



FORO DE DISCUSIÓN 2

Desarrollo de Software para Móviles DSM941 G01T

ELABORADO POR:

Samuel Fernando Calderon Reyes CR202814

Walter Daniel Mejía Palacios MP202829

Docente:

Alexander Alberto Siguenza Campos

Índice

Índice.....	2
Introducción.....	3
Objetivos.....	4
Historia y evolución de Kotlin y Java en Android.....	5
Introducción a Java y su papel en Android.....	5
Emergencia de Kotlin.....	6
Comparación de la evolución de ambos lenguajes en Android.....	7
Breve historia del desarrollo de ambos lenguajes.....	8
Java: Orígenes y evolución.....	8
Kotlin: Nacimiento y desarrollo.....	9
Kotlin y Java: Ventajas y Desventajas.....	11
Ventajas y Desventajas de Java.....	11
Ventajas y Desventajas de Kotlin.....	12
Análisis FODA Kotlin y Java.....	14
Conclusiones.....	20
Bibliografía.....	21

Introducción

La implementación de sistemas de autenticación en aplicaciones Android es un pilar esencial para garantizar la seguridad y personalización de las experiencias de los usuarios. Firebase Authentication se presenta como una solución integral que permite a los desarrolladores incorporar múltiples métodos de inicio de sesión, desde el tradicional correo electrónico y contraseña hasta opciones más modernas como Google, Facebook y Apple. Este proyecto se centra en investigar y desarrollar una aplicación en Android con Kotlin utilizando Firebase, integrando múltiples métodos de autenticación para mejorar la usabilidad y la seguridad de la aplicación. La combinación de estas herramientas modernas busca facilitar el acceso a las aplicaciones y adaptarse a las necesidades de los usuarios.

Objetivos

General: Diseñar e implementar un sistema de autenticación en una aplicación Android utilizando Firebase Authentication con soporte para múltiples métodos, incluyendo correo electrónico, contraseña y autenticación con Google

- Investigar las diferentes opciones de autenticación proporcionadas por Firebase, detallando sus características, ventajas y desventajas.
- Desarrollar una interfaz de usuario funcional y atractiva que integre de manera eficiente los métodos de autenticación seleccionados.
- Documentar el proceso de implementación, proporcionando guías claras y prácticas para el uso de cada método en proyectos futuros.

Investigación sobre Opciones de Autenticación

1. Autenticación por Correo Electrónico y Contraseña

Este método es uno de los más tradicionales y comunes en aplicaciones móviles y permite que los usuarios se registren e inicien sesión usando su correo electrónico y una contraseña. Firebase ofrece soporte completo para este tipo de autenticación, lo cual facilita tanto la creación como la administración de las cuentas de usuario, ya que Firebase se encarga de almacenar de forma segura las credenciales en sus servidores. Además, esta opción ofrece funcionalidades adicionales como la posibilidad de restablecer la contraseña en caso de que el usuario la olvide, así como validaciones de seguridad integradas para verificar que la dirección de correo electrónico es válida.

Es ideal para aplicaciones que buscan un método de inicio de sesión sencillo y familiar para los usuarios, ya que muchas personas prefieren no vincular sus cuentas personales de otras plataformas con una app nueva. No obstante, también tiene sus retos: la dependencia de los usuarios para crear una contraseña suficientemente fuerte es una posible vulnerabilidad, ya que una contraseña débil podría exponer la cuenta. Por eso, se recomienda implementar validaciones en el frontend para asegurar que los usuarios establezcan contraseñas seguras y, si es posible, combinar esta autenticación con otros métodos para añadir una capa de seguridad adicional, como la autenticación multifactorial. Firebase permite configurar y gestionar esta autenticación desde su consola, y el proceso en Kotlin se basa en el uso de FirebaseAuth para registrar, autenticar y manejar las sesiones de usuario de forma bastante directa.

Ventajas:

- **Simplicidad:** Es fácil de implementar y no requiere configuraciones adicionales fuera de Firebase.

- **Seguridad:** Firebase maneja las credenciales de forma segura, reduciendo la exposición de datos sensibles.
- **Coste bajo:** No requiere un proveedor externo para validar o manejar las credenciales.

