Création d'interfaces graphiques en Flutter

Master 2 Informatique Parcours Génie logiciel

Étudiants :
Nouria AINAS
Djouher MOULAHCENE

Encadrants :
Bachar RIMA

27 décembre 2023

Table des matières

1 Introduction :

Dans le cadre de travaux pratiques visant à nous familiariser avec la création d'interfaces graphiques sous Flutter, on a abordé spécifiquement l'utilisation des widgets stateless et stateful, ainsi que la gestion des états à l'aide des Blocs (Business Logic Component). Cette exploration pratique vise à renforcer notre compréhension des concepts fondamentaux nécessaires à la conception d'interfaces interactives et réactives.

2 Exercice 1 : Profile Card

Dans le cadre de l'exercice 1 portant sur la création d'une application de carte de profil, notre objectif est de concevoir un widget représentant le profil d'une personne. Ce widget, dérivé de la classe **StatelessWidget**, regroupe deux éléments essentiels : une photo de profil et des informations relatives au profil telles que le nom, le prénom et l'e-mail. Pour réaliser cela, nous avons utilisé des widgets tels que **Scaffold** pour la barre d'application, **Container** pour la mise en page, **BoxDecoration** pour la décoration, et **Stack** pour superposer plusieurs éléments.

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

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'FirstCard',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ), // ThemeData
      home: const MyHomeCardPage(title: 'FirstCard'),
    ); // MaterialApp
  }
```

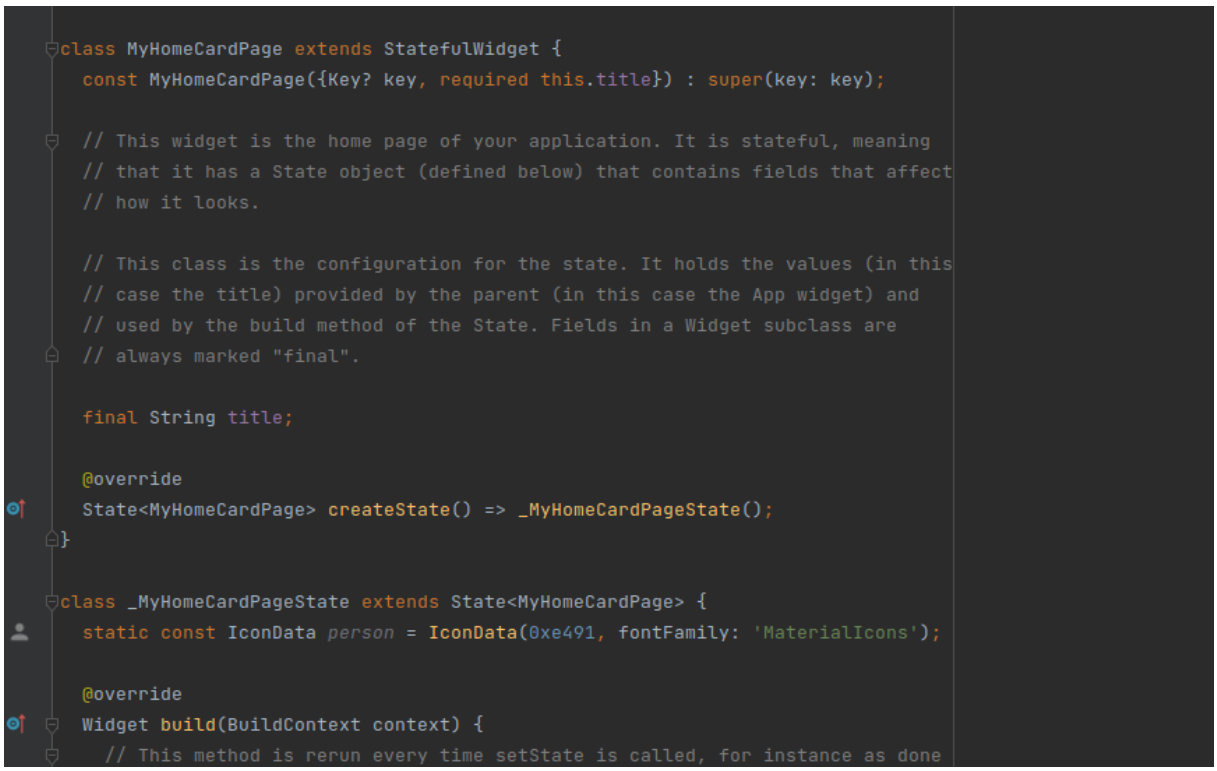
FIGURE 1 – Utilisation de StatelessWidget

