Université De Montpellier
Faculté Des Sciences

Niveau : Master 2

Module : Développement mobile avancé, IoT et embarqué

HAI912I

Programmation Mobile Avancée et IoT Flutter N°2

Supervisé par :
M. Abderrahmane SERIAI

Réalisé par :
Yanis Allouch

2021/2022

Table des matières

Introduction

Dans ce rapport de TP, j'exposerai le travail effectué pour deux applications, l'Application : Questions/Réponses et l'Application : Météo.

D'une part, le but premier de ces exercices est de nous familiariser avec Future, Async, FutureBuilder, ainsi que la gestion des états en utilisant Provider et BLoC (Business Logic Component). D'une autre part, le but seconds de ces exercices est de nous familiariser les BLoC/Cubits, FutureBuilder, les constructeurs nommés « fromJson », etc. Ainsi, je suis amené à organiser mon projet par package : views, data, business_logic, etc. pour le développement de deux applications, l'Application : Questions/Réponses et l'Application : Météo.

L'Application : Questions/Réponses est distingué en deux versions, d'une part, une version utilise Provider pour gérer les états, d'une autre part, une version utilise Cubit pour la gestion des états.

L'Application : Météo est distingué en une seule version. L'application implémente le design indiqué et la classe Network (proposé) encapsulant les appels réseaux REST vers l'API openweathermap.com/api.

1 Application : Questions/Réponses

Le dépôt de l'application dans ces deux adaptations peut être dupliqué depuis le dépôt GitLab publique a l'adresse suivante : https://gitlab.com/yanisallouch/my_questions_reponses_provider.

Avec l'application «Questions/Réponses», nous avons créé un Quiz sur une thématique qui nous intéresse. J'ai ainsi créé un widget de type «StatefulWidget» qui regroupe une image, une question ainsi qu'un ensemble de boutons.

La Figure 1 montre à quoi doit ressembler l'application que nous devons créer.

La Figure 2 et Figure 3 montre à quoi ressemble l'application que j'ai créé.

1.1 Exercice 1

La première étape entreprise pour implémenter la gestion des états par Provider, à été de simplifier le design et de découplée vue et logique métier de mon application. La logique a ensuite été encapsulé par Provider (voir Listing 1).

```dart
import 'package:flutter/material.dart';
import 'package:questionreponses/models/question_model.dart';

class HomepageProvider with ChangeNotifier {
  List<QuestionModel> _questions = [];
  int _index;

  HomepageProvider(this._index, this._questions);

  void nextQuestion() {
    index = (index + 1) % questions.length;
    notifyListeners();
  }

  int get index => _index;

  set index(int value) {
    _index = value;
    notifyListeners();
  }

  List<QuestionModel> get questions => _questions;

  set questions(List<QuestionModel> value) {
    _questions = value;
    notifyListeners();
  }
}
```

Listing 1 – Logique métier encapsulé par Provider

1.2 Exercice 2

Dans l'exercice, j'ai gardé exactement la même application et ai simplement changé l'encapsulation de Provider a Cubit (voir Listing 2), le reste de l'application n'a pas changé et fonctionne de la même façon.

```dart
import 'package:bloc/bloc.dart';
import 'package:meta/meta.dart';
import 'package:questionsreponses/data/data.dart';
import 'package:questionsreponses/models/question_model.dart';

part 'homepage_state.dart';

class HomepageCubit extends Cubit<HomepageState> {
  HomepageCubit(this._index, this._questions) : super(
   HomepageInitial(0, getQuestions()));

  List<QuestionModel> _questions = [];
  int _index = 0;

  void nextQuestion() {
    index = (index + 1) % questions.length;
    emit(HomepageInitial(index, questions));
  }

  int get index => _index;

  set index(int value) {
    _index = value;
    emit(HomepageInitial(index, questions));
  }

  List<QuestionModel> get questions => _questions;

  set questions(List<QuestionModel> value) {
    _questions = value;
    emit(HomepageInitial(index, questions));
  }
}
```

