



AUTENTICACIÓN CON FIREBASE

Aplicaciones Móviles

2DAM

Daniel Rodríguez Fernández

CONTENIDOS:

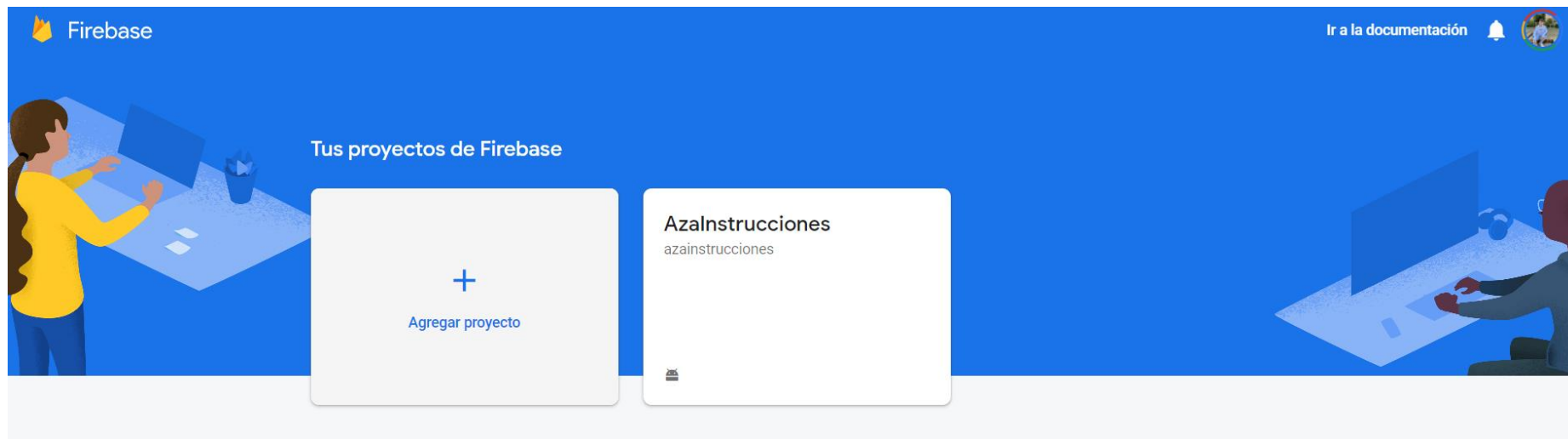
1. CREAR PROYECTO EN FIREBASE
2. CONEXIÓN DE NUESTRO PROYECTO
3. IMPLEMENTACIÓN DE DEPENDENCIAS
4. IMPLEMENTACIÓN DE CÓDIGO
5. FUNCIONAMIENTO DE LA APLICACIÓN

1. CREAR PROYECTO EN FIREBASE

1. CREAR PROYECTO EN FIREBASE

Nos vamos a la web de Firebase → <https://firebase.com>

Una vez estamos en la consola, creamos un nuevo proyecto.



1. CREAR PROYECTO EN FIREBASE

Ahora activamos los métodos de autenticación que vamos a tener disponible en nuestra aplicación. Para agregar un nuevo proveedor hacemos click sobre “Agregar proveedor nuevo”.

The screenshot displays the Firebase Authentication console interface. On the left is a dark sidebar with the 'Authentication' section selected. The main content area is titled 'Authentication' and has tabs for 'Users', 'Sign-in method' (which is active), 'Templates', 'Usage', and 'Settings'. Under the 'Sign-in method' tab, there's a section 'Proveedores de acceso' (Access providers) containing a table with two rows: 'Correo electrónico/contraseña' and 'Google', both marked as 'Habilitado' (Enabled). A blue button 'Agregar proveedor nuevo' (Add new provider) is located at the top right of this section. Below this, the 'Opciones avanzadas' (Advanced options) section is visible, featuring a card for 'Autenticación de varios factores mediante SMS' (Multi-factor authentication via SMS).

