TP1 - Les bases de Flutter

HAI912I - Développement embarqué

Groupe n°8 :

COSSU Arnaud *21908322*
POINTEAU Gabrielle *21917975*
SAID Adam *21905365*

Table des matières

1 Choix de conception

1.1 Diagrammes de classes d'UML

L'idée du TP et de faire une application qui réunis le profil utilisateur et un quiz. Pour cela, nous avons choisis de faire une seule et même application qui gère les deux cas.

Donc on a un Menu, qui permet d'accéder à 2 pages : la page de profil ; la page de quiz.

On peut voir sur la figure 2 à la page 5 l'aspect général de toute l'application. Il detail les différentes dépendances entre les classes du projet.

1.2 Architecture

Au niveau de l'architecture, nous avons plusieurs classes réparties dans les 4 fichiers : `main.dart`, `profil.dart` et `quizz.dart`, `resultat_quiz.dart`.

1.2.1 `main.dart`

Commençons par le fichier `main.dart`. C'est la première page de l'application Flutter réalisé. Il définit le thème de l'application que nous avons choisis, en choissisant des couleurs, des styles personalisés pour chaque texte. On a donc choisis différents éléments à partir de ce code :

```
// Couleurs
const fushia = Color.fromARGB(255, 96, 3, 54);
const fushiaClaire = Color.fromARGB(255, 255, 202, 231);
const fushiaSemiClaire = Color.fromARGB(255, 196, 103, 154);
// Marges / Padding
const margePaddingAll50 = EdgeInsets.all(50);
const margePaddingAll75 = EdgeInsets.all(75);
const margePaddingAll20 = EdgeInsets.all(20);
// Espace
const espaceFixe10 = SizedBox(height: 10);
const espaceFixe20 = SizedBox(height: 20);
const espaceFixe50 = SizedBox(height: 50);
const espaceFixe80 = SizedBox(height: 80);
// Tailles d'objet
const sizeButton = Size(200, 50);
// Theme
const textTitre = TextStyle(fontSize: 24, fontWeight: FontWeight.bold);
const textBody = TextStyle(fontSize: 16);
const textThemePersonel = TextTheme(
  headline6: textTitre, // Exemple de style de texte
  bodyText2: textBody, // Autre exemple de style de texte
);
var themePersonalise = ThemeData(
  primarySwatch: MaterialColor(
```

