

# Autenticación (Anexo)

Descargar estos apuntes [pdf](#) o [html](#)

## Índice

- [Introducción](#)
- ▼ [Modificando el Login para usar Firebase Authentication](#)
  - [Activación del servicio de Authentication](#)
  - [Añadiendo permisos en el `manifest.xml`](#)
  - [Dependencias y Plugins de en Gradle](#)
- ▼ [Modificando el proyecto](#)
  - [Definiendo el proveedor de Firestore con Dagger-Hilt](#)
  - [Implementación con Email y Contraseña](#)
  - [Implementación con varios métodos de acceso](#)
  - [Implementación métodos acceso](#)
  - [Implementación con Google](#)
  - [Implementación con Facebook](#)
  - [Manejo de Sesión en la Aplicación](#)
  - [Cerrar sesión](#)
  - [Restablecer contraseña](#)

# Introducción

- **Autenticación con Firebase Authentication**
  - Documentación oficial: [Firebase documentation](#)
  - Video Tutorial (Inglés): [Firebase](#)
  - Video Tutorial (Inglés): [Philipp Lackner](#)
  - Video Tutorial (Castellano): [AristiDevs](#)
  - Video Tutorial (Castellano): [DevExpert](#)

Firebase Authentication es un servicio que forma parte del ecosistema de Firebase, diseñado para facilitar la implementación de autenticación en aplicaciones móviles y web. Permite gestionar de manera sencilla y segura el acceso de los usuarios a tu aplicación mediante diversas opciones de autenticación.

Con Firebase Authentication, puedes autenticar a los usuarios a través de métodos populares como:

- **Correo electrónico y contraseña:** Los usuarios pueden registrarse y acceder a tu aplicación usando una dirección de correo electrónico y una contraseña.
- **Autenticación de Google, Facebook, Twitter y otros proveedores:** Firebase simplifica la integración con proveedores de identidad como Google, Facebook, Twitter, GitHub, entre otros.
- **Autenticación anónima:** Para aquellos usuarios que prefieren interactuar con la aplicación sin registrarse, Firebase permite el acceso temporal sin necesidad de crear una cuenta.
- **Autenticación con teléfono (SMS):** Los usuarios pueden autenticar su identidad mediante el envío de un código por mensaje de texto, lo cual es útil especialmente en regiones donde los métodos de correo electrónico son menos populares.
- **Autenticación con OAuth y proveedores personalizados:** Firebase también ofrece la posibilidad de integrar autenticación con otros proveedores de identidad personalizados mediante OAuth.

Firebase Authentication se usa para gestionar usuarios, almacenar contraseñas de forma segura y personalizar la experiencia, además de manejar verificación de correo, recuperación de contraseñas y administración de sesiones.

Nosotros nos vamos a centrar en Firebase Authentication, específicamente en la implementación de autenticación mediante correo electrónico y contraseña, así como con proveedores de identidad como Google y Facebook, ya que son las formas de autenticación más utilizadas en aplicaciones modernas.

# Modificando el Login para usar Firebase Authentication

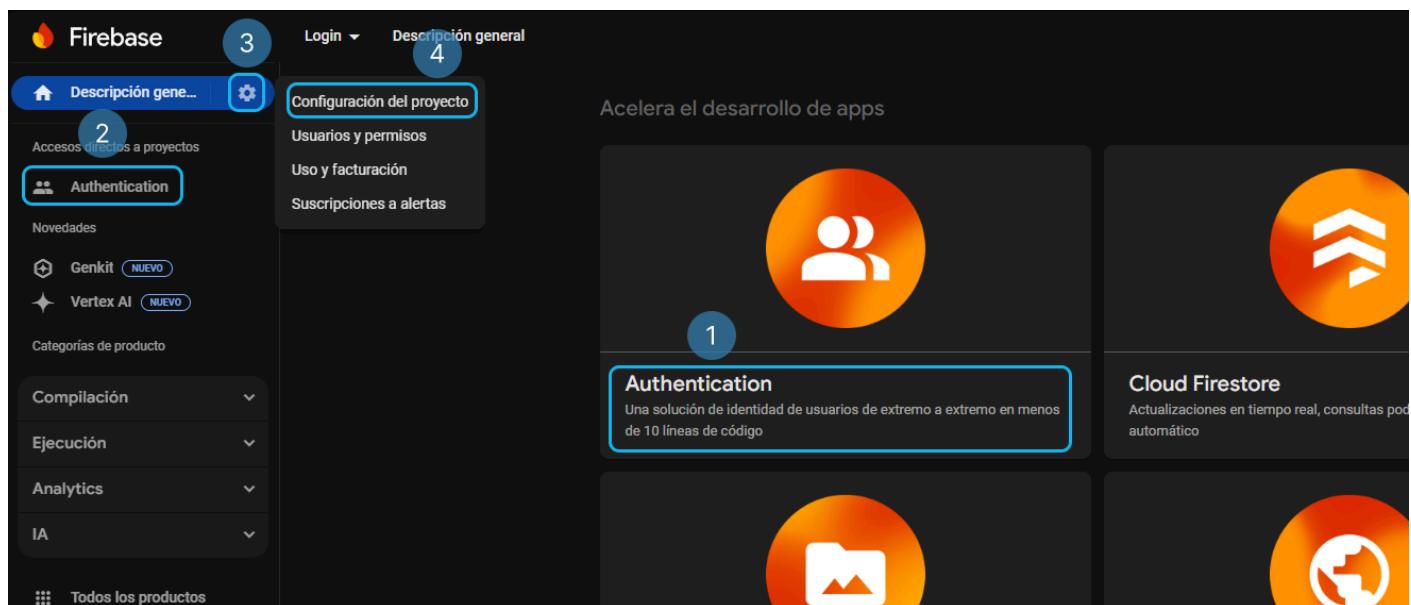
## i Proyecto

Puedes descargar el proyecto del login con todo el código visto en el tema desde el siguiente enlace [LoginFirebase](#)

## Activación del servicio de Authentication

- [Documentación Oficial](#)
- [API de Authentication en Kotlin](#)

Es un servicio flexible y escalable para la gestión de usuarios en aplicaciones móviles, web y servidores. Permite la autenticación mediante múltiples métodos como correo electrónico y contraseña, así como proveedores de identidad como Google y Facebook. Ofrece una integración sencilla y segura, gestionando tareas como la verificación de correo electrónico, recuperación de contraseñas y autenticación multifactor (MFA), todo ello con soporte para múltiples plataformas.