Proveedor	Estado
Correo electrónico/contraseña	Habilitado
Google	Habilitado

2. CONEXIÓN DE NUESTRO PROYECTO

2. CONEXIÓN DE NUESTRO PROYECTO

Ahora una vez hemos creado nuestro proyecto en Firebase vamos a conectar nuestra aplicación móvil.

Para ello seguimos la siguiente ruta desde el menú: Tools > Firebase > Authentication

Una vez estamos aquí hacemos click sobre “Authentication using Google[Kotlin]”



2. CONEXIÓN DE NUESTRO PROYECTO

Ahora vamos al paso 1 de conexión y hacemos click sobre “Connect” y después sobre “Add the firebase Authentication SDK to you API”.

Una vez esta conectado nos mostrará un Check de Connected.

Authenticate using Google [KOTLIN]

You can let your users authenticate with Firebase using their Google Accounts.

[Launch in browser](#)

① Connect your app to Firebase

✔ Connected

② Add the Firebase Authentication SDK to your app

[ADD THE FIREBASE AUTHENTICATION SDK TO YOUR APP](#)

NOTE: After adding the SDK, here are some other helpful configurations to consider:

- **Do you want an easier way to manage library versions?**
You can use the [Firebase Android BoM](#) to manage your Firebase library versions and ensure that your app is always using compatible library versions.

③ Complete your set up in the Firebase console

- If you haven't yet, specify your app's SHA-1 fingerprint. You can do this in your **Project settings** in the Firebase console. Refer to [Authenticating Your Client](#) for details on how to get your app's SHA-1 fingerprint.
- To use an authentication provider, you need to enable it for your Firebase project. Enable [Google](#) in the Sign-in method tab of the Firebase Authentication section of the Firebase console.

④ Integrate Google sign-in into your app

Integrate Google One Tap sign-in into your app by following the steps on the [Sign users in with their saved credentials](#) page. When you configure the `BeginSignInRequest` object, call `setGoogleIdTokenRequestOptions`:

```
signInRequest = BeginSignInRequest.builder()  
    .setGoogleIdTokenRequestOptions()
```


3. IMPLEMENTACIÓN DE DEPENDENCIAS

3. IMPLEMENTACIÓN DE DEPENDENCIAS

Implementamos las siguientes dependencias en build.gradle “Module”:

```
dependencies {  
    implementation 'androidx.core:core-ktx:1.7.0'  
    implementation 'androidx.appcompat:appcompat:1.5.1'  
    implementation 'com.google.android.material:material:1.7.0'  
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'  
    implementation 'com.google.firebase:firebase-auth-ktx:21.1.0'  
    implementation 'com.google.firebase:firebase-analytics-ktx:21.2.0'  
    testImplementation 'junit:junit:4.13.2'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'  
    // FirebaseUI for Firebase Auth  
    implementation 'com.firebaseui:firebase-ui-auth:8.0.2'  
    implementation 'com.google.android.gms:play-services-auth:20.4.0'  
    implementation 'androidx.core:core-ktx:+'  
    // implementation 'com.google.firebase:firebase-auth-ktx:21.1.0'  
}
```

3. IMPLEMENTACIÓN DE DEPENDENCIAS

Ahora en el archivo de build.gradle “Project” implementamos los siguientes plugins:

```
plugins {  
    id 'com.android.application' version '7.3.1' apply false  
    id 'com.android.library' version '7.3.1' apply false  
    id 'org.jetbrains.kotlin.android' version '1.8.0' apply false  
    id 'com.google.gms.google-services' version '4.3.15' apply false  
}  
  
task clean(type: Delete) { delete rootProject.buildDir }
```

4. IMPLEMENTACIÓN DE CÓDIGO

4. IMPLEMENTACIÓN DE CÓDIGO

Durante este proyecto vamos a trabajar solo con la clase Main.

