TP2 : Gestion d'état Avancée (Provider, BLoC), Futures & Streams, Themes & Styling, Assets, and Animations

Le but de ces exercices est de vous familiariser avec des notions plus avancées de Flutter, telles que la gestion avancée de l'état (avec Provider et BLoC), les Futures & Streams, les thèmes, assets, et animations. Ainsi, vous serez amenés à organiser votre projet par package de responsabilité comme ça a été expliqué dans le cours : package presentation (avec des sous package : animations, pages, screens, widgets), package data (avec des sous package : providers, models, repositories), et package business_logic (avec des sous package : blocs, events)

Exercice 1 : Quizz (Evolution)

Avec l'application « Questions/Réponses », vous avez créé un Quizz sur une thématique qui vous intéresse. Vous avez ainsi utilisé un widget de type `StatefulWidget` qui regroupe une image, une question ainsi qu'un ensemble de boutons.

La figure suivante montre à quoi devait ressembler l'application que vous aviez à créer :

Question 1 : Utilisation de Provider

Le premier exercice consiste à utiliser les **Providers** pour la gestion des états plus tôt que l'utilisation initiale de la méthode `setState()`.

Question 2 : Utilisation de BLoC

Le premier exercice consiste à utiliser les **BLoC** pour la gestion des états plus tôt que l'utilisation initiale de la méthode `setState()`.

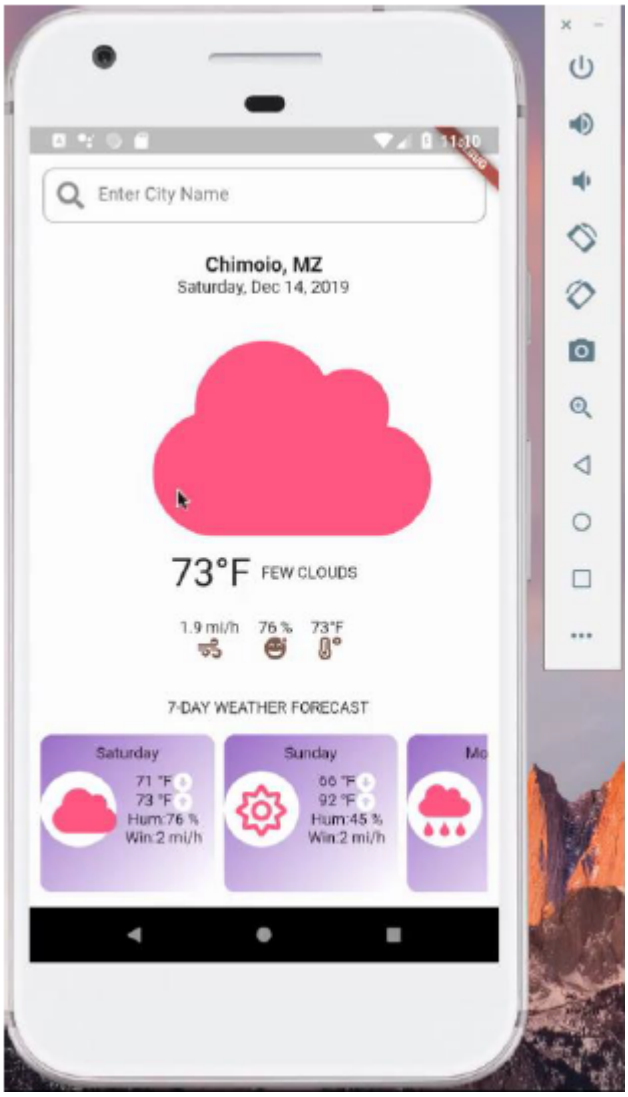
Exercice 2 : Météo

Dans cet exercice, vous allez créer une application Flutter qui permet d'avoir la météo pour une ville donnée. Vous allez ainsi devoir créer un widget de type `StatefulWidget` qui regroupe trois parties :

