Firebase Autenticación Kotlin

OBJETIVO

En esta guía aprenderemos a conectarnos a una Base de Datos Firebase usando Firebase Authentication, para que a futuro puedas implementarlo en tus aplicaciones.

Firebase Authentication

Una aplicación no se encuentra completa si no tiene un mecanismo para identificar a los usuarios que se conectan a la app. Firebase Authentication proporciona servicios de backend, SDK y Biblioteca IU ya elaboradas para usarlo en nuestra Apps.

Admite autenticación mediante contraseñas, números de teléfono, y otros proveedores de identidad como Google, Facebok, Twitter y otros.

Usuarios en Firebase

Un objeto de Usuario en Firebase representan a la cuenta de usuario registrado en nuestro proyecto.

Los usurios en Firebs tienen un conjunto de propiedades: ID único, Dirección de correo, URL de nombre, y una foto; éstos datos se almacenan en la base de datos de usuarios de un proyecto firebase, mediante el SDK (Android, IOS, Web) proporcionado se puede actualizar esta información.

Proveedores de Acceso

Es posible hacer que los usuarios accedan a nuestras apps a través de diversos métodos. Por ejemplo el Acceso con Google o Facebook aunque el más común es mediante la dirección de correo electrónico y contraseña.

El usuario actual

Cuando un usuario accede a nuestra app con sus credenciales, éste pasa a ser el usuario actual de la instancia de `FirebaseAuth`. La instancia de FirebaseAuth consera el estado del usuario, es decir que no se perderá la información del usuario al actualizar la página o reiniciar la App. Cuando el usuario actual sale de la sesión, el usuario activo pasa a ser nulo.

La manera correcta de obtener el usuario actual es mediante el método `getCurrentUser`.

```
FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
if (user != null) {
    // El usuario esta logueado
```

```
} else {
    // el usuario no esta logueado
}
```

Para obtener el perfil del usuario actual debemos usar los métodos de la instancia de `FirebaseUser`.

```
FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
if (user != null) {
    // Name, email address, and profile photo Url
    String name = user.getDisplayName();
    String email = user.getEmail();
    Uri photoUrl = user.getPhotoUrl();

    // Check if user's email is verified
    boolean emailVerified = user.isEmailVerified();

    // The user's ID, unique to the Firebase project. Do NOT use this value to
    // authenticate with your backend server, if you have one. Use
    // FirebaseUser.getIdToken() instead.
    String uid = user.getUid();
}
```

Para obtener el perfil de un usuario de un proveedor en especifico se utiliza el método `getProviderData`.

```
FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
if (user != null) {
    for (UserInfo profile : user.getProviderData()) {
        // Id of the provider (ex: google.com)
        String providerId = profile.getProviderId();

        // UID specific to the provider
        String uid = profile.getUid();

        // Name, email address, and profile photo Url
        String name = profile.getDisplayName();
        String email = profile.getEmail();
        Uri photoUrl = profile.getPhotoUrl();
    }
}
```

Para actualizar la información básica del perfil del usuario actual se utiliza el método `updateProfile`

```
FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

UserProfileChangeRequest profileUpdates = new UserProfileChangeRequest.Builder()
        .setDisplayName("Jane Q. User")
        .setPhotoUri(Uri.parse("https://example.com/jane-q-user/profile.jpg"))
        .build();

user.updateProfile(profileUpdates)
        .addOnCompleteListener(new OnCompleteListener<Void>() {
```

```
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if (task.isSuccessful()) {
                    Log.d(TAG, "User profile updated.");
                }
            }
        });
```

Autenticación mediante email/contraseña

Para usar Autenticación mediante email y contraseña debemos seguir los siguientes pasos:

1. Agregar Firebase a nuestro proyecto.
2. Habilitar el acceso con correo y contraseña desde el firebase console
3. Agregar la dependencia de Firebase-auth desde el build.gradle del módulo de nuestro proyecto

```
dependencies {
   implementation 'com.google.firebase:firebase-auth:20.0.4'
}
```

Para comenzar a trabajar con Firebase Auth debemos obtener la instancia del objeto `FirebaseAuth`, esto lo podemos hacer en el método `onCreate` de nuestra actividad:

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

cuanto inicie la actividad en el método `onStart`, se puede hacer la verificación de que si el usuario inició sesión:

```
@Override
public void onStart() {
    super.onStart();
    // Check if user is signed in (non-null) and update UI accordingly.
    FirebaseUser currentUser = mAuth.getCurrentUser();
    if(currentUser != null){
        reload();
    }
}
```

En caso de que desde nuestra también tenga el soporte de crear cuentas lo podemos hacer con el método `createUserWithEmailAndPassword` de la instancia de `FirebaseAuth`.

```
mAuth.createUserWithEmailAndPassword(email, password)
        .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, "createUserWithEmail:success");
                    FirebaseUser user = mAuth.getCurrentUser();
                    updateUI(user);
                } else {
                    // If sign in fails, display a message to the user.
                    Log.w(TAG, "createUserWithEmail:failure", task.getException());
                    Toast.makeText(EmailPasswordActivity.this, "Authentication
failed.",
                            Toast.LENGTH_SHORT).show();
                    updateUI(null);
                }
            }
        });
```

Finalmente para iniciar sesión con correo electrónico y contraseña debemos ejecutar el método `signInWithEmailAndPassword`

```
  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 with the signed-in user's
information
                    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(EmailPasswordActivity.this, "Authentication
failed.",
                            Toast.LENGTH_SHORT).show();
                    updateUI(null);
                }
            }
        });
```

Para cerrar sesión de un usuario se lo debe hacer de la siguiente manera:

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

Autenticación con FirebaseUI-Auth

FirebaseUI es una biblioteca creada a partir del SDK de Firebase Authentication que proporciona flujos de autenticación directo para usar en nuestra app. Proporciona los siguientes beneficios:

- Soporte de Varios Proveedores: Proporciona flujos de acceso para correo electrónico y contraseña, autenticación telefónica, Acceso con Google, Facebook, Twitter, Microsoft y otros.

- Administración de cuentas, Posibilia flujos para controlas las tareas de administración de cuentas, como la creación de cuentas y restablecimiento de contraseñas.
- Temas personalizados, permite modificar y adaptarlo de acuerdo a nuestra necesidad al ser un proyecto de código abierto.
- FirebaseUI implementa prácticas recomendadas para la autenticación tanto en dispositivos móviles como en aplicaciones web.

Pasos para poner en marcha FirebaseUI en nuestro proyecto

1. Agregar la dependencia FirebaseUI a nuestro proyecto

```
dependencies {
    // ...

    implementation 'com.firebaseui:firebase-ui-auth:6.4.0'

    // Required only if Facebook login support is required
    // Find the latest Facebook SDK releases here: https://goo.gl/Ce5L94
    implementation 'com.facebook.android:facebook-android-sdk:4.x'

    // Required only if Twitter login support is required
    // Find the latest Twitter SDK releases here: https://goo.gl/E5wZvQ
    implementation 'com.twitter.sdk.android:twitter-core:3.x'
}
```