Listing 2 – Logique métier encapsulé par Cubit

1.3 Capture d'écran

FIGURE 1 – Capture d'écran du sujet

FIGURE 2 – Capture d'écran N°1 de mon application

FIGURE 3 – Capture d'écran N°2 de mon application

2 Application : Météo

Le dépôt de l'application peut être dupliqué depuis mon dépôt GitLab publique a l'adresse suivante : https://gitlab.com/yanisallouch/my_first_weather_app. Avec l'application «Météo», nous sommes amenés à créer une application Flutter qui permet d'avoir la météo pour une ville donnée. J'ai ainsi créé un widget de type «StatefulWidget» qui regroupe trois parties : la météo à l'instant T, la météo pour par heure pour les 48 heures et la météo sur 7 jours par tranche de 24h.

La Figure 4 suivante montre à quoi doit ressembler l'application que je dois créer.

La Figure 5 et Figure 6 montre à quoi ressemble l'application que j'ai créé.

Pour ce qui est de la création des models par le site javiercbk.github.io/json_to_dart/, permettant la sérialisation et la dé-sérialisation des JSON échangés avec l'API OWM, c'est avéré être infructueux pour deux raisons impliquant énormément de refactoring manuelle :

1. Les models générés étaient très complexes, profond et incorrect pour un JSON mal strucutré (mapping complet de l'objet JSON),
2. Le model écrit en Dart n'est plus conforme à la spécification du langage Dart 2.0.

Pour les deux raisons énumérées ci-dessus, j'ai préféré implémenter mon propre modèle Weather (voir Listing 3), permettant de dé-sérialiser les échanges avec l'API OWM.

```
class Weather {
  final double temp;
  final double feelsLike;
  final double low;
  final double high;
  final String description;
  final double pressure;
  final double humidity;
  final double wind;
  final String icon;

  Weather(
      {required this.temp,
      required this.feelsLike,
      required this.low,
      required this.high,
      required this.description,
      required this.pressure,
      required this.humidity,
      required this.wind,
      required this.icon});

```

```
23   factory Weather.fromJson(Map<String, dynamic> json) {
24     print(json);
25     return Weather(
26       temp: json['main']['temp'].toDouble(),
27       feelsLike: json['main']['feels_like'].toDouble(),
28       low: json['main']['temp_min'].toDouble(),
29       high: json['main']['temp_max'].toDouble(),
30       description: json['weather'][0]['description'],
31       pressure: json['main']['pressure'].toDouble(),
32       humidity: json['main']['humidity'].toDouble(),
33       wind: json['wind']['speed'].toDouble(),
34       icon: json['weather'][0]['icon'],
35     );
36   }
37 }
```

Listing 3 – Model Weather

2.1 Capture d'écran

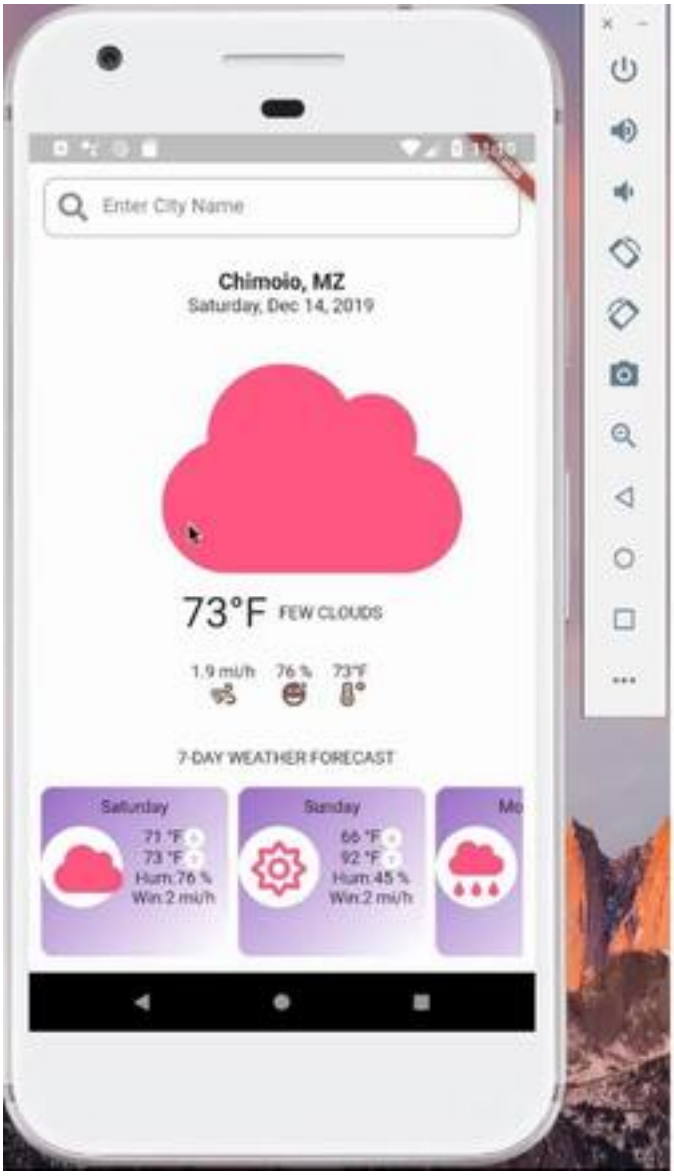
Enter City Name
Chimoio, MZ
Saturday, Dec 14, 2019
73°F FEW CLOUDS
1.9 mi/h
76 %
7-DAY WEATHER FORECAST
Saturday
71 °F
73 °F
Hum:76 %
Win:2 mi/h
Sunday
66 °F
92 °F
Hum:45 %
Win:2 mi/h