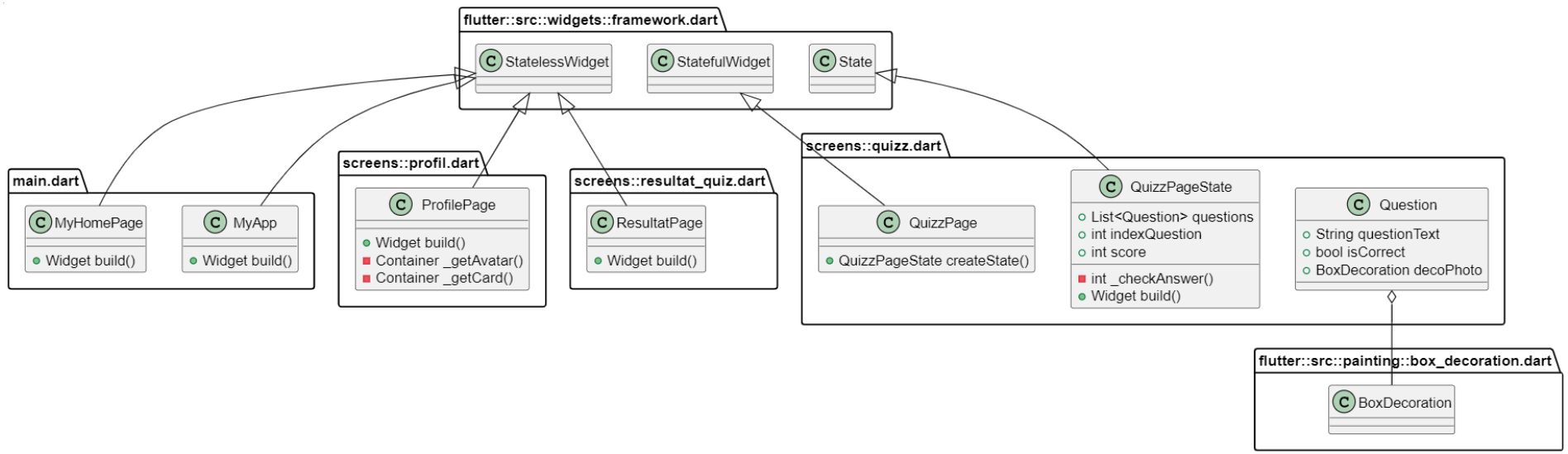

FIGURE 1 – Diagramme UML de dépendances des classes

```
    fushia.value,
    <int, Color>{
      50: fushia.withOpacity(0.1),
      100: fushia.withOpacity(0.2),
      200: fushia.withOpacity(0.3),
      300: fushia.withOpacity(0.4),
      400: fushia.withOpacity(0.5),
      500: fushia.withOpacity(0.6),
      600: fushia.withOpacity(0.7),
      700: fushia.withOpacity(0.8),
      800: fushia.withOpacity(0.9),
      900: fushia,
    },
  ), // Couleur principale de l'application, // Couleur principale de l'application
  visualDensity: VisualDensity.adaptivePlatformDensity,
  fontFamily: 'Roboto', // Police de caractères par défaut
  textTheme: textThemePersonel,
);
```

Ensuite, au niveaux des classes présentes dans `main.dart` :

* La classe `MyApp` est une classe `StatelessWidget` qui a une méthode build pour construire l'interface utilisateur. De plus, elle crée une instance de `MyHomePage` comme page d'accueil. Par ailleurs, elle gère la navigation entre les différentes pages de l'application. On y retrouve ainsi les 4 routes possibles :
  - `/` : route principale du menu
  - `/profil` : route qui amène à lapage sur le profil (cf. Exercice 1)
  - `/quizz` : route qui amène à la page sur le quiz (cf. Exercice 2)
  - `/resultat_quizz` : route après le quiz qui amène au résultat de l'utilisateur sur le quiz.
* La classe `MyHomePage` est une classe `StatelessWidget` et contient la page d'accueil de l'application dont le menu principale. Ce menu est constitué de 2 boutons pour accéder aux pages de profil (ProfilePage) et au quiz (QuizzPage).

1.2.2 `profil.dart`

Une fois qu'on a appuyer sur le bouton de profil, on accède à une nouvelle page : `ProfilPage`. Elle correspond à l'exercice 1 du TP.

Au niveaux des classes on en aura qu'une :

* La classe `ProfilePage` est une classe StatelessWidget avec des méthodes privées _getAvatar et _getCard pour construire différentes parties de la page. Elle affiche des informations de profil comme le nom, l'adresse e-mail et les réseaux sociaux.

  Les deux méthodes privés permettent de :

  - `_getAvatar` : Récupérer l'avatar c'est à la photo de profil de l'utilisateur. Elle permet de centrer, ajuster l'image et de la couper en cercle pour l'affichage.

— `_getCard` : Récuperer les données de la carte d'information. On va pouvoir afficher à l'aide de cette méthode les inforamtions personnelles du projet.

Pour positioner l'avatar par dessus la card on a utiliser un Translate car Positioned permettait de décaler l'itmage mais on ne voyer pas le bas de l'image ainsi avec Translate on le voit correctement.

1.2.3 `quizz.dart`

Une fois qu'on a appuyé sur le bouton de quiz, on accède à une nouvelle page : `QuizzPage`. Elle correspond à l'exercice 2 du TP.

Au niveaux des classes on aura :

* La classe `Question` représente une question du quiz avec des propriétés telles que le texte de la question, si la réponse est correcte et une décoration d'image qui contient l'image du questionnaire.
* La classe `QuizzPage` est une StatefulWidget qui gère l'état du quiz, les questions, l'indice de question, et le score. Elle utilise `_checkAnswer` pour vérifier les réponses des utilisateurs.

  Si la réponse est la bonne, on passe à la question suivante en ajoutant 1 au score sinon on a le même score en passant à la question suivante également. Une fois le quiz terminer on passe aux résultats.
* La classe `QuizzPageState` est la classe d'état de `QuizzPage`. Elle gère les changements d'état liés au quiz, tels que l'incrémentation de l'indice des questions et le calcul du score.

1.2.4 `resultat_quizz.dart`

Une fois qu'on a terminer le quiz, on accède à une nouvelle page : `ResultatPage`.