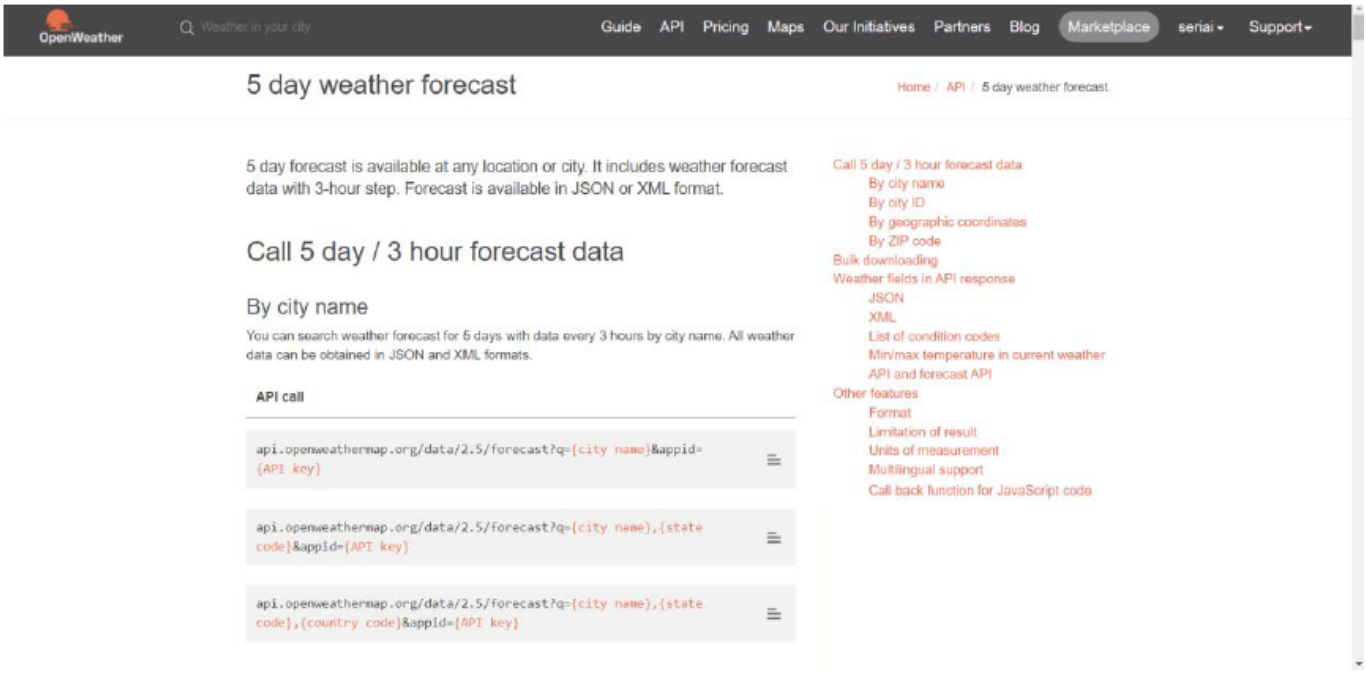
1. Une partie avec un `TextField`,
2. Une partie avec le nom de la ville, la date, une icône, ainsi que les informations concernant la température, la précipitation, et l'humidité.

3. Et une partie avec les prévisions pour au moins 3 ou 4 jours.

La figure suivante montre à quoi doit ressembler votre application créée :



Avant de commencer le développement de votre application, il faut créer un compte sur openweathermap.org, afin de créer une clé API pour que vous puissiez utiliser leurs endpoints REST.



Voici quelques indications pour vous aider à développer votre application :

1. Commencez par créer une classe `WeatherModel` qui représente le modèle des donnés récupérés via l'API. Vous pouvez utiliser la fonctionnalité JSON to Dart afin de transformer les données brutes en classe Dart.
2. Créez une classe qui encapsule les appels réseau et qui fetchent les données depuis l'API et les retournent en JSON
3. Formater la date en utilisant la classe `DateFormat` du package `intl`.
4. Créez vos interfaces graphiques et n'oubliez pas d'ajouter des thèmes et animations et d'utiliser des icônes adaptés aux différentes description de la météo.
5. Vous auriez besoin au moins des paquetages :
    - `http` (^0.13.4)
    - `intl` (^0.17.0)
    - `font_awesome_flutter` (^9.1.0)