FIGURE 4


Montpellier, France
18°
Ciel Dégagé
5.14 km/h
64%
1019.0 hPa
18.52°
18.38°
18.05°
17.68°
Sat
Sun

FIGURE 5

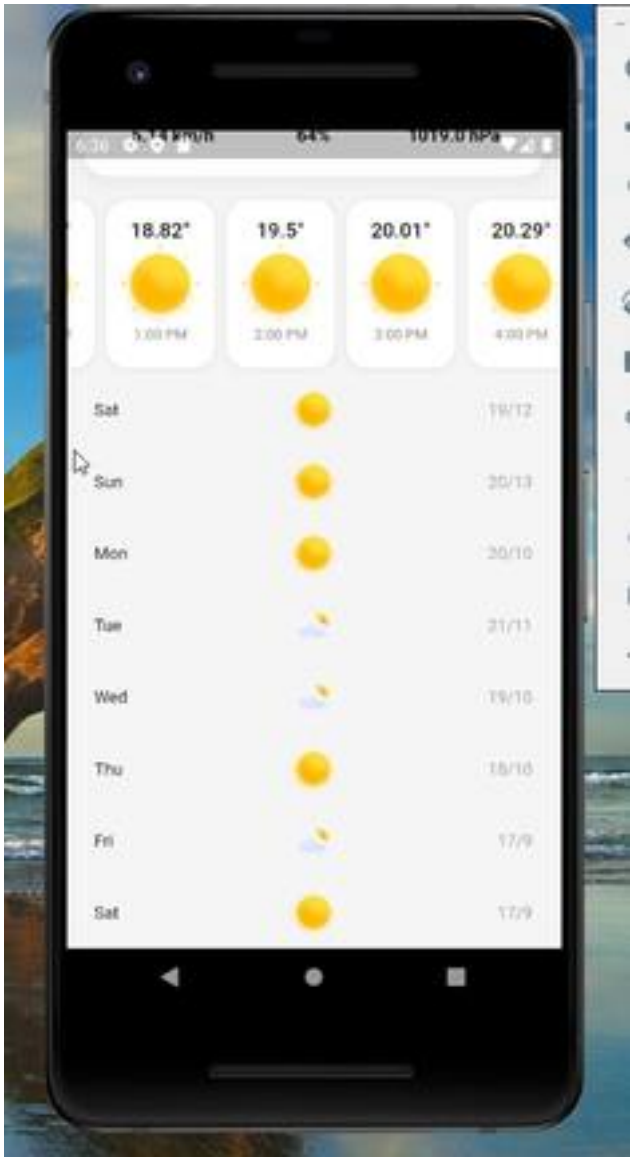

FIGURE 6