Una vez hemos accedido a nuestro proyecto:

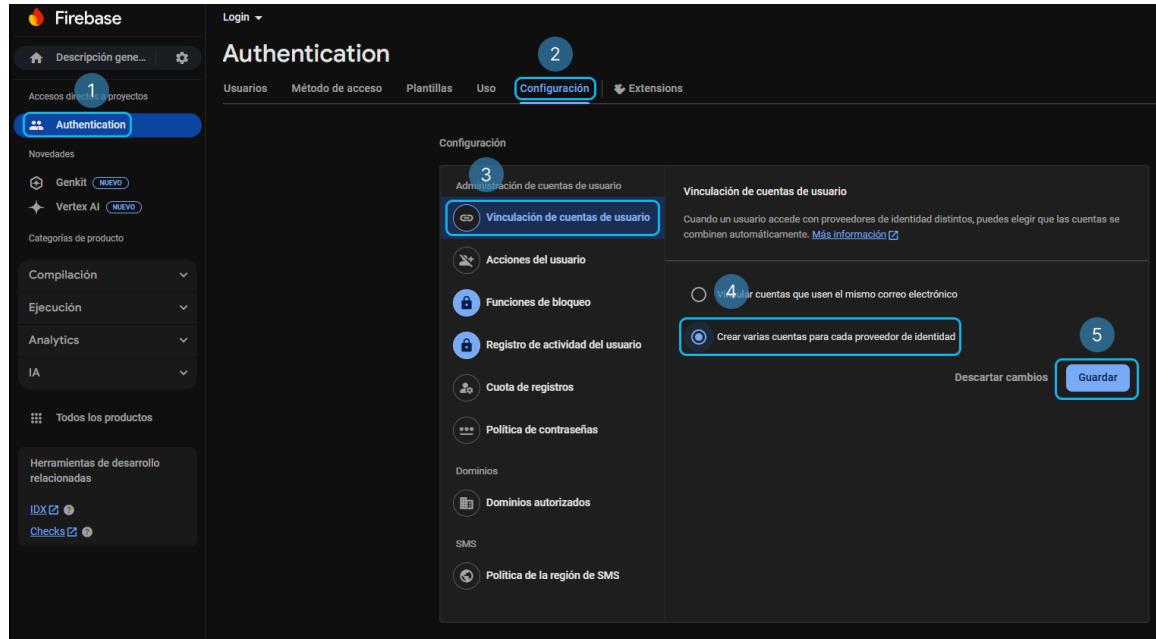
- Marcamos la opción **Authentication** 1 .
- Nos aparecerá la opción 2 de Authentication, y al pulsar seleccionamos Comenzar, lo que nos llevará a los métodos de acceso. Más adelante, veremos cómo agregar los métodos de acceso de Google y Facebook.

- En **3** y **4** debemos acceder a la configuración de nuestro proyecto y desde la misma **debemos** volver a descargar el archivo de configuración **google-services.json**, ya que, si no lo hacemos, no funcionará al intentar registrarse o iniciar sesión.

### Aviso

Al utilizar varios métodos de acceso, si primero registras una cuenta con email y contraseña (por ejemplo, [pepe@gmail.com](mailto:pepe@gmail.com)) y luego intentas registrarla nuevamente mediante el servicio de Google, obtendrás un error.

Para solucionar este problema, debemos seguir estos pasos:



## Añadiendo permisos en el `manifest.xml`

Puesto que Authentication es una BD en la nube, vamos a necesitar permisos de internet. Pare ello, en el archivo `AndroidManifest.xml` deberemos añadir el permiso de acceso a internet para que el servicio pueda acceder al API.

```

<manifest ...>
    ...
    3   <uses-permission android:name="android.permission.INTERNET"/>
    5   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    ...
    9   <application ...>
        <!-- Permitir tráfico http en lugar de https -->
        android:usesCleartextTraffic="true"
        ...
    </application>
</manifest>

```

## Dependencias y Plugins de en Gradle

En el catálogo de versiones `lib.versions.toml` deberemos comprobar que hemos definido tener:

```

# Recuerda además que las entradas van en una sola línea.
# Algunas se han indentado para que se vean mejor.

[versions]
googleServices = "4.4.2"
firebaseBom = "33.11.0"

[libraries]
google-firebase-bom = {
    group = "com.google.firebaseio", name = "firebase-bom", version.ref = "firebaseBom"
}
firebase-auth-ktx = { module = "com.google.firebaseio:firebase-auth-ktx" }

[plugins]
com-google-services = { id = "com.google.gms.google-services", version.ref = "googleServices"

```

En el `build.gradle.kts` raíz del proyecto añadiremos el siguiente plugin:

```

plugins {
    alias(libs.plugins.com.google.services) apply false
}

```

En el `build.gradle.kts` del **módulo de la aplicación** (app) añadiremos:

```
plugins {
    alias(libs.plugins.com.google.services)
}

...
dependencies {
    implementation(platform(libs.google.firebaseio.bom))
    implementation(libs.firebaseio.auth.ktx)
}
```

# Modificando el proyecto

## Definiendo el proveedor de Firestore con Dagger-Hilt

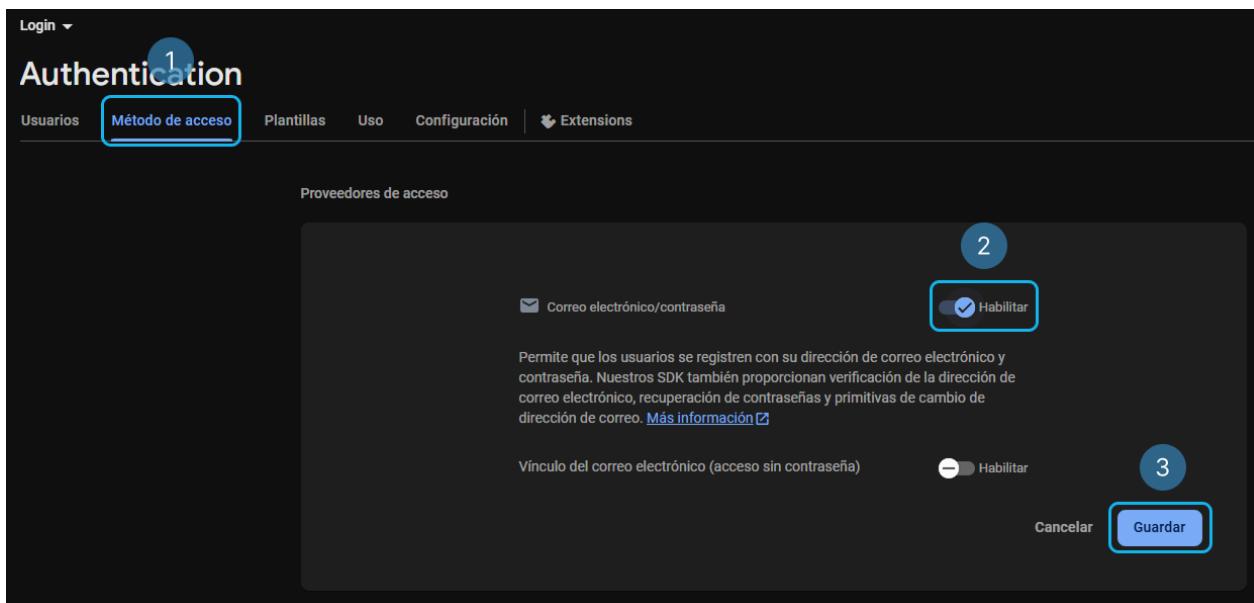
Definiremos el proveedor de Firestore en el archivo  `AppModule.kt` que me creará una instancia de Firestore de acuerdo a la configuración de mi proyecto definida en el archivo `google-services.json` y me la inyectará con **Dagger-Hilt**.

```
@Module
@ModuleIn(SingletonComponent::class)
class AppModule {
    ...
    @Singleton
    @Provides
    fun provideFirebaseAuth(): FirebaseAuth = FirebaseAuth.getInstance()
}
```

## Implementación con Email y Contraseña

### Configuración en Firebase Console

Para utilizar este método de acceso, es necesario activarlo desde la consola de Firebase. A continuación, te explicamos cómo hacerlo:



### Definiendo las clases para operar con Authentication

Definiremos el **paquete** `data.services.authentication` en nuestro proyecto, donde implementaremos el servicio de autenticación que proporcionará las operaciones necesarias para

gestionar el acceso de los usuarios.

