Université de Montpellier
Faculté des science
Département Informatique

Master 2 : Intelligence artificielle et science de données

HAI912I - Développement mobile avancé, Iot et embarqué

TP1 Flutter

Réalisé par :

- Goussem Ayoub

Année universitaire : 2025/2026

Introduction

Dans le cadre de ce TP, j'ai développé une petite application Flutter composée d'une carte de profil et d'un quizz sur le thème des fruits et légumes. Ce projet m'a permis de mettre en pratique la construction d'interfaces avec `StatelessWidget` et `StatefulWidget`, la navigation entre plusieurs écrans et la gestion d'un état simple pour un quizz. Le code source complet est disponible sur mon dépôt GitHub : `https://github.com/ayoubgoussem/tp1-flutter`.

Toutes les images utilisées dans cette application (avatar du profil et illustrations des fruits et légumes) proviennent du site `https://www.flaticon.com/fr/`. Il s'agit d'icônes gratuites sous licence Flaticon, utilisables avec attribution. J'ai donc veillé à respecter les conditions de cette licence dans le cadre de ce TP.

1 Architecture générale

Pour ce TP, j'ai conçu une application Flutter avec une architecture centrée sur le widget racine `MyApp`. Ce widget crée un `MaterialApp` qui définit le thème global de l'application ainsi que la page principale `ProfileHomePage`, utilisée comme écran d'accueil. La navigation entre les différents écrans repose sur la pile de routes gérée par `Navigator` : depuis la page de profil, je pousse une route vers `QuizzPage`, puis cette page peut à son tour pousser une route vers `SummaryPage` pour afficher le récapitulatif du quizz.

J'ai ainsi choisi une structure reposant sur trois écrans indépendants :

— une page de profil (`ProfileHomePage`), de type `StatelessWidget`, dédiée uniquement à la présentation de la carte de profil ;

— une page de quizz (`QuizzPage`), de type `StatefulWidget`, qui contient la logique métier et l'état du quizz ;

— une page de récapitulatif (`SummaryPage`), à nouveau `StatelessWidget`, qui se contente d'afficher les résultats calculés.

2 Choix entre Stateless et Stateful (diagramme de classes)

Dans mon diagramme de classes UML, je représente uniquement les éléments principaux de l'application :

— `MyApp` **: StatelessWidget**
Ce widget configure le `MaterialApp` (thème, titre) et indique `ProfileHomePage` comme page initiale.

— `ProfileHomePage` **: StatelessWidget**
Cette classe affiche la carte de profil (avatar, informations personnelles) et le bouton qui permet d'accéder au quizz via `Navigator.push()`.

— `QuizzPage` **: StatefulWidget**
Elle possède un attribut `title : String` et crée une instance de `_QuizzPageState` via la méthode `createState()`. C'est ce choix d'un widget avec état qui me permet de faire évoluer l'interface au fur et à mesure des réponses.

— `_QuizzPageState` **: State<QuizzPage>**
Cette classe contient tout l'état du quizz :
— `_questions : List<Question>` : la liste des questions à poser ;
— `_currentIndex : int` : l'indice de la question courante ;
— `_userResults : List<bool>` : la liste des réponses (vrai/faux) de l'utilisateur.
Elle expose également les méthodes suivantes :
— `_answer(bool userChoice)` : traite le choix de l'utilisateur, compare avec la bonne réponse et met à jour la liste des résultats ;
— `_showSummary()` : déclenche la navigation vers la page de récapitulatif ;
— `build(BuildContext)` : construit l'interface pour la question courante.
— `Question` (classe modèle)
C'est une petite classe de données avec trois attributs :
— `text : String` : l'énoncé de la question ;
— `isCorrect : bool` : la bonne réponse (vrai ou faux) ;
— `imagePath : String` : le chemin vers l'image illustrant le fruit ou légume.
— `SummaryPage` **: StatelessWidget**
Cette page reçoit en paramètres `questions : List<Question>` et `results : List<bool>`, puis affiche le score final ainsi que le détail question par question.

