

Weather

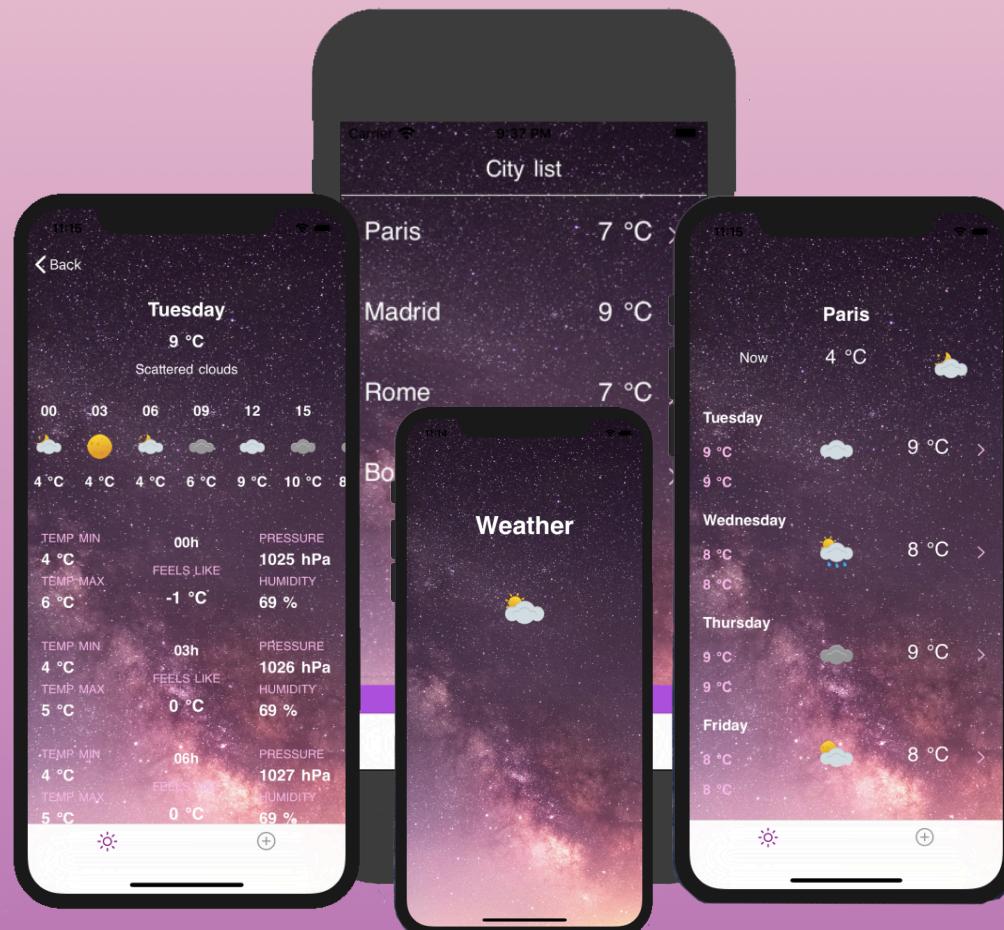


IOS development in swift / Xcode



Weather

A weather application for the city of Paris using
OpenWeather API



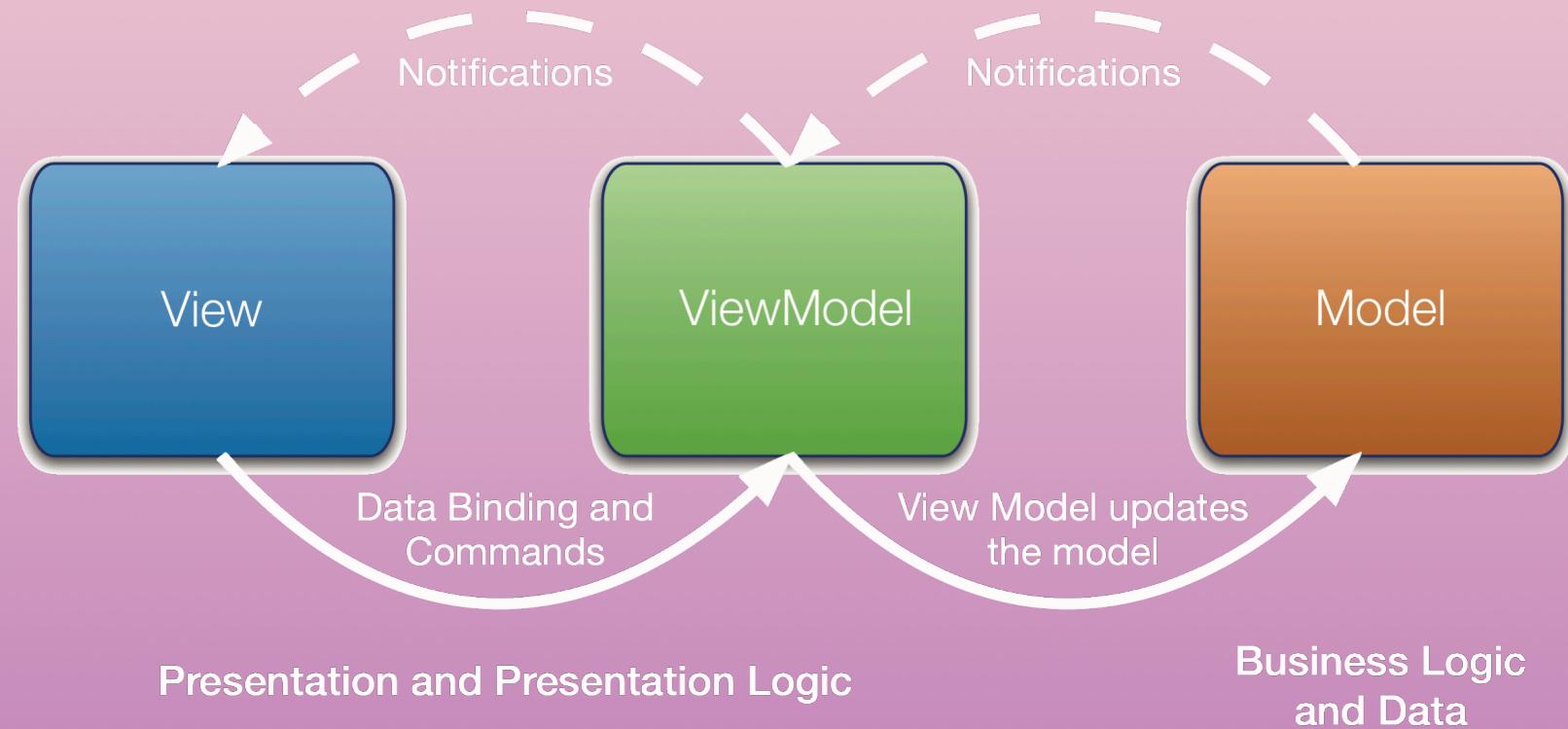


Application simulation





What is MVVM-C pattern ?





The approach used :

- 1 - Creation of two modules : “CityList module” including a detail view “DetailDay” and “SelectCity module” using the Coordinator and the storyboard of each screen view.
- 2 - Implementation of a « UITabBar » with 2 items:
 - . weather
 - . selectCity
- 3 - Creation of the network with an “HTTPClient”, an “HTTPEngine” and a “Token”
- 4 - Creation of the model structure into WeatherItem.swift and adding of WeatherObject attributes into Core data.
- 5 - Writing of WeatherRepository.swift
- 6 - Injection of Core data in the context and the context in the appDelegate.
- 7 - Unit tests
- 8 - UI tests



How does the logic communicate with the view ?

Each screen view is constructed using a ViewController and a ViewModel.

- ViewControllers only exist to control the view, they are clear, simple, and don't know data. ViewControllers subscribe to the reactive variables of ViewModels by the « bind() » function, called in the « viewDidLoad() ».
- ViewModels encapsulate the whole logic of each screen view that doesn't have to be in the ViewController, divided in two parts :
 - **Inputs:** The view subscribes to reactive variables then sends the first input which often is the « viewDidLoad() » : the main input connection for reactive variables.
 - **Outputs:** They are receptacles of events from the corresponding inputs.



What does Reactive Native means ?

Responsive:

The view subscribes to closures, a closure is a variable whose type is a function.

So we create reactive variables in the « ViewModel » and on the « ViewController » side, we subscribe to these closures so to a state.