Desventajas:

- **Dependencia del usuario:** Si los usuarios no eligen contraseñas seguras, existe un riesgo mayor de ataques de fuerza bruta.
- **Menos conveniente que otras opciones:** Muchos usuarios prefieren usar cuentas externas como Google o Facebook para evitar recordar otra combinación de correo y contraseña.

Implementación en Firebase

- **Activar en Firebase Console:** Habilitar la opción de “Email/Password” en la sección de métodos de inicio de sesión.
- **Integración en la aplicación:** En el código, utilizar FirebaseAuth para registrar y autenticar a los usuarios.

Código de ejemplo en Kotlin:

```
1  // Crear cuenta con correo y contraseña
2  auth.createUserWithEmailAndPassword(email, password).addOnCompleteListener { task ->
3      if (task.isSuccessful) {
4          // Registro exitoso
5      } else {
6          // Error en el registro
7      }
8  }
9
10 // Inicio de sesión con correo y contraseña
11 auth.signInWithEmailAndPassword(email, password).addOnCompleteListener { task ->
12     if (task.isSuccessful) {
13         // Inicio de sesión exitoso
14     } else {
15         // Error en el inicio de sesión
16     }
17 }
18
```

2. Autenticación con Proveedores Externos (Google, Facebook, Apple, etc.)

La autenticación a través de proveedores externos permite a los usuarios acceder a la aplicación sin necesidad de crear una cuenta específica, utilizando en cambio sus credenciales de plataformas populares como Google, Facebook o Apple. Esta opción es cada vez más común, ya que ofrece una experiencia de inicio de sesión rápida y conveniente, lo cual suele mejorar la retención de usuarios al reducir el tiempo y esfuerzo que implica crear una cuenta desde cero.

Firebase facilita enormemente la integración con estos proveedores, ya que ofrece un soporte que simplifica el proceso de configuración y autenticación a los usuarios mediante los servidores de cada proveedor externo, lo que significa que las credenciales del usuario no se almacenan directamente en Firebase. Esto añade una capa adicional de confianza, ya que el usuario puede sentirse más seguro al no tener que crear una cuenta específica para la app. A pesar de sus ventajas, este método también implica una dependencia de terceros, lo cual podría ser una desventaja si, por ejemplo, uno de los proveedores experimenta una caída en su servicio. Además, los usuarios deben aceptar el intercambio de datos entre plataformas, por lo que es esencial tener una política de privacidad clara y transparente en la aplicación. Para configurar este método, se habilita la autenticación de proveedores externos en Firebase Console, y se configura cada proveedor por separado, lo que implica registrar la aplicación en plataformas como Google Cloud Console o Facebook Developers, según el caso.

Ventajas

- **Experiencia de usuario optimizada:** Permite a los usuarios evitar crear y recordar una cuenta específica.
- **Mayor seguridad:** Los proveedores de autenticación externos suelen manejar la seguridad y protección de datos.
- **Confianza:** Los usuarios pueden preferir utilizar cuentas conocidas y confiables como Google o Facebook.

Desventajas

- **Dependencia de terceros:** Si un proveedor experimenta un problema, puede afectar el inicio de sesión de los usuarios.
- **Consideraciones de privacidad:** Al utilizar proveedores externos, se requiere consentimiento del usuario para compartir información entre plataformas.

Código de ejemplo en Kotlin para Google Sign-In:

```
1  // Configurar Google Sign-In
2  val gso = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
3      .requestIdToken(getString(R.string.default_web_client_id))
4      .requestEmail()
5      .build()
6
7  val googleSignInClient = GoogleSignIn.getClient(this, gso)
8
9  // Iniciar actividad de autenticación
10 startActivityResult(googleSignInClient.signInIntent, RC_SIGN_IN)
11
12 // Manejar el resultado en onActivityResult
13 val task = GoogleSignIn.getSignedInAccountFromIntent(data)
14 val account = task.getResult(ApiException::class.java)
15 val credential = GoogleAuthProvider.getCredential(account.idToken, null)
16 auth.signInWithCredential(credential)
```


3. Autenticación Anónima

La autenticación anónima de Firebase permite que los usuarios utilicen la aplicación sin registrarse formalmente. Es una opción útil cuando se desea proporcionar a los usuarios una experiencia inicial para explorar la app antes de decidir registrarse completamente. Al iniciar sesión de forma anónima, Firebase crea una cuenta temporal para el usuario que se almacena en su backend, lo cual permite al usuario almacenar datos temporalmente o personalizar su experiencia.