2. Habilitar los métodos de autenticación desde el Firebase Console

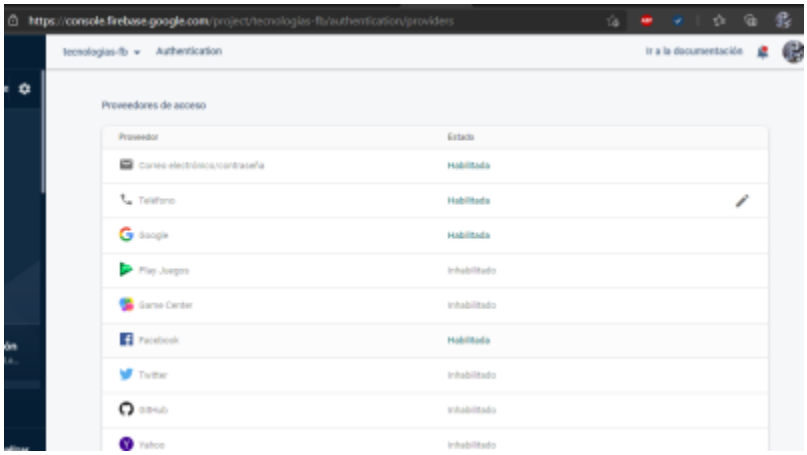

Figure 1: Proveedores de Acceso del Firebase Consoles

3. Si admites método de acceso Google debes agregar las huella SHA-1 de tu app.

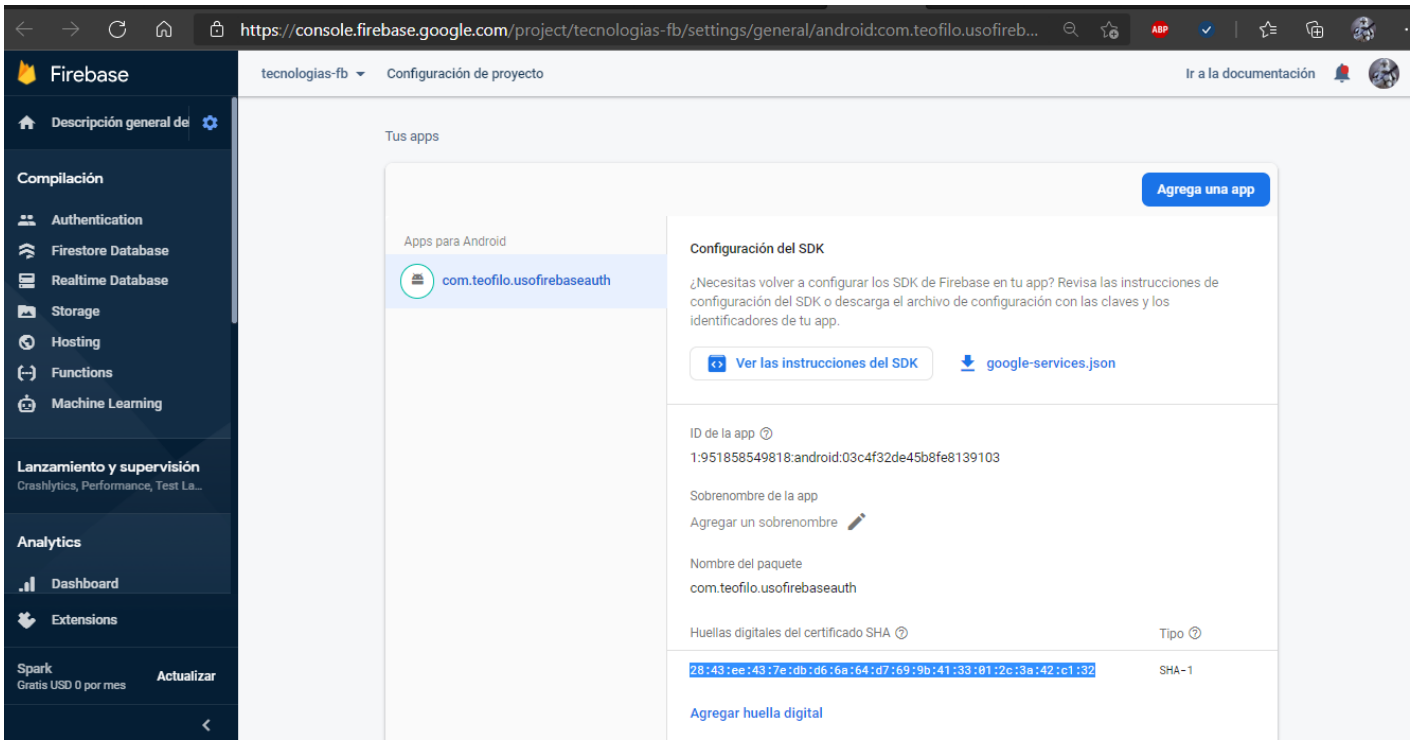

Figure 2: Configuración del SDK Android y huellas digitales

4. Si admites Faceboook o Twitter debes agregar un recurso en strings.xml

```
<resources>
  <!-- Facebook application ID and custom URL scheme (app ID prefixed by 'fb'). -->
  <string name="facebook_application_id" translatable="false">YOUR_APP_ID</string>
  <string name="facebook_login_protocol_scheme"
translatable="false">fbYOUR_APP_ID</string>
  <!-- Twitter consumer key and secret -->
  <string name="twitter_consumer_key" translatable="false">YOUR_CONSUMER_KEY</string>
  <string name="twitter_consumer_secret"
translatable="false">YOUR_CONSUMER_SECRET</string>
</resources>
```

