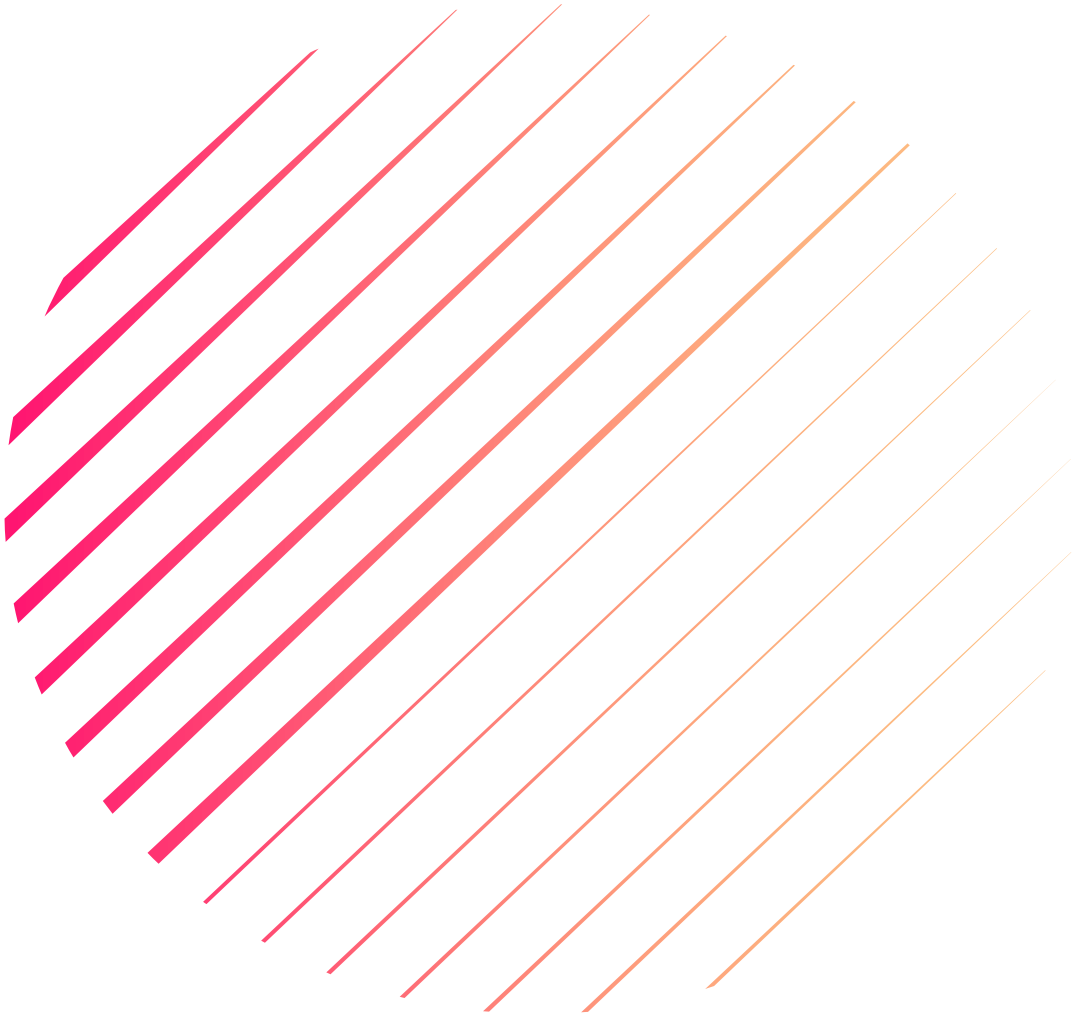
UNIVERSITÉ DES MASCAREIGNES

SAVOIR, C'EST POUVOIR

Université de Limoges

Mental Health Diagnosis

Faculty of Information and Communication Technology



03/02/2024

PMH

Gokhool Diren

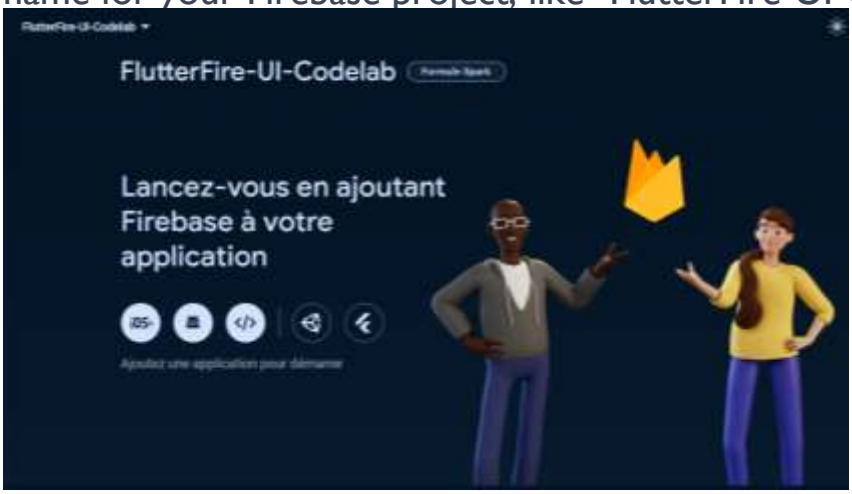
Mr Shiam Beehary

Task-2 Firebase for Flutter

Create and set up a Firebase project

1. Log in to Firebase.

2. On the Firebase console, press "Add Project" (or "Create a project"), then type a name for your Firebase project, like "FlutterFire-UI-Codelab".