Sin embargo, si el usuario elimina la aplicación o decide no registrarse, esta cuenta anónima se perderá, ya que su persistencia es limitada. En el caso de que el usuario elija registrarse más adelante, Firebase permite que esta cuenta anónima se vincule a una cuenta autenticada, conservando así cualquier dato o configuración que el usuario haya establecido. La autenticación anónima puede ser una excelente estrategia para atraer a los usuarios a probar la app y motivarlos eventualmente a registrarse con una cuenta formal.

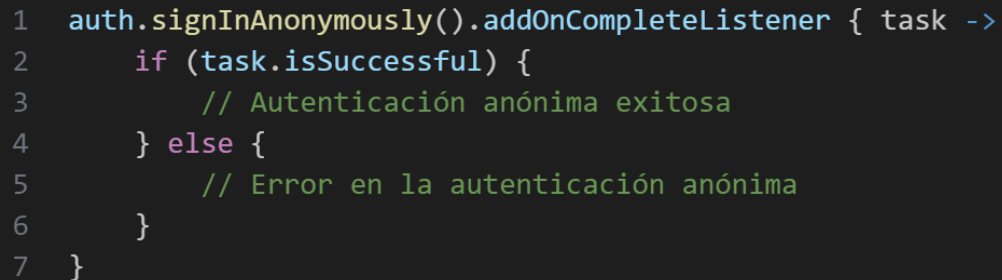
Ventajas

- **Acceso rápido:** Ideal para dar acceso rápido a usuarios que solo necesitan explorar la app sin registrarse.
- **Retención de datos:** Permite guardar los datos de sesión del usuario hasta que decida registrarse.

Desventajas

- **Limitado en persistencia:** Si el usuario desinstala la app o elimina el caché, se pierde la cuenta anónima.
- **Conversión de usuarios:** Requiere incentivar a los usuarios anónimos a registrarse, para poder interactuar mejor con la aplicación.

Código de ejemplo en Kotlin:



```
1  auth.signInAnonymously().addOnCompleteListener { task ->
2      if (task.isSuccessful) {
3          // Autenticación anónima exitosa
4      } else {
5          // Error en la autenticación anónima
6      }
7  }
```

4. Autenticación con Teléfono

La autenticación mediante número de teléfono permite a los usuarios registrarse e iniciar sesión usando su número, verificado a través de un SMS que Firebase envía automáticamente. Este método es particularmente útil en áreas donde el uso del correo electrónico es menos común o cuando los usuarios prefieren un método de autenticación rápido y directo. En este caso, el usuario recibe un código de verificación vía SMS que debe ingresar en la app para completar el proceso de autenticación. Aunque esta opción ofrece una experiencia de autenticación simplificada y, en cierta medida, más segura debido al uso de dos factores (número de teléfono y código SMS), puede resultar costosa en aplicaciones de gran volumen debido al coste de los SMS. Además, depende de que los usuarios tengan señal y disponibilidad de SMS, lo cual podría ser una limitante en ciertas áreas. Para implementarla, es necesario habilitar la

opción en Firebase Console y configurar la verificación de número en la app, gestionando el flujo de envío y recepción del código de forma adecuada.

