Application d'Authentification Flutter avec Firebase

Aperçu du Projet

Cette application Flutter offre une fonctionnalité d'authentification utilisateur de base en utilisant Firebase, permettant la connexion et l'inscription des utilisateurs. L'application inclut trois écrans principaux : **Login**, **Signup**, et **Home**, ainsi qu'une classe `AuthService` dans le dossier `service` pour gérer l'authentification avec Firebase.

Table des Matières

Structure du Projet

Le projet est organisé dans les dossiers et fichiers suivants :

```
lib/
├── main.dart              # Point d'entrée de l'application
├── screens/
│   ├── login.dart         # UI et logique de l'écran de connexion
│   ├── signup.dart        # UI et logique de l'écran d'inscription
│   └── home.dart          # Écran d'accueil après la connexion
└── service/
    └── auth_service.dart  # Logique d'authentification Firebase
```

Configuration de Firebase

1. **Projet Firebase** : Un projet Firebase a été créé et configuré pour l'application, incluant l'activation de **l'authentification par Email/Mot de Passe** dans la Console Firebase.
2. **SDK Firebase** : Les dépendances du SDK Firebase, comme `firebase_auth`, ont été ajoutées au fichier `pubspec.yaml` du projet.
3. **Fichiers de Configuration** : Les fichiers de configuration Firebase pour les plateformes Android (`google-services.json`) et iOS (`GoogleService-Info.plist`) ont été ajoutés pour intégrer Firebase à l'application.

Écrans et Fonctionnalités

Écran de Connexion

- **Fichier** : `screens/login.dart`
- **Description** : Cet écran de connexion présente un formulaire simple avec des champs pour l'email et le mot de passe, ainsi qu'un bouton de connexion. En appuyant sur le bouton, le service `AuthService` est utilisé pour authentifier l'utilisateur. En cas de succès, l'utilisateur est redirigé vers l'écran d'accueil ; sinon, un message d'erreur s'affiche.
- **Code Principal** :

```
AuthService().loginWithEmailAndPassword(email, password);
```

Écran d'Inscription

- **Fichier** : `screens/signup.dart`
- **Description** : L'écran d'inscription permet aux nouveaux utilisateurs de s'inscrire avec un email et un mot de passe. Après avoir saisi les informations et appuyé sur le bouton d'inscription, `AuthService` crée un nouvel utilisateur dans Firebase. En cas de succès, l'utilisateur est redirigé vers l'écran d'accueil.
- **Code Principal** :

```
AuthService().registerWithEmailAndPassword(email, password);
```

Écran d'Accueil

- **Fichier** : `screens/home.dart`
- **Description** : L'écran d'accueil est affiché après une connexion ou une inscription réussie. Il peut être personnalisé pour inclure des informations spécifiques à l'utilisateur. Actuellement, il affiche un message de bienvenue et un bouton de déconnexion, qui permet à l'utilisateur de se déconnecter et de revenir à l'écran de connexion.
- **Code Principal** :

```
AuthService().signOut();
```

AuthService

- **Fichier** : `service/auth_service.dart`
- **Description** : La classe `AuthService` est responsable de l'interaction avec Firebase Authentication. Elle contient des méthodes pour la connexion, l'inscription et la déconnexion des utilisateurs, centralisant la logique Firebase.
- **Méthodes** :
  - `loginWithEmailAndPassword(String email, String password)`: Connecte un utilisateur existant.
  - `registerWithEmailAndPassword(String email, String password)`: Inscrit un nouvel utilisateur.
  - `signOut()`: Déconnecte l'utilisateur actuel et redirige vers l'écran de connexion.

Exemple de Code

```
class AuthService {
  final FirebaseAuth _auth = FirebaseAuth.instance;

  // Connexion avec email et mot de passe
  Future<User?> loginWithEmailAndPassword(String email, String password) async {
    try {
      UserCredential userCredential = await _auth.signInWithEmailAndPassword(
        email: email,
        password: password,
      );
      return userCredential.user;
    } catch (e) {
      print("Erreur de connexion : $e");
      return null;
    }
  }

  // Inscription avec email et mot de passe
  Future<User?> registerWithEmailAndPassword(String email, String password) async
{
    try {
      UserCredential userCredential = await _auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );
      return userCredential.user;
    } catch (e) {
      print("Erreur d'inscription : $e");
      return null;
    }
  }

  // Déconnexion
  Future<void> signOut() async {
    await _auth.signOut();
  }
}
```

Améliorations Futures

1. **Réinitialisation du Mot de Passe** : Implémenter une fonctionnalité de réinitialisation de mot de passe en utilisant l'API de Firebase.
2. **Gestion du Profil** : Ajouter la possibilité de mettre à jour les informations de profil de l'utilisateur, comme le nom d'utilisateur ou la photo de profil.
3. **UI Améliorée** : Améliorer l'interface utilisateur et ajouter des animations pour une meilleure expérience.
4. **Vérification d'Email** : Ajouter une vérification d'email pour s'assurer que l'email de l'utilisateur est valide.

Ce rapport donne une vue d'ensemble structurée des fonctionnalités de l'application Flutter, avec une description détaillée de chaque composant et exemple de code.