Dentro de este paquete, crearemos las clases que manejarán la autenticación en Firebase. Estas incluirán la definición de la interfaz del servicio, su implementación y la gestión de excepciones relacionadas con la autenticación.

La estructura del paquete quedará organizada de la siguiente manera:



Con esta estructura, centralizaremos la lógica de autenticación, facilitando la gestión de usuarios mediante Firebase Authentication.

## Métodos de autenticación con email y contraseña

```
class AuthService @Inject constructor(
    private val firebaseAuth: FirebaseAuth,
    @ApplicationContext private val context: Context
) {
    ...
    fun signUp(email: String, password: String): Task<AuthResult> =
        firebaseAuth.createUserWithEmailAndPassword(email, password)
    fun signIn(email: String, password: String): Task<AuthResult> =
        firebaseAuth.signInWithEmailAndPassword(email, password)
    fun logOut() {
        firebaseAuth.signOut()
    }
}
```

En esta clase **AuthService**, definimos dos métodos clave para la autenticación de usuarios mediante **email y contraseña** usando Firebase:

- **signup** : Este método se encarga de registrar a un nuevo usuario en Firebase utilizando su email y contraseña. Utiliza el método `createUserWithEmailAndPassword` de FirebaseAuth, el cual devuelve una tarea (Task) que, al completarse, indica si el registro fue exitoso.
- **signIn** : Este método permite que un usuario existente inicie sesión con su email y contraseña. Utiliza el método `signInWithEmailAndPassword` de FirebaseAuth, también devolviendo una tarea ( Task ) que nos informa si la autenticación fue correcta.

Ambos métodos devuelven un objeto `Task<AuthResult>`, lo que nos permite manejar el resultado de las operaciones de autenticación y tomar acciones según si fueron exitosas o no.

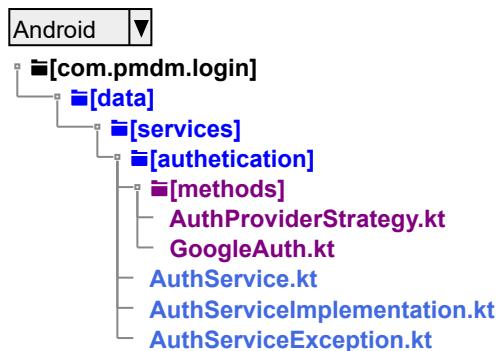
```
class AuthServiceImplementation @Inject constructor(
    private val authService: AuthService
) {
    private val logTag: String = "AuthServiceImplementation"
    ...
    suspend fun signIn(email: String, password: String): FirebaseUser {
        val errorMessage: String = "Error while trying to sign in."
        try {
            val response: Task<AuthResult> = authService.signIn(email, password)
            val user: FirebaseUser? = response.await().user
            return user ?: throw AuthServiceException(errorMessage)
        } catch (e: Exception) {
            Log.e(logTag, "Error: ${e.localizedMessage}")
            throw AuthServiceException(errorMessage)
        }
    }

    suspend fun signUp(email: String, password: String): FirebaseUser {
        val errorMessage: String = "Error while trying to sign up."
        try {
            val response: Task<AuthResult> = authService.signUp(email, password)
            val user: FirebaseUser? = response.await().user
            return user ?: throw AuthServiceException(errorMessage)
        } catch (e: Exception) {
            Log.e(logTag, "Error: ${e.localizedMessage}")
            throw AuthServiceException(errorMessage)
        }
    }
}
```

## Implementación con varios métodos de acceso

Ahora definiremos el **patrón Strategy** para poder, en el futuro. Este patrón nos permite reducir código, eliminar duplicación y adaptar el código de manera eficiente cuando necesitemos implementar métodos de autenticación como `Google` o `Facebook`. De esta forma, el código será

más modular, fácil de mantener y escalar, permitiendo añadir nuevos proveedores de autenticación sin grandes cambios en la estructura.



Mediante esta clase abstracta `AuthProviderStrategy`, definimos la estructura básica para los diferentes métodos de acceso, como Google, Facebook, etc. Como la mayoría de sus métodos son comunes, podemos evitar la repetición de código. La clase contiene un método abstracto `getCredential(idToken)` que debe ser implementado por las clases hijas para obtener las credenciales específicas de cada proveedor de autenticación.

### Nota

Se recomienda ejecutar la aplicación en un dispositivo real en lugar de un emulador, ya que este último suele presentar problemas.

Además, incluye un método `signIn(credential)` que maneja el proceso de inicio de sesión de manera uniforme, utilizando un `Task` de Firebase. Si la autenticación es exitosa, devuelve el `FirebaseUser`, y si ocurre un error, lanza una excepción personalizada (`AuthServiceException`) con un mensaje de error predefinido.

```

abstract class AuthProviderStrategy(
    protected val errorMessage: String
) {
    abstract fun getCredential(idToken: String): AuthCredential
    suspend fun signIn(credential: Task<AuthResult>): FirebaseUser {
        return try {
            val user: FirebaseUser? = credential.await().user

            user ?: throw AuthServiceException(errorMessage)
        } catch (e: Exception) {
            throw AuthServiceException(errorMessage)
        }
    }
}

```

## Implementación métodos acceso

```

class AuthService @Inject constructor(
    private val firebaseAuth: FirebaseAuth,
    @ApplicationContext private val context: Context
) {
    fun getCredential(idToken: String, authProvider: AuthProviderStrategy): Task<AuthResult>
        val credential: AuthCredential = authProvider.getCredential(idToken)

        return firebaseAuth.signInWithCredential(credential)
    }
}