Au niveaux des classes on aura une classe :

* La classe `ResultatPage` est une classe `StatelessWidget` avec une méthode build pour construire l'interface utilisateur de la page de résultats. Elle affiche le score de l'utilisateur et un message en fonction de son score.

L'architecture suit le modèle Flutter avec des classes `Stateless` et `StatefulWidget` pour gérer l'interface utilisateur et l'état de l'application.

1.3 Workflow

1. L'utilisateur lance l'application, et la classe `MyApp` est chargée en tant que point d'entrée.
2. La classe `MyApp` définit le thème de l'application et configure les routes, associant chaque route à une classe de page spécifique.
3. L'utilisateur arrive sur la page d'accueil (`MyHomePage`) qui affiche un menu avec des boutons pour accéder au profil (`ProfilePage`) ou au quiz (`QuizzPage`).
4. Lorsque l'utilisateur clique sur le bouton "Votre profil", la navigation conduit à la classe ProfilePage, qui affiche les informations du profil de l'utilisateur.

5. Si l'utilisateur clique sur le bouton "Quizz", la navigation le conduit à la classe `QuizzPage`, qui affiche une série de questions avec des images associées. L'utilisateur répond aux questions en sélectionnant Vrai ou Faux.
6. Après avoir répondu à toutes les questions, la classe `QuizzPage` passe à la classe ResultatPage pour afficher le score obtenu dans le quiz. Un commentaire est affiché en fonction du score.
7. La classe `ResultatPage` permet à l'utilisateur de revenir à la page d'accueil pour explorer d'autres fonctionnalités de l'application.