Lo primero que hacemos es implementar las variables necesarias para el binding y en inicio de sesión.

```
//Variable para el binding
private lateinit var binding: ActivityMainBinding

//Variable para inicio de sesion
private lateinit var firebaseAuth: FirebaseAuth
private lateinit var authStateListener: FirebaseAuth.AuthStateListener
```

4. IMPLEMENTACIÓN DE CÓDIGO

Ahora creamos los onCreate para crear el binding y el inicio de la autenticación.

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    binding = ActivityMainBinding.inflate(layoutInflater)  
    setContentView(binding.root)  
    configAuth() //Configuración del método de autenticación  
}
```

4. IMPLEMENTACIÓN DE CÓDIGO

A continuación implementamos el método `configAuth` que nos sirve para inicializar las variables de autenticación y los listeners.

```
private fun configAuth() {  
    //inicialiar las variables  
    firebaseAuth = FirebaseAuth.getInstance()  
  
    //iniciamos el listener para cuando nos autentiquemos  
    authStateListener = FirebaseAuth.AuthStateListener {  
  
        if (it.currentUser != null) { //si el usuario ya esta autenticado  
            supportActionBar?.title =  
                it.currentUser?.displayName //ponemos el nombre del usuario en la toolbar  
            binding.textInit.visibility = View.VISIBLE //haer visible...  
        } else {  
            //si el usuario no esta autenticado entonces  
            //crear la lista de todas las formas de autenticacion  
            val providers = arrayListOf(  
                AuthUI.IdpConfig.EmailBuilder().build(), //email  
                AuthUI.IdpConfig.GoogleBuilder().build()  
            ) //google  
  
            //lanzar el intent para mostrar todas las formas de logueo  
            resultLauncher.launch{//este bloque es el intent para mostrar el logeado  
                AuthUI.getInstance()  
                    .createSignInIntentBuilder()  
                    .setAvailableProviders(providers)  
                    .setIsSmartLockEnabled(false)  
                    .build()  
            }  
        }  
    }  
}
```

4. IMPLEMENTACIÓN DE CÓDIGO

Ahora implementamos el intent donde lanzamos la autenticación y comprobamos que se ha realizado de forma correcta.

```
private var resultLauncher =
    registerForActivityResult(ActivityResultContracts.StartActivityForResult()) {
        val response = IdpResponse.fromResultIntent(it.data)