J'insiste sur le fait que j'utilise un `StatefulWidget` uniquement pour la partie quizz, car l'interface doit être reconstruite après chaque réponse (changement d'index et nouvelle question). À l'inverse, le profil et le récapitulatif ne dépendent pas d'un état mutable local, ce qui justifie l'utilisation de `StatelessWidget`.

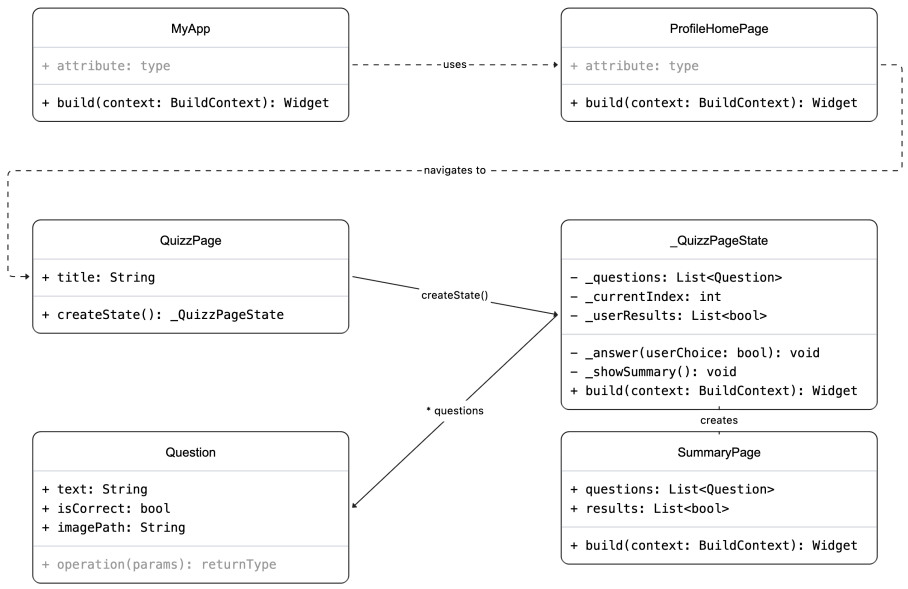


FIGURE 2 – Diagramme de classes UML

3 Workflow

Le fonctionnement global peut se décrire comme ceci :

1. Au lancement de l'application, `MyApp.build()` instancie le `MaterialApp` et affiche `ProfileHomePage` comme premier écran.
2. Sur `ProfileHomePage`, l'utilisateur clique sur le bouton "Aller au quizz". Ce clic déclenche un appel à `Navigator.push()` avec une `MaterialPageRoute` qui crée `QuizzPage`.
3. `QuizzPage.createState()` retourne une instance de `_QuizzPageState`. Dans l'état, j'initialise la liste de cinq objets `Question` (textes, réponses correctes, images).
4. Pour chaque question :
   — `_QuizzPageState.build()` sélectionne `_questions[_currentIndex]` et affiche l'image et le texte correspondants.
   — L'utilisateur appuie sur le bouton `VRAI` ou `FAUX`, ce qui appelle la méthode `_answer(userChoice)`.
   — `_answer()` compare `userChoice` avec `question.isCorrect`, ajoute `true` ou `false` dans `_userResults`, puis :
      — si ce n'est pas la dernière question, incrémente `_currentIndex` et appelle `setState()` pour reconstruire l'interface avec la question suivante ;
      — si c'était la dernière question, appelle `_showSummary()`.
5. La méthode `_showSummary()` pousse une nouvelle route vers `SummaryPage` en lui passant la liste `_questions` et la liste `_userResults`. La page de récapitulatif calcule alors un score global et affiche, pour chaque question, une icône verte (bonne réponse) ou rouge (mauvaise réponse).