```

```

class AuthServiceImplementation @Inject constructor(
    private val authService: AuthService
) {
    suspend fun signInWithProvider(
        idToken: String,
        authProvider: AuthProviderStrategy
): FirebaseUser {
        val credential = authService.getCredential(idToken, authProvider)

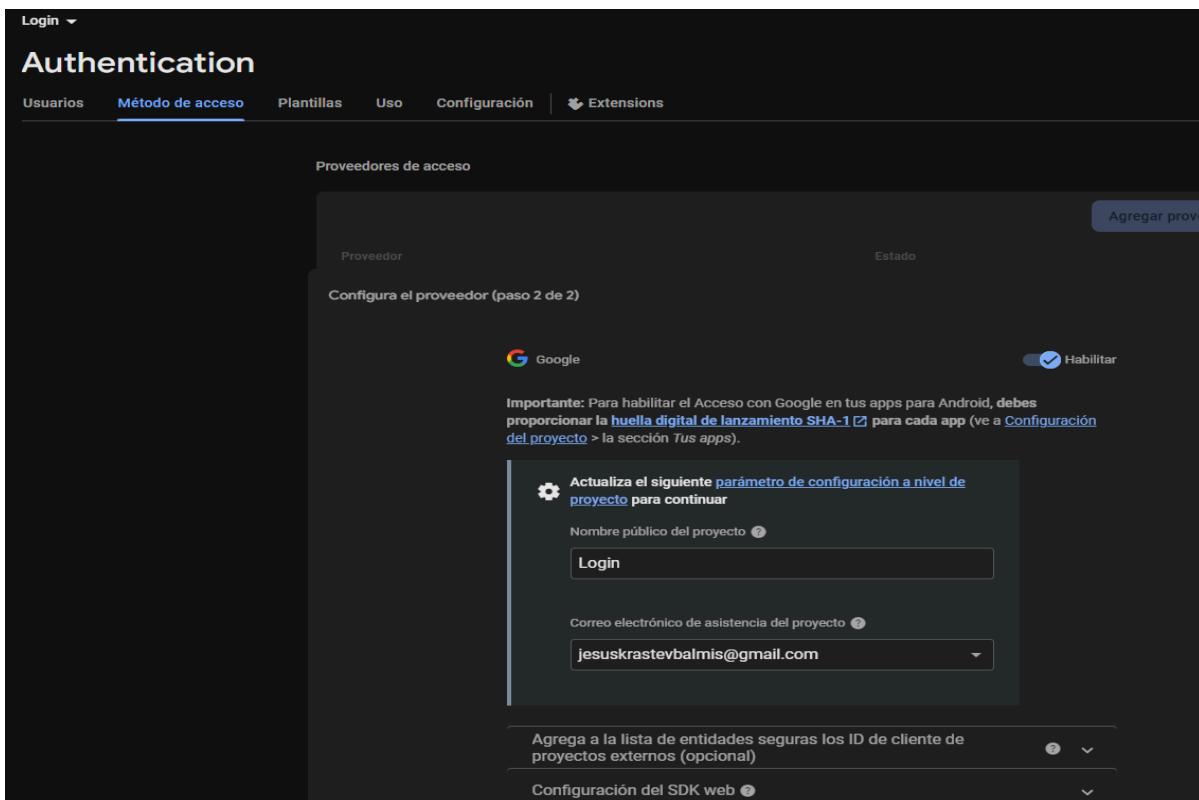
        return authProvider.signIn(credential = credential)
    }
}

```

# Implementación con Google

## Configuración en Firebase Console

Al agregar el método de acceso de Google, debes pulsar en Agregar Nuevo Proveedor, seleccionar la opción Google y luego elegir el correo por defecto que nos permite Firebase. Nos pedirá volver a descargar el fichero de configuración `google-services.json` para poder usar su método de acceso. En la siguiente imagen puedes observar cómo se activa este método de acceso:



## Dependencias y Plugins de en Gradle

```
[versions]
playServices = "21.3.0"

[libraries]
play-services-auth = { module = "com.google.android.gms:play-services-auth", version.ref = "p
```

En el `build.gradle.kts` del **módulo de la aplicación** (app) añadiremos:

```
...
dependencies {
    implementation(libs.play.services.auth)
}
```

## Definiendo los métodos

En este caso, debemos definir un launcher para poder usar la autenticación de **Google**. A este launcher le tenemos que pasar un método para gestionar el inicio de sesión desde el **ViewModel**. Además, debemos definir otros dos métodos: uno para manejar el inicio de sesión cuando el usuario ya tiene una cuenta registrada, y otro para registrar al usuario en caso contrario.

```
@Composable
fun registerGoogleLauncher(
    signInWithGoogle: (idToken: String, onSignIn: () -> Unit, onSignUp: () -> Unit) -> Unit,
    onSignIn: () -> Unit,
    onSignUp: () -> Unit
): ManagedActivityResultLauncher<Intent, ActivityResult> {
    return rememberLauncherForActivityResult(ActivityResultContracts.StartActivityForResult()) {
        if (result.resultCode == Activity.RESULT_OK) {
            val task: Task<GoogleSignInAccount> =
                GoogleSignIn.getSignedInAccountFromIntent(result.data)
            try {
                val account: GoogleSignInAccount = task.getResult(ApiException::class.java)!!
                signInWithGoogle(account.idToken!!, onSignIn, onSignUp)
            } catch (e: Exception) {
                e.printStackTrace()
                Log.d("GoogleLauncher", "Error: ${e.message}")
            }
        } else {
            Log.d("GoogleLauncher", "Error: ${result.resultCode}")
        }
    }
}
```

Creamos la clase encargada de implementar la autenticación con Google.

```
class GoogleAuth(
    errorMessage: String = "Error while trying to sign in with Google."
): AuthProviderStrategy(errorMessage) {
    override fun getCredential(idToken: String): AuthCredential = GoogleAuthProvider.getCredential(idToken, null)
}
```

Es necesario agregar un método en la clase AuthService para obtener la cuenta de Google al iniciar el launcher.

```
class AuthService @Inject constructor(
    private val firebaseAuth: FirebaseAuth,
    @ApplicationContext private val context: Context
) {
    fun getGoogleAccount(): GoogleSignInClient {
        val idToken = GoogleSignInOptions
            .Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
            .requestIdToken(context.getString(R.string.default_web_client_id))
            .requestEmail()
            .build()

        return GoogleSignIn.getClient(context, idToken)
    }
}
```



### Nota

El string resource `default_web_client_id` generará un error. Para solucionarlo, es necesario compilar el proyecto, lo que permitirá que el recurso se genere automáticamente.

```
class AuthServiceImplementation @Inject constructor(
    private val authService: AuthService
) {
    ...
    fun getGoogleAccount(): GoogleSignInClient {
        val errorMessage: String = "Error while trying to get Google account."

        try {
            return authService.getGoogleAccount()
        } catch (e: Exception) {
            Log.e(logTag, "Error: ${e.localizedMessage}")
            throw AuthServiceException(errorMessage)
        }
    }
}
```

# Implementación con Facebook

## Aviso

La autenticación con Facebook puede ser **compleja**, ya que requiere varios pasos y, además, la documentación está completamente **desactualizada**. En esta guía, te guiaremos paso a paso para implementarla de manera que todo funcione correctamente.

Documentación oficial:

- [Firebase](#)
- [Facebook](#)

## Configuración en Firebase Console

Para que funcione el inicio de sesión con Facebook, primero debes configurar el SHA1 en la consola de Firebase. Para ello, ejecuta el siguiente comando en la terminal del proyecto o corre una tarea de Gradle.

```
$ ./gradlew signingReport
```

Al ejecutarlo, se mostrará un resultado. Debes copiar el código SHA1.

The screenshot shows the Firebase console interface. On the left, there's a sidebar with various project management options like 'Descripción general', 'Accesos directos a proyectos', 'Authentication', 'Firestore Database', 'Novedades', 'Genkit', 'Vertex AI', and 'Categorías de producto'. The main area is titled 'Configuración de proyecto' and shows a list of users ('jesuskrastev@almis.com'). Below this, it lists 'Tus apps' and shows details for an 'App para Android' named 'Login' with package name 'com.pmdm.login'. It includes sections for 'Configuración del SDK' (with a link to instructions and a download button for 'google-services.json'), 'ID de la app' (showing '1:1059052377624:android:bb0eda93322cca6d3956d'), 'Sobrenombre de la app' (set to 'Login'), 'Nombre del paquete' (set to 'com.pmdm.login'), and 'Huellas digitales del certificado SHA' (listing the SHA1 key 'df:5c:2d:fb:13:d3:c1:8a:da:7d:0f:3f:d8:78:c5:65:18:14:ec' with a note that it's SHA-1). There are also buttons for 'Agregar huella digital' and 'Quitar esta app'.