1.4 Diagramme de séquences

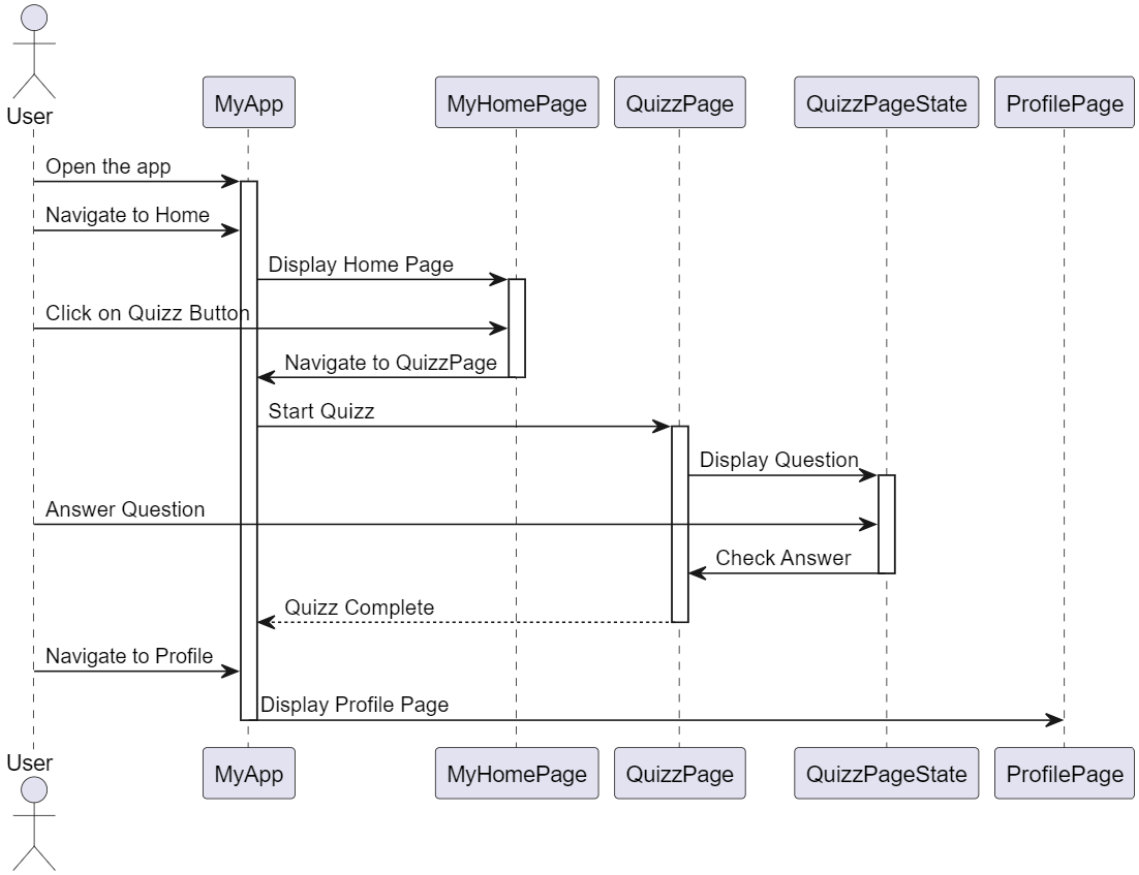

FIGURE 2 – Diagramme de séquence

2 Snapshot de notre application

On va pouvoir montrer le fil directeur de l'application à partir de snapshot de l'application.

On commence par le menu (à gauche) et la page de profil (à droite) :

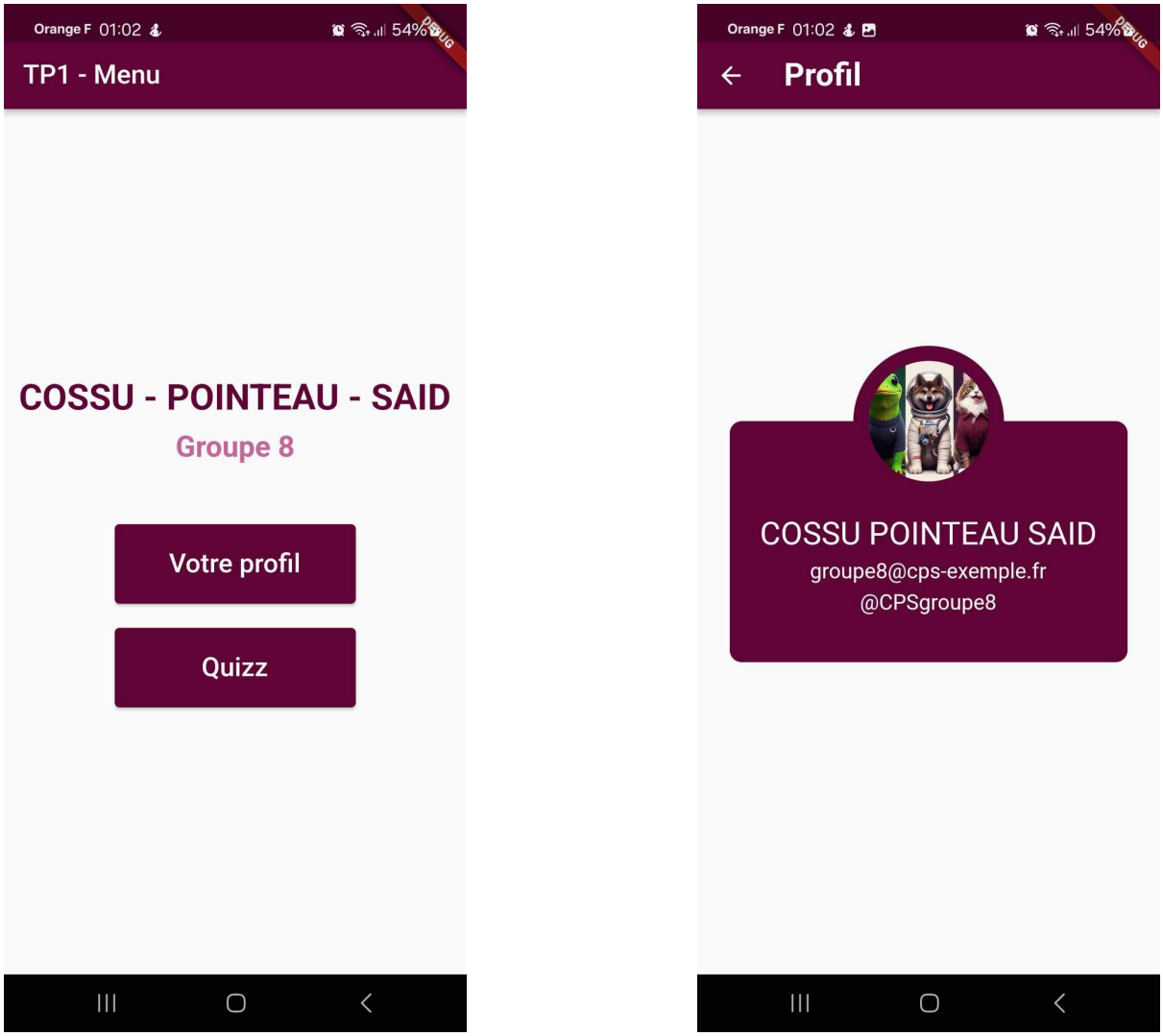

FIGURE 3 – Pages d'affichage de menu et de profil

Puis on va avoir les pages de quiz où il y aura 2 pages de question et 2 page de résultat à la fin ensuite montrant les différentes possibilités.

