1. Contexte et présentation

Cette application mobile (écrite en **Dart** et utilisant **Flutter**) permet de réaliser diverses fonctionnalités en lien avec l'univers pharmaceutique. On y retrouve notamment :

- Une authentification (login, signup).
- L'accès à un **Home** où sont listés différents éléments (médicaments, rappels, etc.).
- Des pages spécialisées pour la consultation de médicaments ou la gestion des rappels.
- Un historique des actions de l'utilisateur.
- Des fonctionnalités de stockage (par ex. mémorisation des rappels pour prendre des médicaments).

2. Structure du projet Flutter

Voici l'arborescence simplifiée :

makefile

CopierModifier

```
C:.
 ├── app_layout.dart
 ├── main.dart
 ├── route.dart
|
 ├── assets
|    ├── boy.png
|    ├── fondBurger.jpg
|    ├── readme1.png
|    ├── readme2.png
|    ├── readme3.png
|    ├── woman.png
|    └── logo
|        ├── brand.png
```

```
|       ├── logo120x120.png
|       └── logo512x512.png
|
├── components
|   └── reminder_dialog.dart
|
├── models
|   └── reminder.dart
|
├── pages
|   ├── historique_page.dart
|   ├── home_page.dart
|   ├── login_page.dart
|   ├── main_page.dart
|   ├── medicament_page.dart
|   ├── profile_page.dart
|   ├── rappels_page.dart
|   └── signup_page.dart
|
├── services
|   └── auth_service.dart
|
└── utils
    └── reminder_storage.dart
```

2.1 Fichiers racine

- **main.dart** : Point d'entrée de l'application Flutter.
    - Appel de runApp(MyApp()).
    - Configuration générale (thème, initialisation de packages, etc.).
- **app_layout.dart** :
    - widget à la racine qui gère la structure globale. (Scaffold)
    - inclut l'AppBa et le Drawer.
- **route.dart** :
    - Fichier où sont définies les routes.
    - Sert à naviguer entre les différentes **pages** (/login, /home, /profile, etc.).

2.2 Dossier assets

- Contient toutes les ressources statiques (images, logos, etc.).
- Sous-dossier logo pour regrouper les différentes tailles de logos (icônes d'app, logo marketing, etc.).

2.3 Dossier components

- **reminder_dialog.dart** : exemple d'un **widget** réutilisable (affiche un pop-up de rappel).
- Les composants (ou widgets) partagés et génériques sont placés ici (boutons personnalisés, cards, dialogues, etc.).

2.4 Dossier models

- **reminder.dart** : modèle de données pour un rappel.
- Les classes de modèles décrivent la structure de vos objets métiers.
- Possible d'y ajouter des méthodes de sérialisation (JSON) si besoin.

2.5 Dossier pages

Il s'agit des **écrans** principaux de l'application.

1. **historique_page.dart** : Historique d'actions, de rappels, de commandes, etc.
2. **home_page.dart** : Écran d'accueil après connexion (accès rapide aux fonctionnalités, résumé, etc.).

3. **login_page.dart** : Écran de connexion, champ email/mot de passe, etc.
4. **signup_page.dart** : Écran d'inscription pour nouveaux utilisateurs.
5. **main_page.dart** : Page d'acceuil.
6. **medicament_page.dart** : Liste, Filtre et détail de médicaments.
7. **profile_page.dart** : Gère l'affichage et la modification du profil utilisateur.
8. **rappels_page.dart** : Page dédiée à la gestion/paramétrage des rappels (heure, notification, etc.).

2.6 Dossier services

- **auth_service.dart** :
    - Contient la logique de connexion/inscription.
    - Gère la session utilisateur.

2.7 Dossier utils

- **reminder_storage.dart** :
    - Logique de stockage local (shared_preferences) pour les rappels.
    - Gère la persistance de données (sauvegarde, chargement, mise à jour).

3. Cycle de vie d'une fonctionnalité

Pour illustrer :

1. **Modèle** : Reminder (défini dans models/reminder.dart).
2. **Service** : La logique associée (planifier des notifications locales, enregistrer un rappel dans la base locale.
3. **Widget** : L'écran rappels_page.dart affiche la liste de rappels et propose l'ajout/la suppression.
4. **Dialog** : Pour ajouter un rappel, on peut appeler un composant modal du dossier components/ (reminder_dialog.dart).
5. **Navigation** : Gérée via route.dart; l'utilisateur peut passer de home_page.dart → rappels_page.dart → etc.