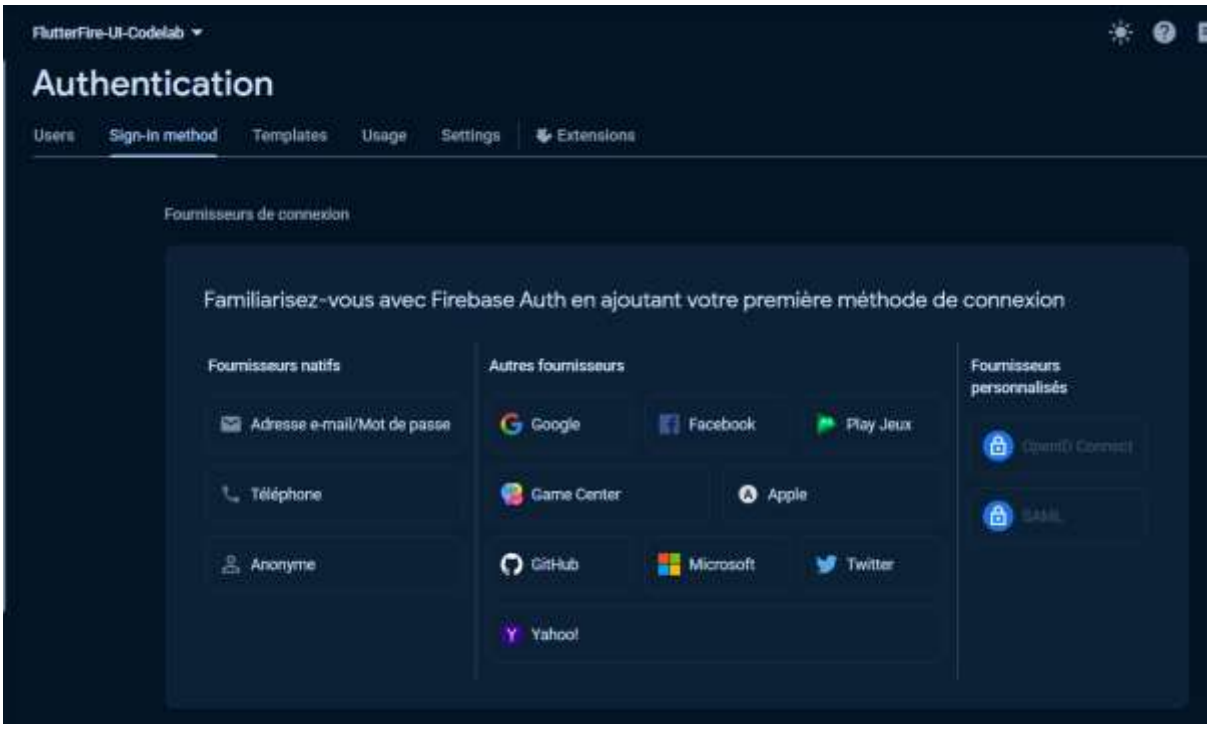


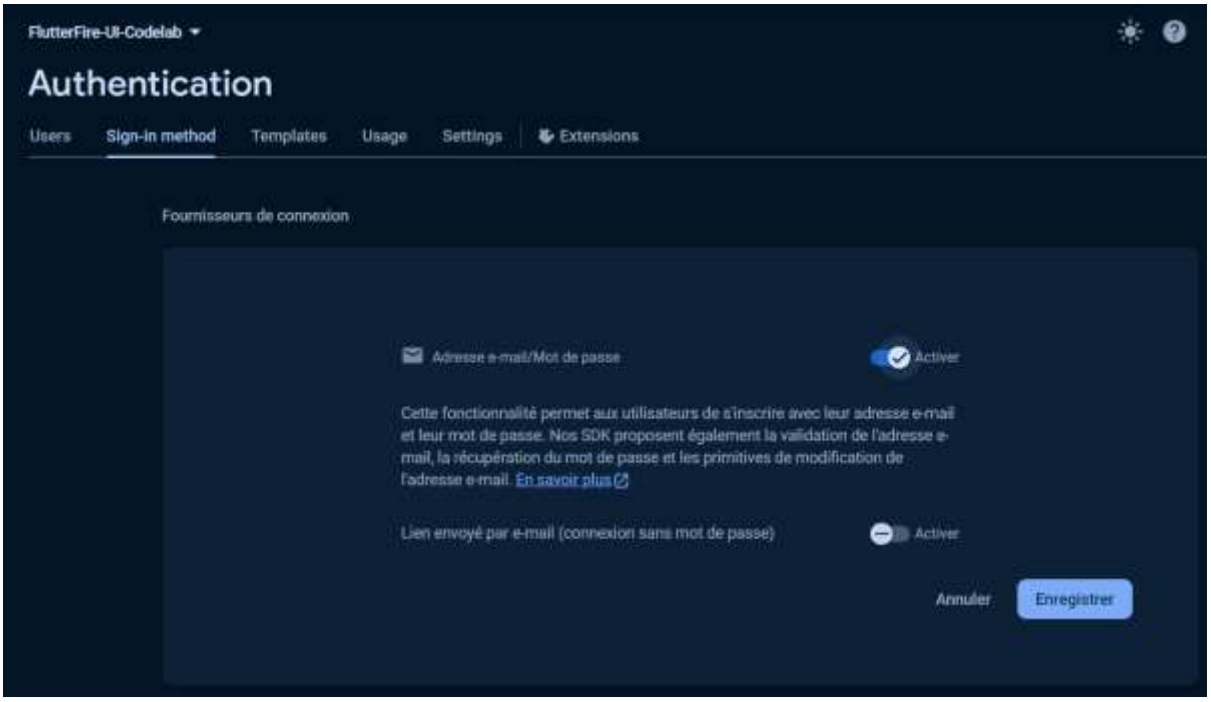
Enable email sign-in for Firebase Authentication

To enable user sign-in for the web app, we'll start with the Email/Password method.

