FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Mobile Computing
5th year of the Integrated Masters in Informatics and Computing Engineering (MIEIC)

Xamarin app for weather consulting
Portugal Weather Seer

Group elements:

Diogo Luis Rey Torres - 201506428 - up201506428@fe.up.pt
Filipe Oliveira e Sousa Ferreira de Lemos - 201200689 - ee12140@fe.up.pt

December 18, 2019

Index

1. Introduction

This project consists of a weather application and was developed using the MVVM architecture.

The application needs to fetch the weather data for a specific city and display it in a user friendly way. When a user starts the application, he can add cities to his favorite list or view a quick summary of the that days weather for the cities on that list. By clicking on each city, he will be presented with a more detailed description of the current weather. Moreover, it is also possible to check tomorrow's forecast, which displays the weather amplitudes, a detailed description and an hourly temperature graph. With this in mind, the system implements all the necessary requirements for its operation presenting them in an intuitive and pleasant way for the final user.

In the next sections, the application will be described in finer detail, namely its architecture, implementation details and relevant issues. This report concludes with a small reflection about the work done and possible future enhancements.

2. Architecture

2.1. Description

This application is based on a Client-Server architecture that allows clients and servers to communicate over a computer network, where the server shares their resources with clients [1]. The sharing information between the server and clients is made through a REST protocol, where clients make requests they are interested in, and the server responds accordingly.

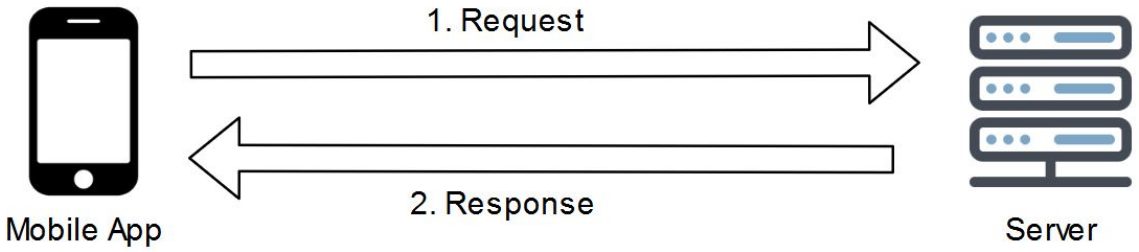

Figure 1: System Architecture

The used architecture consists of a mobile application (Xamarin App) and an external server (OpenWeatherMap). The using of Xamarin allows us to build modern & performant iOS and Android apps with C# and .NET [2]. Also, using the OpenWeatherMap as API we can easily provide weather data, including current weather data and forecasts to our users [3].

2.2. Weather Application

This application is based on *Xamarin Framework* that presents to the user a way to check the current weather or the following day forecast of his favorite cities. It uses a weather API to retrieve and synchronize data about those cities weather. Furthermore, we reutilize some code provided by a Sample in Xamarin Docs [4].

2.2.1. Code Structure

The weather application is divided into 3 projects: WeatherApp, Android and iOS. The first one contains the dependencies and the C# code. The remainings projects contain the resources (assets) and some specific code to run in different devices.