        if (it.resultCode == RESULT_OK) {
            val user = FirebaseAuth.getInstance().currentUser //datos del usuario identificado
            if (user != null) {
                Toast.makeText(this, "Bienvenido", Toast.LENGTH_SHORT).show()
            }
        } else {
            if (response == null) { //el usuario a pulsado hacia atras para salir de la APP
                Toast.makeText(this, "Adios....", Toast.LENGTH_SHORT).show()
                finish()
            } else { //se debe tratar los errores de conexion
                response.error?.let {
                    if (it.errorCode == ErrorCodes.NO_NETWORK) {
                        Toast.makeText(this, "Sin red", Toast.LENGTH_SHORT).show()
                    } else {
                        Toast.makeText(
                            this,
                            "Código de error: ${it.errorCode}",
                            Toast.LENGTH_SHORT
                        ).show()
                    }
                }
            }
        }
    }
}
```


4. IMPLEMENTACIÓN DE CÓDIGO

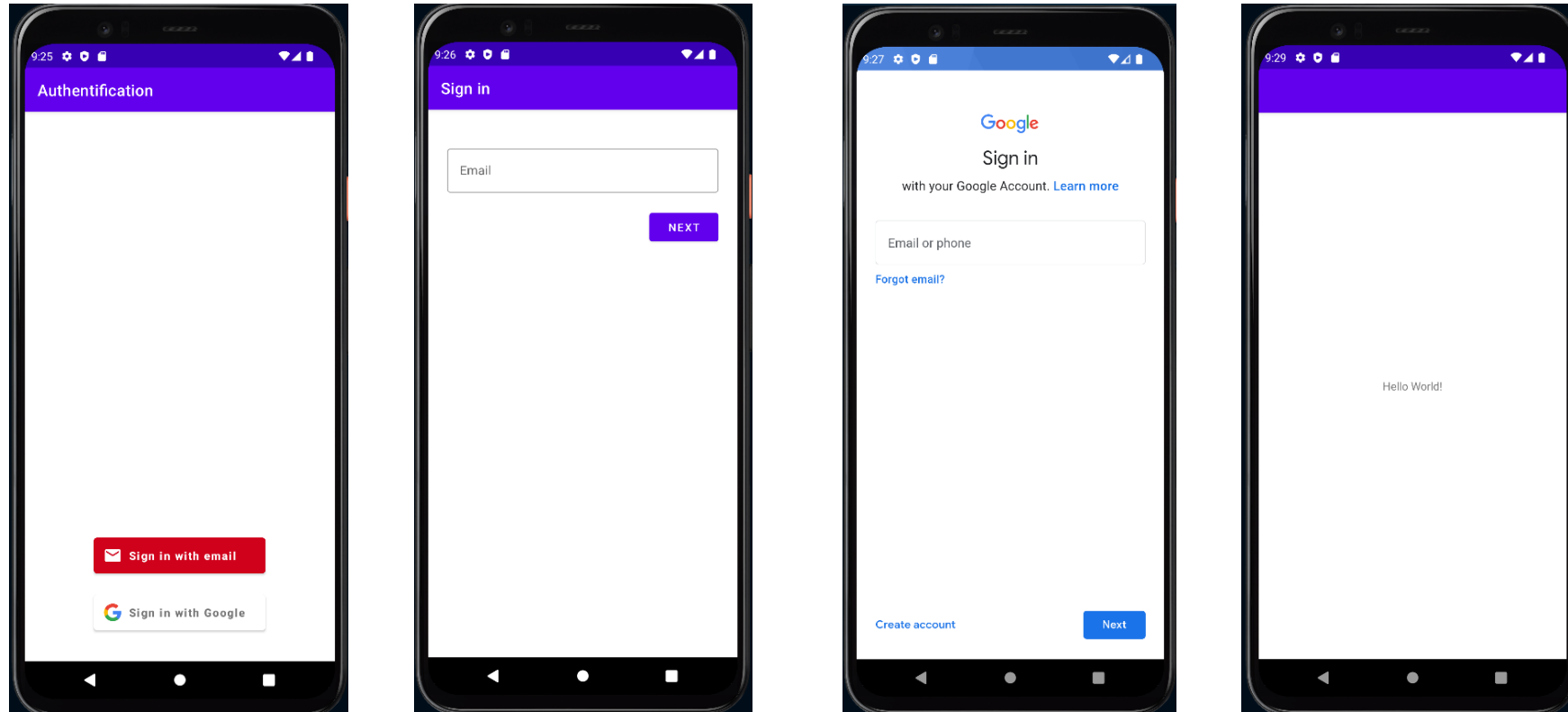
Tenemos que tener en cuenta que una aplicación puede tener varios ciclos de vida, por lo tanto implementamos los métodos necesarios para controlar los ciclos de vida de nuestra aplicación.

```
override fun onResume() {  
    super.onResume()  
    firebaseAuth.addAuthStateListener(authStateListener)  
}  
  
override fun onPause() {  
    super.onPause()  
    firebaseAuth.removeAuthStateListener(authStateListener)  
}
```

5. FUNCIONAMIENTO DE LA APLICACIÓN

5. FUNCIONAMIENTO DE LA APLICACIÓN

Aquí podemos ver el funcionamiento de la aplicación y su utilización:



¡GRACIAS!

Sigue mis proyectos:

GitHub: <https://github.com/idanirf>

Linkedin:

<https://www.linkedin.com/in/danielrodriguezfernandez03002/>

Web: <https://idanirf.github.io>