1. Go to the Firebase console, find the Build section on the left.

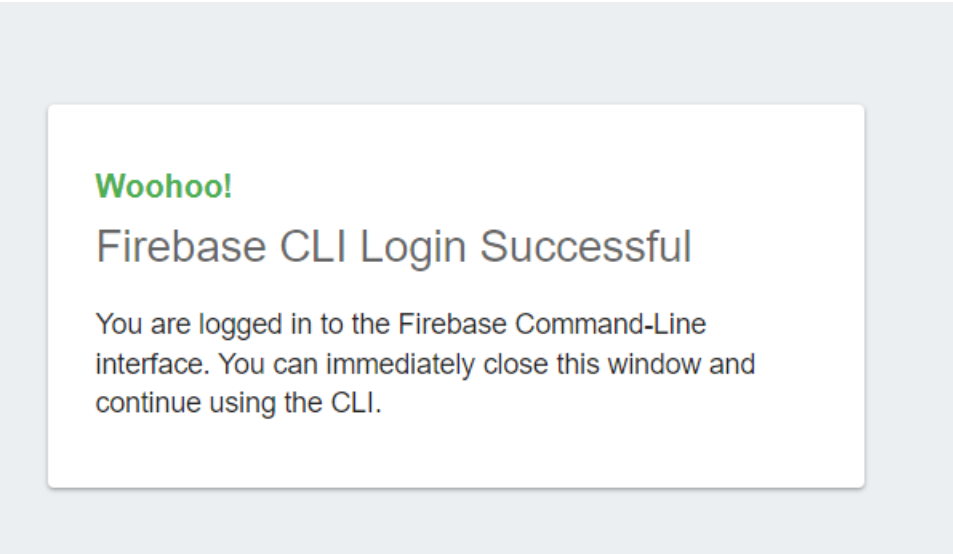
2. Select Authentication, then click Get Started, and navigate to the Sign-in method tab.


FlutterFire-UI-Codelab
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
Fournisseurs de connexion
Familiarisez-vous avec Firebase Auth en ajoutant votre première méthode de connexion
Fournisseurs natifs
Adresse e-mail/Mot de passe
Téléphone
Anonyme
Autres fournisseurs
Google
Facebook
Play Jeux
Game Center
Apple
GitHub
Microsoft
Twitter
Yahoo!
Fournisseurs personnalisés


FlutterFire-UI-Codelab
Authentication
Users
Sign-in method
Templates
Usage
Settings
Extensions
Fournisseurs de connexion
Adresse e-mail/Mot de passe
Activer
Cette fonctionnalité permet aux utilisateurs de s'inscrire avec leur adresse e-mail et leur mot de passe. Nos SDK proposent également la validation de l'adresse e-mail, la récupération du mot de passe et les primitives de modification de l'adresse e-mail. En savoir plus
Lien envoyé par e-mail (connexion sans mot de passe)
Activer
Annuler
Enregistrer

Set up Flutter app

Before starting, we need to download the initial code and set up the Firebase CLI. Log in to Firebase using your Google account with this command: If you encounter any issues in the VS Code terminal, try again in a new terminal.



The displayed list should match the Firebase projects listed in the Firebase console. Install the FlutterFire CLI, a tool that simplifies Firebase installation across all supported platforms in a Flutter app, built on top of the Firebase CLI. First, install the CLI. Then, add the Firebase project to the Flutter app.

Configure FlutterFire to generate the required Dart code for using Firebase in the Flutter app. Run the command:
flutterfire configure
This command prompts to select the Firebase project and platforms to set up. Alternatively, you can set the default project in the firebase directory in the root of the Flutter project, allowing flutterfire configure to automatically select the default project. Then, choose the platforms you want to use. This codelab includes steps to configure Firebase Authentication for Flutter on web, iOS, and Android, but you can set up your Firebase project to use all options.

In the Flutter app, when inspecting the text editor, FlutterFire CLI generates a new file named firebase_options.dart. This file includes a class called FirebaseOptions, containing static variables holding the Firebase configuration for each platform. As we chose all platforms during the flutterfire configure command, static values like web, android, ios, and macos are visible.

Initialize Firebase

To utilize the added packages and DefaultFirebaseOptions.currentPlatform, update the code within the main function in the main.dart file.

Add initial Firebase UI Auth page

Firebase UI for Auth offers widgets representing complete screens in the app, managing various authentication flows like Sign In, Registration, Forgot Password, and User Profile. To begin, let's add a landing page to the app serving as an authentication guard for the main application.