## Configuración Facebook y Gradle

Para comenzar, necesitamos registrarnos en la plataforma **Facebook Developers**. Una vez registrados, crearemos un nuevo proyecto y lo configuraremos. Luego, en Firebase, agregaremos el proveedor de Facebook.

Detalles de la aplicación Casos de uso Empresa Finalizar

**Nombre de la aplicación**  
Este es el nombre de la aplicación que se mostrará en la página "Mis aplicaciones" y que estará asociado a tu identificador de la aplicación. Puedes cambiarlo más tarde en la configuración.

Login 5/30

**Correo electrónico de contacto de la aplicación**  
Esta es la dirección de correo electrónico que usaremos para ponernos en contacto contigo en relación con tu aplicación. Asegúrate de añadir una dirección que consultes habitualmente. Es posible que te contactemos para informarte sobre políticas, restricciones de la aplicación o formas de recuperar la aplicación si esta se ha eliminado o está en peligro.

jesuskrastevbalmis@gmail.com

Cancelar Siguiente

**Crear una aplicación**

Detalles de la aplicación Casos de uso Empresa Finalizar

**Añadir casos de uso**

**Create & manage app ads with Meta Ads Manager**  
Promote your mobile app and drive installs. Create and manage campaigns that encourage users to download and install your app. Does not include access to Marketing API. [Obtén más información.](#)

**Acceder a la API de Threads**  
Usa la API de Threads para autenticar usuarios, recuperar información de usuario, publicar hilos, responder a hilos, administrar la configuración de respuestas o recopilar insights para un perfil de Threads que poseas o que administres en nombre de otras personas. [Obtén más información.](#)

**Integrar contenidos de Threads en otros sitios web**  
Usa la API de Threads oEmbed para integrar el contenido de las publicaciones de Threads, como fotos y vídeos, en otros sitios web. [Obtén más información.](#)

**Inicia un juego en Facebook**  
Inicia un juego que los jugadores puedan encontrar y disfrutar directamente en su feed o en las conversaciones de Messenger, tanto en el ordenador como en dispositivos móviles. [Obtén más información.](#)

**El inicio de sesión con Facebook te permite solicitar y autenticar los datos de los usuarios**  
Nuestro caso de uso más habitual. Una forma segura y rápida de que los usuarios inicien sesión en tu aplicación (o juego) y esta les pida permiso para acceder a sus datos y personalizar así su experiencia. [Obtén más información.](#)

**¿Buscas otra cosa?**  
Si necesitas alguna opción que no aparece más arriba, selecciona "Otro" para ver más.

Otro Tu app se creará en la experiencia anterior. Luego deberás elegir entre todos los permisos, funciones y productos disponibles.

Cancelar Anterior Siguiente

## Selecciona un tipo de aplicación

Una vez que hayas creado la aplicación, ya no podrás cambiar su tipo. [Más información](#)

### Empresa



Crea o administra activos comerciales como páginas, eventos, grupos, anuncios, Messenger, WhatsApp e Instagram mediante los permisos, las funciones y los productos empresariales disponibles.

### Consumidor



Conecta a tu aplicación permisos y productos para el consumidor, como el inicio de sesión con Facebook.

[Cancelar](#)

[Siguiente](#)

## Nombre de la aplicación

Este es el nombre de la aplicación que se mostrará en la página "Mis aplicaciones" y que estará asociado a tu identificador de la aplicación. Puedes cambiarlo más tarde en la configuración.

Login

5/30

## Correo electrónico de contacto de la aplicación

Esta es la dirección de correo electrónico que usaremos para ponernos en contacto contigo en relación con tu aplicación. Asegúrate de añadir una dirección que consultes habitualmente. Es posible que te contactemos para informarte sobre políticas, restricciones de la aplicación o formas de recuperar la aplicación si esta se ha eliminado o está en peligro.

jesuskrastevbalmis@gmail.com

## Porfolio empresarial · Opcional

Conectar un porfolio empresarial a tu aplicación solo es necesario para determinados productos y permisos. Te pediremos que conectes un porfolio empresarial cuando solicites acceso a dichos productos y permisos.

No se ha seleccionado ningún porfolio empresarial



Al continuar, aceptas las [Condiciones de la plataforma de Meta](#) y las [Políticas para desarrolladores](#).

[Cancelar](#)

[Anterior](#)

[Crear aplicación](#)

Ve a Información Básica, copia el Identificador de la aplicación y pégalo en la consola de Firebase.

Luego, haz clic en Mostrar, copia el texto que aparece y pégalo en el último campo de Firebase Console. Finalmente, haz clic en Guardar.

Login ▾ Identificador de la aplicación 994286956135230 Modo de la aplicación: En desarrollo En producción Tipo de aplicación: Consumidor

**1** Configuración de la apl... ▾

**2** Información básica

Identificador de la aplicación  
994286956135230

Nombre para mostrar  
Login

Dominios de la aplicación

Espacio de nombres

Correo electrónico de contacto  
jesuskastevbalmis@gmail.com

URL de la política de privacidad  
Política de privacidad del cuadro de diálogo de inicio de sesión y la in

URL de las Condiciones del servicio  
Condiciones del servicio del cuadro de diálogo de inicio de sesión y la

Eliminación de datos de usuario  
URL de las instrucciones para la eliminación de datos

También puedes proporcionar un enlace

Icono de la aplicación (1024 x 1024)

1024 x 1024

Categoría

Habilita el método de acceso con Facebook desde la consola de Firebase, copia los códigos correspondientes, pégalos en su lugar y luego guarda los cambios.

Login ▾

## Authentication

Usuarios Método de acceso Plantillas Uso Configuración Extensions

Proveedores de acceso

Agregar proveedor nuevo

Proveedor	Estado
Correo electrónico/contraseña	Habilitado
Google	Habilitado

**1** Facebook

**2** ID de la app  
994286956135230

**3** Secreto de app  
13bae140d34e8f9dbe4a711d3d0b790b

Para completar la configuración, agrega este URI de redireccionamiento de OAuth a la configuración de tu app de Facebook. [Más información](#)

**4** https://login-f8106.firebaseio.com/\_/auth/handler

Borrar proveedor Cancelar Guardar

**∞ Meta**

Identificador de la aplicación 994286956135230 Modo de la aplicación: En desarrollo En producción Tipo de aplicación: Consumidor

**Panel**

- Acciones requeridas
- Configuración de la apl...**
- Información básica
- Opciones avanzadas
- Roles de la aplicación
- Alertas
- Revisión de la aplicación
- Productos **Añadir producto**
- Registro de actividad

Identificador de la aplicación: 994286956135230 Clave secreta de la aplicación: 13bae140d34e8f9dbe4a711d3d0b790b Restablecer

Nombre para mostrar: Login Espacio de nombres:

Dominios de la aplicación: Correo electrónico de contacto: jesuskastevbalmis@gmail.com

URL de la política de privacidad: Política de privacidad del cuadro de diálogo de inicio de sesión y la in Eliminación de datos de usuario: URL de las instrucciones para la eliminación de datos: También puedes proporcionar un enlace

URL de las condiciones del servicio: Condiciones del servicio del cuadro de diálogo de inicio de sesión y la icono de la aplicación (1024 x 1024): 1024 x 1024

**∞ Meta**

Identificador de la aplicación 994286956135230 Modo de la aplicación: En desarrollo En producción Tipo de aplicación: Consumidor

**Panel**

- Acciones requeridas
- Configuración de la aplicación
- Roles de la aplicación
- Alertas
- Revisión de la aplicación
- Productos **Añadir producto**
- Registro de actividad

0 % del límite usado Ver detalles Usuarios limitados

**Añade productos a tu aplicación**

Simplificamos el proceso de creación de aplicaciones al mostrar los productos y permisos necesarios para crear el tipo de aplicación que has seleccionado.

