Universidad Don Bosco

Dirección de Educación a Distancia

Ingeniería en Ciencias de la Computación

Desarrollo de Software para Móviles DSM941 G01T

Foro 2

Implementando un acceso a una aplicación móvil con Firebase

Presentado por

José Ernesto Sorto González

Repositorio: devJosesorto/DMSFireBasev

domingo 24, noviembre del 2024

Contenido

Introducción

La autenticación es una de las características fundamentales en cualquier aplicación moderna que interactúa con usuarios. Firebase, como una plataforma Backend-as-a-Service (BaaS), ofrece una solución robusta y escalable para implementar diferentes métodos de autenticación en aplicaciones Android. Esto simplifica la gestión de usuarios y se integra de manera fluida con otros servicios de Firebase, como Firestore o Realtime Database.

En esta investigación, exploraremos las opciones de autenticación disponibles en Firebase y nos enfocaremos en la implementación de dos métodos clave: autenticación por correo electrónico y contraseña y Google Sign-In. Documentaremos las características de cada método, sus ventajas y desventajas, y proporcionaremos una guía detallada para su implementación en un proyecto Android utilizando Kotlin.

El propósito principal es proporcionar una visión completa y práctica que facilite a los desarrolladores la incorporación de estas funcionalidades, mejorando la seguridad y la experiencia de usuario en sus aplicaciones. Esta introducción servirá como base para entender las capacidades de Firebase Authentication y las mejores prácticas para su uso.

Historia y Evolución:

Firebase comenzó su trayectoria en 2011 como una plataforma de base de datos en tiempo real enfocada en sincronización y almacenamiento de datos en la nube. En 2014, Google adquirió Firebase con el propósito de convertirlo en una solución integral para el desarrollo de aplicaciones modernas. Uno de los componentes fundamentales que surgió de esta adquisición fue Firebase Authentication, diseñado para simplificar la autenticación y gestión de usuarios en aplicaciones móviles y web.

Etapas Clave en la Evolución de Firebase Authentication

Inicio como una plataforma de base de datos (2011 - 2014):

Firebase comenzó como una solución de base de datos en tiempo real, centrada en la sincronización de datos entre clientes y servidores.

Durante este periodo, no existía un sistema nativo de autenticación en Firebase. Los desarrolladores solían implementar sus propios sistemas de autenticación personalizados, lo que requería un esfuerzo adicional para garantizar seguridad y escalabilidad.

Adquisición por Google y primera generación de Firebase Authentication (2014 - 2016):

Google amplió las capacidades de Firebase, introduciendo Firebase Authentication como un servicio independiente.

La primera versión de Firebase Authentication ofrecía soporte básico para autenticación por correo electrónico y contraseña, así como integración con Google Sign-In y otros proveedores populares como Facebook y Twitter.

Este enfoque facilitó a los desarrolladores integrar la autenticación con servicios populares sin necesidad de construir sistemas personalizados desde cero.

Unificación y expansión de servicios (2016 - 2018):

En 2016, Google unificó sus herramientas de desarrollo de aplicaciones bajo la marca Firebase. Esto incluyó la integración de Google Cloud Identity Platform con Firebase Authentication.

Se añadieron más métodos de autenticación, incluyendo:

Autenticación anónima.

Inicio de sesión con número de teléfono (OTP).

Soporte mejorado para OAuth 2.0.

Firebase se posicionó como una plataforma líder para aplicaciones móviles, gracias a su facilidad de uso, compatibilidad con múltiples plataformas y escalabilidad.

Introducción de características avanzadas (2018 - 2021):

Verificación de identidad avanzada: Firebase Authentication comenzó a soportar medidas avanzadas de seguridad como reCAPTCHA y verificación en dos pasos (2FA).

Herramientas de personalización: Los desarrolladores obtuvieron la capacidad de personalizar flujos de inicio de sesión y gestionar cuentas de usuarios a través de las APIs de Firebase Admin SDK.

Soporte para Apple Sign-In: Para cumplir con los requisitos de la App Store, Firebase añadió soporte nativo para el inicio de sesión con Apple.

