Diplomado en Desarrollo de Software

Parte V

Programación Móvil Avanzada

Ruthford Jay / rjay@uninorte.edu.co

Programación Móvil Avanzada

Firebase

Objetivos de la Sesión

Al finalizar esta sesión estarás en capacidad de:

- Entender los conceptos de la autenticación usando los servicios de Firebase
- Desarrollar una aplicación usando los servicios de autenticación de Firebase

Firebase

Firebase es una plataforma de desarrollo de apps Backend-as-a-Service (BaaS) que proporciona servicios de backend hospedados como una base de en tiempo real, almacenamiento en la nube, autentificación, reportes de fallos, machine learning, configuración remota, y hospedaje para tus ficheros estáticos.

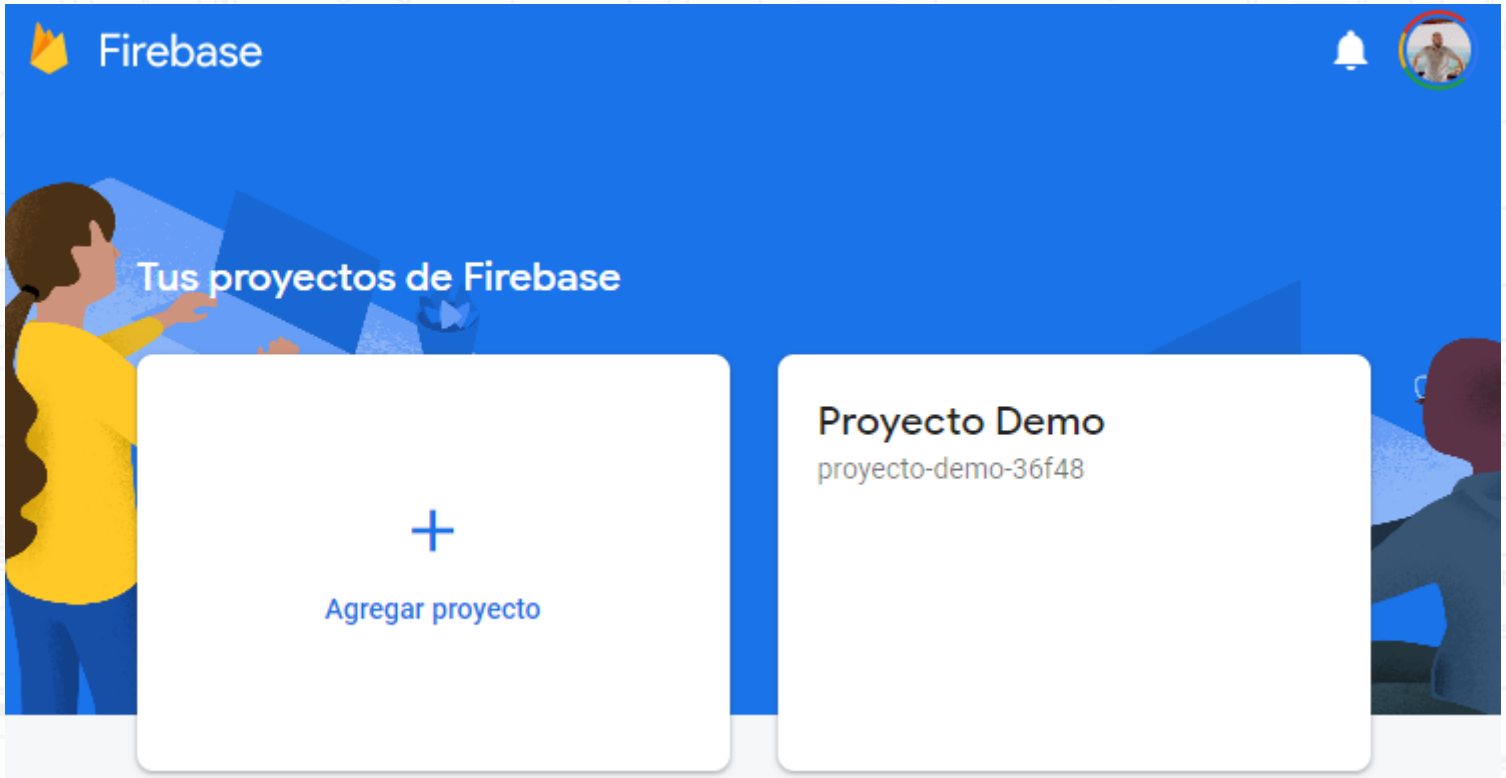
FlutterFire – Firebase Core

FlutterFire es un conjunto de librerías que da soporte a aplicaciones desarrolladas en Flutter a las funcionalidades de Firebase. Primero vamos a instalar **firebase_core**, la que nos permite realizar la primera conexión al backend.

```
$ flutter pub add firebase_core
```

Firebase Console

Es necesario crear un proyecto en Firebase con el cual podamos conectar a nuestra aplicación, esto se puede realizar desde el Firebase Console



Archivos de Configuración

Al crear una nueva aplicación de Android, el "certificado de firma de depuración SHA-1" es opcional; sin embargo, es necesario para autenticación con los servicios de Google. Para generar un certificado, ejecute el siguiente comando ubicado en la carpeta Android de su proyecto:

```
.\gradlew.bat signingReport
```

Y copie el SHA1 de la clave de **debug**. Esto genera dos claves variantes. Puede copiar el 'SHA1' que pertenece a la opción de clave variante **debugAndroidTest**.

Una vez que se haya registrado su aplicación de Android, descargue el archivo de configuración de Firebase Console (el archivo se llama google-services.json). Agrega este archivo al directorio android/app dentro de tu proyecto Flutter.

Soporte para Android en Firebase

Después de crear un nuevo proyecto es necesario agregar soporte para aplicaciones Android.

Como parte del proceso de va a requerir el nombre del paquete, el mismo debe coincidir con el nombre del paquete del proyecto local que se creó cuando inició el proyecto Flutter.

El nombre del paquete actual se puede encontrar en el archivo Gradle de su módulo (nivel de aplicación), generalmente android/app/build.gradle, sección defaultConfig (nombre de paquete de ejemplo: co.edu.uninorte.demoproject).