Check authentication state

Before displaying a sign-in screen, we must verify if the user is authenticated. This is typically done by listening to FirebaseAuth's authStateChanges using the Firebase Auth plugin. In the provided code sample, the MaterialApp constructs an AuthGate widget in its build method (a custom widget, not part of FlutterFire UI). This widget should be updated to incorporate the authStateChanges stream. The authStateChanges API returns a Stream containing either the current user (if signed in) or null (if not). To monitor this state in our app, we can utilize Flutter's StreamBuilder widget, passing the stream to it. StreamBuilder dynamically updates based on the latest data snapshot from the provided Stream.

Sign-In screen

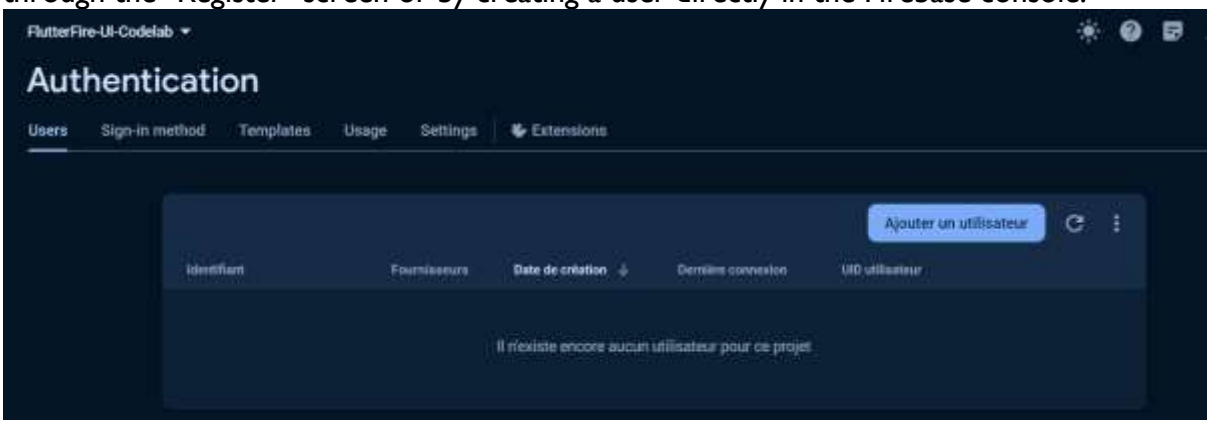
The SignInScreen widget from FlutterFire UI offers the following features:
1. Enables users to sign in.
2. Provides a "Forgot password?" option for users to reset their password via a form.
3. Offers a "Register" option for users not yet registered, leading them to a sign-up form.

Customizing the sign-in screen involves several builders:

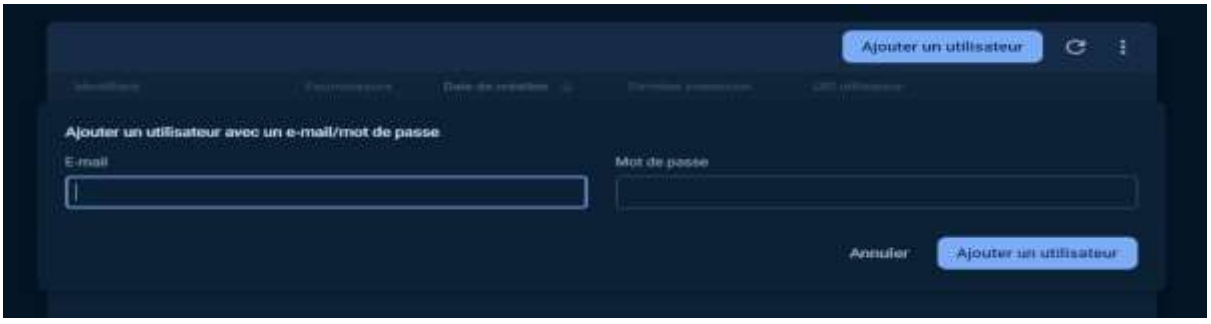
1. Header Builder: This builder allows you to customize the header of the sign-in screen.
2. Subtitle Builder: Use this builder to customize the subtitle or additional information displayed on the sign-in screen.
3. Footer Builder: Customize the footer of the sign-in screen using this builder.
4. Side Builder: This builder lets you customize the side elements or additional components on the sign-in screen.