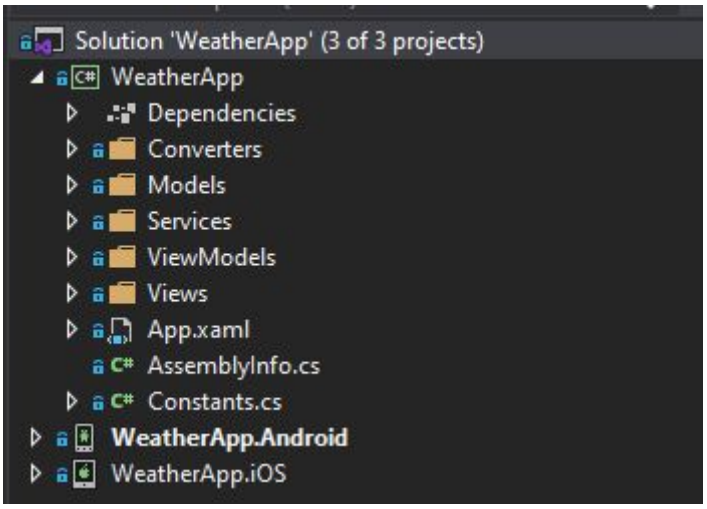

Figure 2: Weather App Overview

The C# code is divided into 5 packages: *converters*, *models*, *services*, *viewModels,* and *views*. Also, the code was developed by following the Model-View-ViewModel pattern. This pattern helps to separate the business and presentation logic of the application from its user interface (UI). Maintaining a clean separation between application logic and the UI helps to address numerous development issues and can make an application easier to test, maintain, and evolve [6]. There are three core components in the MVVM pattern: the model, the view, and the view model. Each serves a distinct purpose that will be detailed in the next sections.

2.2.1.1. Converters Package

The **converters** package was created to host some functions to ease the value conversion between the data present in *ViewModels* and binded into the *Views*. To handle these conversions, the classes that have specialized code need to implement the *IValueConverter* interface [7].

2.2.1.2. Models Package

The **models** package is responsible for creating, managing and saving the entities used during the app. The entities are CityInfo, GraphEntry, and WeatherForecast.

2.2.1.3. Services Package

The **services** package was created to easily host new services to the app. The current service is the **Weather API** service.

The **Weather API** service also consists of 2 classes: *Constants and WeatherAPI*.

The *WeatherAPI* contains methods to retrieve the weather and other information related to icons. The weather data is directly deserialized using *JsonConvert* into the correspondent model.

The *Constants* class contains some configurations used like the server endpoints and the server key.

2.2.1.4. ViewModels Package

The **ViewModels** package was created to host an intermediary between the *View* and the *Model*, called the *ViewModel.*

This package contains a *ViewModelBase* that sets up the events through the *INotifyPropertyChanged* interface to notify the view when a property in a model changed. Furthermore, it also contains the *FavoriteViewModel* that extends the *ViewModelBased*. Moreover, it has all the information needed by the views.

In order to keep the *ViewModel* without buttons' handlers and to allow the *ViewModel* to be more independent of a particular user interface, a *command* interface is used [8].

2.2.1.5. Views Package

The **views** package contains the different views used in the application. Each view contains a XAML file to define the visual contents of a page and the respective C# file to define some additional information [9].

The navigation between views is made with the help of the *NavigationPage* class. The *NavigationPage* class provides a hierarchical navigation where the user is able to navigate through the pages, forwards and backwards [10].

2.2.2. Local Storage

Some data is persisted locally in the application internal storage [11]. Every time a city is added to the favorite city list, it is saved to a file "data.txt", which is accessed every time the app restarts. This avoids losing the favorite list every time the application is closed.

2.2.3. External Libraries

Alongside Xamarin, some libraries were used, most noticeably:

- **SkiaSharp** is an open source 2D graphics system for .NET and C# [12] and was used to draw the temperature graph.
- **Newtonsoft.Json** is an high-performance JSON framework for .NET [13] and was used to parse the data retrieved by the OpenWeatherMap

2.3. OpenWeatherMap

OpenWeatherMap is an online service that provides weather data, including current weather data, forecasts, and historical data to the developers of web services and mobile applications [18].

2.3.1. Endpoints Used

The table below shows all the endpoints that the server provides. For the first two routes, we have to provide some extra parameters to fetch the data:

- "?q={cityName},PT" - to retrieve the weather from portuguese cities
- “&units=metric” - to retrieve temperature in celsius
- “&APPID={Constants.OpenWeatherMapApiKey}” - the API key

Type	URL	Description
GET	https://api.openweathermap.org/data/2.5/forecast	5 day forecast includes weather data every 3 hours [14]
GET	https://api.openweathermap.org/data/2.5/weather	Access current weather data [15]
GET	http://openweathermap.org/img/wn/	Access weather icons

3. Functionalities included

3.1. Weather App

- View the user’s favorite cities list;
- View the today’s detailed city weather;
- View tomorrow's forecast;
- View Activity Indicator[16] when fetching data from the API or loading from local storage;
- Add cities to the favorite list;
- Remove cities from the favorite list;
- Update favorite cities weather and forecast;
- Load/Store favorite cities list.

