

.:: Login con Cuenta de Google y Firebase en Android Studio con Java ::.

-= By Surflaweb =-



12 de Octubre del 2020.

Youtube: <https://bit.ly/30XnPyJ>

Github: <https://github.com/alcarazolabs>



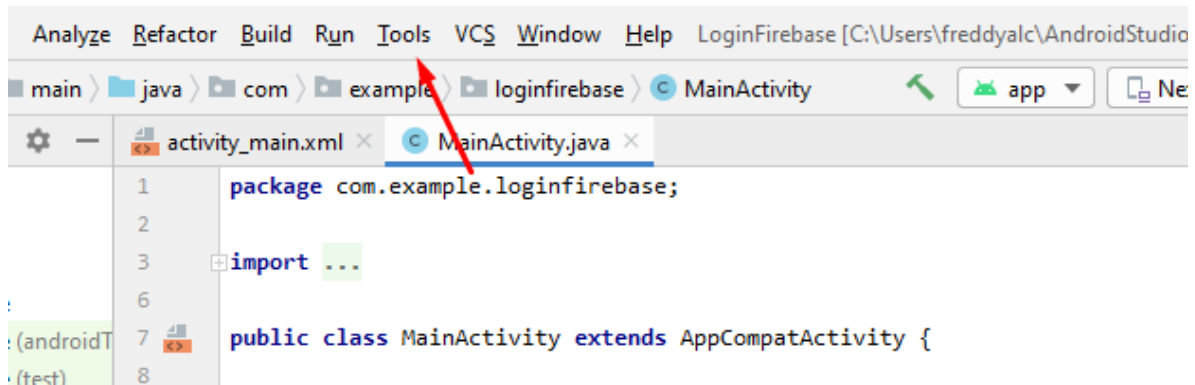
Presentación:

En esta guía, manual, bitácora como lo quisiéramos llamar se creó con el objetivo de hacer inicio de sesión mediante una cuenta de Google en una aplicación Android. Vamos a hacer **sign in/iniciar sesión, log out/cerrar sesión y eliminar cuenta**. Finalmente se tendrá en la base de datos firebase todos los usuarios autenticados mediante su cuenta de Google en su aplicación.

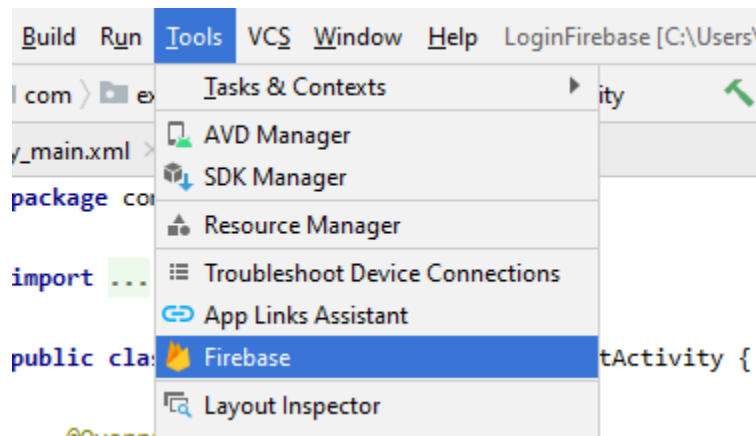
Configurar Firebase en el proyecto.

Nos olvidamos de como estará diseñada la aplicación, solo vamos a usar un botón para que el usuario le de click y se habrá un menú y el usuario elija su cuenta de Google. No se preocupen del diseño. Primero configuremos nuestro proyecto firebase.

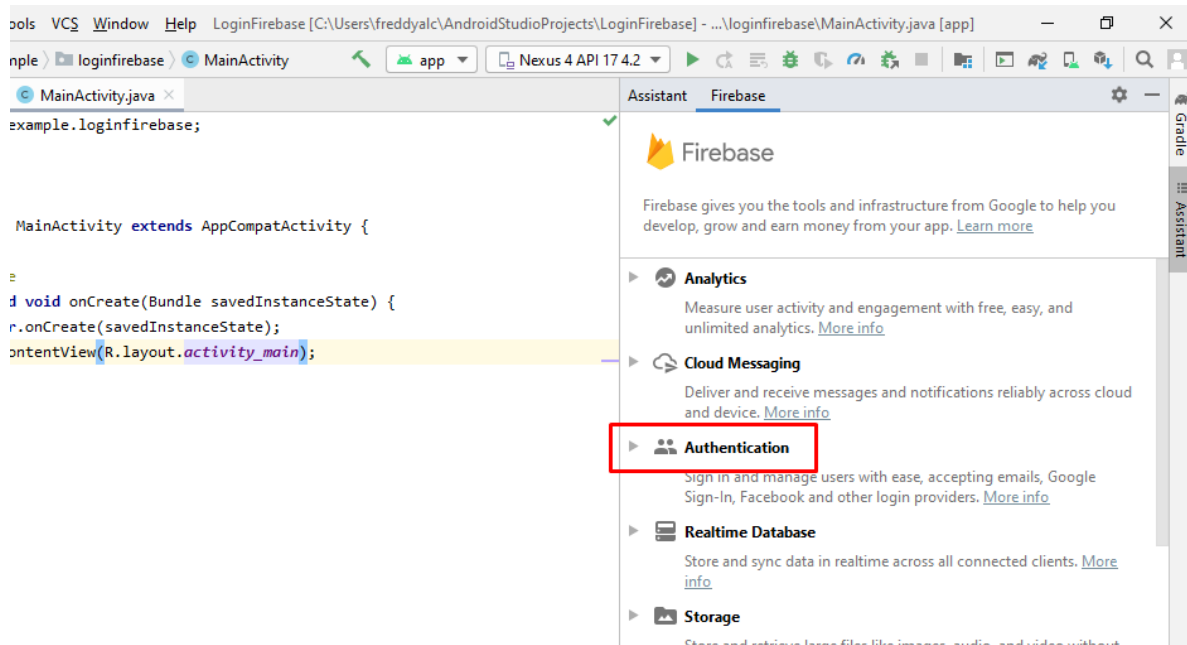
Creada la aplicación/proyecto android “firebaseLogin” vamos a irnos a la opción de “tools” de Android studio.



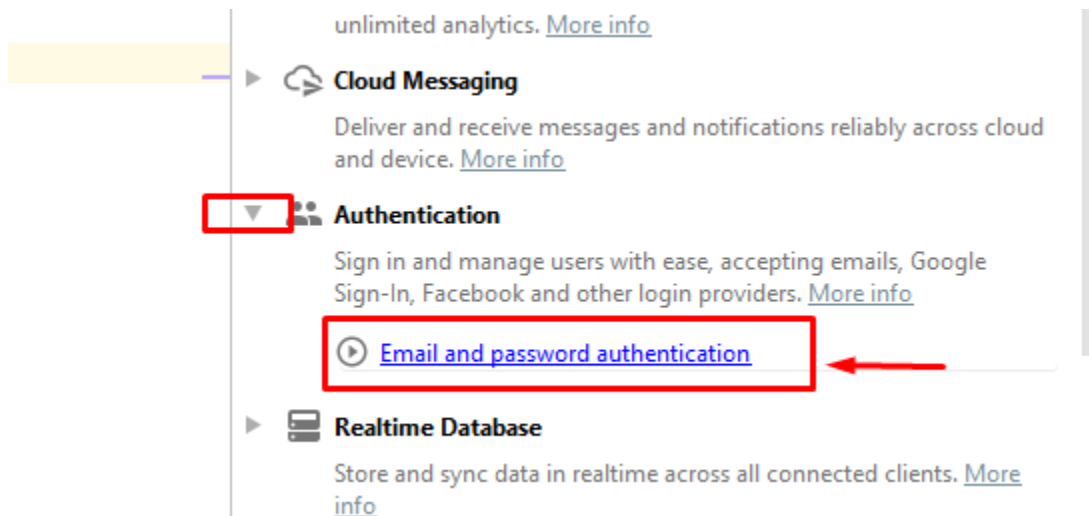
Ahora elegimos “Firebase”:



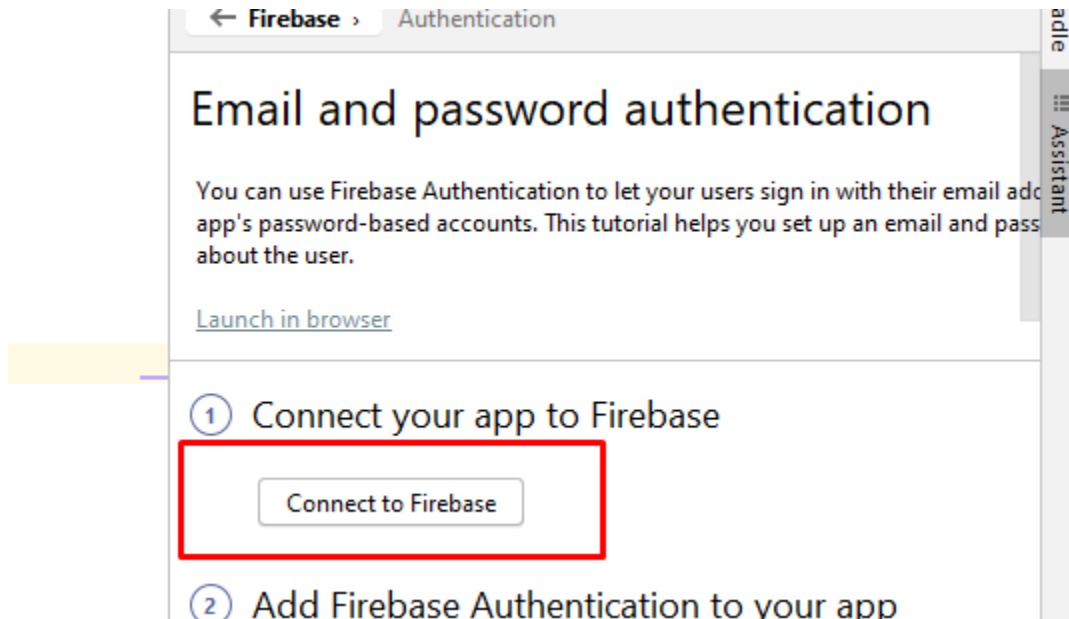
Ahora se nos abre un menú en la parte derecha de Android studio, elegimos “Autenticación”:



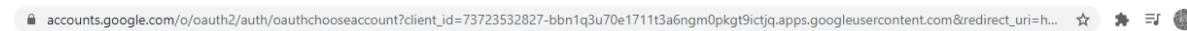
Hacer click en “Email and password authentication”;



Luego en “Connect to Firebase”:



Luego nos lleva a la siguiente página ahí debemos de “iniciar sesión” con una cuenta de Google que usará firebase.



Luego le damos en “Permitir”:

siguiente.

- Ve a y administre sus datos en los servicios de Google Cloud Platform. ⓘ
- Permite ver y administrar tus aplicaciones implementadas en Google App Engine. ⓘ
- Ve a y administre sus tareas en Actions on Google. ⓘ
- 🔥 Ver y administrar todos tus datos y opciones de configuración de Firebase ⓘ

Asegúrate de que Android Studio sea de confianza

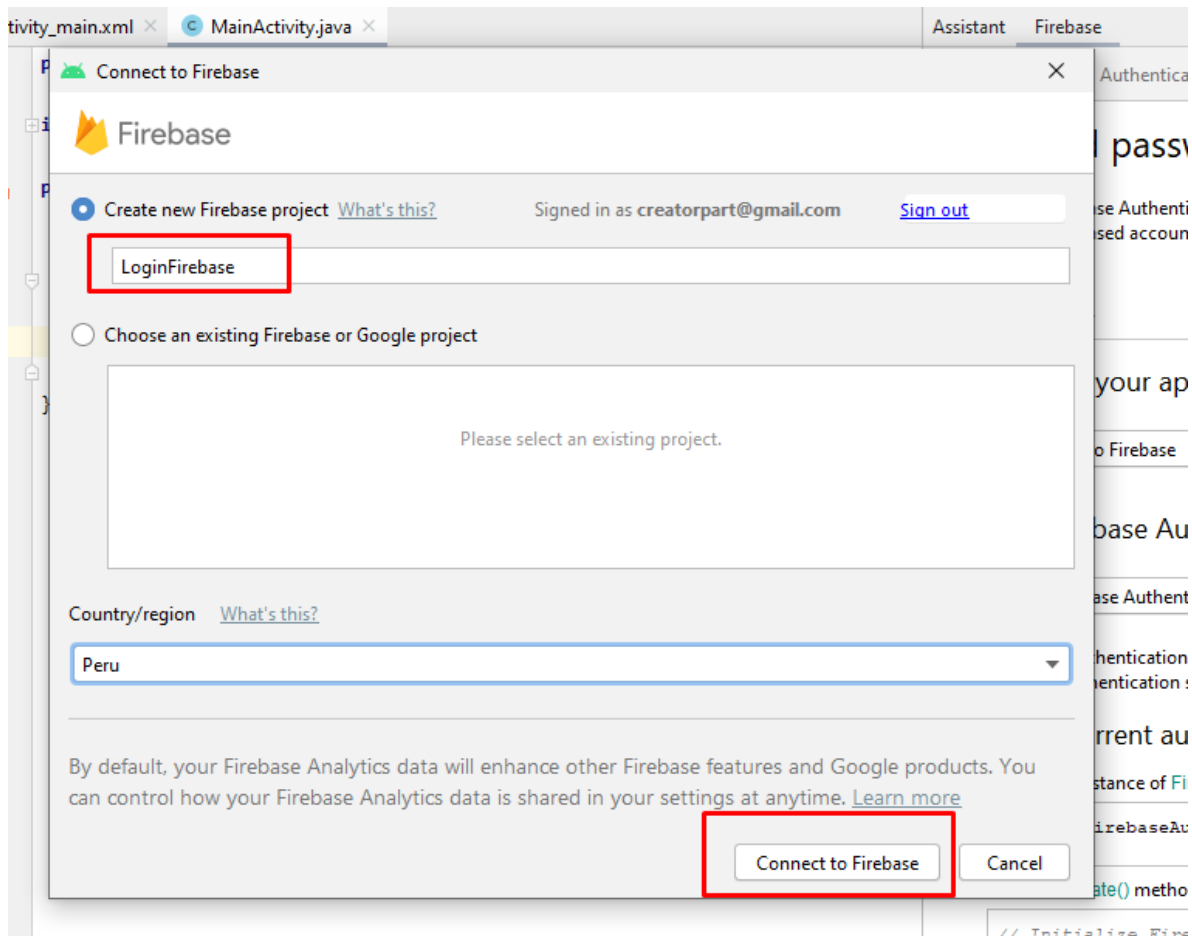
Es posible que compartas información confidencial con este sitio o app. Para obtener más información sobre la forma en que Android Studio administrará tus datos, consulta sus condiciones del servicio y políticas de privacidad. Puedes ver o quitar el acceso a través de tu [Cuenta de Google](#) en cualquier momento.

