# TP1 : Création d'interfaces graphiques en Flutter

## Objectives

Le but de ces exercices et de vous familiariser avec la création d'interfaces graphiques en Flutter en abordant les widgets stateless/stateful ainsi que la gestion de state via les composants **BLoC** (**Business Logic Component**).

## Exercice 1 : Profile Card

Le but de cette exercice est de créer une application représentant le profile d'une personne. Le profile sera ainsi un widget regroupant deux parties : (1) un avatar du profile (*photo*) ; (2) les informations concernant le profile (*e.g., nom, prénom, email, compte réseau social, ...*).

Voici un exemple à quoi doit ressembler votre application

Notre application est un stateless widget héritant de la classe `StatelessWidget` et aura un squelette ressemblant au suivant :

```
// importation du paquetage pour utiliser Material Design
import 'package:flutter/material.dart';

void main() => runApp(MyApp()); // point d'entrée

// Le widget racine de notre application
class MyApp extends StatelessWidget {
    const MyApp({Key? key}) : super(key: key);

    @override
    Widget build(BuildContext context) {
        return MaterialApp( // une application utilisant Material Design
            title: 'My First Flutter App',
            theme: ThemeData(...), // données relatives au thème choisi
            home: const ProfileHomePage(),  // le widget de la page d'accueil
            ...
        );
    }
}
```

Pour la page d'accueil, on pourrait utiliser les widgets suivants :

1. `Scaffold` avec son attribut `appBar` de type `AppBar` afin d'avoir la barre d'application.
2. `Container` qui combine des widgets courants de peinture, de positionnement et de dimensionnement.
3. `BoxDecoration` qui permet de décorer d'autres widgets.
4. `Stack` qui permet facilement de superposer plusieurs widgets enfants : *e.g., une image et un text superposés avec un background dégradé et un boutton*

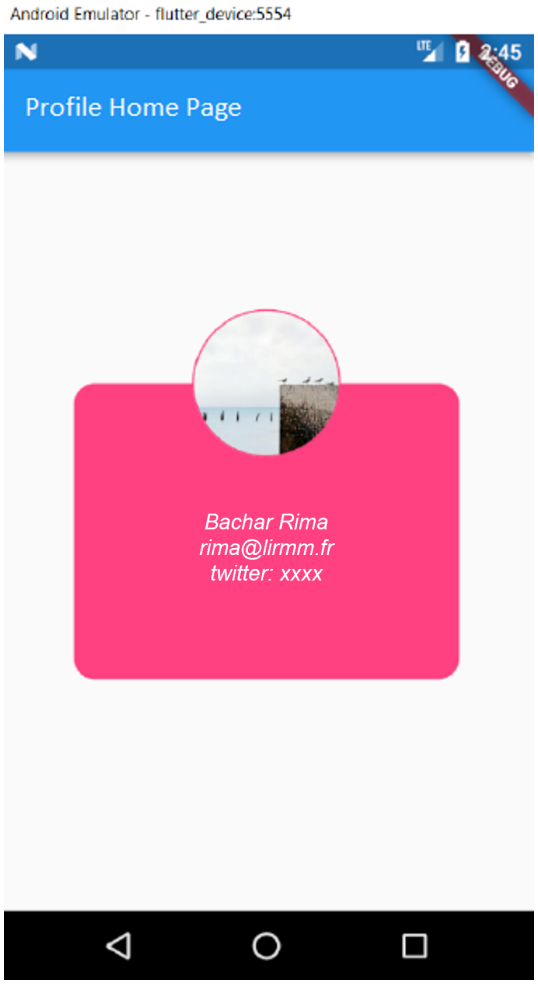


Figure 1: Application Profile Card

*attaché dessous.*

5. ...

Enfin votre page d'accueil devrait possèder un squelette ressemblant au suivant :

```
// Le widget de notre page d'accueil
class ProfileHomePage extends StatelessWidget {
    const ProfileHomePage({Key? key}) : super(key: key);

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text("Profile Card"),
                centerTitle: false,
                ...
            ),
            body: Container(
                alignment: Alignment.center,
                child: Stack(...),
                ...
            ),
        );
    }

    Container _getCard() {...}
    Container _getAvatar() {...}
}
```

## Exercice 2 : Quizz

Le but de cette exercice est de créer une application permettant de répondre à des questions d'un quizz sur une thématique donnée.

Voici un exemple à quoi doit ressembler votre application

Notre application est un stateless widget héritant de la classe `StatelessWidget` comme l'application précédente. Toutefois, la page du Quizz est un stateful widget `QuizzPage` héritant de la classe `StatefulWidget`. Pour celle-ci, on devrait utiliser ainsi la classe `State<QuizzPage>` pour gérer son état selon le squelette suivant :

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

    final String title;
```


Figure 2: Application Quizz

```
    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(...),
            backgroundColor: Colors.blueGrey,
            body: Container(...)
        );
    }

    // l'état du widget est créé à partir de la classe le réifiant
    @Override
    State<QuizzPage> createState() => SomeQuizzPageState();

    // à invoquer à chaque endroit ou l'on souhaite
    // modifier l'état afin de relancer la méthode build()
    setState(() {
        // changement de l'état du widget
    })

    bool _checkAnswer(bool userChoice, BuildContext context){...}
    Question _nextQuestion() {...}
}

class SomeQuizzPageState extends State<QuizzPage> {
    ...
}
```

Une question du quizz doit être modélisée comme suit :

```
class Question {
    String questionText;
    bool isCorrect;

    Question({required this.questionText, required this.isCorrect});
}
```

## Liens utiles

1. Catalogue des widgets Flutter : familiariser vous avec les différents widgets, leurs domaines d'usage, leurs attributs et méthodes, etc. afin de savoir les utiliser correctement.
2. Flutter Tutorial for Beginners : jusqu'à la vidéo 25 sera suffisant pour l'exercice 1.
3. BLoC - from Zero to Hero : jusqu'à la vidéo 4 sera suffisant pour l'exercice 2.