Create a user

All the code for the screen is complete now. Before signing in, we must create a user. This can be achieved through the "Register" screen or by creating a user directly in the Firebase console.



1. Click the "Add user" button.



Enter an email address and password for the new user, then click "Add user".

Profile Screen

FlutterFire UI also offers a ProfileScreen widget, which provides significant functionality in just a few lines of code.

Add ProfileScreen widget

Navigate to the home.dart file in your text editor and replace the existing code with the following:
The updated code includes a callback passed to the IconButton.onPressed method. When the IconButton is pressed, the application generates a new anonymous route and navigates to it. This route will display the ProfileScreen widget returned from the MaterialPageRoute.builder callback.

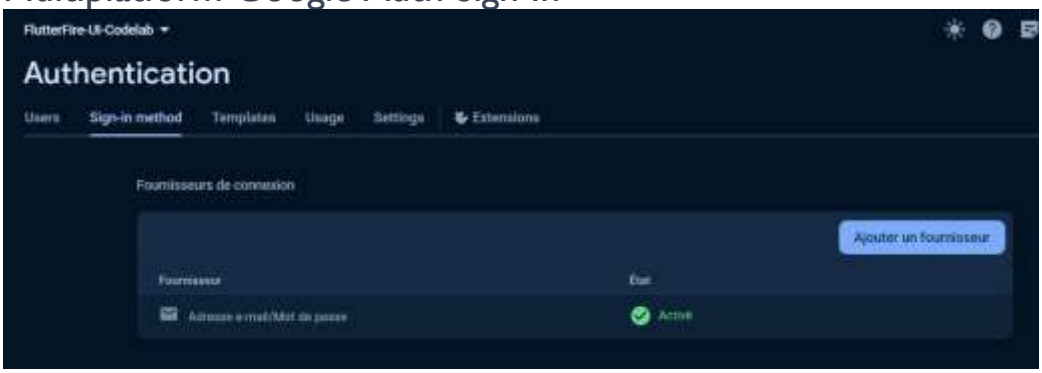
Signing Out

Now, if we press the "Sign out" button, the app will not change. It will sign the user out, but it will not be Once back in the AuthGate widget, update home.dart. Use ProfileScreen.actions to handle authentication changes. For instance, SignedOutAction triggers when the user signs out, navigating back. With only one route showing sign-in or home based on authentication, signing out takes you to the SignIn page. Then, customize the Profile Page.

Add Children

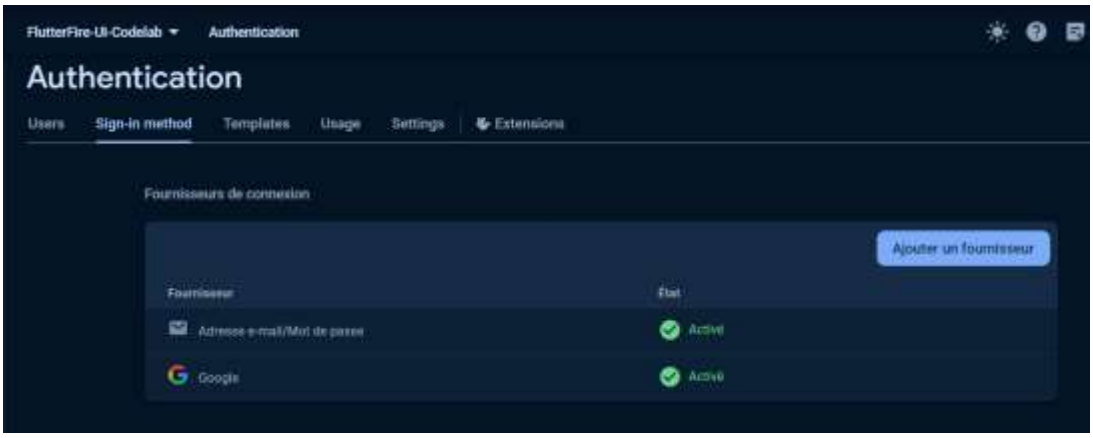
The ProfileScreen widget has an optional argument named children. It takes a list of widgets, which will be vertically aligned inside a Column widget within the ProfileScreen. These widgets will be placed above the "Sign out" button. Modify the code in home.dart to include the company logo, resembling the sign-in screen.