[Más información sobre los riesgos](#)

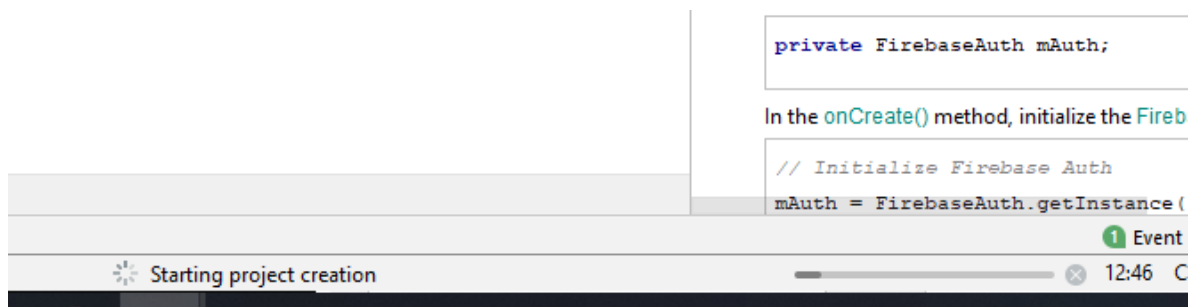
[Cancelar](#)

[Permitir](#)

Luego en Android Studio nos saldrá la siguiente ventana la cual creará el proyecto firebase:

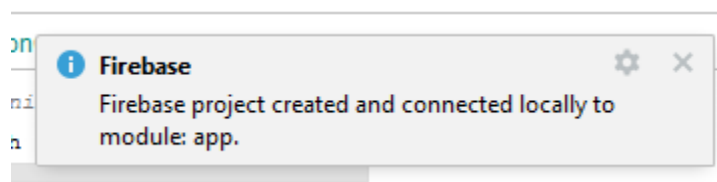


Luego en la parte inferior de Android studio veremos lo siguiente:



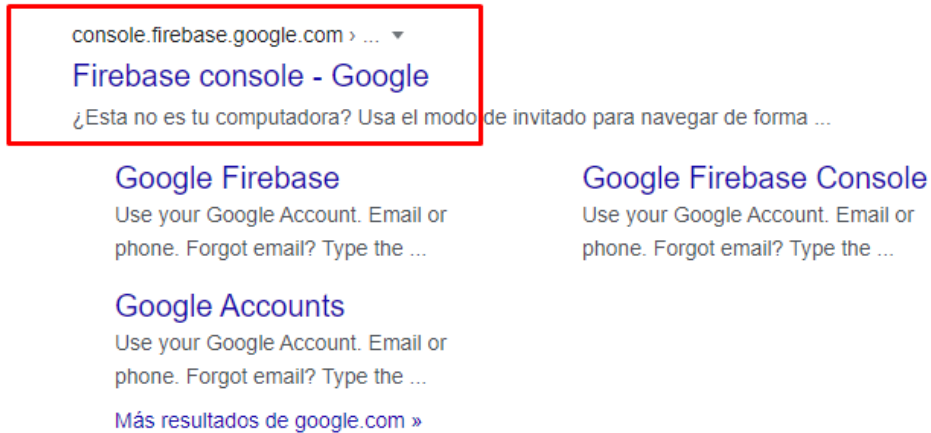
Eso significa que el proyecto “Login Firebase” se está creando en firebase.

Al final veremos un mensaje:

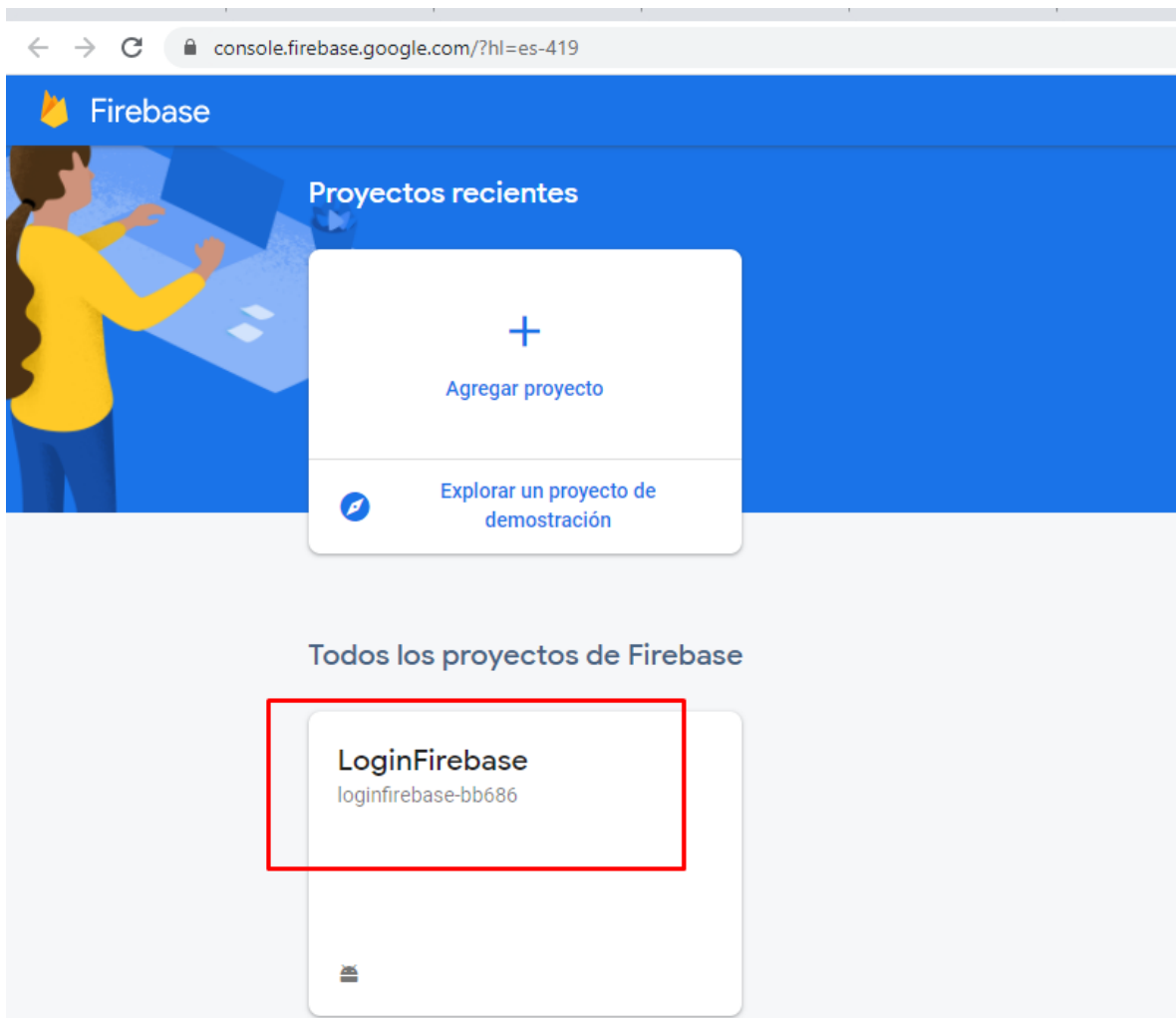


Ahora vamos a verificar en la consola de firebase:

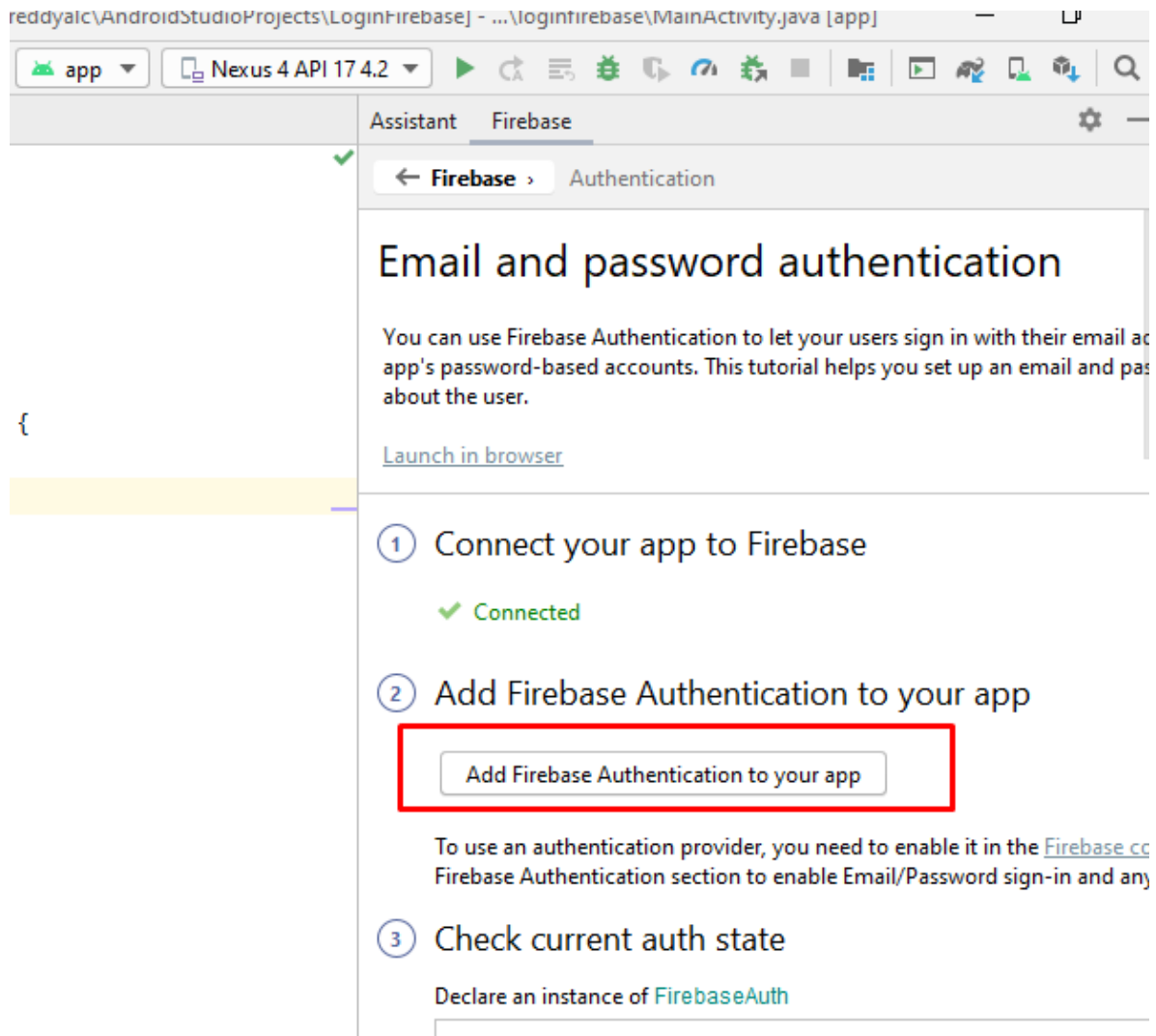
<https://console.firebase.google.com/?hl=es-419>



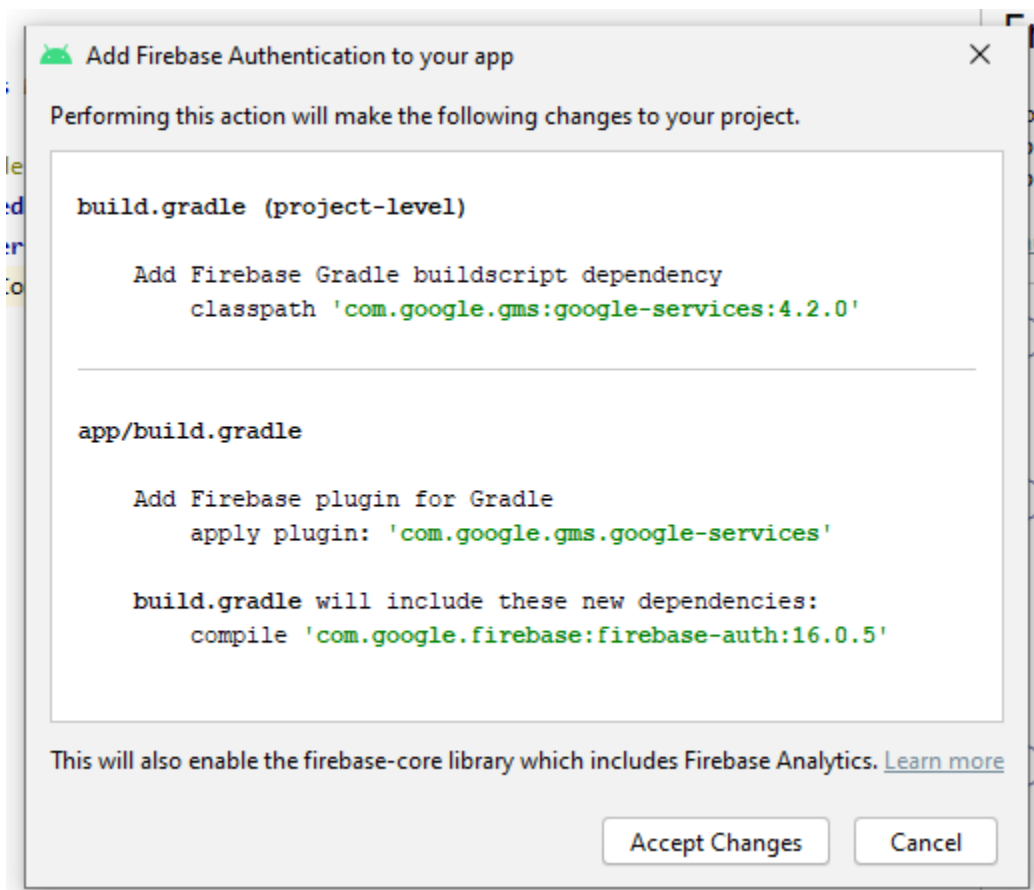
En efecto vemos el proyecto creado:



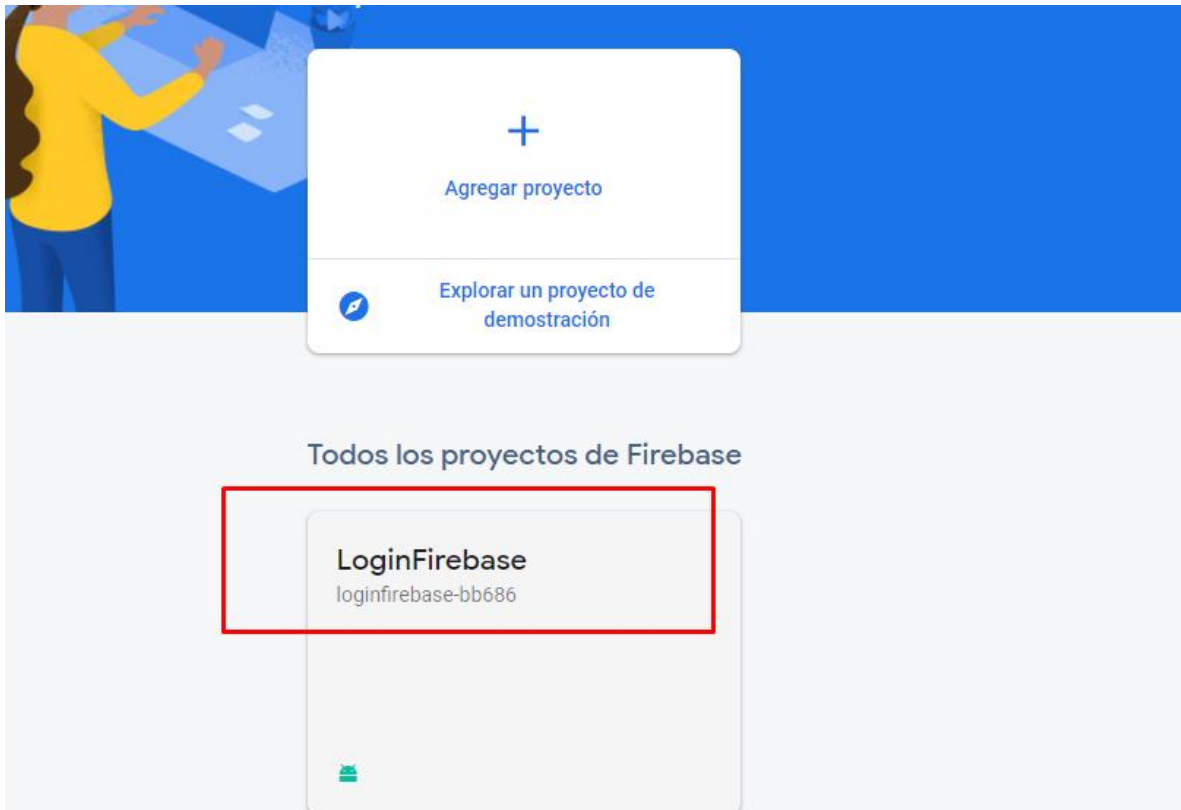
Ahora volvemos a android studio y agregaremos las dependencias de autenticación de firebase:



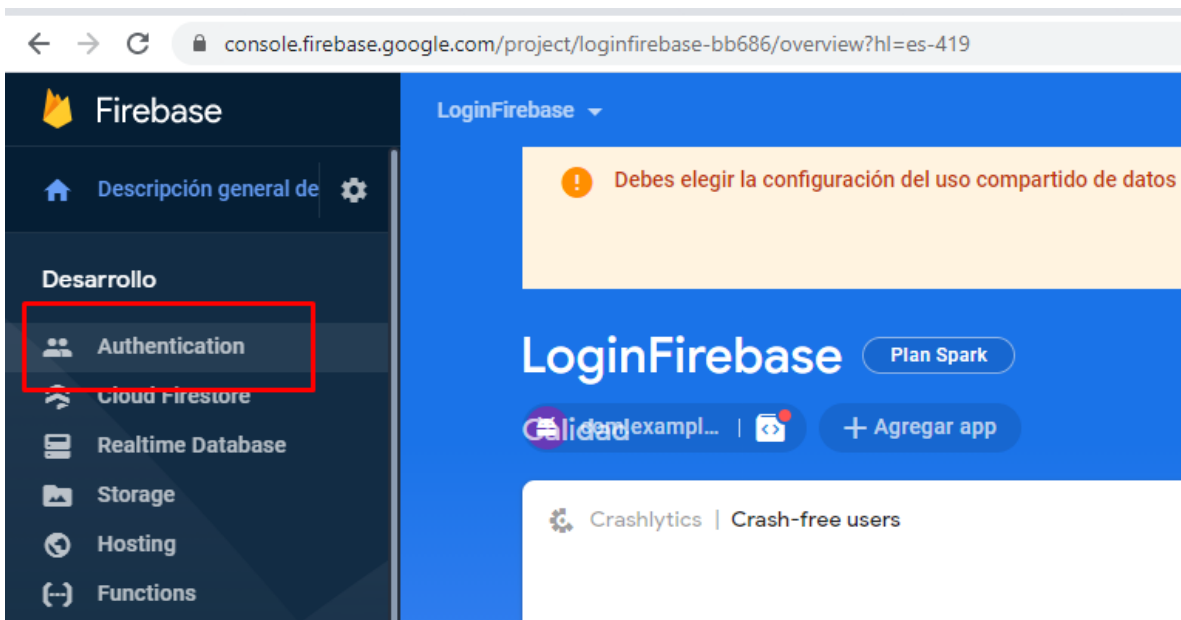
Aceptamos los cambios:



Ahora en la consola firebase necesitamos de habilitar la autenticación mediante Google, para eso abrimos el proyecto en la consola firebase:










Luego nos vamos a “autenticación”:



Luego a “Sign-in method” o método de acceso:

Elegimos Google:

Proveedores de acceso	
Proveedor	Estado
 Correo electrónico/contraseña	Inhabilitado
 Teléfono	Inhabilitado
 Google	Inhabilitado
 Play Juegos	Inhabilitado
 Game Center	Inhabilitado
 Facebook	Inhabilitado
 Twitter	Inhabilitado

☒ Habilitar

El acceso de Google está configurado automáticamente en tus apps web y de iOS conectadas. Para configurar el acceso de Google en tus apps de Android, debes agregar la [huella digital SHA1](#) para cada app en la [Configuración de proyecto](#).

⚙️ Actualiza la siguiente [configuración de nivel de proyecto](#) para continuar

Nombre público del proyecto ⓘ

project-287268525790

Correo electrónico de asistencia del proyecto ⓘ

██████████@gmail.com ▼

Incluir ID de clientes de proyectos externos en la lista de elementos seguros (opcional) ⓘ ▼

Configuración de SDK web ⓘ ▼

Cancelar

Guardar

Luego nos vamos a “Configuración del proyecto”:

← → ↻ console.firebase.google.com/project/loginfirebase-bb686/authentication/providers?hl=es-419

Firestore

Descripción general de **Configuración del proyecto**

Usuarios y permisos

Uso y facturación

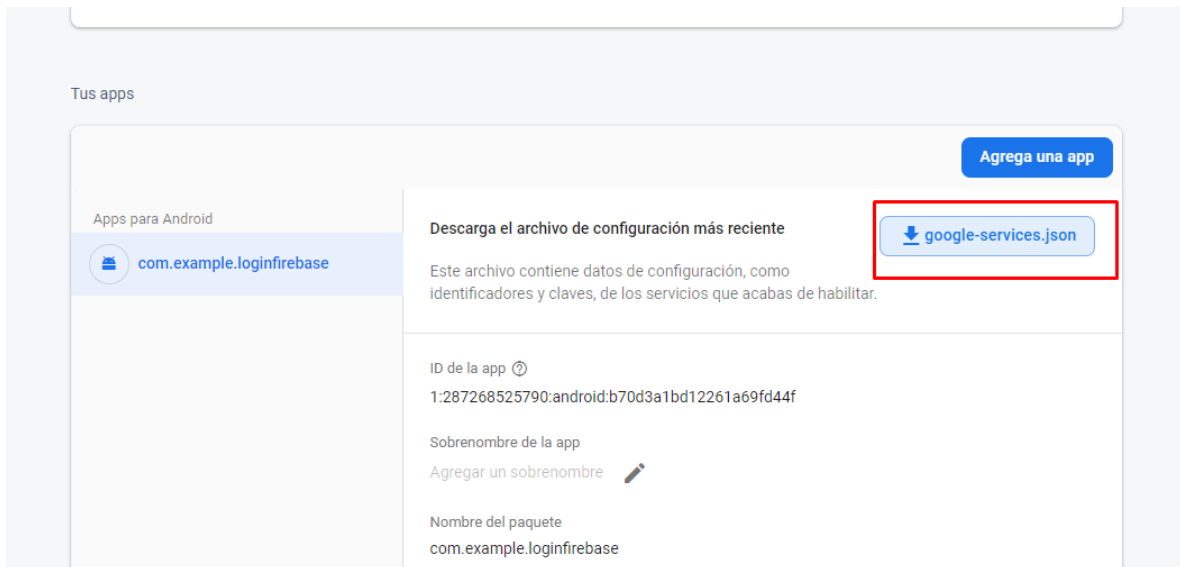
Desarrollo

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

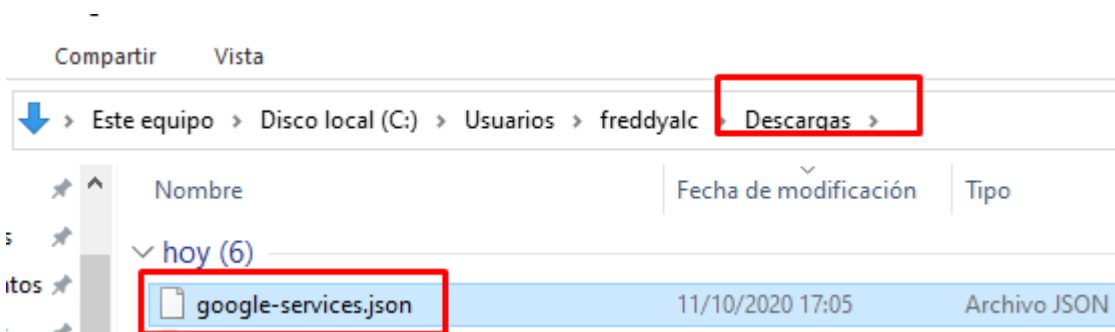
Proveedores de acceso

Proveedor	Estado
✉ Correo electrónico/contraseña	Inhabilitado
☎ Teléfono	Inhabilitado
🌐 Google	Habilitado

Luego hacemos scroll para ver más abajo y nos descargamos el archivo **“google-services.json”**

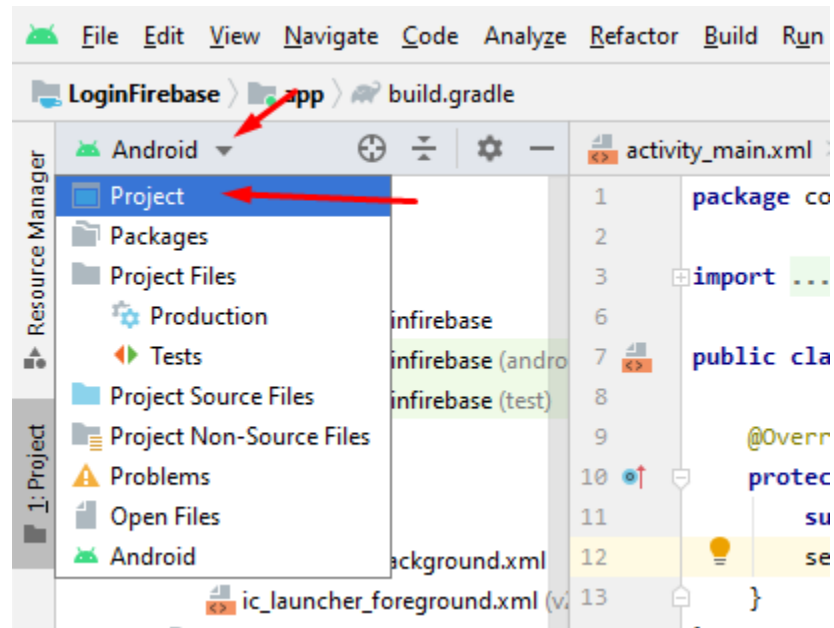


Ahora copiamos el archivo:

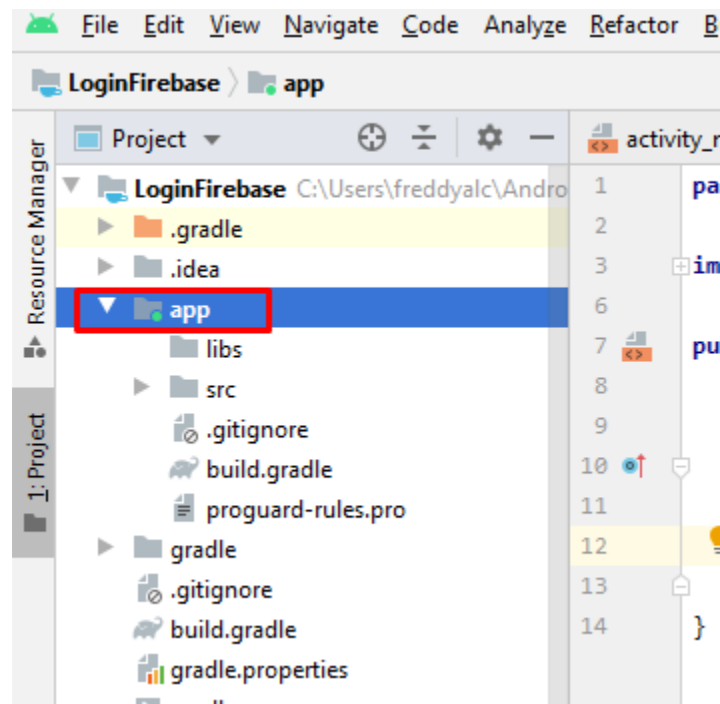


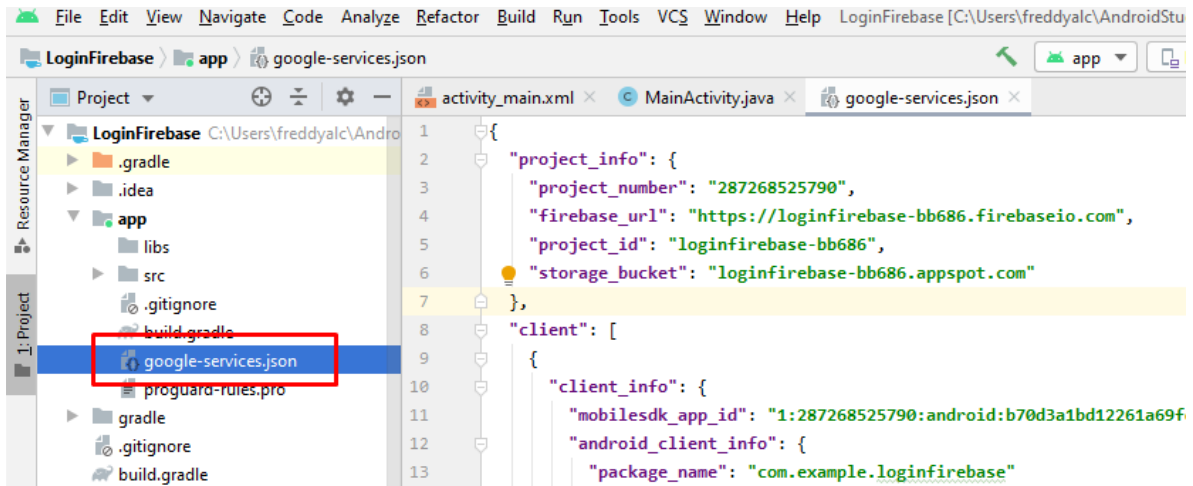
Y lo vamos a pegar dentro del proyecto Android:

Hacemos click en el menú “Android” del proyecto y luego click en “Project”

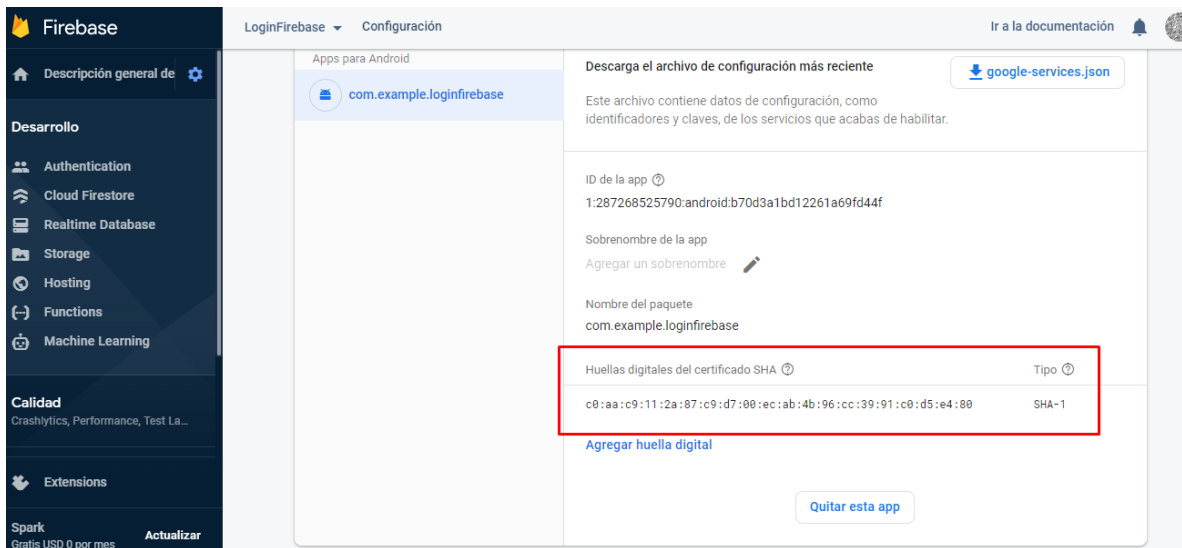


Ahora dentro de la carpeta “App” pegamos el archivo:

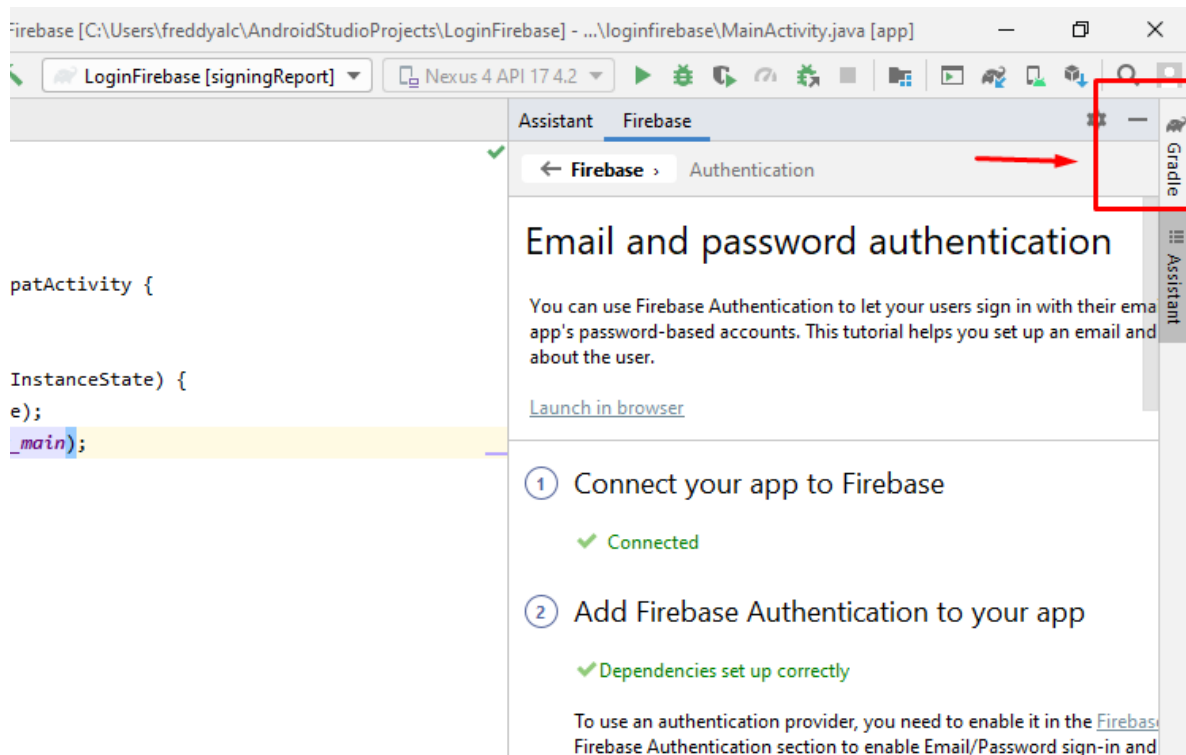




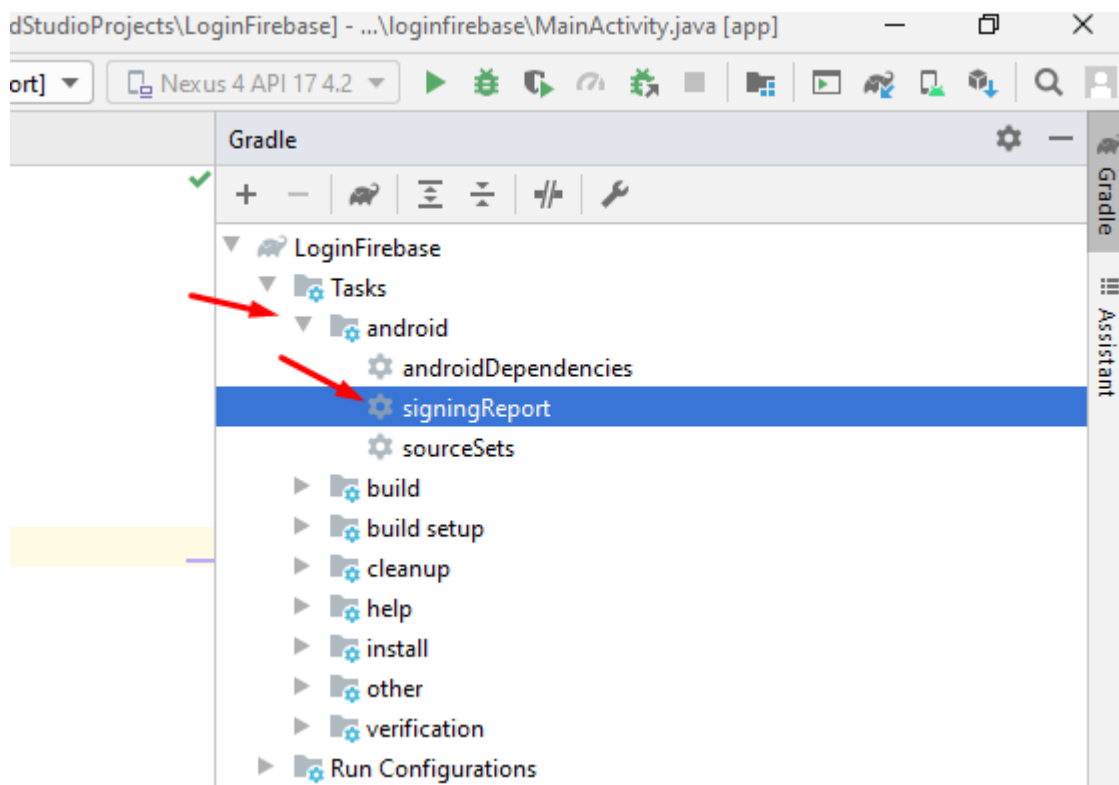
Con eso el proyecto ya esta listo para codificar e implementar las clases de firebase. Sin embargo, aquí se a omitido un paso “el de generar la clave SHA-1” esta fue tomada del proyecto antes cuando creamos el proyecto firebase desde Android studio, si nos fijamos en la consola de firebase en la configuración del proyecto:



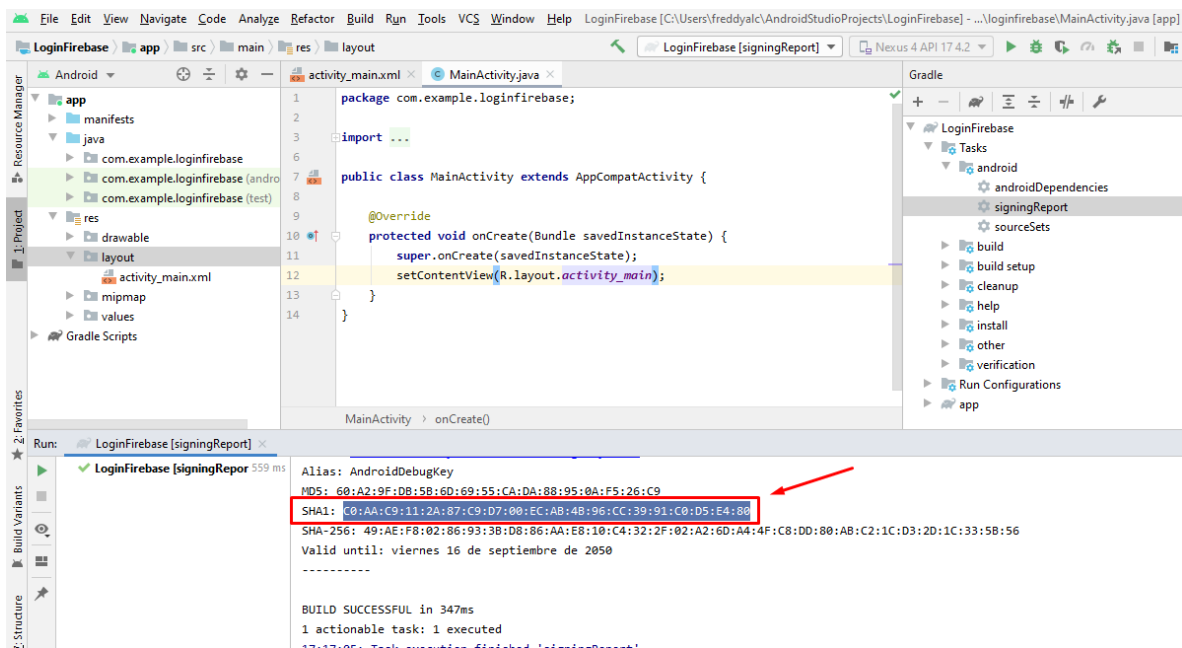
Ya tenemos la huella digital SHA-1. En caso de no tenerla debemos de generarla desde Android studio, para eso nos vamos a la pestaña de “Gradle”:



Luego doble click en Signing Report:



Como vemos ahí se genera:



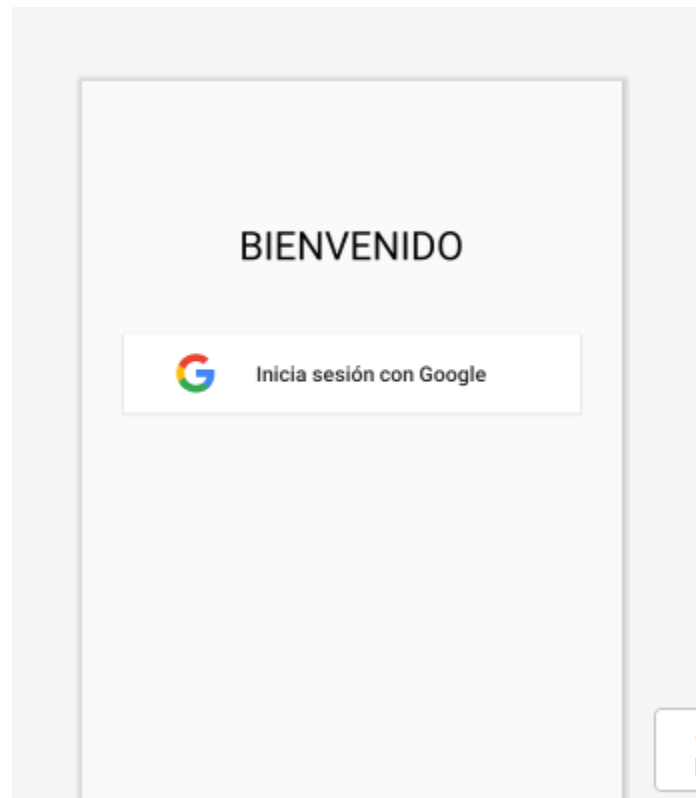
Si comparamos es la misma SHA-1 que tenemos en la consola de firebase. En caso de no tenerla en la consola de firebase, la copiamos y la agregamos en firebase.

Nombre del paquete	com.example.loginfirebase	
Huellas digitales del certificado SHA ?	Tipo ?	
c0:aa:c9:11:2a:87:c9:d7:00:ec:ab:4b:96:cc:39:91:c0:d5:e4:80	SHA-1	
Agregar huella digital		
Quitar esta app		

Codificar el proyecto:

Primero diseñamos el layout xml “activity_main” con un botón para iniciar sesión o “Sign in” y un textview con un texto “Bienvenido”:

```
res/layout/foreground.xml
layout
  activity_main.xml
minman
```



Esos elementos están dentro de un **ConstraintLayout** el botón tiene una propiedad:

```
android:drawableStart="@drawable/ic_google"
```

Para poner el icono al lado izquierdo del texto. El código xml de ese drawable llamado “ic_google” es el siguiente:

```

<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="28.895dp"
    android:height="29.489dp"
    android:viewportWidth="28.895"
    android:viewportHeight="29.489">
    <path
        android:pathData="M1.5889,8.133a12.776,12.776 0,0 1,2.448 -
3.522A14.243,14.243 0,0 1,12.2549 0.218,14.237 14.237,0 0,1
24.3249,3.575c0.221,0.19 0.276,0.3 0.035,0.537C23.1249,5.321 21.9119,6.55
20.6889,7.772c-0.125,0.125 -0.21,0.278 -0.433,0.074A8.468,8.468 0,0 0,8.8739
8.1,9.4 9.4,0 0,0 6.4249,11.834c-0.076,-0.05 -0.157,-0.095 -0.229,-
0.15Q3.8929,9.91 1.5889,8.133Z"
        android:fillColor="#ea4335"
        android:fillType="evenOdd"/>
    <path
        android:pathData="M6.392,17.5983a9.876,9.876 0,0 0,1.806 3.083,8.677
8.677,0 0,0 7.724,2.958 9.348,9.348 0,0 0,3.953 -1.378c0.117,0.1 0.228,0.216
0.351,0.312q2.139,1.67 4.28,3.335a12.255,12.255 0,0 1,-5.522 3.018,14.849
14.849,0 0,1 -13.5,-2.725 13.69,13.69 0,0 1,-3.9 -4.866Z"
        android:fillColor="#34a853"
        android:fillType="evenOdd"/>
    <path
        android:pathData="M24.507,25.9087q-2.141,-1.667 -4.28,-3.335c-0.123,-0.1 -
0.234,-0.208 -0.351,-0.312a7.487,7.487 0,0 0,2.3 -2.718,9.438 9.438,0 0,0 0.5,-
1.372c0.1,-0.322 0.067,-0.448 -0.332,-0.444 -2.375,0.02 -4.75,0.01 -7.126,0.01 -
0.5,0 -0.5,0 -0.5,-0.521 0,-1.61 0.008,-3.22 -0.007,-4.83 0,-0.31 0.052,-0.43
0.4,-0.429q6.571,0.019 13.143,0c0.236,0 0.385,0.017 0.426,0.3a16.167,16.167 0,0
1,-1.9 10.874A10.856,10.856 0,0 1,24.507 25.9087Z"
        android:fillColor="#4285f4"
        android:fillType="evenOdd"/>
    <path
        android:pathData="M6.391,17.5984 L1.576,21.3404A13.339,13.339 0,0 1,0.13
16.6834a14.571,14.571 0,0 1,1.235 -8.135c0.066,-0.142 0.149,-0.277 0.224,-
0.415q2.3,1.775 4.61,3.55c0.072,0.055 0.152,0.1 0.229,0.15A9.315,9.315 0,0
0,6.391 17.5984Z"
        android:fillColor="#fbbc04"
        android:fillType="evenOdd"/>
</vector>

```

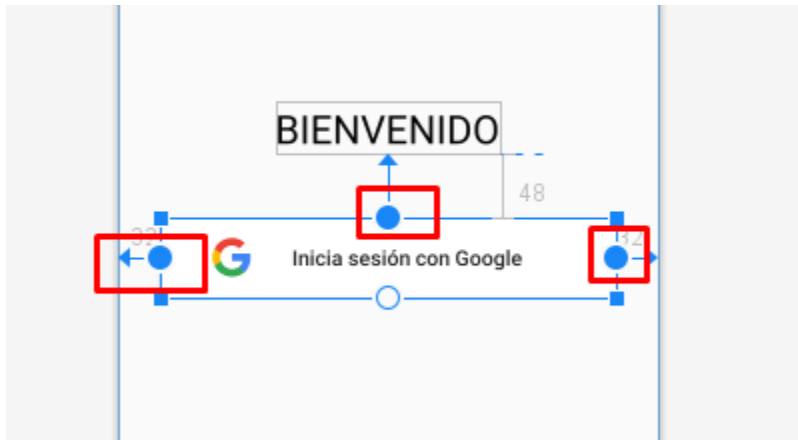
Les dejo unas propiedades extras del botón:

```

android:paddingStart="40dp"
android:paddingEnd="40dp"
android:text="Inicia sesión con Google"
android:textAllCaps="false"
android:textSize="16sp"

```

No olviden de hacer los constraints, es muy fácil:



Además, no olvidar de poner un id al botón por que lo vamos a utilizar desde el main activity:

```
android:id="@+id/btnSignIn"
```

Programar el MainActivity.java

Lo primero que vamos a hacer es crear dos variables arriba del método onCreate();

```
//Variable para gestionar FirebaseAuth  
private FirebaseAuth mAuth;  
//Agregar cliente de inicio de sesión de Google  
private GoogleSignInClient mGoogleSignInClient;
```

```
public class MainActivity extends AppCompatActivity {  
    //Variable para gestionar FirebaseAuth  
    private FirebaseAuth mAuth;  
    //Agregar cliente de inicio de sesión de Google  
    private GoogleSignInClient mGoogleSignInClient;  
    |  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```

Sin embargo, al crear la variable **mGoogleSignInCliente** vemos que no existe la clase **GoogleSignInCliente** para eso necesitamos de agregar la siguiente dependencia:

```
implementation 'com.google.firebase:firebase-auth:19.4.0'  
implementation 'com.google.android.gms:play-services-auth:18.1.0'
```

```
implementation 'com.google.firebase:firebase-auth:19.4.0'  
implementation 'com.google.android.gms:play-services-auth:18.1.0'
```

Fuente <https://firebase.google.com/docs/auth/android/google-signin#java> 4

Ahora hacemos “Alt+ENTER” en las letras rojas de la clase “GoogleSignInCliente” para importar la clase:

```
public class MainActivity extends AppCompatActivity {  
    //Variable para gestionar FirebaseAuth  
    private FirebaseAuth mAuth;  
    //Agregar cliente de inicio de sesión de Google  
    private GoogleSignInClient mGoogleSignInClient;  
    @Override  
    import com.google.android.gms.auth.api.signin.GoogleSignInClient;  
    import com.google.firebase.auth.FirebaseAuth;  
  
    public class MainActivity extends AppCompatActivity {  
        //Variable para gestionar FirebaseAuth  
        private FirebaseAuth mAuth;  
        //Agregar cliente de inicio de sesión de Google  
        private GoogleSignInClient mGoogleSignInClient;
```

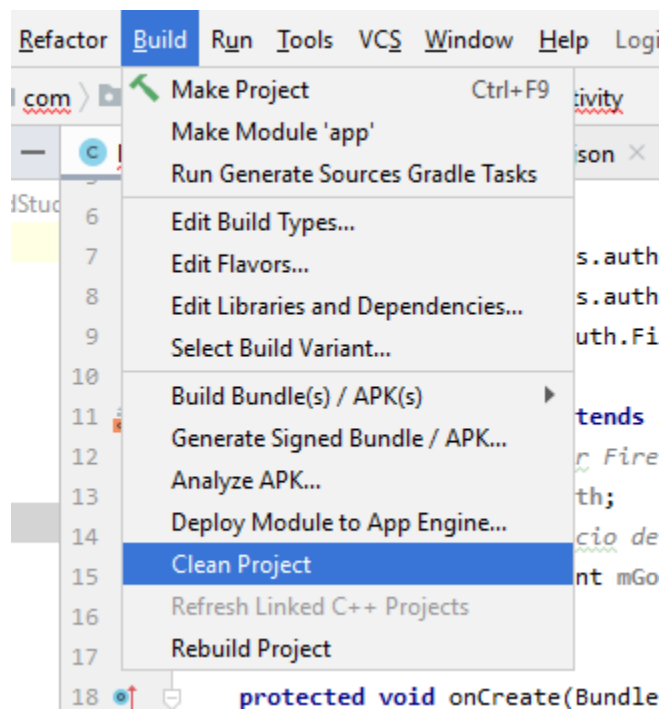
Lo siguiente es configurar “GoogleSignInOptions”:

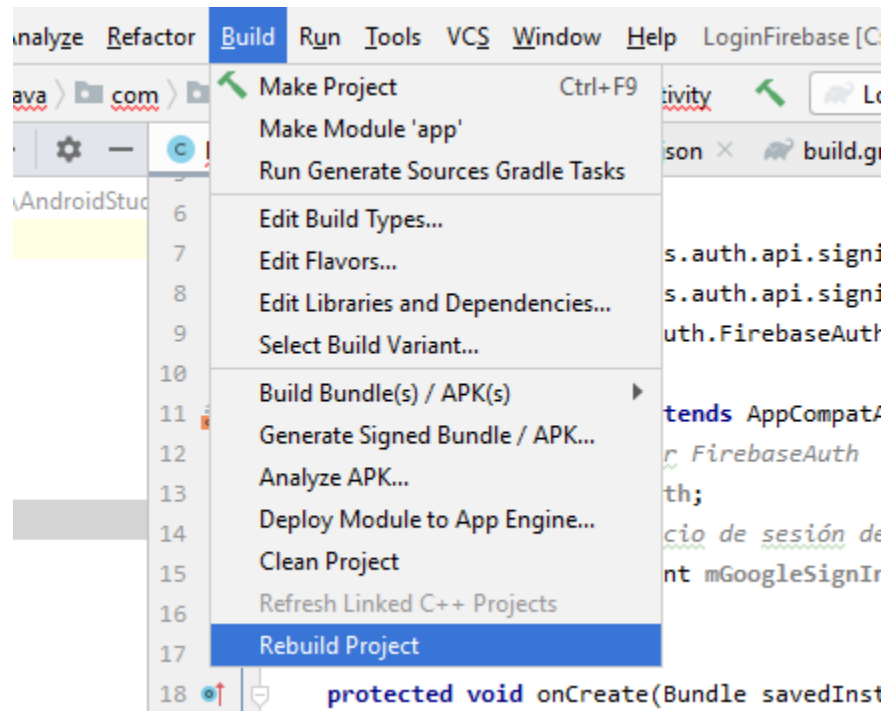
```
// Configurar Google Sign In
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestIdToken(getString(R.string.default_web_client_id))
    .requestEmail()
    .build();
```