```dart
return Scaffold(
  appBar: AppBar(
    // Here we take the value from the MyHomePage object that was created by
    // the App.build method, and use it to set our appbar title.
    title: Text(widget.title),
  ), // AppBar
  body: Container(
    alignment: Alignment.center,
    child: Stack(
      clipBehavior: Clip.none,
      children: <Widget>[
        Container(
          width: 290,
          height: 190,
          color: Colors.pink,
          child: Center(
            child: Column(
              children: [
                Container(
                  width: 0,
                  height: 70,
                ), // Container
                Container(
                  child: Text("Djouher Nouria",
                      textAlign: TextAlign.center,
                      textScaleFactor: 1.0,
                      style: TextStyle(
                        color: Colors.white,
```

FIGURE 2 – Utilisation des widgets

La page d'accueil, définie dans `ProfileHomePage`, intègre ces composants, offrant ainsi une structure claire et esthétique. De plus, des méthodes telles que `getCard()` et `getAvatar()` ont été implémentées pour encapsuler la logique de création de la carte de profil et de l'avatar, respectivement.
Ce squelette de l'application offre une base solide pour explorer la création d'interfaces graphiques en utilisant Flutter, mettant en avant des concepts tels que les widgets Stateless et les différentes possibilités de mise en page.

La figure suivante illustre l'implémentation de la page d'accueil.



```dart
class MyHomeCardPage extends StatefulWidget {
  const MyHomeCardPage({Key? key, required this.title}) : super(key: key);

  // This widget is the home page of your application. It is stateful, meaning
  // that it has a State object (defined below) that contains fields that affect
  // how it looks.

  // This class is the configuration for the state. It holds the values (in this
  // case the title) provided by the parent (in this case the App widget) and
  // used by the build method of the State. Fields in a Widget subclass are
  // always marked "final".

  final String title;

  @override
  State<MyHomeCardPage> createState() => _MyHomeCardPageState();
}

class _MyHomeCardPageState extends State<MyHomeCardPage> {
  static const IconData person = IconData(0xe491, fontFamily: 'MaterialIcons');