**App Events**: Comprende cómo la gente interacciona con tu negocio en las aplicaciones, los dispositivos, las plataformas y los sitios web. **Configurar**

**Audience Network**: Monetiza tu aplicación y aumenta los ingresos con publicidad de anunciantes de Meta. **Configurar**

**Inicio de sesión con Facebook**: El producto de inicio de sesión social número uno del mundo. **Configurar**

**Webhooks**: Suscríbete para recibir en tiempo real actualizaciones y notificaciones sobre cambios sin tener que llamar a la API. **Configurar**

**Fundraisers**: Create and manage fundraisers for charities. **Configurar**

## Authentication

Usuarios    Método de acceso    Plantillas    Uso    Configuración    Extensions

Proveedores de acceso

Proveedor Estado

- Correo electrónico/contraseña Habilitado
- Google Habilitado
- Facebook**  Habilitar

ID de la app  
994286956135230

Secreto de app  
13bae140d34e8f9dbe4a711d3d0b790b

Para completar la configuración, agrega este URI de redireccionamiento de OAuth a la configuración de tu app de Facebook. [Más información](#)

[https://login-f8106.firebaseio.com/\\_/auth/handler](https://login-f8106.firebaseio.com/_/auth/handler)

Borrar proveedor    Cancelar    Guardar

Copia el código del método de acceso de Facebook desde la consola de Firebase y pégalo en el campo indicado en la imagen.

Identificador de la aplicación 994286956135230

Modo de la aplicación: En desarrollo  En producción

Tipo de aplicación: Consumidor

Configuración del cliente de OAuth

Acceso del cliente de OAuth  Activar el flujo de identificador de acceso del cliente de OAuth estándar. Asegura tu aplicación y evita su uso incorrecto especificando qué URI de redirección del identificador de acceso quieras permitir mediante las siguientes opciones. Desactiva esta opción globalmente si no la utilizas. [?]

Acceso de OAuth web  Activa el acceso del cliente de OAuth basado en web. [?]

Aplicar HTTPS  Aplica el uso de HTTPS para los URI de redireccionamiento y el SDK para JavaScript. Muy recomendado. [?]

Forzar la reautenticación OAuth web  Si se activa, solicita a las personas que introduzcan su contraseña de Facebook para iniciar sesión en el sitio web. [?]

Acceso de OAuth de navegador insertado  Activa los URI de redireccionamiento de la vista web para el acceso del cliente de OAuth. [?]

Usar modo estricto para URI de redireccionamiento  Permite únicamente los redireccionamientos que coinciden exactamente con los URI de redireccionamiento de OAuth válidos. Opción altamente recomendada. [?]

URI de redireccionamiento de OAuth válidos  
El campo "redirect\_uri" especificado manualmente que se usa con el inicio de sesión en la web debe coincidir exactamente con uno de los URI de esta lista. El SDK para JavaScript también usa esta lista para los navegadores en la aplicación que bloquean las ventajas emergentes. [?]

[https://login-f8106.firebaseio.com/\\_/auth/handler](https://login-f8106.firebaseio.com/_/auth/handler)

Copiar en el portapapeles

## Dependencias y Plugins de en Gradle

```
[versions]
facebook = "18.0.2"

[libraries]
facebook-android-sdk = { module = "com.facebook.android:facebook-android-sdk", version.ref =
```

En el `build.gradle.kts` del **módulo de la aplicación** (app) añadiremos:

```
...
dependencies {
    implementation(libs.facebook.android.sdk)
}
```

## Ajustes finales

The screenshot shows the Facebook Meta developer console interface. On the left, there's a sidebar with various options like Panel, Acciones requeridas, Configuración de la aplicación, Roles de la aplicación, Alertas, and Revisión de la aplicación. Below that, it says 'Inicio de sesión con Facebook' with a 'Guía de inicio rápido' button highlighted with a blue circle. The main area shows three steps for setting up an Android app: 1. Descargar el SDK de Facebook para Android, 2. Importar el SDK de Facebook, and 3. Informarnos sobre el proyecto de Android. Step 3 is currently selected and highlighted with a blue circle. In the 'Nombre del paquete' field, 'com.pmdm.login' is entered. In the 'Nombre de clase de actividad predeterminado' field, 'com.pmdm.login.MainActivity' is entered. A blue circle is also placed over the 'Save' button at the bottom right.

Instala el siguiente programa desde el enlace, descomprime el archivo en tu ordenador y ejecuta el comando indicado a continuación.

#### 4. Añadir los hashes de clave de desarrollo y activación

Para garantizar la autenticidad de las interacciones entre tu aplicación y Facebook, debes proporcionarnos el hash de clave de Android de tu entorno de desarrollo. Si tu aplicación ya se ha publicado, también deberás añadir el hash de clave de activación.

##### Generar un hash de clave de desarrollo

Tendrás un hash de clave de desarrollo único de cada entorno de desarrollo de Android.

##### Mac OS

Necesitarás la herramienta de administración de claves y certificados ([keytool](#)) del kit de desarrollo de Java.

Para generar un hash de clave de desarrollo, abre una ventana de terminal y ejecuta el siguiente comando:

```
keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore | openssl sha1 -binary |
```

[Copiar código](#)

##### Windows

Necesitarás lo siguiente:

- Herramienta de administración de claves y certificados ([keytool](#)) del kit de desarrollo de Java
- Biblioteca OpenSSL para Windows ([openssl-for-windows](#)) de Google Code Archive.

Para generar un hash de clave de desarrollo, ejecuta el comando siguiente en un símbolo del sistema del carpeta del SDK de Java:

```
keytool -exportcert -alias androiddebugkey -keystore "C:\Users\USERNAME\.android\debug.keystore" | "PAT
```

[Copiar código](#)

Este comando generará un hash de clave único de 28 caracteres de tu entorno de desarrollo. Cópialo y pégalo en el campo siguiente. Deberás proporcionar un hash de clave de desarrollo del entorno de desarrollo de cada persona que trabaje con tu aplicación.