```
16
17
18 @Override
19 protected void onCreate(Bundle savedInstanceState) {
20     super.onCreate(savedInstanceState);
21     setContentView(R.layout.activity_main);
22
23     // Configurar Google Sign In
24     GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
25         .requestIdToken(getString(R.string.default_web_client_id))
26         .requestEmail()
27         .build();
28 }
```

MainActivity.java

Sin embargo vemos que no existe el String “default_web_cliente_id” a pesar de ya haber agregado el “Google-services.json” para solucionar eso vamos a darle “Clean-Rebuilt” al proyecto:





Y listo:

```
// Configurar Google Sign In
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestIdToken(getString(R.string.default_web_client_id))
    .requestEmail()
    .build();
```

Si no se quita el error de color rojo, necesitamos borrar esa línea de código desde `.requestIdToken(getString(R.string.default_web_client_id))` y volverla a pegar..

Lo siguiente es crear on GoogleSignIn con las opciones especificadas:

```
// Crear un GoogleSignInClient con las opciones especificadas por gso.
mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
```

```

22
23 // Configurar Google Sign In
24 GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
25     .requestIdToken(getString(R.string.default_web_client_id))
26     .requestEmail()
27     .build();
28 // Crear un GoogleSignInClient con las opciones especificadas por gso.
29 mGoogleSignInClient = GoogleSignIn.getClient( activity: this, gso);
30
31

```

Hasta aquí se realizado la configuración básica de acuerdo a :

<https://developers.google.com/identity/sign-in/android/sign-in>

Lo siguiente es agregar un método llamado “signIn()” dentro del mainActivity, este método será llamado cuando le hagamos click al botón “btnSignIn”

```

private void signIn() {
    Intent signInIntent = mGoogleSignInClient.getSignInIntent();
    startActivityForResult(signInIntent, RC_SIGN_IN);
}

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // Configurar Google Sign In
    GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken(getString(R.string.default_web_client_id))
        .requestEmail()
        .build();

    // Crear un GoogleSignInClient con las opciones especificadas por gso.
    mGoogleSignInClient = GoogleSignIn.getClient( activity: this, gso);
}

private void signIn() {
    Intent signInIntent = mGoogleSignInClient.getSignInIntent();
    startActivityForResult(signInIntent, RC_SIGN_IN);
}

```

Para que no nos marque error esa variable llamada “RC_SIGN_IN” la creamos con un valor numérico cualquiera, les recomiendo que le pongan 1.

```

//Constante
int RC_SIGN_IN = 1;

```



```
public class MainActivity extends AppCompatActivity {
```

```
    //Constante
```

```
    int RC_SIGN_IN = 1;
```

```
    //Variable para gestionar FirebaseAuth
```

```
    private FirebaseAuth mAuth;
```

```
    //Agregar cliente de inicio de sesión de Google
```

```
    private GoogleSignInClient mGoogleSignInClient;
```

```
    @Override
```

Lo siguiente es referenciar nuestro botón de “SignIn” y asignarle el evento onClick luego dentro del método onClick llamamos al método **signIn()**

```
    btnSignIn = findViewById(R.id.btnSignIn);  
    btnSignIn.setOnClickListener(new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            signIn();  
        }  
    });
```

```
    private Button btnSignIn;
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
```

```
        btnSignIn = findViewById(R.id.btnSignIn);
```

```
        btnSignIn.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
```

```
            public void onClick(View v) {
```

```
                signIn();
```

```
            }
```

```
        });
```

```
        // Configurar Google Sign In
```

Luego creamos una instancia de “**FirebaseAuth**”:

```
// Inicializar Firebase Auth
mAuth = FirebaseAuth.getInstance();
```

```
35    });
36    // Configurar Google Sign In
37    GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DI
38    .requestIdToken(getString(R.string.default_web_client_id))
39    .requestEmail()
40    .build();
41    // Crear un GoogleSignInClient con las opciones especificadas por gso.
42    mGoogleSignInClient = GoogleSignIn.getClient( activity, this, gso);
43
44    // Inicializar Firebase Auth
45    mAuth = FirebaseAuth.getInstance();
46  }
47  private void signIn() {
```

Lo siguiente es agregar el método “onActivityResult()”: este método esta debajo de onCreate();

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    //Resultado devuelto al iniciar el Intent de GoogleSignInApi.signInIntent (...);
    // Result returned from Launching the Intent from GoogleSignInApi.signInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
        if(task.isSuccessful()){

            try {
                // Google Sign In was successful, authenticate with Firebase
                GoogleSignInAccount account = task.getResult(ApiException.class);
                Log.d(TAG, "firebaseAuthWithGoogle:" + account.getId());
                firebaseAuthWithGoogle(account.getIdToken());
            } catch (ApiException e) {
                // Google Sign In fallido, actualizar GUI
                Log.w(TAG, "Google sign in failed", e);
            }

        }else{
            Log.d(TAG, "Error, login no exitoso:" + task.getException().toString());
            Toast.makeText(this, "Ocurrio un error. "+task.getException().toString(),
            Toast.LENGTH_LONG).show();
        }
    }
}
```

```
}  
}
```

```
@Override  
public void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
    //Resultado devuelto al iniciar el Intent de GoogleSignInApi.getSignInIntent (...);  
    // Result returned from Launching the Intent from GoogleSignInApi.getSignInIntent(...);  
    if (requestCode == RC_SIGN_IN) {  
        Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);  
        if(task.isSuccessful()){  
            try {  
                // Google Sign In was successful, authenticate with Firebase  
                GoogleSignInAccount account = task.getResult(ApiException.class);  
                Log.d(TAG, msg: "firebaseAuthWithGoogle:" + account.getId());  
                firebaseAuthWithGoogle(account.getIdToken());  
            } catch (ApiException e) {  
                // Google Sign In fallido, actualizar GUI  
                Log.w(TAG, msg: "Google sign in failed", e);  
            }  
        }  
    } else {  
        Log.d(TAG, msg: "Error, login no exitoso:" + task.getException().toString());  
        Toast.makeText( context: this, text: "Ocurrio un error. "+task.getException().toString(), Toast.LENGTH_LONG).show();  
    }  
}
```

Crear constante "TAG":

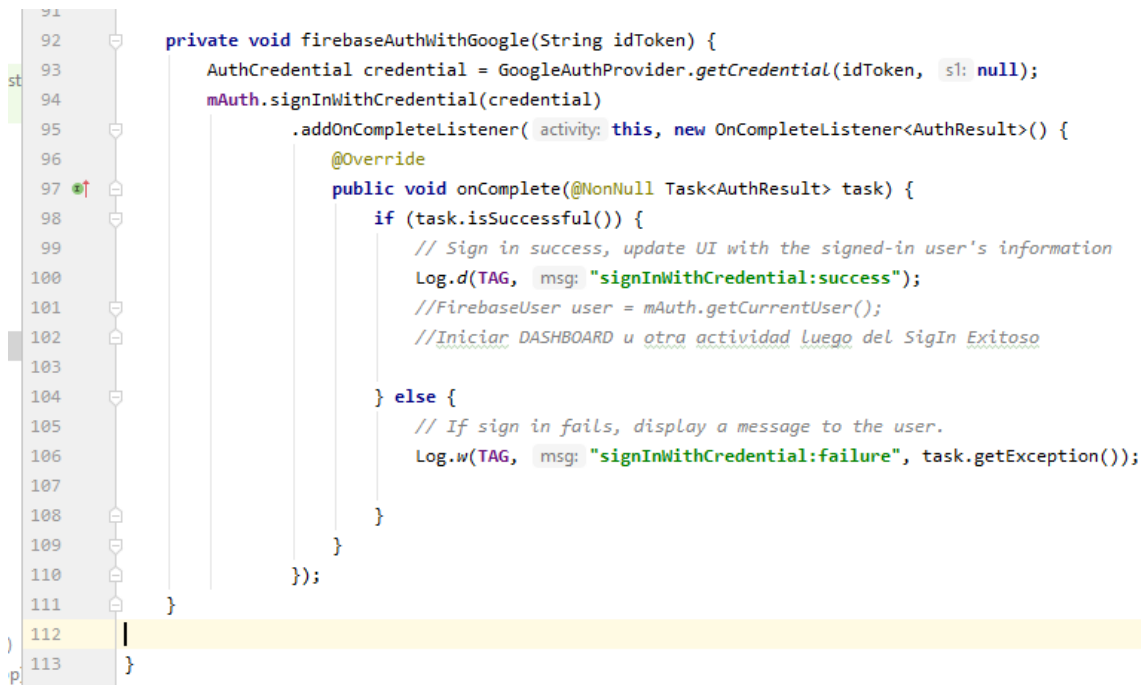
```
String TAG = "GoogleSignIn";
```

```
public class MainActivity extends AppCompatActivity {  
    //Constante  
    int RC_SIGN_IN = 1;  
    String TAG = "GoogleSignIn";  
}
```

Lo siguiente es agregar un último método llamado “`firebaseAuthWithGoogle()`” debajo de `onActivityResult()` para completar el inicio de sesión mediante Google Auth. Este método recibe el token del usuario y obtiene su credencial. Leer el código del método este habla por sí solo.

```
private void firebaseAuthWithGoogle(String idToken) {
    AuthCredential credential = GoogleAuthProvider.getCredential(idToken, null);
    mAuth.signInWithCredential(credential)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    // Sign in success, update UI with the signed-in user's information
                    Log.d(TAG, "signInWithCredential:success");
                    //FirebaseUser user = mAuth.getCurrentUser();
                    //Iniciar DASHBOARD u otra actividad luego del SigIn Exitoso

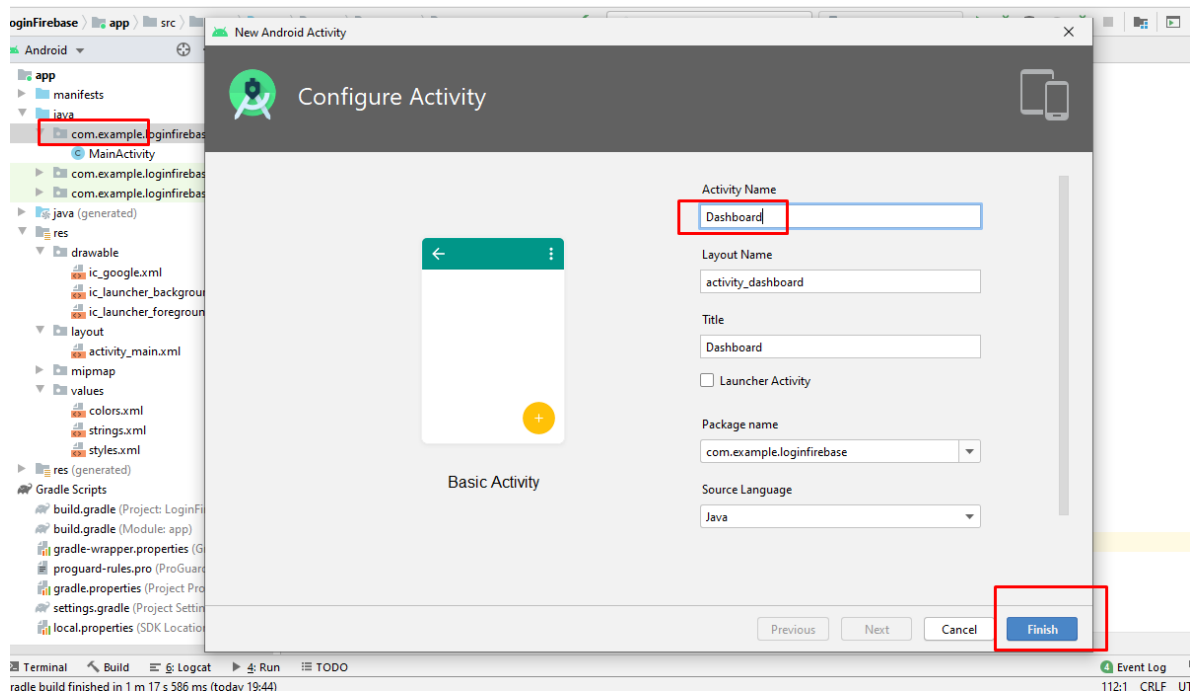
                } else {
                    // If sign in fails, display a message to the user.
                    Log.w(TAG, "signInWithCredential:failure", task.getException());
                }
            }
        });
}
```



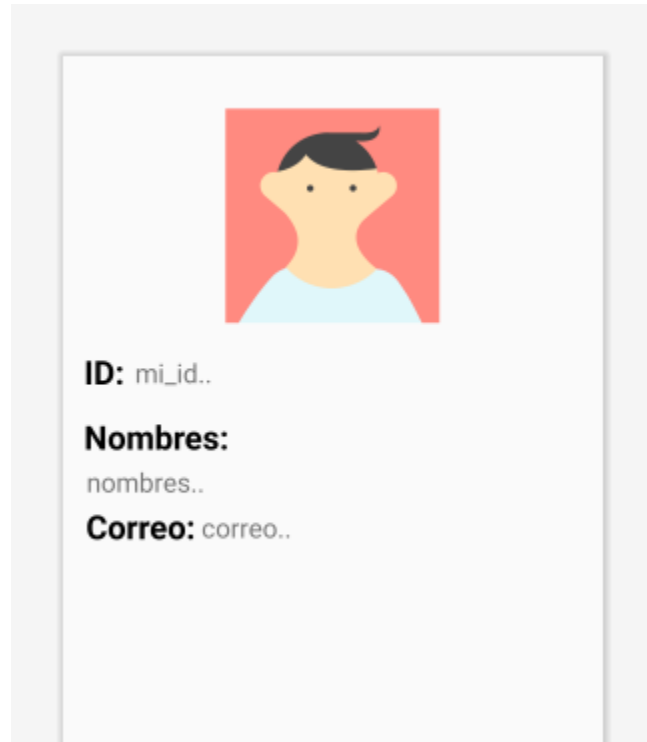
```
91
92 private void firebaseAuthWithGoogle(String idToken) {
93     AuthCredential credential = GoogleAuthProvider.getCredential(idToken, null);
94     mAuth.signInWithCredential(credential)
95         .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
96             @Override
97             public void onComplete(@NonNull Task<AuthResult> task) {
98                 if (task.isSuccessful()) {
99                     // Sign in success, update UI with the signed-in user's information
100                     Log.d(TAG, msg: "signInWithCredential:success");
101                     //FirebaseUser user = mAuth.getCurrentUser();
102                     //Iniciar DASHBOARD u otra actividad luego del SigIn Exitoso
103
104                 } else {
105                     // If sign in fails, display a message to the user.
106                     Log.w(TAG, msg: "signInWithCredential:failure", task.getException());
107                 }
108             }
109         });
110     }
111 }
112
113 }
```