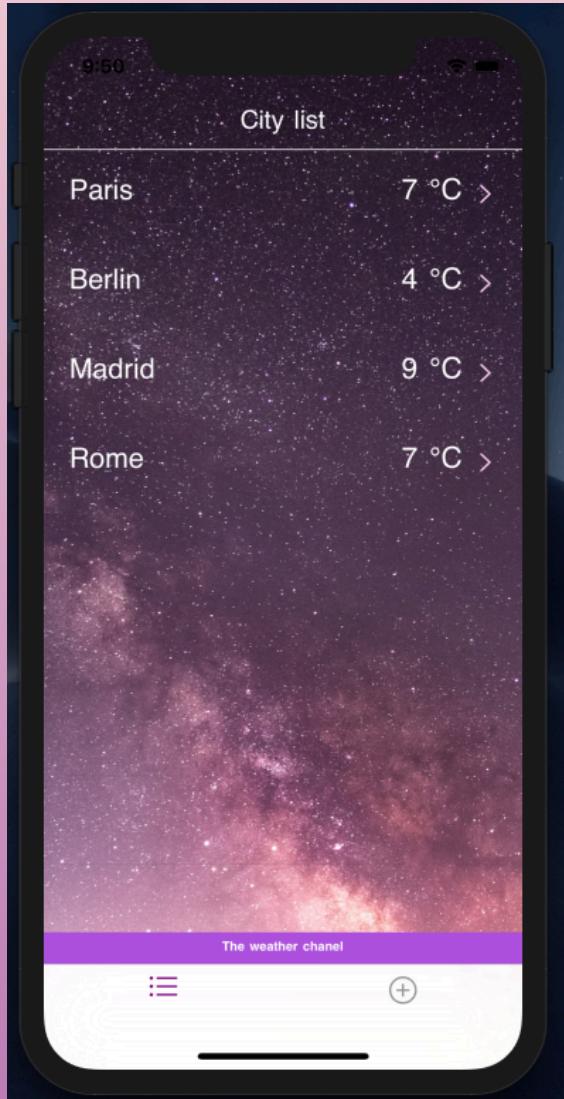
Native:

It means that the application is coding without an external library.
Different from MVVM RX for example.



Inputs / Outputs of each view

CityListView



```
// MARK: - Output

var visibleWeatherItems: (([WeatherItem]) -> Void)?

var isLoading: ((Bool) -> Void)?

var labelState: ((Bool) -> Void)?

var labelText: ((String) -> Void)?

var urlText: ((String) -> Void)?

var navBarTitle: ((String) -> Void)?

// MARK: - Input

func viewDidLoad() {

}

func didSelectCity(at index: Int) {

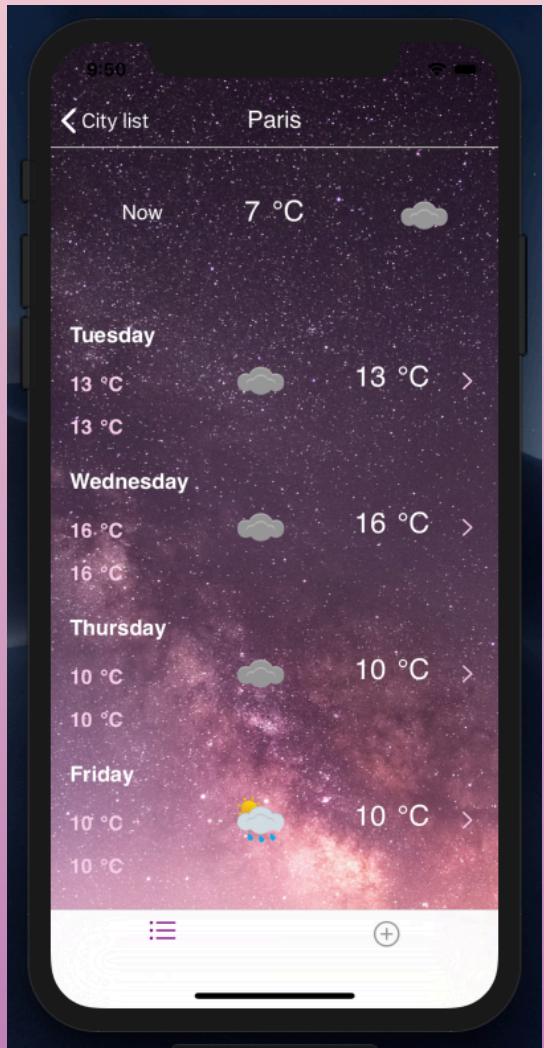
}

func didPressDeleteCity(at index: Int) {
```