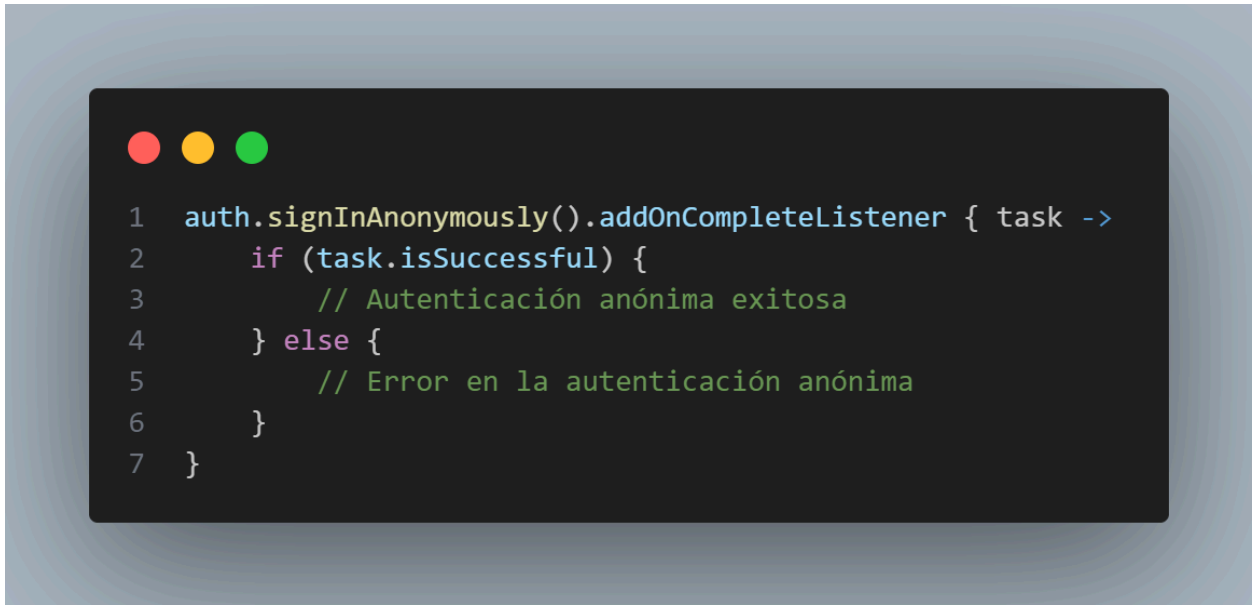
Ventajas

- **Acceso simple:** Es ideal para áreas donde el correo electrónico no es común o donde los usuarios prefieren autenticarse mediante su teléfono.
- **Verificación adicional:** SMS proporciona una autenticación de dos factores (2FA) que aumenta la seguridad.

Desventajas

- **Costo de SMS:** Los envíos de SMS pueden generar costos en aplicaciones con gran cantidad de usuarios.
- **Disponibilidad de red:** Requiere que el usuario tenga servicio de SMS activo y puede ser menos fiable en áreas de mala cobertura.

Código de ejemplo en Kotlin:

A screenshot of a code editor with a dark background and light-colored text. At the top left of the editor, there are three colored circles: red, yellow, and green. The code is written in Kotlin and is as follows:

```
1  auth.signInAnonymously().addOnCompleteListener { task ->
2      if (task.isSuccessful) {
3          // Autenticación anónima exitosa
4      } else {
5          // Error en la autenticación anónima
6      }
7  }
```