Ce scénario illustre bien le cycle typique en Flutter : interaction utilisateur → mise à jour de l'état dans `_QuizzPageState` → appel à `setState()` → reconstruction de l'interface.

4 Choix d'interface et de thème

Pour l'interface utilisateur, j'ai fait plusieurs choix visant à garder une identité visuelle cohérente entre les écrans.

Page de profil

Sur la page de profil, j'utilise un `Column` centré afin de regrouper visuellement l'avatar, la carte de profil et le bouton d'accès au quizz. La carte de profil est implémentée avec un `Container` décoré via `BoxDecoration` (fond blanc, coins arrondis, ombre), ce qui rappelle une carte Material. Les informations (téléphone, email, LinkedIn, GitHub) sont alignées avec des `Row` contenant une `Icon` suivie d'un `Text`, ce qui rend la lecture plus claire. Le bouton "Aller au quizz" reprend exactement le même style que la carte (même largeur, même rayon et même ombre) pour donner l'impression d'un ensemble homogène.

Page de quizz

Pour la page de quizz, je réutilise le même thème de couleurs : une `AppBar` bleu clair, un fond bleu très pâle et une carte blanche centrale contenant la question. Chaque question affiche un titre du type « Question i/n », une image de fruit ou de légume, puis l'énoncé centré. Les boutons `VRAI` et `FAUX` sont eux aussi intégrés dans des `Container` stylés comme des mini-cartes, de manière à rester cohérents avec le bouton de la page principale.

10:29

Questions/Réponses

Question 1/5

L'avocat est classé botaniquement comme un fruit.

VRAI FAUX

10:29

Questions/Réponses

Question 2/5

La rhubarbe est un fruit car on en mange les tiges sucrées.

VRAI FAUX

10:29

Questions/Réponses

Question 3/5

La banane est une baie au sens botanique

VRAI FAUX

10:29

Questions/Réponses

Question 4/5

Le poivron rouge contient en moyenne plus de vitamine C qu'une orange.

VRAI FAUX

10:30

Questions/Réponses

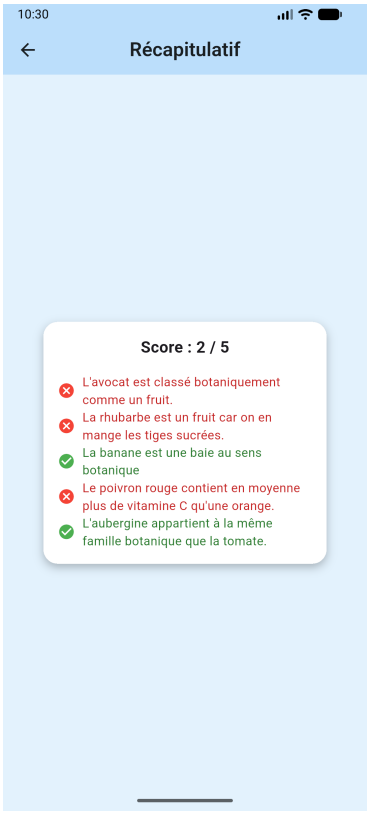
Question 5/5

L'aubergine appartient à la même famille botanique que la tomate.

VRAI FAUX

Page de récapitulatif

La page de récapitulatif reprend le même gabarit de carte centrale : un bloc blanc arrondi avec ombre sur le fond bleu clair. En haut de la carte, j'affiche le score global `Score : x / 5`, puis j'énumère les questions avec, pour chacune, une icône `check` verte ou `cancel` rouge selon que la réponse de l'utilisateur était correcte ou non.



Conclusion

Ce TP m'a permis de mettre en pratique les concepts de base de Flutter, aussi bien du point de vue de la construction d'interfaces que de la gestion de l'état. En partant d'une simple carte de profil, j'ai progressivement structuré l'application autour de trois écrans distincts (profil, quizz, récapitulatif) en utilisant de manière appropriée les widgets `StatelessWidget` et `StatefulWidget`.