All requested features have been implemented as well as some extra features. Also, all the features were tested manually in Android and Windows. Note that this application was made for the portuguese capitals of districts.

4.Screen Captures illustrating the main sequences of use

The App flow starts by opening the *Homepage* (Figure 3). Here the user can add cities to his favorite list. An example of a favorite list with some entries can be seen in Figure 4. This list can be updated by pressing the SYNC button on the application navigation bar.

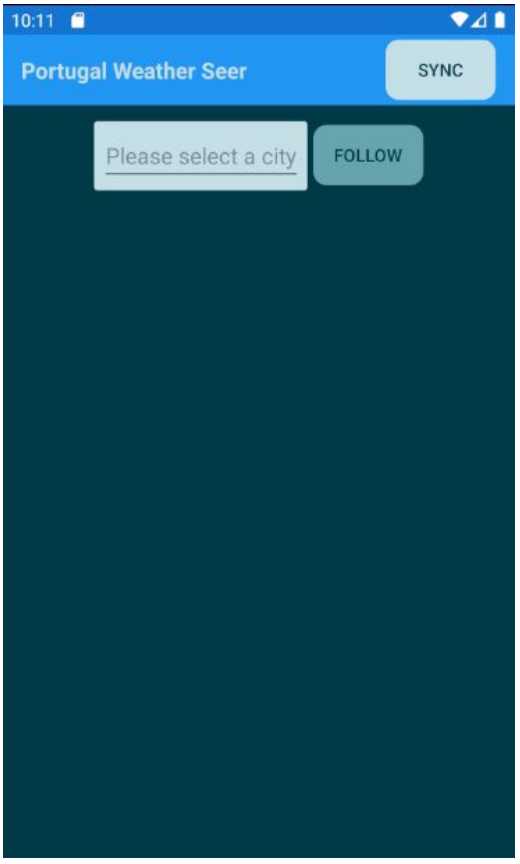

Fig. 3 - Homepage

Fig. 4 - Homepage with filled favorite list

Each list row displays the city name, its current temperature and an icon describing the current weather. Pressing the cross removes the element from the list, while pressing the element row takes the user to today's detailed weather page (Figure 5).

Fig 5 - Today's detailed weather

From this page, the user can ask for the detailed forecast of the following day by pressing the button "Tomorrow". This page provides a scrollable view with the hourly forecast, the weather amplitudes and a temperature graph - (Figure 6,7,8).

Fig. 6 - Tomorrow Forecast 1

Fig. 7 - Tomorrow Forecast 2

Fig. 8 - Tomorrow Forecast 3

The same application was tested on windows UWP and can be seen in Figure 9.