  @override
  Widget build(BuildContext context) {
    // This method is rerun every time setState is called, for instance as done
```

FIGURE 3 – HomePage

La figure suivante illustre le rendu final.

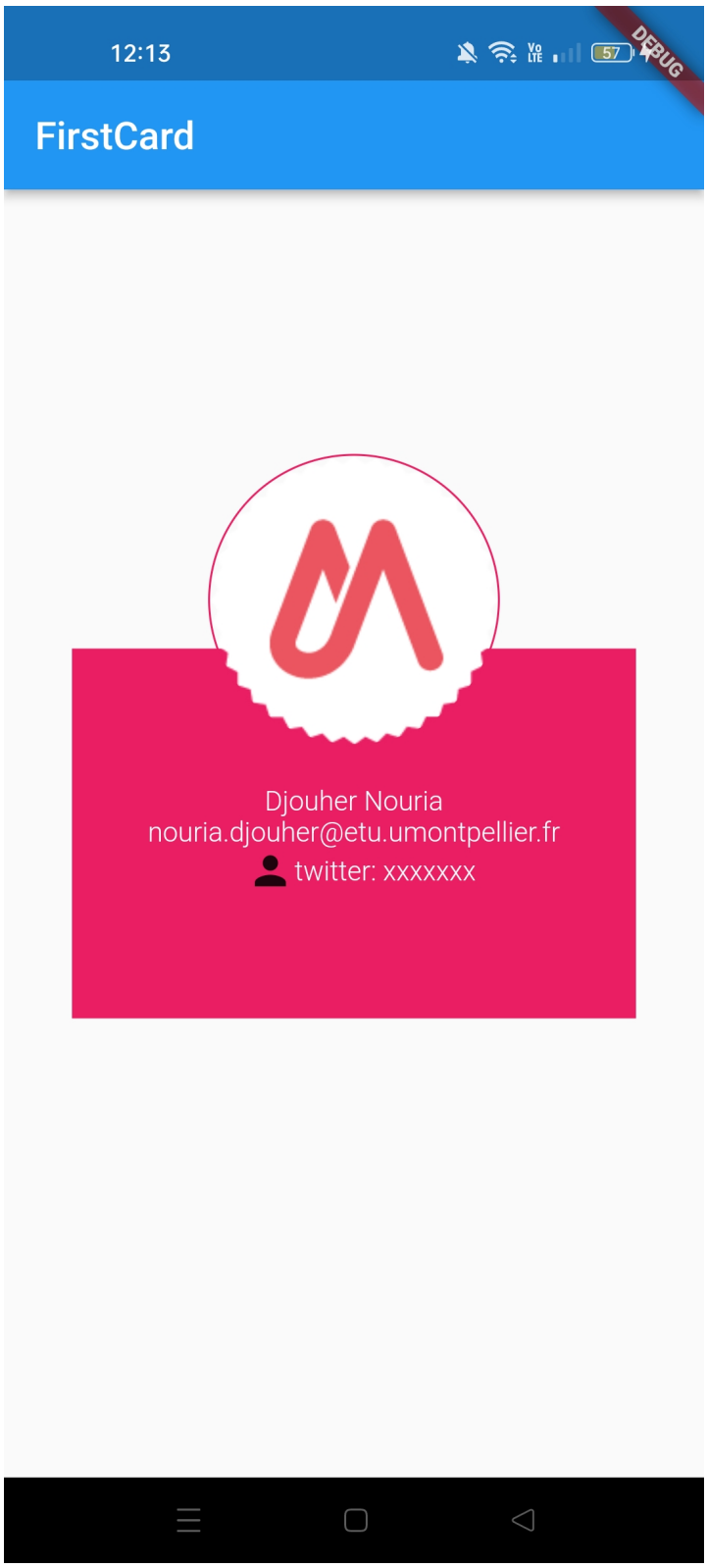


FIGURE 4 – Profile Card

3 Exercice 2 : Quizz

Dans le cadre de notre deuxième exercice, nous avons développé une application Quiz avec Flutter, en mettant en œuvre le concept de **StatefulWidget** pour gérer dynamiquement l'interface utilisateur. L'objectif principal de cette réalisation était de permettre à l'utilisateur de répondre de manière interactive à des questions portant sur une thématique spécifique.

3.1 Structure de l'Application

L'application se compose principalement de trois écrans clés : la page d'accueil (`HomePage`), l'écran de jeu (`PlayQuiz`), et l'écran de résultats (`Result`). La transition entre ces écrans est gérée de manière fluide grâce à l'utilisation de StatefulWidget.

3.2 Page d'Accueil

La page d'accueil (`HomePage`) présente une image de fond avec un bouton "Commencer le quiz". Lorsque l'utilisateur appuie sur ce bouton, il est redirigé vers l'écran de jeu (`PlayQuiz`) pour commencer le quiz.La figure suivante illustre l'écran.