Configuración del Proyecto en Flutter

Para permitir que Firebase use la configuración en Android, se debe aplicar el complemento **'google-services'** en el proyecto. Esto requiere la modificación de dos archivos en el directorio **android/app/**.

Primero, agregue el complemento 'google-services' como una dependencia dentro del archivo **android/build.gradle**:

```
buildscript {
    // ...
    dependencies {
        // Otras depedencias
        classpath 'com.google.gms:google-services:x.x.xx'
    }
}
```

NOTA: Las versiones pueden cambiar, el paso a paso actualizado puede ser accedido desde el firebase console.

Configuración del Proyecto en Flutter

Por último, ejecute el complemento agregando lo siguiente debajo de la línea aplicar complemento: **'com.android.application'**, dentro del archivo **/android/app/build.gradle**:

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'
apply plugin: 'kotlin-android'
```

Inicialización FlutterFire

Antes de que se pueda usar cualquiera de los servicios de Firebase, es necesario inicializar **FlutterFire**. El paso de inicialización es asincrónico, lo que significa que deberá evitar cualquier uso relacionado con FlutterFire hasta que se complete la inicialización.

Para inicializar **FlutterFire**, llama al método **initializeApp** en la clase Firebase:

```
await Firebase.initializeApp();
```

El método es asincrónico y devuelve un futuro, por lo que debe asegurarse de que se haya completado antes de mostrar su aplicación principal.

Inicialización FlutterFire

```dart
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp();

  runApp(const MyApp());
}
```

Inicialización FlutterFire

```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(
          title: const Text('FlutterFire'),
        ),
        body: const Center(
          child: Text(
            'FlutterFire App',
          ),
        ),
      ),
    );
  }
}
```

Autenticación

Firebase Authentication proporciona servicios de backend y SDK fáciles de usar para autenticar a los usuarios en su aplicación. Admite la autenticación mediante contraseñas, números de teléfono, proveedores de identidad federada populares como Google, Facebook y Twitter, y más.

Para lograr esto, debemos seguir estos pasos:

- Agregar dependencia
- Descargar dependencia
- Reconstruir la aplicación