Evolución reciente y enfoque en seguridad (2021 - Presente):

Firebase Authentication ha continuado mejorando su seguridad y usabilidad, con características como:

Compatibilidad con claves de seguridad (FIDO2).

Mejores integraciones con Google Identity Services.

Soporte ampliado para autenticación multifactor.

Google también ha fortalecido la infraestructura subyacente para garantizar un mayor rendimiento y confiabilidad en aplicaciones de gran escala.

Ventajas y Desventajas:

Ventajas

Fácil Integración:

Firebase Authentication se integra fácilmente con otras herramientas de Firebase como Firestore, Realtime Database y Cloud Functions.

Los SDK de Firebase están diseñados para minimizar la complejidad, especialmente para desarrolladores con experiencia básica en Kotlin o Android.

Soporte Multiplataforma:

Funciona en Android, iOS y aplicaciones web, lo que permite crear soluciones de autenticación consistentes en múltiples plataformas.

Los mismos métodos de autenticación pueden ser usados en distintas plataformas sin reconfiguraciones complejas.

Seguridad Incorporada:

Firebase gestiona automáticamente aspectos críticos como el hash de contraseñas, verificaciones de correo electrónico y protección contra ataques como el phishing.

Ofrece soporte para autenticación multifactor (MFA) y medidas adicionales como reCAPTCHA para evitar abusos.

Escalabilidad:

Firebase Authentication es adecuado tanto para aplicaciones pequeñas como para aquellas con millones de usuarios.

Al ser parte de la infraestructura de Google Cloud, está diseñado para manejar picos de tráfico sin problemas.

Soporte para Múltiples Proveedores:

Incluye autenticación con correo electrónico y contraseña, Google, Facebook, Twitter, Apple, Microsoft y otros proveedores basados en OAuth 2.0.

Permite autenticación con número de teléfono (OTP) y opciones anónimas para usuarios temporales.

Panel de Control Centralizado:

Desde la consola de Firebase, puedes administrar usuarios, visualizar estadísticas y realizar acciones como restablecer contraseñas o eliminar cuentas.

Documentación y Comunidad Activa:

Firebase tiene una excelente documentación oficial y una comunidad activa, lo que facilita la resolución de problemas y el aprendizaje.

Desventajas

Dependencia de Firebase:

Utilizar Firebase Authentication implica una dependencia directa de la infraestructura de Google. Si el servicio experimenta interrupciones, puede afectar a la autenticación en la aplicación.

Migrar a otro proveedor de autenticación puede ser costoso y complicado.

Límites en la Personalización:

Aunque permite cierta personalización de los flujos de autenticación, los desarrolladores tienen menos control sobre los detalles del backend.

Para aplicaciones que necesitan flujos de autenticación completamente personalizados, Firebase puede ser limitado.

Costos Escalables:

Aunque Firebase ofrece un nivel gratuito generoso, los costos pueden aumentar significativamente para aplicaciones con usuarios activos masivos o con autenticación por SMS (número de teléfono).

Restricciones de Configuración Regional:

Algunos métodos de autenticación, como SMS (OTP), pueden no estar disponibles o tener limitaciones en ciertas regiones.

Comportamiento Variable entre Proveedores de OAuth:

Aunque Firebase soporta múltiples proveedores como Google, Facebook y Apple, las configuraciones de estos pueden variar y requieren ajustes específicos en sus respectivas consolas de desarrollador.

Complejidad en Aplicaciones Grandes:

Para aplicaciones con requisitos de seguridad avanzados o integraciones complejas (como sistemas SSO personalizados o compliance con GDPR), Firebase Authentication puede no ser suficiente.

Falta de Control Directo sobre el Backend:

No puedes acceder directamente a las contraseñas de los usuarios, ya que Firebase gestiona todo el cifrado. Esto puede ser una ventaja desde el punto de vista de la seguridad, pero una limitación si necesitas migrar usuarios a otra plataforma.

Pasos de Implementación: Autenticación con Correo Electrónico y Google Sign-In:

A continuación, se describen los pasos detallados para implementar Firebase Authentication (correo electrónico/contraseña y Google Sign-In) en un proyecto Android utilizando IntelliJ IDEA.