Ya podríamos probar la aplicación, pero falta algo más. Si nos fijamos en la línea 102 del método `firebaseAuthWithGoogle` hay un comentario que e dejado que es lugar donde deberíamos de iniciar o abrir una actividad para terminar este proceso ya que el usuario ahí ya esta logueado exitosamente. Lo que resta entonces es hacer una nueva actividad para mostrar la información del usuario:

Creamos una nueva actividad llamada “Dashboard”:



Al archivo xml del activity **dashboard** lo diseñamos de la siguiente manera:



Para poder mostrar la imagen del usuario necesitamos de implementar una librería que permita cargar imágenes mediante URL, para eso utilizaremos “Glide” por lo tanto agregamos en la dependencia/gradle glide:

```
implementation 'com.github.bumptech.glide:glide:4.11.0'
```

El Código del dashboard Activity sería el siguiente;

```
public class Dashboard extends AppCompatActivity {
    //Variable para gestionar FirebaseAuth
    private FirebaseAuth mAuth;
    private TextView txtid, txtnombres, txtemail;
    private ImageView imagenUser;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dashboard);
        imagenUser = findViewById(R.id.imagenUser);
        txtid = findViewById(R.id.txtId);
        txtnombres = findViewById(R.id.txtNombres);
        txtemail = findViewById(R.id.txtCorreo);
        // Inicializar Firebase Auth
        mAuth = FirebaseAuth.getInstance();
        FirebaseUser currentUser = mAuth.getCurrentUser();
    }
}
```

```

        //set datos:
        txtid.setText(currentUser.getUid());
        txtnombres.setText(currentUser.getDisplayName());
        txtemail.setText(currentUser.getEmail());
        //cargar imagen con glide:
        Glide.with(this).load(currentUser.getPhotoUrl()).into(imagenUser);
    }
}

```

```

14
15 public class Dashboard extends AppCompatActivity {
16     //Variable para gestionar FirebaseAuth
17     private FirebaseAuth mAuth;
18     private TextView txtid, txtnombres, txtemail;
19     private ImageView imagenUser;
20     @Override
21     protected void onCreate(Bundle savedInstanceState) {
22         super.onCreate(savedInstanceState);
23         setContentView(R.layout.activity_dashboard);
24         imagenUser = findViewById(R.id.imagenUser);
25         txtid = findViewById(R.id.txtId);
26         txtnombres = findViewById(R.id.txtNombres);
27         txtemail = findViewById(R.id.txtCorreo);
28         // Inicializar Firebase Auth
29         mAuth = FirebaseAuth.getInstance();
30         FirebaseUser currentUser = mAuth.getCurrentUser();
31         //set datos:
32         txtid.setText(currentUser.getUid());
33         txtnombres.setText(currentUser.getDisplayName());
34         txtemail.setText(currentUser.getEmail());
35         //cargar imagen con glide:
36         Glide.with( activity: this).load(currentUser.getPhotoUrl()).into(imagenUser);
37     }
38 }
39

```

REFERENCIAR VISTAS

INSTANCIA DE FIREBASE Y OBTENEMOS EL USUARIO ACTUAL.

ESTABLECER DATOS EN LAS VISTAS

Ya tenemos preparada la actividad **“dashboard”** para recibir los datos lo que resta es regresar a **“MainActivity”** dentro del método `firebaseAuthWithGoogle()` agregar las líneas de código para abrir la actividad del **“dashboard”** luego de que se ha validado la credencial del usuario:

```

Intent dashboardActivity = new Intent(MainActivity.this, Dashboard.class);
startActivity(dashboardActivity);
MainActivity.this.finish();

```

```

91
92 private void firebaseAuthWithGoogle(String idToken) {
93     AuthCredential credential = GoogleAuthProvider.getCredential(idToken, null);
94     mAuth.signInWithCredential(credential)
95         .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
96             @Override
97             public void onComplete(@NonNull Task<AuthResult> task) {
98                 if (task.isSuccessful()) {
99                     // Sign in success, update UI with the signed-in user's information
100                     Log.d(TAG, msg: "signInWithCredential:success");
101                     //FirebaseUser user = mAuth.getCurrentUser();
102                     //Iniciar DASHBOARD u otra actividad Luego del SigIn Exitoso
103                     Intent dashboardActivity = new Intent( packageContext: MainActivity.this, Dashboard.class);
104                     startActivity(dashboardActivity);
105                     MainActivity.this.finish();
106
107                 } else {
108                     // If sign in fails, display a message to the user.
109                     Log.w(TAG, msg: "signInWithCredential:failure", task.getException());
110                 }
111             }
112         });
113     }
114 }

```

Además, agregamos el método “onStart()” al mainActivity para verificar ahí si el usuario ya está logueado en caso de ser así enviarlo al dashboard, de esa manera se evita que el usuario regrese al mainActivity a iniciar sesión:

```

@Override
protected void onStart() {
    FirebaseUser user = mAuth.getCurrentUser();
    if(user!=null){ //si no es null el usuario ya esta Logueado
        //mover al usuario al dashboard
        Intent dashboardActivity = new Intent(MainActivity.this, Dashboard.class);
        startActivity(dashboardActivity);
    }
    super.onStart();
}

```

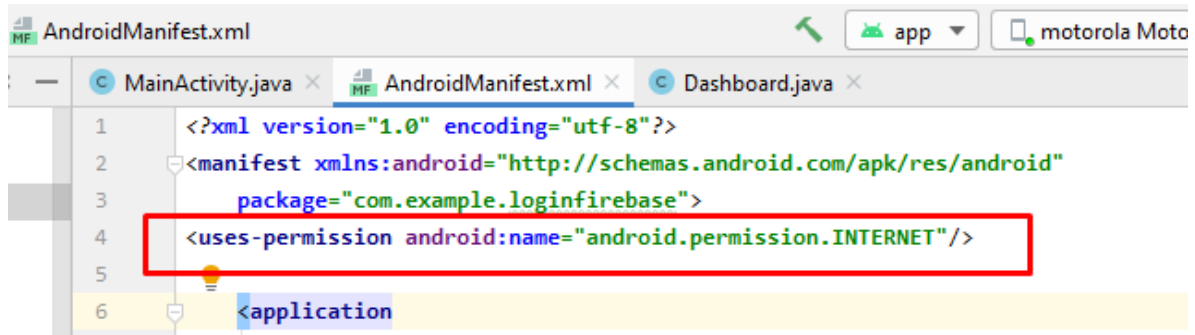
```

@Override
protected void onStart() {
    FirebaseUser user = mAuth.getCurrentUser();
    if(user!=null){ //si no es null el usuario ya esta Logueado
        //mover al usuario al dashboard
        Intent dashboardActivity = new Intent( packageContext: MainActivity.this, Dashboard.class);
        startActivity(dashboardActivity);
    }
    super.onStart();
}

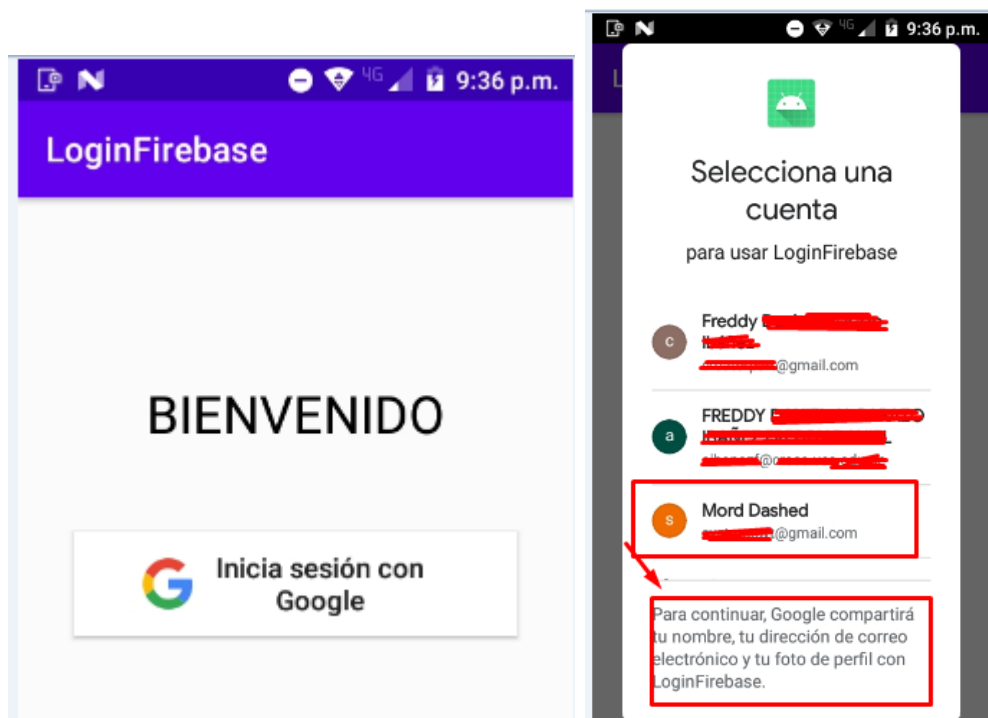
```

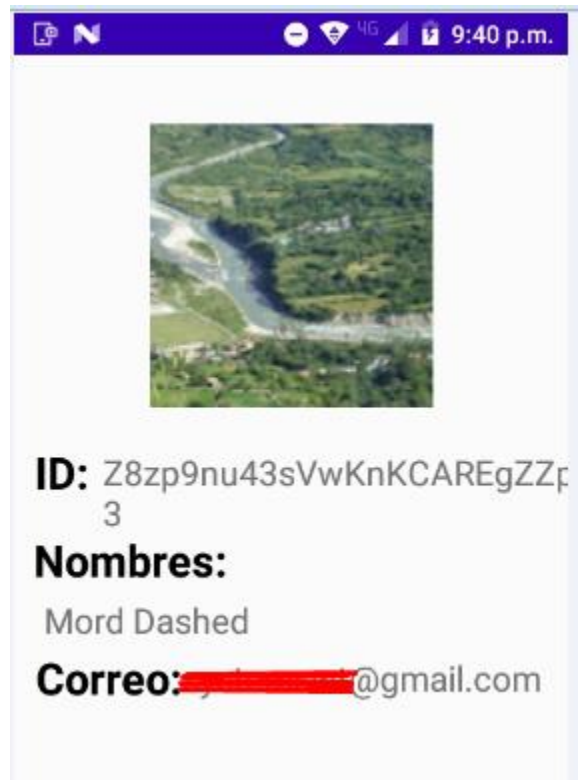

Para probar solo resta agregar el permiso de Internet en el AndroidManifest:

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



Prueba:



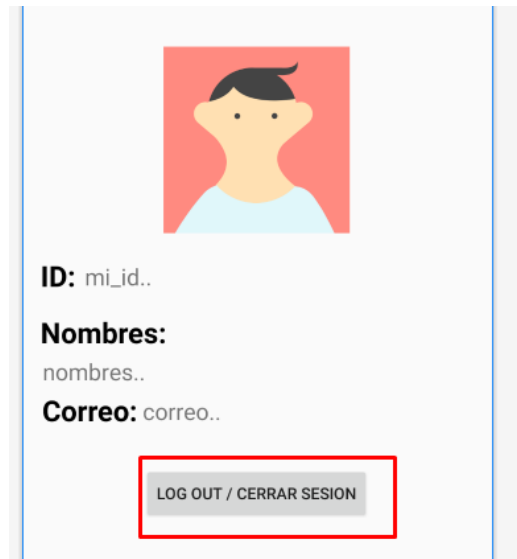


Verificamos nuestros usuarios en Firebase Console:

A screenshot of the Firebase Authentication console. The left sidebar shows the 'Desarrollo' (Development) section with options like Authentication, Cloud Firestore, Realtime Database, Storage, Hosting, Functions, and Machine Learning. The main area is titled 'Authentication' and has tabs for 'Users', 'Sign-in method', 'Templates', and 'Usage'. The 'Users' tab is active, showing a search bar and a table of users. The table has columns for 'Identificador', 'Proveedores', 'Creado', 'Accediste a tu cuenta', and 'UID de usuario'. A single user is listed with a redacted email address, the Google provider, and creation/access dates of 11 oct. 2020. The UID is 'QxDR0pGub6fJ7tq2E4M71kufPww2'. At the bottom, it shows 'Filas por página: 50' and '1 a 1 de 1'.

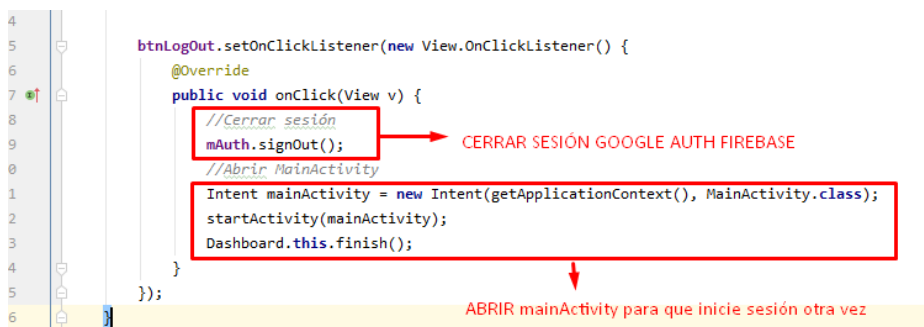
Lo que resta ahora es hacer “Log out” o “Cerrar Sesión”

- Agregamos un botón al dashboard para cerrar sesión.



