidireccionales.

<https://firebase.google.com/docs/database/android/structure-data>

# Demo!

- Aplicación para la creación y lectura de usuarios en Firebase
  1. Crear Layout
  2. Programar botón para salvar datos básicos
  3. Crear objeto de usuario y persistirlo
  4. Consultar una vez
  5. Suscribirse a cambios

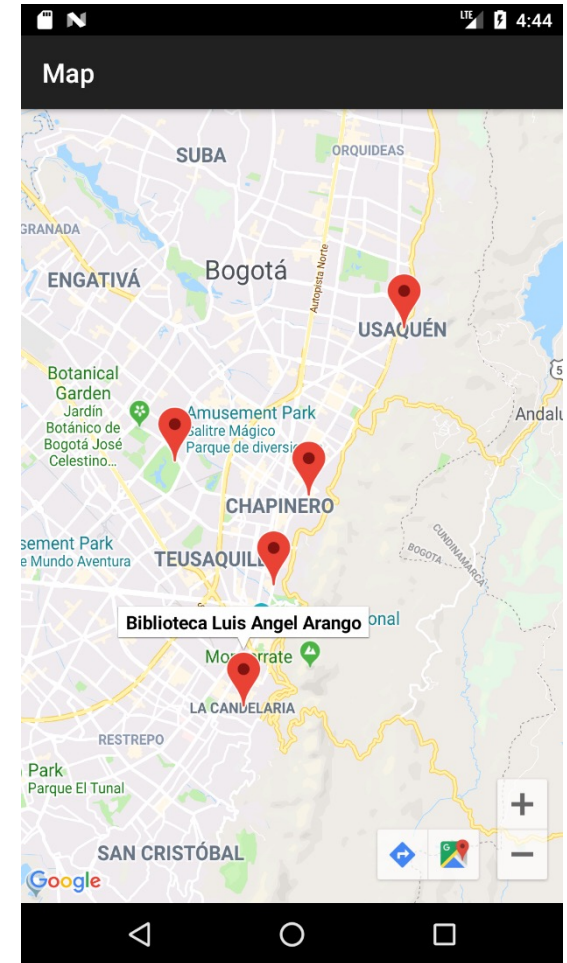


The screenshot shows a mobile application interface for 'ClaseFB'. At the top, there is a teal header bar with the text 'ClaseFB' and a small icon of a notepad and pencil. Below the header, there are three text input fields labeled 'name...', 'last name...', and 'age...'. A grey button labeled 'SAVE DATA' is positioned below the input fields. At the bottom of the screen, there is a list of user data: 'Carlos Parra - 38', 'Pedro Perez - 48', and 'Julian Alvarado - 28'. The status bar at the very top shows the time as 10:33 and various system icons. The bottom of the screen features a black navigation bar with standard Android navigation icons.



# Ejercicio 4

- Utilice el archivo JSON disponible en BrightSpace e impórtelo en su base de datos de Firebase desde la consola web.
- Construya una actividad con un mapa. Puede usar la misma actividad del taller 2.
- Utilizando Firebase, consulte **una sólo vez** las localizaciones guardadas en la base de datos. Para cada localización, cree marcadores con un ícono personalizado y muéstrellos en un mapa.
  - Para cargar las localizaciones, defina una clase que tenga los atributos y los tipos de cada localización en el archivo JSON, de esta manera puede cargar los objetos usando:  
`DataSnapshot.getValue(ClaseLocalizacion.class)`



# Ejercicio 5

- Agregue al usuario definido en el ejercicio 3 la información de su localización (latitud y longitud).
- Recupere la ubicación actual del usuario suministrada por el GPS del dispositivo y guárdela en Firebase, de acuerdo al intervalo definido.
- Modifique la aplicación del ejercicio 2 para que además de las posiciones cargadas desde el archivo JSON, se cree un marcador para cada usuario autenticado en la aplicación.
  - Para esto debe leer los datos del path que contiene todas las localizaciones de usuarios en modo suscripción a cambios.
  - Pruebe la aplicación con varios usuarios en varios dispositivos y cambie los valores del emulador (*i.e.* simule un cambio de posición geográfica del dispositivo) para ver el movimiento del marcador en el mapa.

# Storage

Firebase

# Firebase Storage

- Storage


## Set up Cloud Storage

1 Secure rules for Cloud Storage

2 Set Cloud Storage location

By default, your rules allow all reads and writes from authenticated users.

After you define your data structure, **you will need to write rules to secure your data.**

[Learn more](#) 

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

CancelNext

# Storage

- Desde el asistente de Android Studio

① Connect your app to Firebase

✓ Connected

② Add Cloud Storage to your app

Add Cloud Storage to your app

③ Create a reference

Declare a [StorageReference](#) and initialize it in the [onCreate](#) method.

```
private StorageReference mStorageRef;
```

```
mStorageRef = FirebaseStorage.getInstance().getReference
```

# Upload a file

```
private void uploadFile(){
    Uri file = Uri.fromFile(new File("path/to/images/image.jpg"));
    StorageReference imageRef = mStorageRef.child("images/profile/userid/image.jpg");

    imageRef.putFile(file)
        .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                // Get a URL to the uploaded content
                Log.i("FBApp", "Successfully upload image");
            }
        })
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception exception) {
                // Handle unsuccessful uploads
                // ...
            }
        });
}
```

<https://firebase.google.com/docs/storage/android/start>

# Download a File

```
private void downloadFile() throws IOException {
    File localFile = File.createTempFile("images", "jpg");
    StorageReference imageRef = mStorageRef.child("images/image.jpg");
    imageRef.getFile(localFile)
        .addOnSuccessListener(new OnSuccessListener<FileDownloadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {
                // Successfully downloaded data to local file
                // ...
                Log.i("FBApp", "successfully downloaded");
                //UpdateUI using the localFile
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception exception) {
                // Handle failed download
                // ...
            }
        });
}
```