##### Generar un hash de clave de activación

Las aplicaciones Android deben firmarse digitalmente con una clave de activación para que puedas subirlas a la tienda. Para

generar un hash de tu clave de activación, ejecuta el siguiente comando en Mac o Windows, sustituyendo el alias de la clave de



Projects Search About

Project



openssl-for-windows

Source

Issues

Wikis

Downloads

File	Summary + Labels	Uploaded	Size
openssl-0.9.8k_WIN32.zip	openssl-0.9.8k WIN32 <span>Featured</span>	Jul 23, 2009	1.25MB
openssl-0.9.8k_X64.zip	openssl-0.9.8k X64 <span>Featured</span>	Jul 23, 2009	1.42MB
openssl-0.9.8e_X64.zip	openssl-0.9.8e X64	Jul 23, 2009	1.25MB
openssl-0.9.8e_WIN32.zip	openssl-0.9.8e WIN32	Jul 23, 2009	1.08MB
openssl-0.9.8d_X64.rar	openssl-0.9.8d X64	Jul 23, 2009	1.32MB
openssl-0.9.8d_WIN32.rar	openssl-0.9.8d win32	Jul 23, 2009	1.16MB

4. Añadir los hashes de clave de desarrollo y activación

Para garantizar la autenticidad de las interacciones entre tu aplicación y Facebook, debes proporcionarnos el hash de clave de Android de tu entorno de desarrollo. Si tu aplicación ya se ha publicado, también deberás añadir el hash de clave de activación.

Generar un hash de clave de desarrollo

Tendrás un hash de clave de desarrollo único de cada entorno de desarrollo de Android.

Mac OS

Necesitarás la herramienta de administración de claves y certificados ([keytool](#)) del kit de desarrollo de Java.

Para generar un hash de clave de desarrollo, abre una ventana de terminal y ejecuta el siguiente comando:

```
keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore | openssl sha1 -binary |
```

Windows

Necesitarás lo siguiente:

- Herramienta de administración de claves y certificados ([keytool](#)) del kit de desarrollo de Java
- Biblioteca OpenSSL para Windows ([openssl-for-windows](#)) de Google Code Archive.

Para generar un hash de clave de desarrollo, ejecuta el comando siguiente en un símbolo del sistema de la carpeta del SDK de Java:

```
keytool -exportcert -alias androiddebugkey -keystore "C:\Users\USERNAME\.android\debug.keystore" | "C:\Users\USERNAME\Desktop\openssl\bin\openssl" sha1 -binary | "C:\Users\USERNAME\Desktop\openssl\bin\openssl" base64
```

Ejecuta el siguiente comando, copia y pega la ruta donde está ubicado el programa mencionado arriba. Además, ten en cuenta que debes usar **android** como contraseña para evitar errores.

```
$ keytool -exportcert -alias androiddebugkey -keystore  
"C:\Users\Jesus\.android\debug.keystore" |  
"C:\Users\Jesus\Desktop\openssl\bin\openssl" sha1 -binary |  
"C:\Users\Jesus\Desktop\openssl\bin\openssl" base64
```

Al ejecutar el comando, se mostrará un código que debes ingresar en el campo indicado en la siguiente imagen.

```
C:\Windows\System32\cmd.exe  
C:\Users\Jesus>keytool -exportcert -alias androiddebugkey -keystore "C:\Users\Jesus\.android\debug.keystore"  
"C:\Users\Jesus\Desktop\openssl\bin\openssl" sha1 -binary | "C:\Users\Jesus\Desktop\openssl\bin\openssl" base64  
Introduzca la contraseña del almacén de claves: android  
31wt5gsT08GK2n0PP9h4xWUYFOw=  
C:\Users\Jesus>
```

The screenshot shows the Meta dashboard interface. On the left, there's a sidebar with various options like 'Panel', 'Acciones requeridas', 'Configuración de la aplicación', etc. The main area is titled 'Windows' and contains instructions for generating key hashes. It includes two code snippets for generating SHA-1 and SHA-256 hashes using 'keytool' and 'openssl'. A blue arrow points from the text 'Copiar código' in the first snippet down to the 'Hashes de clave' input field, which contains the copied SHA-1 hash value.

En este paso, haz clic en Siguiente. No es necesario activar esta opción.

This screenshot shows step 5 of the quick start guide. It asks if you want to enable single sign-on for Android notifications. There are two buttons: 'No' (selected) and 'Sí'. Below the buttons, it says 'Se iniciará desde las notificaciones de Android'. At the bottom right are 'Save' and 'Siguiente' buttons.

Por último, solo debes seguir los pasos indicados en la guía para añadir todo lo necesario al manifest, y con eso, todo debería estar configurado correctamente.

## 6. Editar los recursos y el manifiesto



Si utilizas la versión 5.15 o posterior del SDK de Facebook para Android, no es necesario que añadas una actividad o un filtro de intención en las pestañas personalizadas de Chrome. Esta funcionalidad está incluida en el SDK.



Al integrar el inicio de sesión con Facebook, determinados eventos de la aplicación se registran y recopilan automáticamente para el **Administrador de eventos**, a menos que desactives el registro automático de eventos de la aplicación. Para obtener información más detallada sobre los datos que se recopilan y cómo desactivar el registro automático de eventos de la aplicación, consulta [Registro automático de eventos de la aplicación](#).

Crea cadenas para el identificador de tu aplicación de Facebook y para habilitar las pestañas personalizadas de Chrome. Añade también [FacebookActivity](#) a tu manifiesto de Android.

1. Abre el archivo `/app/res/values/strings.xml`.
2. Añade los elementos `string` con los nombres `facebook_app_id`, `fb_login_protocol_scheme` y `facebook_client_token`, y establece los valores de tu **identificador de la aplicación** e **identificador de acceso del cliente**. Por ejemplo, si tu identificador de la aplicación es `1234` y tu identificador de acceso del cliente es `56789`, el código tendrá un aspecto similar al siguiente:

```
<string name="facebook_app_id">1234</string>
<string name="fb_login_protocol_scheme">fb1234</string>
<string name="facebook_client_token">56789</string>
```

[Copiar código](#)

3. Abre el archivo `/app/manifest/AndroidManifest.xml`.
4. Añade los elementos `meta-data` al elemento `application` para tu identificador de la aplicación e identificador de acceso del cliente:

```
<application android:label="@string/app_name" ...>
    ...
    <meta-data android:name="com.facebook.sdk.ApplicationId" android:value="@string/facebook_app_id" />
    <meta-data android:name="com.facebook.sdk.ClientToken" android:value="@string/facebook_client_token" />
```

[Copiar código](#)

Finalmente, nuestro archivo `strings.xml` debe quedar de la siguiente manera:

```
<resources>
    <string name="app_name">Login</string>
3     <string name="facebook_app_id">994286956135230</string>
    <string name="fb_login_protocol_scheme">fb994286956135230</string>
5     <string name="facebook_client_token">d6d901ca140a50bcc0311b05c195af9</string>
</resources>
```

## ⚡ Resumen

Todo este proceso se detalla en un [tutorial](#) paso a paso. El tutorial de configuración termina en el minuto **18:20**.