5. El Acceso, el flujo de acceso de FirebaseUI consiste en crear un Intent de acceso con los métodos de acceso de nuestra preferencia:

```
// Choose authentication providers
List<AuthUI.IdpConfig> providers = Arrays.asList(
        new AuthUI.IdpConfig.EmailBuilder().build(),
        new AuthUI.IdpConfig.PhoneBuilder().build(),
        new AuthUI.IdpConfig.GoogleBuilder().build(),
        new AuthUI.IdpConfig.FacebookBuilder().build(),
        new AuthUI.IdpConfig.TwitterBuilder().build());

// Create and launch sign-in intent
startActivityForResult(
        AuthUI.getInstance()
                .createSignInIntentBuilder()
```

```
                .setAvailableProviders(providers)
                .build(),
        RC_SIGN_IN);
```

Cuando se complete el flujo de acceso debemos recibir el resultado en el método `onActivityResult`:

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

    if (requestCode == RC_SIGN_IN) {
        IdpResponse response = IdpResponse.fromResultIntent(data);

        if (resultCode == RESULT_OK) {
            // Successfully signed in
            FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
            // ...
        } else {
            // Sign in failed. If response is null the user canceled the
            // sign-in flow using the back button. Otherwise check
            // response.getError().getErrorCode() and handle the error.
            // ...
        }
    }
}
```

Para cerrar sesión FirebaseUI tambien proporiciona métodos prácticos para desconectar de todos los proveedores de identidad especificados:

```
AuthUI.getInstance()
        .signOut(this)
        .addOnCompleteListener(new OnCompleteListener<Void>() {
            public void onComplete(@NonNull Task<Void> task) {
                // ...
            }
        });
```

Autenticación con Google

De forma separada sin usar la libreria FirebaseUI es posible acceder a Firebase usando Google u otros métodos de autenticación mencionados anteriormente. En este apartado realizaremos la autenticación con Google para ello debemos añadir la dependencia `com.google.android.gms:play-services-auth` de forma manual o automática desde el menú Tools→Firebase de Android Studio

```
dependencies {
    // Import the BoM for the Firebase platform
    implementation platform('com.google.firebase:firebase-bom:26.8.0')

    // Declare the dependency for the Firebase Authentication library
    // When using the BoM, you don't specify versions in Firebase library dependencies
```

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

    // Also declare the dependency for the Google Play services library and specify
its version
    implementation 'com.google.android.gms:play-services-auth:19.0.0'
}
```

Debemos especificar la huella SHA-1 de nuestra app. Para obtener la huella podemos usar keytool:

```
C:\Users\contenidos>"c:\Program Files\Java\jdk1.8.0_281\bin\keytool.exe" -list -v -
alias androiddebugkey -keystore %USERPROFILE%\.android\debug.keystore
Introduzca la contraseña del almacén de claves:
Nombre de Alias: androiddebugkey
Fecha de Creación: 25-feb-2021
Tipo de Entrada: PrivateKeyEntry
Longitud de la Cadena de Certificado: 1
Certificado[1]:
Propietario: C=US, O=Android, CN=Android Debug
Emisor: C=US, O=Android, CN=Android Debug
Número de serie: 1
Válido desde: Thu Feb 25 07:30:32 BOT 2021 hasta: Sat Feb 18 07:30:32 BOT 2051
Huellas digitales del certificado:
         SHA1: 28:43:EE:43:7E:DB:D6:6A:64:D7:69:9B:41:33:01:2C:3A:42:C1:32
         SHA256:
61:81:F8:80:94:AB:9E:80:38:38:2C:74:F8:6F:F5:12:92:65:37:B7:9F:A1:88:80:E5:E3:5C:D8:5E
:C9:C3:93
Nombre del algoritmo de firma: SHA1withRSA (débil)
Algoritmo de clave pública de asunto: Clave RSA de 2048 bits
Versión: 1
```

o ejecutar el signReport de gradle desde android studio.