Inputs / Outputs of each view

WeekView



```
// MARK: - Outputs

var navBarTitle: ((String) -> Void)?

var nowText: ((String) -> Void)?

var visibleItems: (([WeatherItem]) -> Void)?

var tempText: ((String) -> Void)?

var iconText: ((String) -> Void)?

var isLoading: ((Bool) -> Void)?

// MARK: - Inputs

func viewDidLoad() {

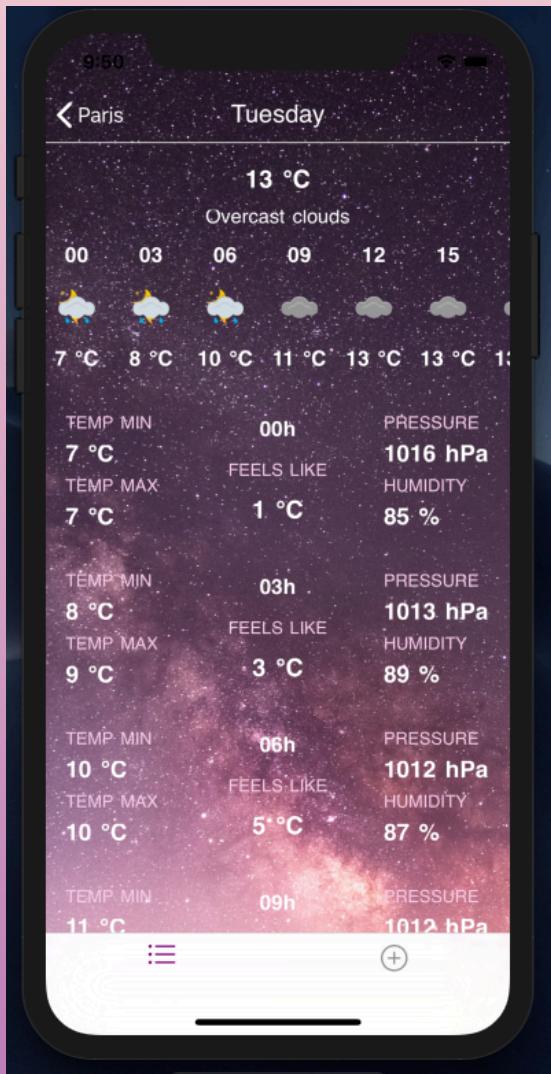
}

func didSelectWeatherDay(at index: Int) {
```



Inputs / Outputs of each view

DetailDayView



```
// MARK: - Outputs

var navBarTitle: ((String) -> Void)?

var visibleItems: (([WeatherItem]) -> Void)?

var tempText: ((String) -> Void)?

var descriptionText: ((String) -> Void)?

// MARK: - Inputs

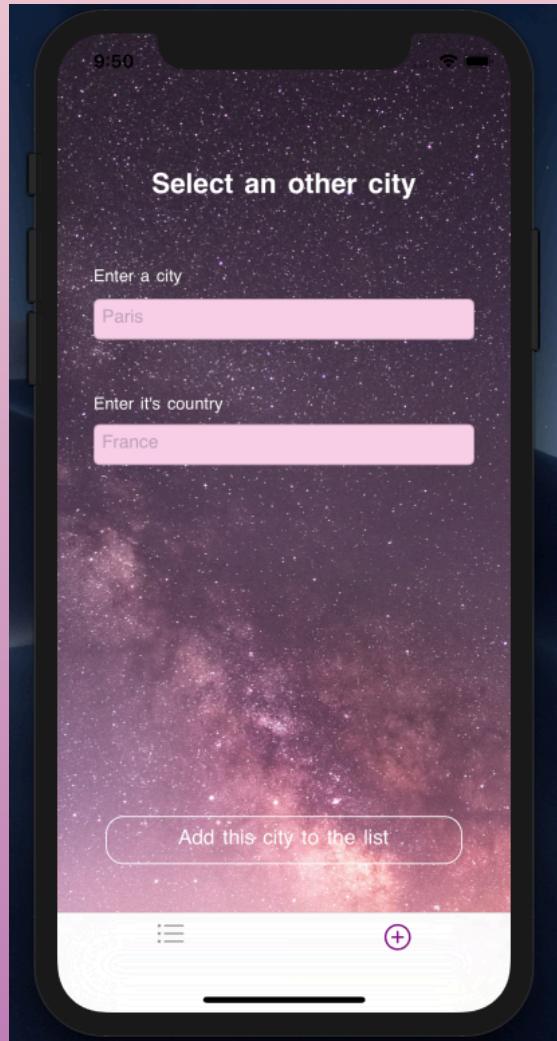
func viewDidLoad() {

}
```



Inputs / Outputs of each view

SelectCityView



```
// MARK: - Outputs

var titleText: ((String) -> Void)?

var cityText: ((String) -> Void)?

var cityPlaceHolder: ((String) -> Void)?

var countryText: ((String) -> Void)?

var countryPlaceHolder: ((String) -> Void)?

var addButtonText: ((String) -> Void)?

// MARK: - Inputs

func viewDidLoad() {

}

func addbuttonNormalState() {

}

func didPressAddCity(nameCity: String, country: String) {
```



The OpenWeather API

The url used of the OpenWeather API is receiving data of 5 days weather every 3 hours.