1. Configuración del Proyecto Firebase

Crear un Proyecto en Firebase

1. Ve a la consola de Firebase.
2. Crea un nuevo proyecto y sigue las instrucciones.

Registrar la App en Firebase

1. En la consola de Firebase, selecciona tu proyecto.
2. Haz clic en "Añadir app" y selecciona **Android**.
3. Introduce el nombre del paquete de tu aplicación (puedes encontrarlo en `AndroidManifest.xml` en tu proyecto IntelliJ).
4. Descarga el archivo `google-services.json` cuando Firebase lo solicite.

Configurar Métodos de Autenticación

1. Ve a la pestaña `Authentication > Sign-in method`.
2. Habilita **Correo Electrónico/Contraseña** y **Google** como métodos de autenticación.

2. Configuración del Archivo `google-services.json`

Colocar el Archivo en tu Proyecto

Mueve el archivo `google-services.json` a la carpeta `app` en tu proyecto IntelliJ.

Agregar el Complemento de Google Services

1. Abre el archivo `build.gradle.kts` del nivel del proyecto y añade:

```
plugins {
    id("com.google.gms.google-services") version "4.4.2" apply false
}
```

5. Abre el archivo `build.gradle.kts` del módulo `app` y añade:

```
plugins {
```

```
    id("com.google.gms.google-services")
}
```

3. Agregar Dependencias de Firebase

Abre el archivo `build.gradle.kts` del módulo `app` y añade las siguientes dependencias:

```
dependencies {
    // Firebase Authentication
    implementation("com.google.firebase:firebase-auth-ktx")

    // Google Sign-In
    implementation("com.google.android.gms:play-services-auth:20.7.0")
}
```

Sincroniza Gradle: Ve a **File > Sync Project with Gradle Files**.

4. Configuración del Proyecto en IntelliJ

Agregar default_web_client_id a strings.xml

1. Abre el archivo `res/values/strings.xml`.
2. Añade:

```
<resources>
    <string name="app_name">DMSFirebaseV</string>
    <string name="default_web_client_id">TU_CLIENT_ID_GENERADO</string>
</resources>
```

Configurar Permisos en `AndroidManifest.xml`

Asegúrate de que tu archivo `AndroidManifest.xml` incluya los permisos necesarios:

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

5. Implementación de la Lógica de Autenticación

Pantalla de Login

Crea un archivo llamado `LoginScreen.kt` en tu proyecto y añade el siguiente código:

```
package com.example.firebaseauth

import android.content.Intent
```