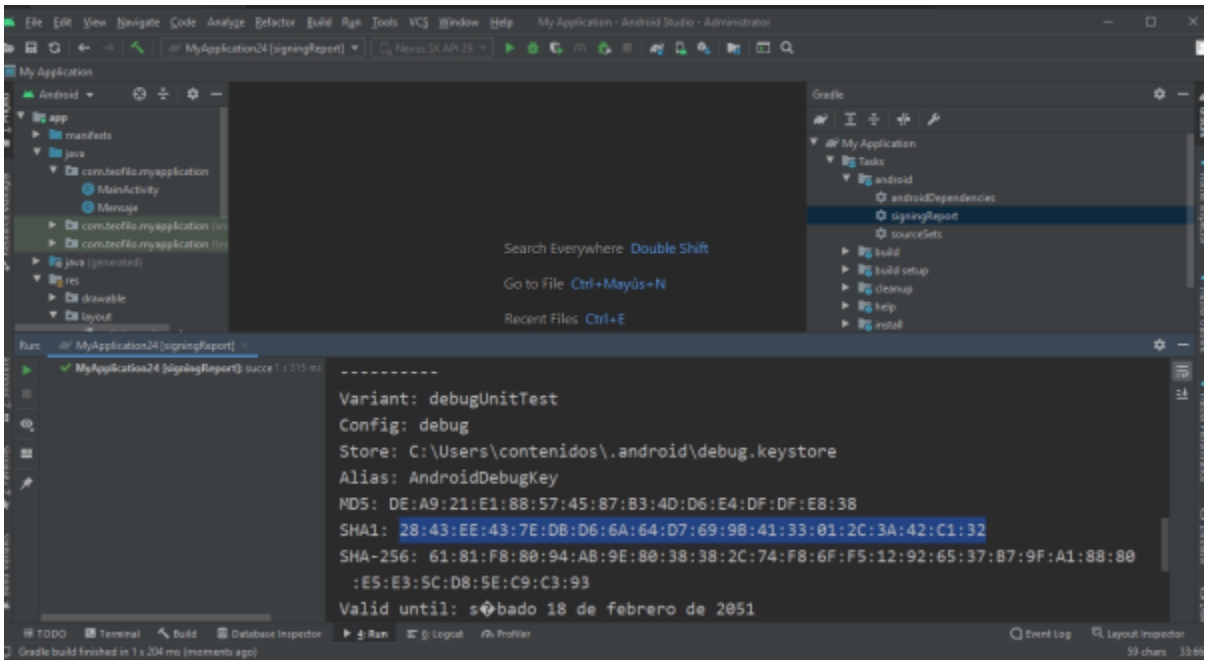

Figure 3: Información de la huella digital de nuestro almacen de claves

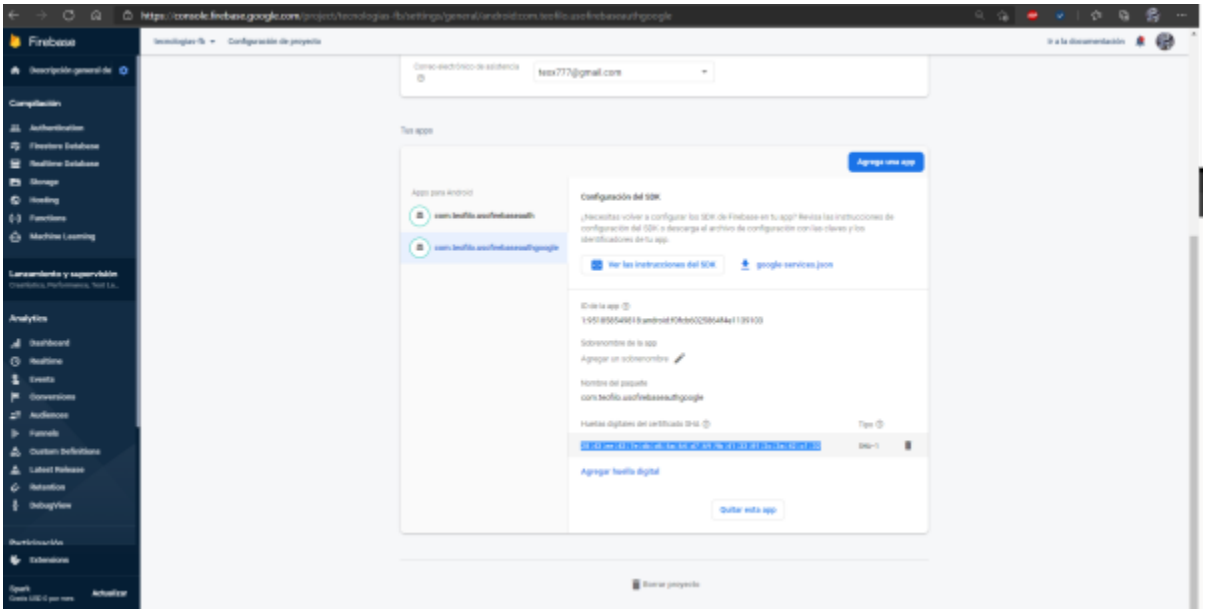

Figure 4: Agregando la huella digital a la configuración de nuestro proyecto firebase

La configuración del inicio de Google se guarda en un objeto de tipo `GoogleSignInClient`

```
private GoogleSignInClient mGoogleSignInClient;
```

Dentro del método `onCreate` configuramos el inicio de sesión de Google, dentro del cual debemos solicitar a Google Auth el `requestIdToken` para poder autentificar con el token obtenido en Firebase.

```
        GoogleSignInOptions gso = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                .requestIdToken(getString(R.string.default_web_client_id))
                .requestEmail()
                .build();
        mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
```

Donde R.string.default_web_client_id es el client_id type 3 ubicado en el archivo google-services.json, que se genera de forma automática, si es que no se generase por alguna razón, debemos copiar el valor desde el archivo google-services.json

Para lanzar el intent de inicio de sesión de google se lo puede lanzar de la siguiente forma:

```
  int RC_SIGN_IN = 100;
    private void iniciarConGoogle() {
        Intent signInIntent = mGoogleSignInClient.getSignInIntent();
        startActivityForResult(signInIntent, RC_SIGN_IN);
    }
```

y el método onActivityResult quedará de la siguiente forma:

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

```
        // Result returned from launching the Intent from
GoogleSignInApi.getSignInIntent(...);
        if (requestCode == RC_SIGN_IN) {
            Task<GoogleSignInAccount> task =
GoogleSignIn.getSignedInAccountFromIntent(data);
            try {
                // Google Sign In was successful, authenticate with Firebase
                GoogleSignInAccount account = task.getResult(ApiException.class);
                String idToken = account.getIdToken();


                //listo para autenticar con firebase despues de haber autenticado en
Google