Documentación de Implementación de Autenticación por Correo Electrónico y Google

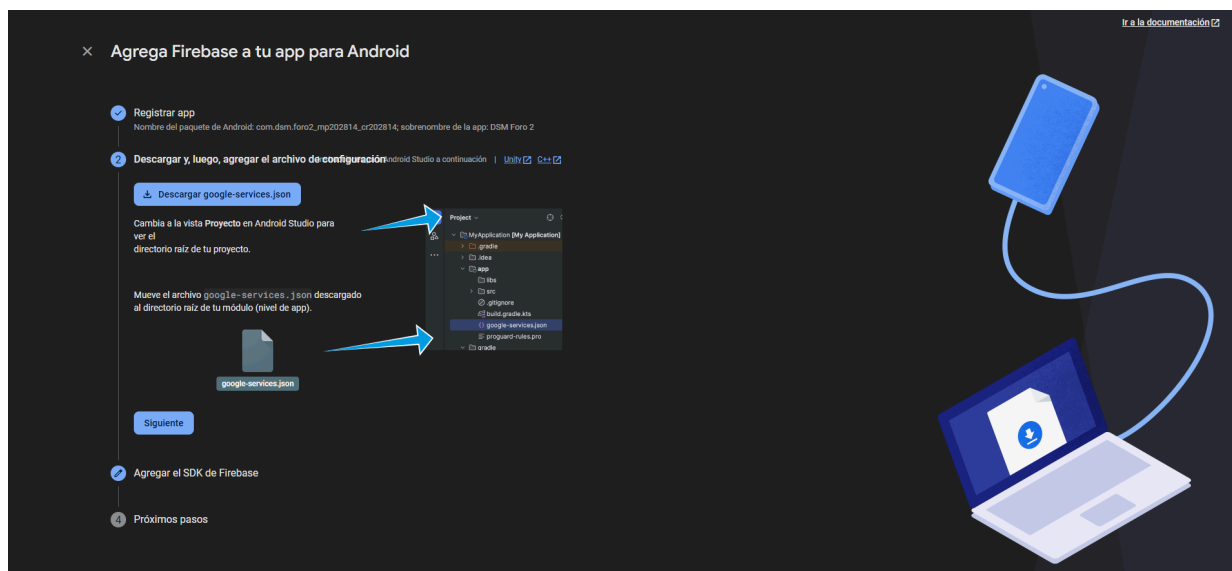
Agregar Login con Firebase Authentication

1. Creación de un Proyecto en Firebase

- Acceder a Firebase Console.
- Crear un proyecto nuevo y configurar un nombre para el mismo.

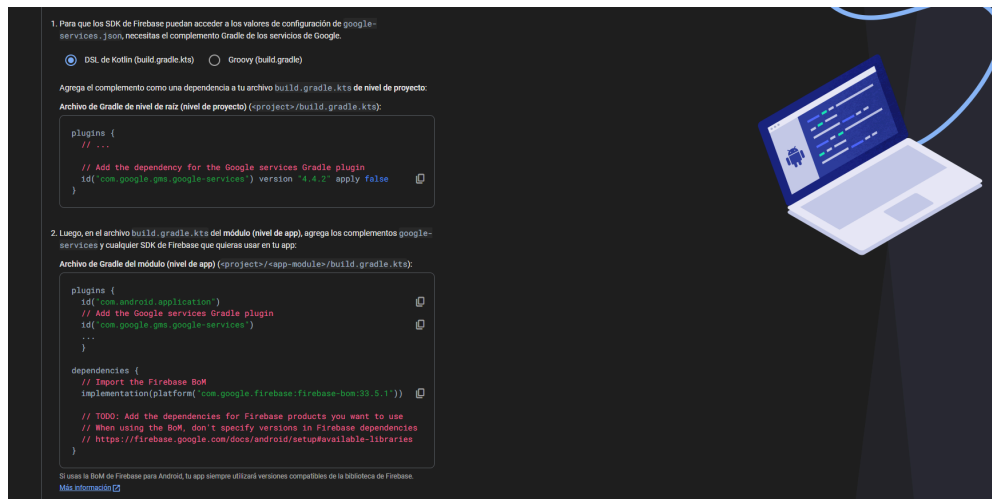
2. Vinculación de la App Android con Firebase

- Seleccionar el proyecto recién creado en la consola.
- Hacer clic en "Agregar App" y elegir la opción "Android".
- Ingresar el nombre del paquete de la aplicación (este se encuentra en el archivo `AndroidManifest.xml`).
- Descargar el archivo `google-services.json` que proporciona Firebase y colocarlo en la carpeta `app` del proyecto.