## Definiendo los métodos

```
class FacebookAuth(
    errorMessage: String = "Error while trying to sign in with Facebook."
): AuthProviderStrategy(errorMessage) {
    override fun getCredential(idToken: String): AuthCredential = FacebookAuthProvider.getCre}
```

```

@HiltViewModel
class LoginViewModel @Inject constructor(
    private val validadorLogin: ValidadorLogin,
    private val authService: AuthServiceImplementation
) : ViewModel() {

    ...
    private fun onSuccessFacebookLogin(
        token: String,
        onSignIn: () -> Unit,
        onSignUp: () -> Unit
    ) {
        viewModelScope.launch {
            try {
                authService.signInWithProvider(token, FacebookAuth())
                informacionEstadoState = InformacionEstadoUiState.Oculta()
                onSignIn()
            } catch (e: AuthServiceException) {
                informacionEstadoState = InformacionEstadoUiState.Error(
                    mensaje = "Error al iniciar sesion con facebook",
                    onDismiss = { informacionEstadoState = InformacionEstadoUiState.Oculta() }
                )
                Log.d("AuthenticationViewModel", "Error: ${e.stackTraceToString()}")
            }
        }
    }

    private fun signInWithFacebook(
        onSignIn: () -> Unit,
        onSignUp: () -> Unit,
        activityOwner: ActivityResultRegistryOwner
    ) {
        val callbackManager = CallbackManager.Factory.create()

        informacionEstadoState = InformacionEstadoUiState.Informacion(
            muestraProgreso = true,
            mensaje = "Iniciando sesion con facebook"
        )
        LoginManager.getInstance().registerCallback(callbackManager,
            object : FacebookCallback<LoginResult> {
                override fun onCancel() {
                    informacionEstadoState = InformacionEstadoUiState.Oculta()
                }
            }
        )
    }
}

```

```

        override fun onError(error: FacebookException) {
            informacionEstadoState = InformacionEstadoUiState.Error(
                mensaje = "Error al iniciar sesión con facebook",
                onDismiss = { informacionEstadoState = InformacionEstadoUiState.Oculto })
        }
    }

    override fun onSuccess(result: LoginResult) {
        val token: String = result.accessToken.token
        onSuccessFacebookLogin(
            token = token,
            onSignIn = onSignIn,
            onSignUp = onSignUp
        )
    }
}

)
LoginManager.getInstance().logInWithReadPermissions(
    activityResultRegistryOwner = activityOwner,
    callbackManager = callbackManager,
    permissions = listOf("email", "public_profile")
)
}

fun onLoginEvent(loginEvent: LoginEvent) {
    when(loginEvent) {
        is LoginEvent.OnSignInWithFacebookClick -> {
            signInWithFacebook(
                onSignIn = loginEvent.onSignIn,
                onSignUp = loginEvent.onSignUp,
                activityOwner = loginEvent.activityOwner
            )
        }
    }
}
}

```

## Nota

Al cerrar sesión después de haber iniciado con la autenticación de Facebook, al volver a iniciar sesión no se mostrará la opción para seleccionar entre varias cuentas en el

dispositivo. En su lugar, se iniciará sesión automáticamente con la cuenta utilizada la primera vez.

## Manejo de Sesión en la Aplicación

Para determinar si el usuario está iniciado sesión y navegar dependiendo del resultado hacia la pantalla de autenticación o dentro de la app deberemos usar una pantalla de carga.

```
@Composable
fun LoadingScreen(
    modifier: Modifier = Modifier
    ...
) {
    LaunchedEffect(key1 = Unit) {
        onLoadingEvent(
            LoadingEvent.OnManejarSesion(
                onNavigateToLogin = onNavigateToLogin,
                onNavigateToAccount = onNavigateToAccount
            )
        )
    }
    ...
}
```

```

@HiltViewModel
class LoadingViewModel @Inject constructor(
    private val authService: AuthServiceImplementation
): ViewModel() {
    private fun manejarSesion(
        onNavigateToAccount: () -> Unit,
        onNavigateToLogin: () -> Unit
    ) {
        viewModelScope.launch {
            val isUserLoggedIn: Boolean = authService.isUserLoggedIn()
            if(isUserLoggedIn) onNavigateToAccount() else onNavigateToLogin()
        }
    }

    fun onLoadingEvent(event: LoadingEvent) {
        when(event) {
            is LoadingEvent.OnManejarSesion -> {
                manejarSesion(
                    onNavigateToAccount = event.onNavigateToAccount,
                    onNavigateToLogin = event.onNavigateToLogin
                )
            }
        }
    }
}

```

```

@Composable
fun LoginNavHost() {
    val navController: NavHostController = rememberNavController()
    val vmLoading: LoadingViewModel = hiltViewModel()

    NavHost(
        navController = navController,
        startDestination = LoadingRoute
    ) {
        ...
    }
}

```

## Cerrar sesión

Para cerrar sesión en cualquier proveedor de servicio, podemos hacer lo siguiente:

```
class AuthService @Inject constructor(
    private val firebaseAuth: FirebaseAuth,
    @ApplicationContext private val context: Context
) {
    ...
    fun logOut() {
        firebaseAuth.signOut()
        getGoogleAccount().signOut()
        LoginManager.getInstance().logOut()
    }
}
```

```
class AuthServiceImplementation @Inject constructor(
    private val authService: AuthService
) {
    fun logOut() = authService.logOut()
}
```

## Restablecer contraseña

Para finalizar, permitiremos al usuario restablecer su contraseña en caso de que la haya olvidado.

```

@HiltViewModel
class ForgotPasswordViewModel @Inject constructor(
    private val authService: AuthServiceImplementation,
    private val forgotPasswordValidador: ForgotPasswordValidador
): ViewModel() {

    ...
    fun onForgotPasswordEvent(forgotPasswordEvent: ForgotPasswordEvent) {
        when(forgotPasswordEvent) {
            is ForgotPasswordEvent.EnviarEmail -> {
                if(!forgotPasswordValidador.hayError) {
                    viewModelScope.launch {
                        informacionEstadoState = InformacionEstadoUiState.Informacion(
                            muestraProgreso = true,
                            mensaje = "Enviando email"
                        )
                        authService.sendPasswordResetEmail(emailState)
                        informacionEstadoState = InformacionEstadoUiState.Exitoso("Se ha enviado el correo")
                        delay(SnackbarDuration.Short.toMillis())
                        informacionEstadoState = InformacionEstadoUiState.Oculta()
                    }
                } else {
                    informacionEstadoState = InformacionEstadoUiState.Error(
                        mensaje = "Email no valido",
                        onDismiss = { informacionEstadoState = InformacionEstadoUiState.Oculta() }
                    )
                }
            }
        }
    }
}

```

```

class AuthService @Inject constructor(
    private val firebaseAuth: FirebaseAuth,
    @ApplicationContext private val context: Context
) {
    fun sendPasswordResetEmail(email: String): Task =
        firebaseAuth.sendPasswordResetEmail(email)
}

```

```
class AuthService @Inject constructor(
    private val firebaseAuth: FirebaseAuth,
    @ApplicationContext private val context: Context
) {
    suspend fun sendPasswordResetEmail(email: String) = withContext(Dispatchers.IO) {
        try {
            authService.sendPasswordResetEmail(email).await()
        } catch (e: Exception) {
            Log.e(logTag, "Error: ${e.localizedMessage}")
            throw AuthServiceException("Error while trying to send password reset email.")
        }
    }
}
```