4. Configuration et exécution

4.1 Prérequis

- **Flutter SDK** installé (vérifier via flutter doctor).
- **Dart** >= 2.xx (inclus avec Flutter).
- **Android Studio**, **Visual Studio Code**, ou autre IDE/éditeur compatible.

4.2 Lancement de l'application

1. S'assurer que les dépendances dans pubspec.yaml sont à jour :

bash

CopierModifier

flutter pub get

2. Lancer l'émulateur (Android ou iOS) ou brancher un appareil réel.
3. Démarrer l'application :

bash

CopierModifier

flutter run

4. En mode **release** (pour un build final) :

bash

CopierModifier

flutter build apk   # Pour Android

flutter build ios   # Pour iOS (nécessite Xcode, provisioning, etc.)

5. Gestion de la navigation (dans route.dart)

configuration :

import 'package:flutter/material.dart';

import 'package:medicall/pages/historique_page.dart';

import 'package:medicall/pages/home_page.dart';

import 'package:medicall/pages/main_page.dart';

import 'pages/medicament_page.dart';

```
import 'pages/signup_page.dart';
import 'pages/login_page.dart';
import 'pages/rappels_page.dart';
import 'pages/profile_page.dart';
import 'app_layout.dart'; // Import du layout

Map<String, WidgetBuilder> appRoutes = {
  '/home': (context) => const HomePage(),

  '/menu': (context) => const AppLayout(
      child: MenuPage()), // rajouter une page pour decrire l'application

  '/rappels': (context) =>
      AppLayout(child: ReminderPage()), // Enveloppé dans AppLayout
  '/profile': (context) =>
      const AppLayout(child: ProfilePage()), // Enveloppé dans AppLayout

  '/signup': (context) => const SignUpPage(),
  '/login': (context) => const LoginPage(),

  '/medicaments': (context) =>
      const AppLayout(child: MedicamentListPage()), // Enveloppé dans AppLayout

  '/historique': (context) =>
      const AppLayout(child: HistoriqueListPage()), // Enveloppé dans AppLayout
};
```

6. Authentification (dans auth_service.dart)

- compare l'email et le password fournis.
- la méthode renvoie true si et seulement si l'email et le mot de passe correspondent exactement à ces valeurs prédéfinies ; sinon, elle renvoie false.

Code :

class AuthService { bool login(String email, String password) { return email == "test@example.com" && password == "password"; } }

7. Stockage local (dans reminder_storage.dart)

- Utilisation de SharedPreferences

Code :

```
import 'dart:convert';

import 'package:shared_preferences/shared_preferences.dart';

import '../models/reminder.dart';


class ReminderStorage {

 static const String _reminderKey = 'reminders';


 // Sauvegarder la liste des rappels dans SharedPreferences

 static Future<void> saveReminders(List<Reminder> reminders) async {

  final prefs = await SharedPreferences.getInstance();

  final List<String> remindersJson = reminders.map((reminder) {

   return json.encode({

    'name': reminder.name,

    'time': reminder.time,

    'recurrence': reminder.recurrence,

   });

  }).toList();

  await prefs.setStringList(_reminderKey, remindersJson);
```

```
  }

  // Charger les rappels depuis SharedPreferences
  static Future<List<Reminder>> loadReminders() async {
    final prefs = await SharedPreferences.getInstance();
    final List<String>? remindersJson = prefs.getStringList(_reminderKey);

    if (remindersJson == null) {
      return [];
    }

    return remindersJson.map((reminderJson) {
      final data = json.decode(reminderJson);
      return Reminder(
        name: data['name'],
        time: data['time'],
        recurrence: data['recurrence'],
      );
    }).toList();
  }
}
```

9. Conclusion

Cette application Flutter est organisée de manière **classique** et **scalable** :

- **main.dart** : initialisation de l'app.
- **route.dart** : gestion centralisée de la navigation.
- **Pages** : chaque écran dans un fichier dédié.
- **Components** : widgets réutilisables.
- **Models** : structures de données (Reminder).
- **Services** : logique métier / appels API (AuthService).
- **Utils** : fonctions/outils divers, comme le stockage local.