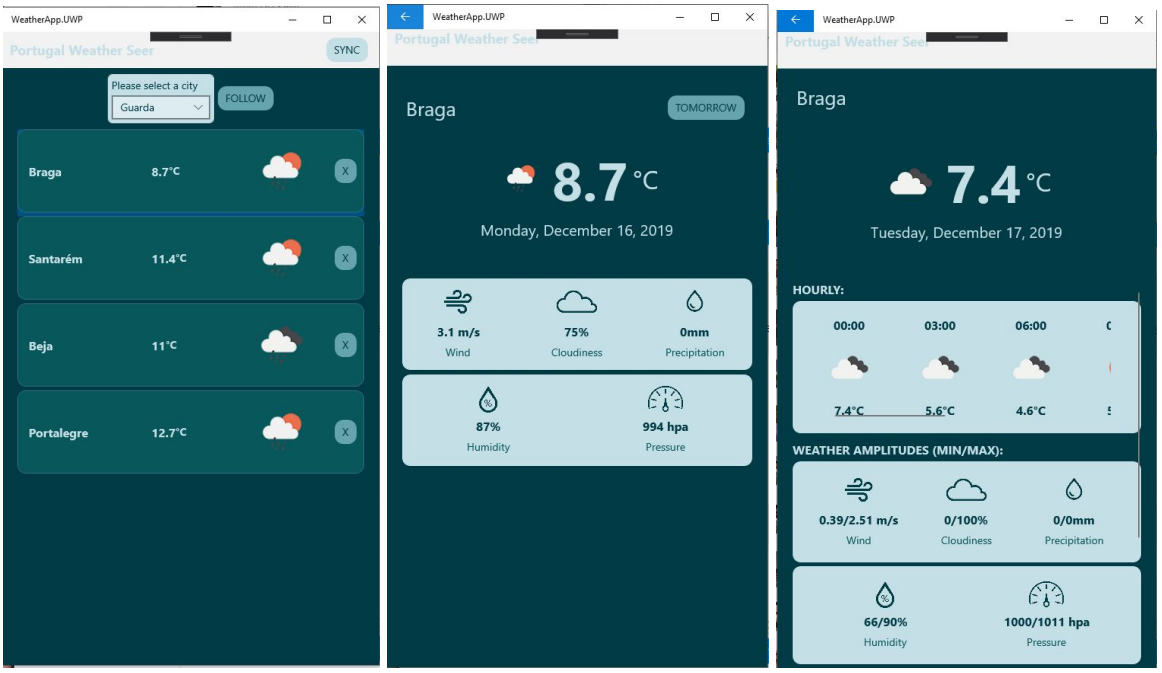

Fig.9 - Application running on UWP

5. Conclusion

With this work, the group realized the importance of a well structured architecture in the development process in order to achieve a good final result. We were able to learn the basic operation of applications developed with *Xamarin Framework* as well as the main problems associated with them (compatibility between different environments, dependencies, communication between views and navigation across the application).

The final project provides an application that presents to the user a way to check the current weather or the following day forecast of his favorite cities.

Despite this, as always, there can be enhancements done to further elevate its quality and performance. In this case, we can develop automated tests to verify that all functionalities are operational, as well improving the design to allow screen rotation.

6. Bibliography

[1] “Client–Server Model.” *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Client%E2%80%93server_model&oldid=929177360.

[2] *What Is Xamarin. Forms? - Xamarin*. https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin-forms.

[3] *Current Weather and Forecast - OpenWeatherMap*. https://openweathermap.org/

[4] *Xamarin.Forms - Weather App - Code Samples*. https://docs.microsoft.com/en-us/samples/xamarin/xamarin-forms-samples/weather/

[5] *Devcrux/WeatherApp*. 2019. *GitHub*, https://github.com/devcrux/WeatherApp

[6] *The Model-View-ViewModel Pattern - Xamarin*. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

[7] *Xamarin.Forms Binding Value Converters - Xamarin*. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/data-binding/converters

[8] *The Xamarin.Forms Command Interface - Xamarin*. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/data-binding/commanding

[9] *Getting Started with XAML - Xamarin*. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/xaml/xaml-basics/get-started-with-xaml

[10] *Hierarchical Navigation - Xamarin*. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/navigation/hierarchical

[11] *File Handling in Xamarin.Forms - Xamarin*. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/data-cloud/data/files

[12] *SkiaSharp Graphics in Xamarin.Forms - Xamarin*. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/graphics/skiasharp/

[13] *Json.NET - Newtonsoft*. https://www.newtonsoft.com/json

[14] *5 Day Weather Forecast - OpenWeatherMap*. https://openweathermap.org/forecast5

[15] *Current Weather Data - OpenWeatherMap*. https://openweathermap.org/current

[16] *Activity Indicator in Xamarin.Forms - Xamarin*. https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/activityindicator

[17] *Mobile Computing 2019/20*. https://paginas.fe.up.pt/~apm/CM/

[18] “OpenWeatherMap.” *Wikipedia*, https://en.wikipedia.org/w/index.php?title=OpenWeatherMap&oldid=915459277

7. Resources

7.1. Software Used

- *Rider provided by Jetbrains*
- *Visual Studio 2019* provided by *Microsoft*

7.2. Setup Project

Run the projects in the visual studio 2019.