```
$ flutter pub add firebase_auth
```

Autenticación

Firebase Auth proporciona muchos métodos y utilidades que te permiten integrar la autenticación segura en tu aplicación Flutter nueva o existente. En muchos casos, necesitará conocer el estado de autenticación de su usuario, por ejemplo, si está conectado o desconectado.

Firebase Auth le permite suscribirse en tiempo real a este estado a través de un Stream. Una vez llamada, la secuencia proporciona un evento inmediato del estado de autenticación actual del usuario y luego proporciona eventos posteriores cada vez que cambia el estado de autenticación.

Hay tres métodos para escuchar los cambios de estado de autenticación:

- authStateChanges()
- idTokenChanges()
- userChanges()

Autenticación

Authentication State - authStateChanges(): Para suscribirse a estos cambios, llame al método authStateChanges() en su instancia **FirebaseAuth.**

Los eventos se activan cuando ocurre lo siguiente:

- Inmediatamente después de que se haya registrado el oyente.
- Cuando un usuario inicia sesión.
- Cuando se cierra la sesión del usuario actual.

Autenticación

Authentication State - idTokenChanges(): Para suscribirse a estos cambios, llame al método idTokenChanges() en su instancia **FirebaseAuth.**

Los eventos se activan cuando ocurre lo siguiente:

- Inmediatamente después de que se haya registrado el listener .
- Cuando un usuario inicia sesión.
- Cuando se cierra la sesión del usuario actual.
- Cuando hay un cambio en el token del usuario actual.

Autenticación

Authentication State - userChanges(): Para suscribirse a estos cambios, llame al método userChanges() en su instancia **FirebaseAuth**.

Los eventos se activan cuando ocurre lo siguiente:

- Inmediatamente después de que se haya registrado el listener .
- Cuando un usuario inicia sesión.
- Cuando se cierra la sesión del usuario actual.
- Cuando hay un cambio en el token del usuario actual.

Autenticación

Authentication State - userChanges():

Los eventos se activan cuando ocurre lo siguiente:

- Cuando `FirebaseAuth.instance.currentUser` se llaman a los siguientes métodos proporcionados por :
  - `reload()`
  - `unlink()`
  - `updateEmail()`
  - `updatePassword()`
  - `updatePhoneNumber()`
  - `updateProfile()`

Autenticación

Estado de Autenticación Persistente: Los SDK de Firebase para todas las plataformas brindan soporte listo para usar para garantizar que el estado de autenticación de su usuario se mantenga durante los reinicios de la aplicación o las recargas de la página.

En plataformas nativas como Android e iOS, este comportamiento no se puede configurar y el estado de autenticación del usuario se mantendrá en el dispositivo entre los reinicios de la aplicación. El usuario puede borrar los datos almacenados en caché de las aplicaciones a través de la configuración del dispositivo que borrará cualquier estado existente que se esté almacenando.

Autenticación Correo y Contraseña

El correo electrónico / contraseña es un método de inicio de sesión de usuario común para la mayoría de las aplicaciones.

Esto requiere que el usuario proporcione una dirección de correo electrónico y una contraseña segura.

Los usuarios pueden registrar nuevas cuentas con un método llamado `createUserWithEmailAndPassword()` o iniciar sesión en una cuenta existente con `signInWithEmailAndPassword()`.

Autenticación Correo y Contraseña

Registro: Para crear una nueva cuenta en su proyecto de Firebase, llame al método `createUserWithEmailAndPassword()` con la dirección de correo electrónico y la contraseña del usuario.

El método es una operación de dos pasos; primero creará la nueva cuenta (si aún no existe y la contraseña es válida) y luego iniciará automáticamente la sesión del usuario en esa cuenta. Si está escuchando cambios en el estado de autenticación , se enviará un nuevo evento a sus listeners.

Autenticación Correo y Contraseña

Inicio de sesión: Para iniciar sesión en una cuenta existente, llame al método `signInWithEmailAndPassword()`. Una vez que tenga éxito, si está escuchando los cambios en el estado de autenticación , se enviará un nuevo evento a sus listeners.

Componente práctico

Aplicación demo de Autenticación con Firebase