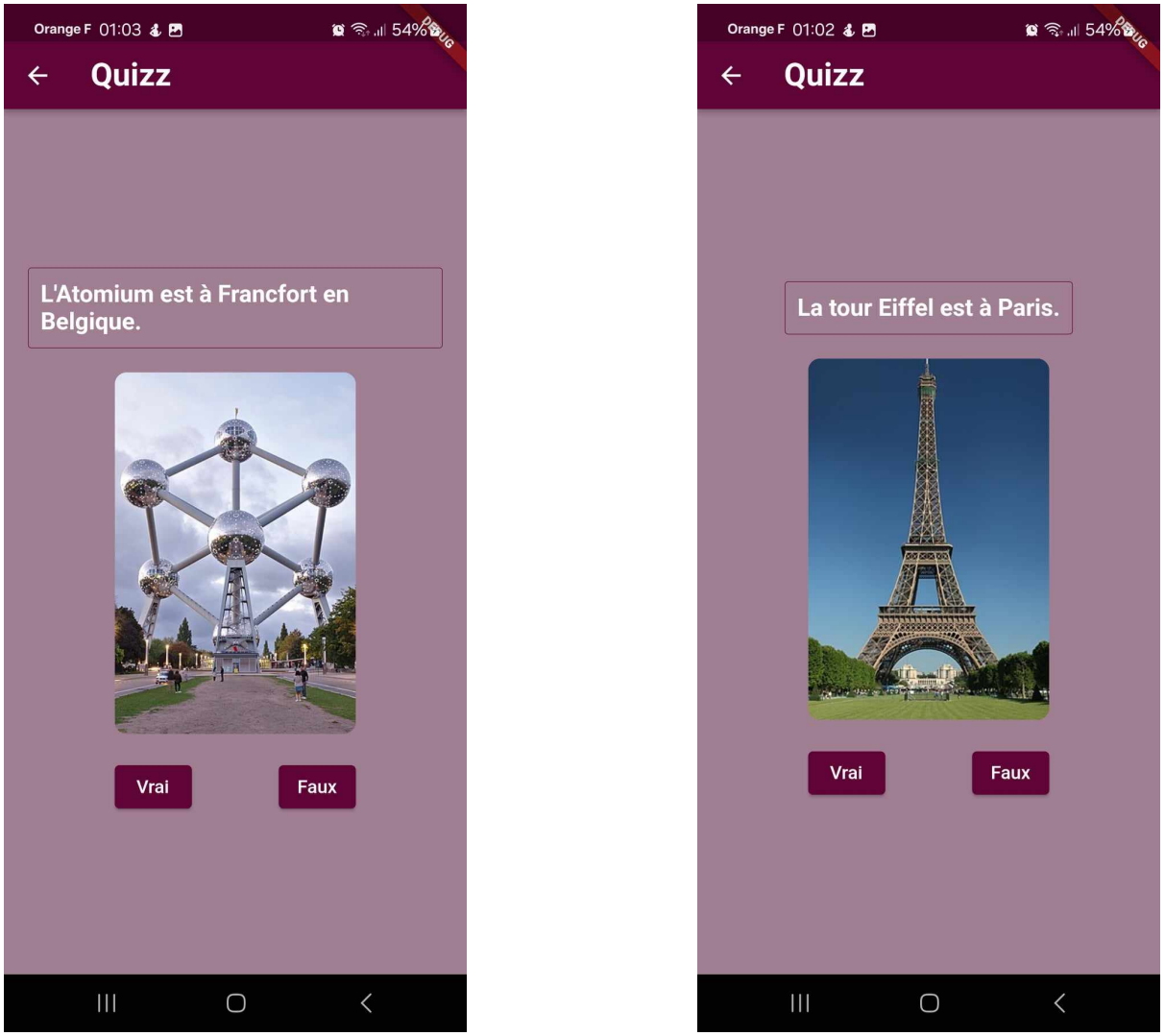

FIGURE 4 – Pages d'affichage du quiz

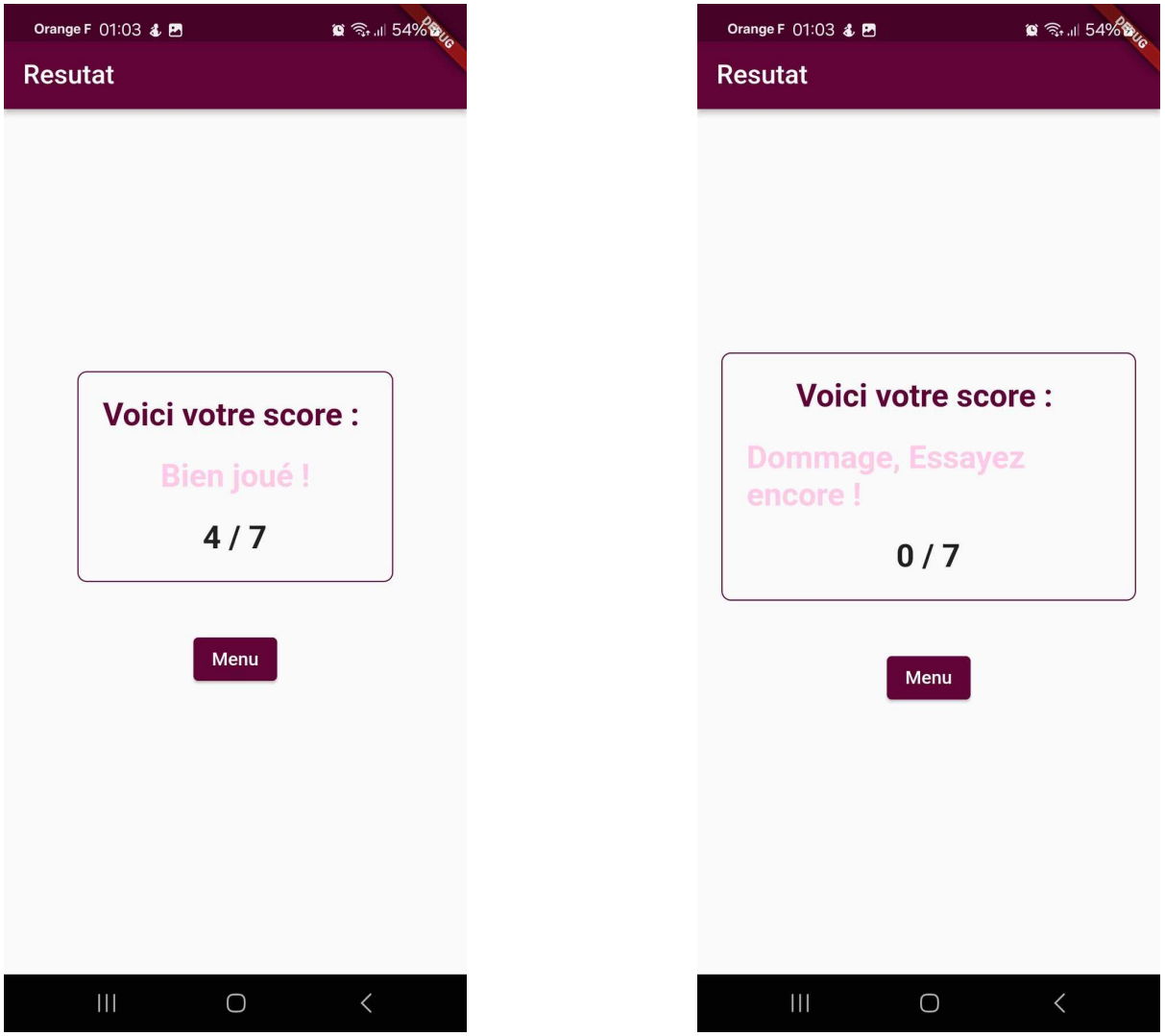

FIGURE 5 – Pages d'affichage du resultat

3 Conclusion

Ainsi, pour ce 1er TP, nous avons vu les bases de Flutter avec l'utilisation des state pour le quiz ainsi que le positionnement des widgets entre eux pour le profil.