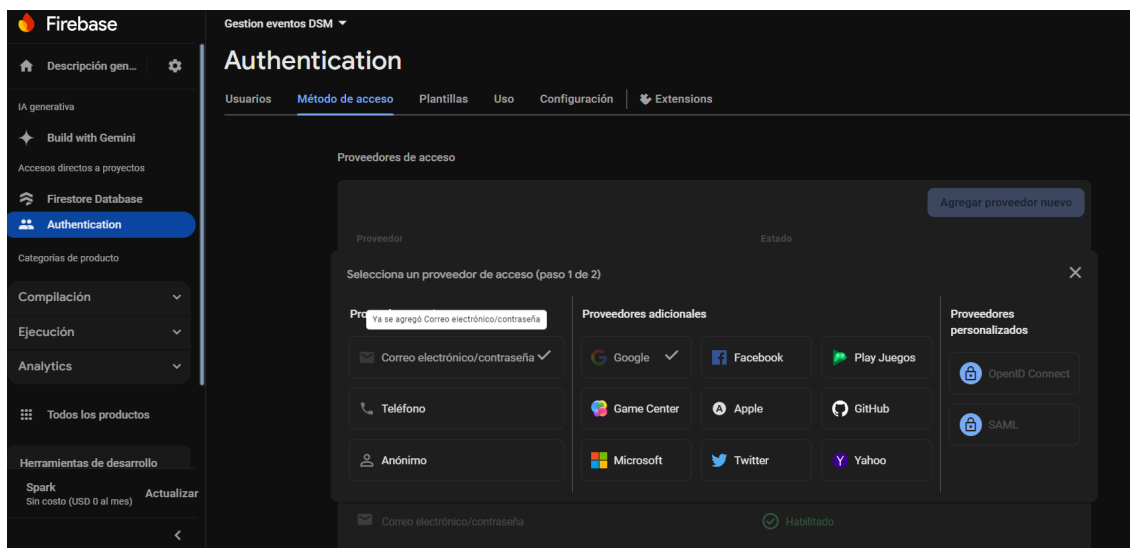
3. Configuración de las dependencias de Firebase

- En el archivo **build.gradle** de nivel raíz, incluir **classpath 'com.google.gms:google-services:4.3.15'**
- En el archivo **build.gradle** de nivel módulo (**app**), agregar: **implementation 'com.google.firebase:firebase-auth:22.1.1'**
apply plugin: 'com.google.gms.google-services'



4. Activación del Proveedor de Correo y Contraseña en Firebase

- En la consola de Firebase, navegar a **Authentication > Métodos de inicio de sesión**.
- Activar el proveedor de "Correo electrónico y contraseña".