Multiplatform Google Auth Sign In



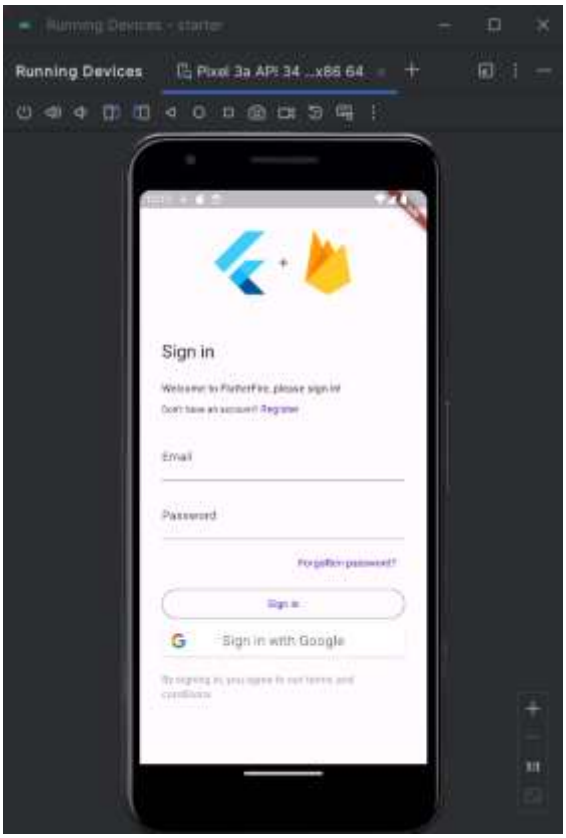
- Select "Google".

- Toggle the switch labeled "Enable", and press "Save"
- Confirm that the Google sign-in provider has been added.

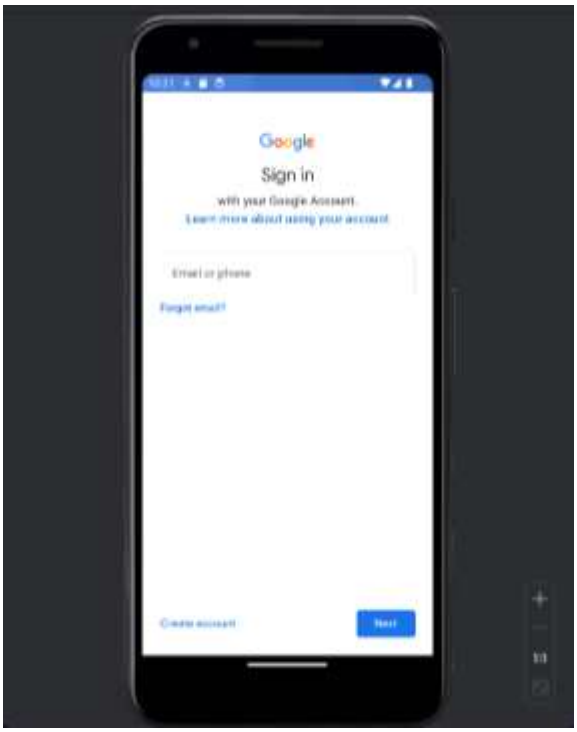


To add a Google sign-in button, insert GoogleProvider(clientId: "YOUR_WEBCLIENT_ID") in the SignInScreen widget configuration within auth_gate.dart. Navigate to Firebase Console's Authentication providers page, select Google, and copy the "Web client ID" value from the "Web SDK configuration" panel. Return to your text editor and update GoogleProvider's instance in auth_gate.dart with this copied ID.

Outcome




Forgotten password


Google
Sign in
with your Google Account
Email or phone
Next


Google
Ajouter un numéro de téléphone ?
Par exemple, votre numéro sera utilisé pour
Fonctionnement