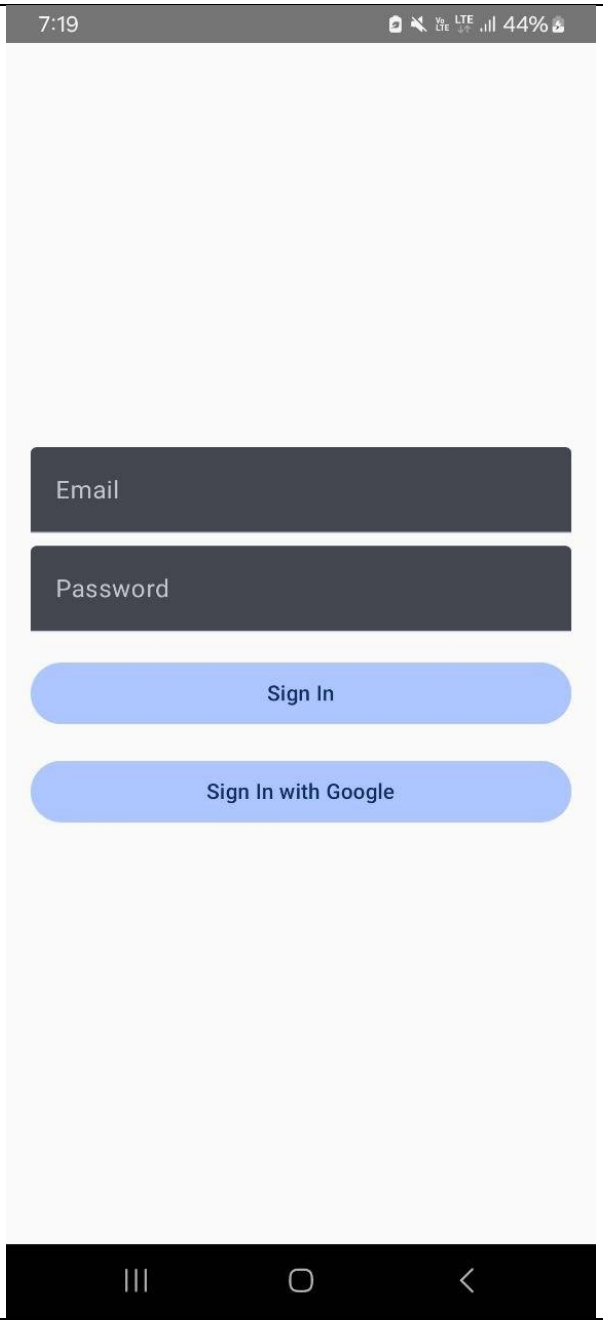
```
import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.unit.dp
import com.google.android.gms.auth.api.signin.GoogleSignIn
import com.google.android.gms.auth.api.signin.GoogleSignInClient
import com.google.android.gms.auth.api.signin.GoogleSignInOptions
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.GoogleAuthProvider

@Composable
fun LoginScreen() {
    val context = LocalContext.current
    val auth = FirebaseAuth.getInstance()

    var email by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var message by remember { mutableStateOf("") }

    // Configurar Google Sign-In
    val                            googleSignInOptions                                =
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken(context.getString(R.string.default_web_client_id))
        .requestEmail()
        .build()
    val googleSignInClient: GoogleSignInClient = GoogleSignIn.getClient(context,
googleSignInOptions)

    // UI de Login
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        TextField(
            value = email,
            onValueChange = { email = it },
            label = { Text("Email") },
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(8.dp))
        TextField(
            value = password,
            onValueChange = { password = it },
            label = { Text("Password") },
            visualTransformation = PasswordVisualTransformation(),
            modifier = Modifier.fillMaxWidth()
        )
        Spacer(modifier = Modifier.height(16.dp))
```

```
        Button(onClick = {
            auth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener { task ->
                    message = if (task.isSuccessful) {
                        "Login Successful"
                    } else {
                        "Error: ${task.exception?.message}"
                    }
                }
        }) {
            Text("Sign In")
        }

        Spacer(modifier = Modifier.height(16.dp))

        Button(onClick = {
            val signInIntent = googleSignInClient.signInIntent
            context.startActivity(signInIntent)
        }) {
            Text("Sign In with Google")
        }

        Spacer(modifier = Modifier.height(8.dp))
        Text(message)
    }
}
```