FIGURE 5 – Page d'Accueil

3.3 Écran de Jeu

L'écran de jeu (`PlayQuiz`) affiche une question à la fois, avec des options "Vrai" et "Faux" pour chaque question. Un minuteur animé indique le temps restant pour répondre

à chaque question. Lorsque l'utilisateur répond, l'application vérifie la réponse, attribue des points, et passe à la question suivante.Les figures suivantes illustrent les écrans.

FIGURE 6 – Écran 1 de Jeu

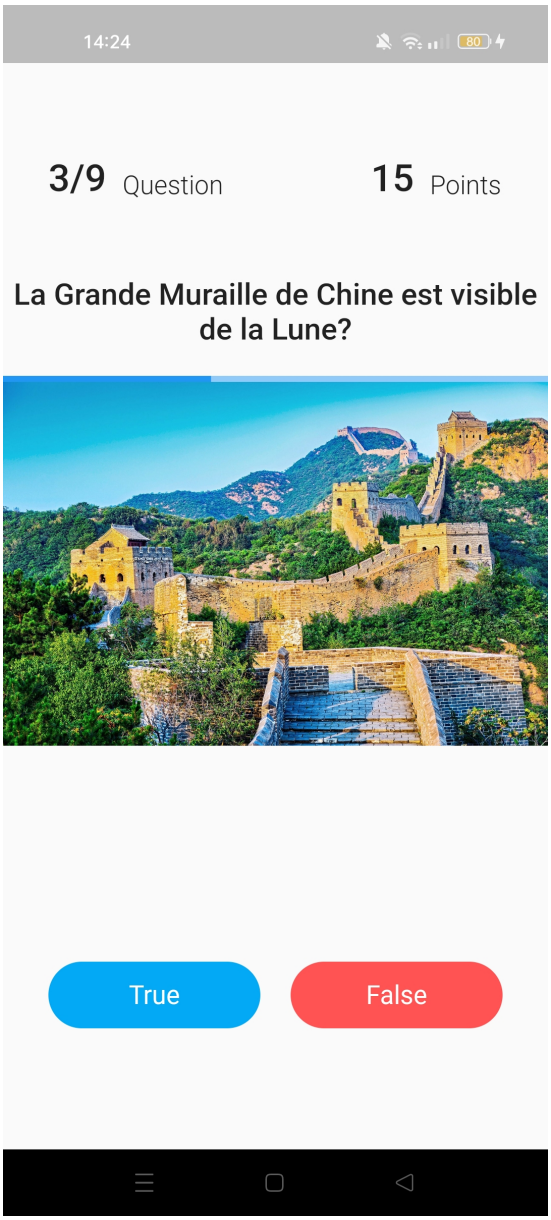


FIGURE 7 – Écran 3 de Jeu

3.4 Écran de Résultats

Une fois le quiz terminé, l'utilisateur est redirigé vers l'écran de résultats (`Result`) qui affiche le score obtenu, le nombre de réponses correctes, incorrectes et non tentées. De plus, un message personnalisé en fonction du score est affiché, offrant une expérience utilisateur engageante.La figure suivante illustre l'écran.

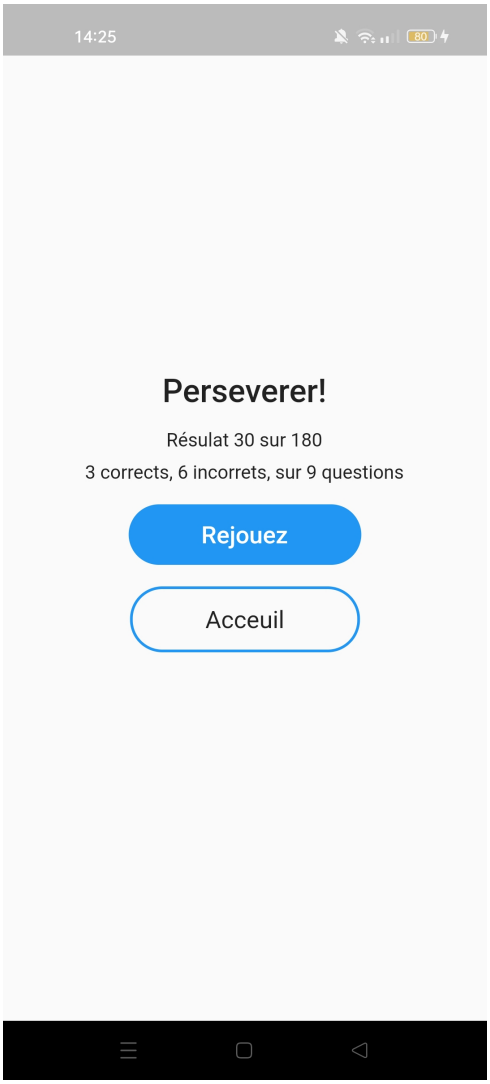


FIGURE 8 – Écran de Résultats

3.4.1 Lien (vidéos) d'éxécution de l'exercice 2 :

Pour illustrer un exemple d'exécution, vous pouvez regarder la vidéo en suivant le lien ci-dessous :
Regarder la vidéo de démonstration Exercice 2