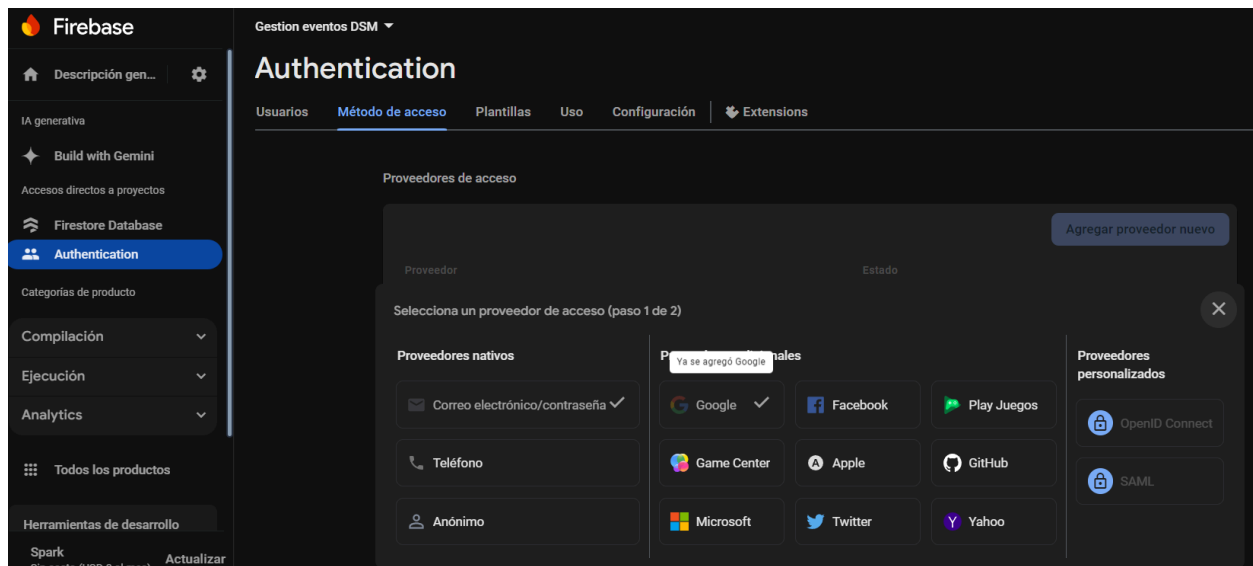


Agregar Login con Firebase Google Authentication

Teniendo en cuenta que los pasos anteriores ya fueron realizados:

1. Configuración Inicial en Firebase Console

- Ve a la Consola de Firebase y selecciona tu proyecto.
- Navega a **Authentication > Métodos de inicio de sesión**.
- **Activa el proveedor Google.**
- **Configura el ID de cliente de OAuth 2.0. Esto se hace automáticamente si tienes tu proyecto registrado en Firebase.**



2. Agregar Dependencias

- Agregar las dependencias necesarias en tu archivo `build.gradle` de nivel de módulo:
`implementation 'com.google.firebase:firebase-auth:22.1.1'`
`implementation 'com.google.android.gms:play-services-auth:20.7.0'`
- Aplica el plugin de Google Services:
`apply plugin: 'com.google.gms.google-services'`

3. Configurar Google Sign-In en tu Proyecto Android

- Ve al archivo `AndroidManifest.xml` y agrega
`<meta-data`
`android:name="com.google.android.gms.wallet.api.enabled"`
`android:value="true" />`

- En tu `strings.xml`, agrega el nombre de tu proyecto o ID de cliente (se obtiene desde el archivo de `google_services.json`):

```
<string  
name="default_web_client_id">YOUR_CLIENT_ID.apps.googleusercontent.com</string>
```

google_services.json:

```
42     "services": {  
43         "appinvite_service": {  
44             "other_platform_oauth_client": [  
45                 {  
46                     "client_id": "55670096915-cc3ugkshjnjha0qjmop7vecp2d25u7dp.apps.googleusercontent.com",  
47                     "client_type": 3  
48                 }  
49             ]  
50         }  
51     }
```

strings.xml:

```
1 <resources>  
2     <string name="app_name">Foro2_MP202814_CR202814</string>  
3     <string name="default_web_client_id">55670096915-cc3ugkshjnjha0qjmop7vecp2d25u7dp.apps.googleusercontent.com</string>  
4
```

Repositorio de github

<https://github.com/DSM-MP202829-CR202814/foro2-login>

Conclusiones

Firestore Authentication es una herramienta versátil y confiable para implementar sistemas de autenticación en aplicaciones móviles, proporcionando métodos diversos como correo electrónico, contraseña y autenticación con Google. La incorporación de estas opciones optimiza la experiencia del usuario al adaptarse a diferentes preferencias, mejorando la accesibilidad y seguridad de las aplicaciones. La combinación de Firestore con Kotlin resulta en un entorno de desarrollo eficiente, que simplifica la implementación de características avanzadas y fomenta la creación de soluciones modernas para el manejo de usuarios. Además, la adecuada documentación del proceso de desarrollo no solo asegura la replicabilidad en futuros proyectos, sino que también promueve el aprendizaje continuo y el uso eficiente de estas tecnologías en la industria del desarrollo móvil.

Bibliografía

- Google Developers. (2024). *Get started with Firebase Authentication*. Recuperado de <https://firebase.google.com/docs/auth>
- JetBrains. (2024). *Kotlin for Android development*. Recuperado de <https://developer.android.com/kotlin>
- Oracle Corporation. (2024). *Java vs. Kotlin: Benefits in Android development*. Recuperado de <https://docs.oracle.com/javase/tutorial>
- Bloch, J. (2008). *Effective Java*. Addison-Wesley Professional.
- Jemerov, D., & Isakova, S. (2017). *Kotlin in Action*. Manning Publications.
- TechCrunch. (2019). *The rise of Kotlin: A new era for Android developers*.