- Referenciamos el botón en la actividad “**Dashboard.java**” y le asignamos el evento onClick con el siguiente código:

```
btnLogout.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        //Cerrar session con Firebase  
        mAuth.signOut();  
        //Abrir MainActivity  
        Intent mainActivity = new Intent(getApplicationContext(),  
MainActivity.class);  
        startActivity(mainActivity);  
        Dashboard.this.finish();  
    }  
});
```



Ya podemos probar y veremos que iniciamos sesión y podemos cerrar sesión. Pero sin embargo no cerramos sesión con Google solo con firebase, para cerrar sesión con Google también debemos de agregar los siguiente en la actividad “**Dashboard.java**”:

```
//Variables opcionales para desloguear de google tambien
private GoogleSignInClient mGoogleSignInClient;
private GoogleSignInOptions gso;
```

```
31
32 //Variables opcionales para desloguear de google tambien
33 private GoogleSignInClient mGoogleSignInClient;
34 private GoogleSignInOptions gso;
35
36 @Override
37 protected void onCreate(Bundle savedInstanceState) {
```

Luego dentro del método **onCreate** de la misma actividad “**Dashboard**”:

```
//Configurar las gso para google signIn con el fin de luego desloguear de google
gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestIdToken(getString(R.string.default_web_client_id))
    .requestEmail()
    .build();
mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
```

```
57 //Configurar las gso para google signIn con el fin de luego desloguear de google
58 gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
59     .requestIdToken(getString(R.string.default_web_client_id))
60     .requestEmail()
61     .build();
62 mGoogleSignInClient = GoogleSignIn.getClient(activity: this, gso);
63
64
65 btnLogout.setOnClickListener(new View.OnClickListener() {
```

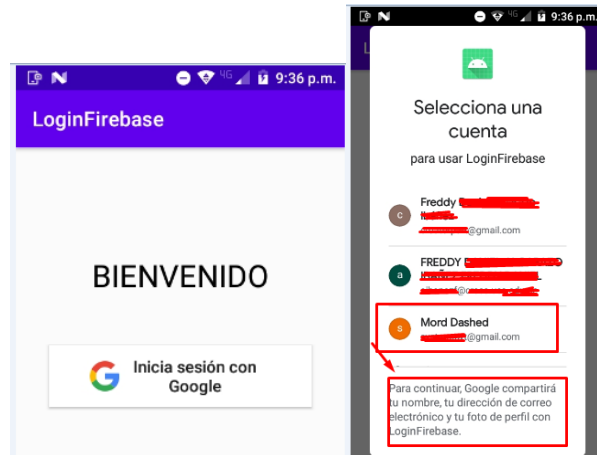
Finalmente, dentro del método **onClick** del botón de cerrar sesión agregamos lo siguiente luego del código de cerrar sesión con firebase:

```
//Cerrar sesión con google tambien: Google sign out
mGoogleSignInClient.signOut().addOnCompleteListener(new OnCompleteListener<Void>() {
    @Override
    public void onComplete(@NonNull Task<Void> task) {
        //Abrir MainActivity con SigIn button
        if(task.isSuccessful()){
            Intent mainActivity = new Intent(getApplicationContext(), MainActivity.class);
            startActivity(mainActivity);
            Dashboard.this.finish();
        }else{
            Toast.makeText(getApplicationContext(), "No se pudo cerrar sesión con google",
            Toast.LENGTH_LONG).show();
        }
    }
});
```



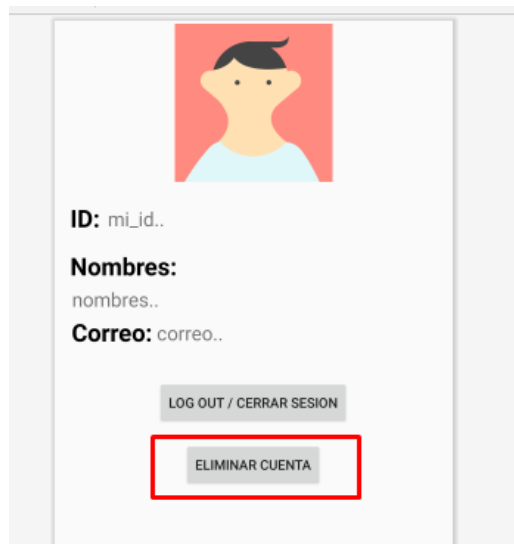
✓ **Probar que el usuario puede hacer log out o cerrar sesión.**

De esa manera el usuario puede desloguearse y volver a iniciar sesión con su cuenta:



El siguiente paso es hacer que el usuario elimine su cuenta:

Lo primero que hacemos es **agregar un boton eliminar** al xml del dashboard:



Ahora vamos a programar el **evento onClick** del boton “**btnEliminarCuenta**” dentro de **Dashboard.class**

Antes de eliminar el usuario debemos de “**re-autenticarlo**” usando sus credenciales en este caso es el token:

```
btnEliminarCuenta.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //obtener el usuario actual
        final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        // Get the account
        GoogleSignInAccount signInAccount =
        GoogleSignIn.getLastSignedInAccount(getApplicationContext());
        if (signInAccount != null) {
            AuthCredential credential =
            GoogleAuthProvider.getCredential(signInAccount.getIdToken(), null);
            //Re-autenticar el usuario para eliminarlo
            user.reauthenticate(credential).addOnCompleteListener(new OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if (task.isSuccessful()) {
                        //Eliminar el usuario
                        user.delete().addOnSuccessListener(new OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                Log.d("dashBoard", "onSuccess:Usuario Eliminado");
                                //llamar al metodo signOut para salir de aqui
                                signOut();
                            }
                        });
                    } else {
                        Log.e("dashBoard", "onComplete: Error al eliminar el usuario",
task.getException());
                    }
                }
            });
        } else {
            Log.d("dashBoard", "Error: reAuthenticateUser: user account is null");
        }
    }
}); //fin onClick
```

- ✓ Firebase exige autenticación reciente antes de acciones sensibles como la eliminación de la cuenta, por lo que estoy usando "user.reauthenticate (credencial)" antes de "user.delete ()"

- ✓ Firebase Authentication almacenará en caché un token de autenticación para el usuario cuando inicie sesión. Esto evita tener que autenticar cada pequeña interacción que el usuario hace con otros servicios proporcionados por Firebase. Este token se actualiza automáticamente de forma periódica, pero hasta entonces, el SDK asume que el token representa al usuario. Pero el token caducará después de un tiempo es por eso que hacemos la re-autenticación.

```
94 btnEliminarCuenta.setOnClickListener(new View.OnClickListener() {
95     @Override
96     public void onClick(View v) {
97         //obtener el usuario actual
98         final FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
99         // Get the account
100         GoogleSignInAccount signInAccount = GoogleSignIn.getLastSignedInAccount(getApplicationContext());
101         if (signInAccount != null) {
102             AuthCredential credential = GoogleAuthProvider.getCredential(signInAccount.getIdToken(), null);
103             //Re-autenticar el usuario para eliminarlo
104             user.reauthenticate(credential).addOnCompleteListener(new OnCompleteListener<Void>() {
105                 @Override
106                 public void onComplete(@NonNull Task<Void> task) {
107                     if (task.isSuccessful()) {
108                         //Eliminar el usuario
109                         user.delete().addOnSuccessListener(new OnSuccessListener<Void>() {
110                             @Override
111                             public void onSuccess(Void aVoid) {
112                                 Log.d( tag: "dashBoard", msg: "onSuccess:Usuario Eliminado");
113                                 //Llamar al metodo signOut para salir de aqui
114                                 signOut();
115                             }
116                         });
117                     } else {
118                         Log.e( tag: "dashBoard", msg: "onComplete: Error al eliminar el usuario", task.getException());
119                     }
120                 }
121             });
122         }
123     }
124 }
```

Luego de la eliminación exitosa en la línea **114** llamamos al método **signOut()**:

```
private void signOut() {
    //sign out de firebase
    FirebaseAuth.getInstance().signOut();
    //sign out de "google sign in"
    mGoogleSignInClient.signOut().addOnCompleteListener(new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            //regresar al login screen o MainActivity
            //Abrir mainActivity para que inicie sesión o sign in otra vez.
            Intent IntentMainActivity = new Intent(getApplicationContext(), MainActivity.class);
            IntentMainActivity.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(IntentMainActivity);
            Dashboard.this.finish();
        }
    });
}
```


Antes de probar vamos a hacer un ligero cambio en el **mainActivity** para tener un control optimo de que si el usuario es null “quiere decir que no esta logueado o no existe” o diferente de null “que existe, esta logueado”. Estavamos validando dentro del método **onStart()** del mainAcitivity de que si el usuario estaba logueado con la condición “**user!=null**” esa condición la vamos a manejar de otra forma en el método onCreate usando **FirebaseAuth.AuthStateListener**:

Código antiguo en el onStart():

```
19  @Override
20  protected void onStart() {
21      FirebaseUser user = mAuth.getCurrentUser();
22      if(user!=null){ //si no es null el usuario ya esta Logueado
23          //mover al usuario al dashboard
24          Intent dashboardActivity = new Intent( packageContext MainActivity.this, Dashboard.class);
25          startActivity(dashboardActivity);
26      }
27      super.onStart();
28  }
```

Entonces lo primero haremos es poner en la cabecera del **mainActivity** lo siguiente:

```
//Variable mAuthStateListener para controlar el estado del usuario:
private FirebaseAuth.AuthStateListener mAuthStateListener;
```

```
35
36      //Variable mAuthStateListener para controlar el estado del usuario:
37      private FirebaseAuth.AuthStateListener mAuthStateListener;
38
39      private Button btnSignIn;
40      @Override
41      protected void onCreate(Bundle savedInstanceState) {
```

Luego en algún lugar dentro del método **onCreate()** agregamos lo siguiente:

```
//Controlar el estado del usuario
 mAuthStateListener = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        if (firebaseAuth.getCurrentUser() != null){ //si no es null redirigir
            Intent intentDashboard = new Intent(getApplicationContext(), Dashboard.class);
            intentDashboard.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intentDashboard);
        }
    }
};
```

Básicamente con ese Código controlamos al usuario, en caso de que sea diferente de null lo mandamos a la actividad **Dashboard** y en caso de ser nulo permanece dentro de la actividad y realizara **sign in**.

Ahora el código del método **onStart()** solamente tiene lo siguiente:

```
@Override
protected void onStart() {

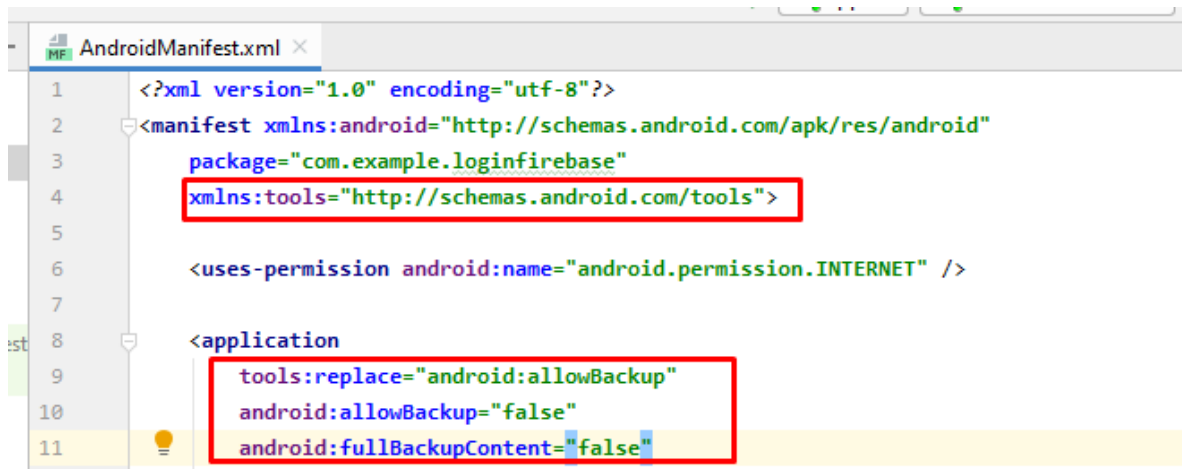
    mAuth.addAuthStateListener(mAuthStateListener);
    super.onStart();
}
```

Lo último que vamos a agregar son unas directivas dentro del **AndroidManifest.xml** de la aplicación. Vamos a agregar:

```
tools:replace="android:allowBackup"
android:allowBackup="false"
android:fullBackupContent="false"
```

Esas directivas garantizan que la información de identidad/autenticación del usuario no se conserve después de desinstalar la aplicación.

```
xmlns:tools="http://schemas.android.com/tools"
```



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.example.loginfirebase"
4     xmlns:tools="http://schemas.android.com/tools">
5
6     <uses-permission android:name="android.permission.INTERNET" />
7
8     <application
9         tools:replace="android:allowBackup"
10         android:allowBackup="false"
11         android:fullBackupContent="false">
```

Listo!

Eso es todo. Probar la aplicación.

Conclusión:

Hemos visto como integrar firebase en nuestra aplicación Android, hemos logrado que el usuario inicie sesión, cierre sesión y elimine su cuenta. Hemos aprendido que antes de hacer una operación sensible como la de eliminar un usuario firebase recomienda que re-autentiquemos el usuario ya que el token puede estar vencido, aunque se regenera periódicamente, por lo que se recomienda que el usuario vuelva a iniciar sesión antes de eliminarlo.

No se olviden de visitar el canal del youtube y seguirme en github también.