The model structure « WeatherItem » has been created to show the time, temperature, icon image and plus, the temperature min/max, pressure, humidity, feelsLike and sky description.



What is Core Data ?

Core Data is a local database, provided with an object oriented API.

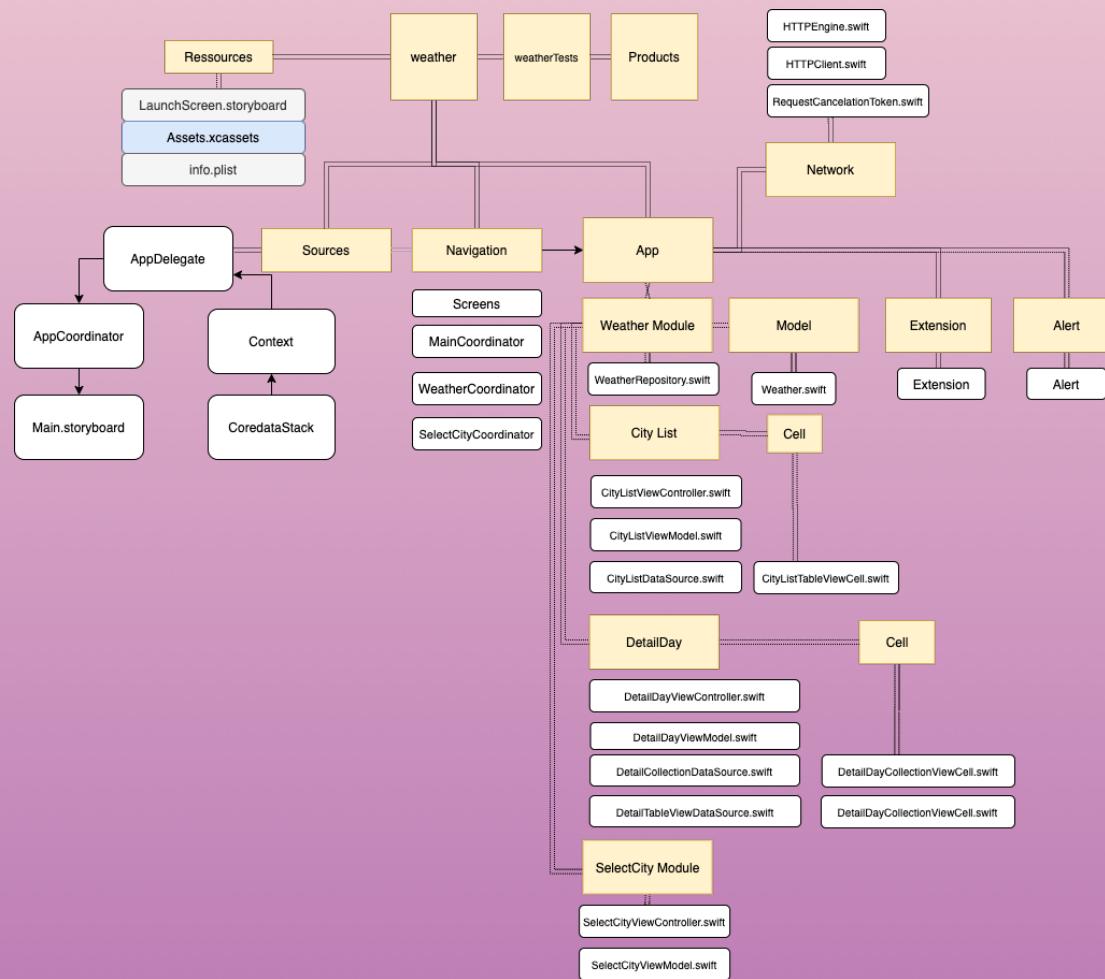
In this application, we just needed to create entity's name object in the « weather.xcdatamodeld » file and add its attributes.

A single entity comprising 10 attributes with « String » type has been created to save current weather data inside the phone.

WeatherObject includes: nameCity, descriptionWeather, feelsLikeWeather, humidityWeather, iconWeather, pressureWeather, tempMaxWeather, tempMinWeather, tempWeather, timeWeather.



The architecture diagramme and layer's description





Thanks for your attention