                firebaseAuthWithGoogle(account.getIdToken()); //autentica conFirebase
            } catch (ApiException e) {
                //fallo al autenticar con google
                //actualizar la UI de acuerdo a nuestra necesidad si no se autenticó
correctamente.

            }
        }
    }
```

El método para autenticar con firebase queda de la siquiente forma:

```
 private FirebaseAuth mAuth;
    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();
                           // updateUI(user);
                            Snackbar.make(findViewById(R.id.main_content),"Bienvenido
"+user.getEmail(), BaseTransientBottomBar.LENGTH_SHORT).setAction("OK",null).show();
                        } else {
                            // If sign in fails, display a message to the user.
                            //Log.w(TAG, "signInWithCredential:failure",
task.getException());
                            Snackbar.make(findViewById(R.id.main_content),"No se pudo
autenticar a firebase con el token de google proporcionado",
BaseTransientBottomBar.LENGTH_SHORT).setAction("OK",null).show();

                            // updateUI(null);
                        }
                    }
```

```
                });
    }
```

Finalmente podemos usar el componente com.google.android.gms.common.SignInButton que representa el boton de inicio de sesión de google:

```
<com.google.android.gms.common.SignInButton
      android:id="@+id/btnGoogle"
      android:layout_width="wrap_content"
      android:layout_height="0dp"
      android:layout_marginTop="64dp"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.538"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
```

En el archivo de contexto se puede añadir un evento de tale forma que ejecute el método **iniciarConGoogle**

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

Práctico Nro. 18

Agregar el soporte de inicio de sesión con usuario y contraseña u otros métodos de autenticación al proyecto desarrollado en el laboratorio anterior. De tal forma que permite acceder a la actividad de listado de paises solo a usuarios autenticados.

https://github.com/F1852D160/FirebaseAuthEjemplo.git