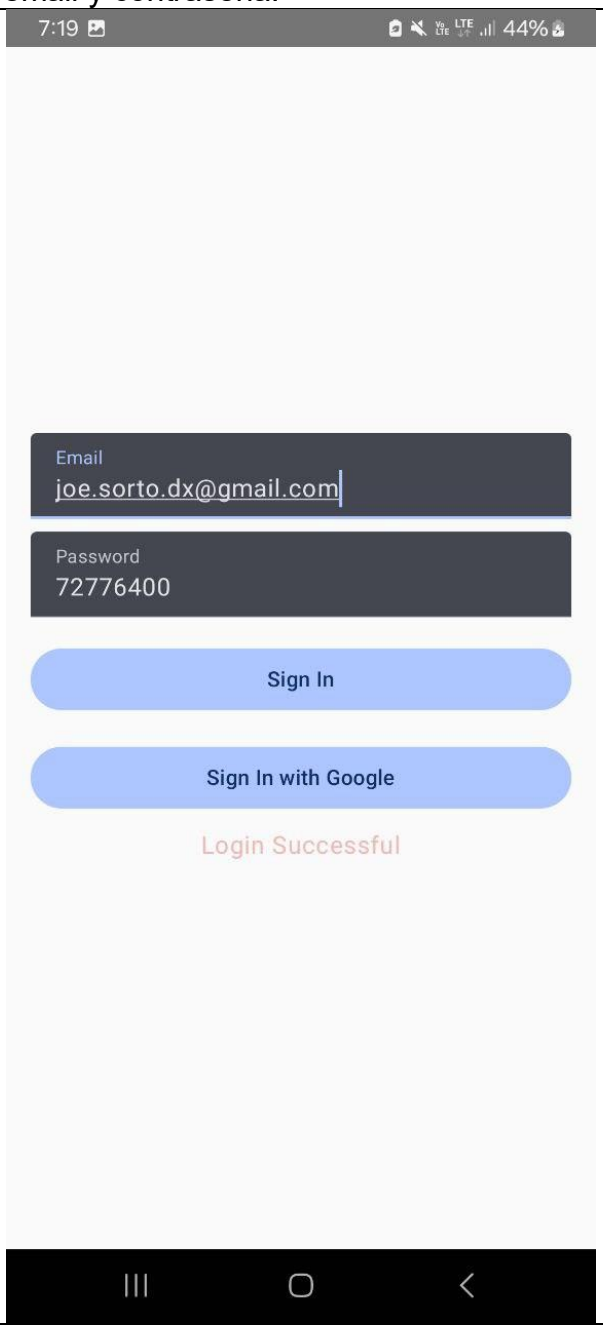
6. Verificación del Usuario en Firebase

1. Ve a la consola de Firebase.
2. En `Authentication > Users`, verifica que los usuarios aparezcan después de iniciar sesión.

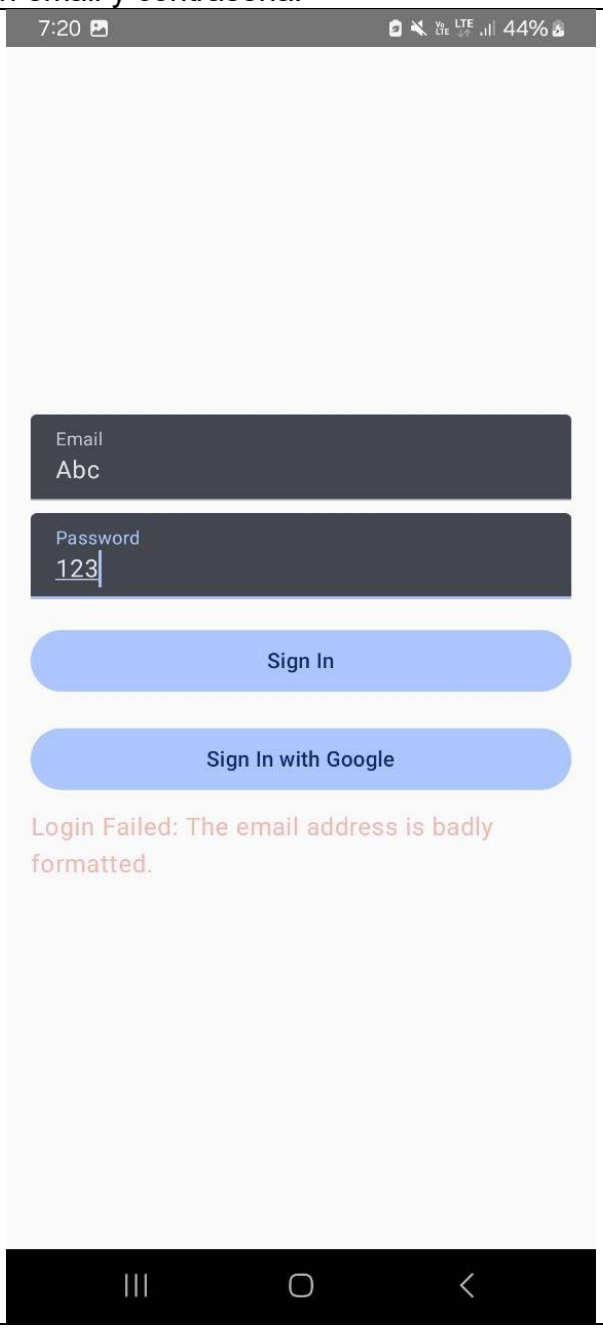
Anexos

Pantalla de Login


Login validado con email y contraseña.



Login denegado con email y contraseña.



Login mediante autentificación de Google.