City Weather Project

System requirements

and

design document

Name: Mehmet

Surname : IZBUL

Table of Contents

1. Introduction

1.1. Purpose

The purpose of this document is to build an application that gets the city weather forecast and design a web API to add weather information for each city available in TUI Musement's catalogue.

1.2. Intended Audience

This project is for the TUI Musement hiring and engineering team to be evaluated as a job position candidate selection step.

1.3. Project Scope

The project scope consists of 2 sub-projects.

- The first sub-project is to implement a CLI application that displays the city weather for next 2 days.
- The second sub-project is to design a web API to store/retrieve weather forecast for a specific city to/from TUI Musement’s database.

2. CLI Application Implementation

2.1. Description

The CLI application:

- gets the list of the cities from TUI Musement's API
- displays the forecast for next 2 days

Example Output:

```
Processed city Milan | Heavy rain - Partly cloudy
Processed city Rome | Sunny – Sunny
```

There is no database needed for implementing the CLI.

2.2. Design and Implementation Principles

- The methods are implemented as *static* methods to be able to used in "Main Method" without assigning them into an object.
- The global variables are defined as *static* to be able to use them inside *static* functions.
- Third party libary "Newtonsoft" is included to project to parse asynchronus call JSON response content into .NET objects.
- API calls are implemented with *await* operator.
- Try-Catch architecture is used to provide ruboustness.

2.3. Application Class Entities

The implemented class entities and relevant property types to be mapped to the API response content is as follows:

- City
    - Name (string)
    - Latitude (string)
    - Longitude (string)
- Weather
    - Location (Location)
    - Forecast (Forecast)
- Location
    - Name (string)
- Forecast
    - ForecastDay[] (array)
- ForecastDay
    - Day (Day)
- Day
    - Condition (Condition)
- Condition
    - Text (string)

2.4. REST API Calls

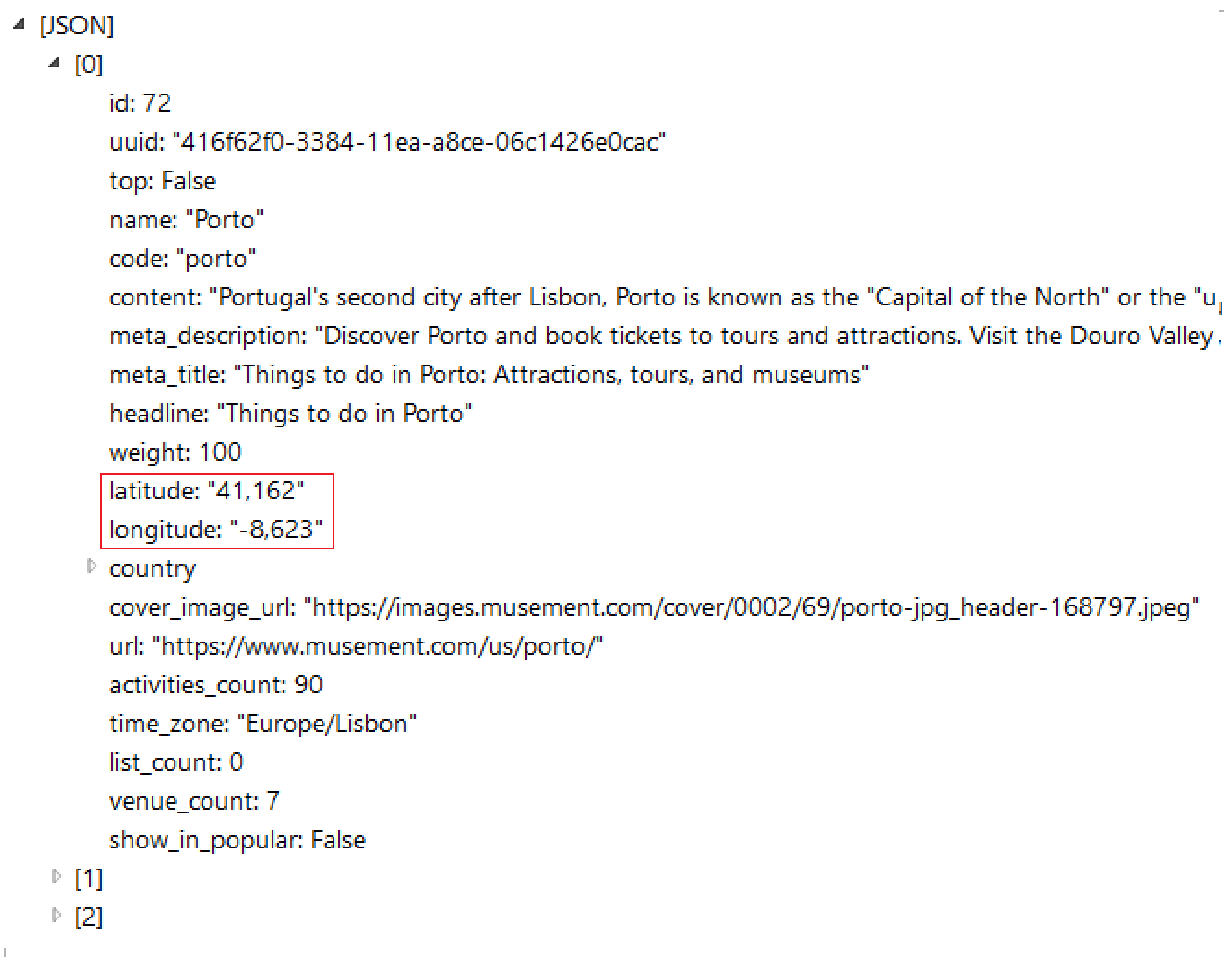
There are 2 different REST API GET requests to 2 different locations to get the desired output. The “Latitude” and “Longitude” values should be retrieved first, then to be used for “weatherapi” API call parameters.

- “Latitude” and “Longitude” is get from the TUI Musement's API with “GET /api/v3/cities” endpoint
- The weather condition information is being retrieved from http://api.weatherapi.com/v1/forecast.json?key=[your-key]&q=[lat],[long]&days=[weatherDays]

Separate methods are implemented for separate calls mentioned above.

2.4.1. *GetCityAsync()* method

This method gets the cities from the TUI musement "GET /api/v3/cities" endpoint with an *asynchronous GET* call, *deserialize* the JSON response into implemented class list (List<City>) and returns it. Sample API response content is as below:

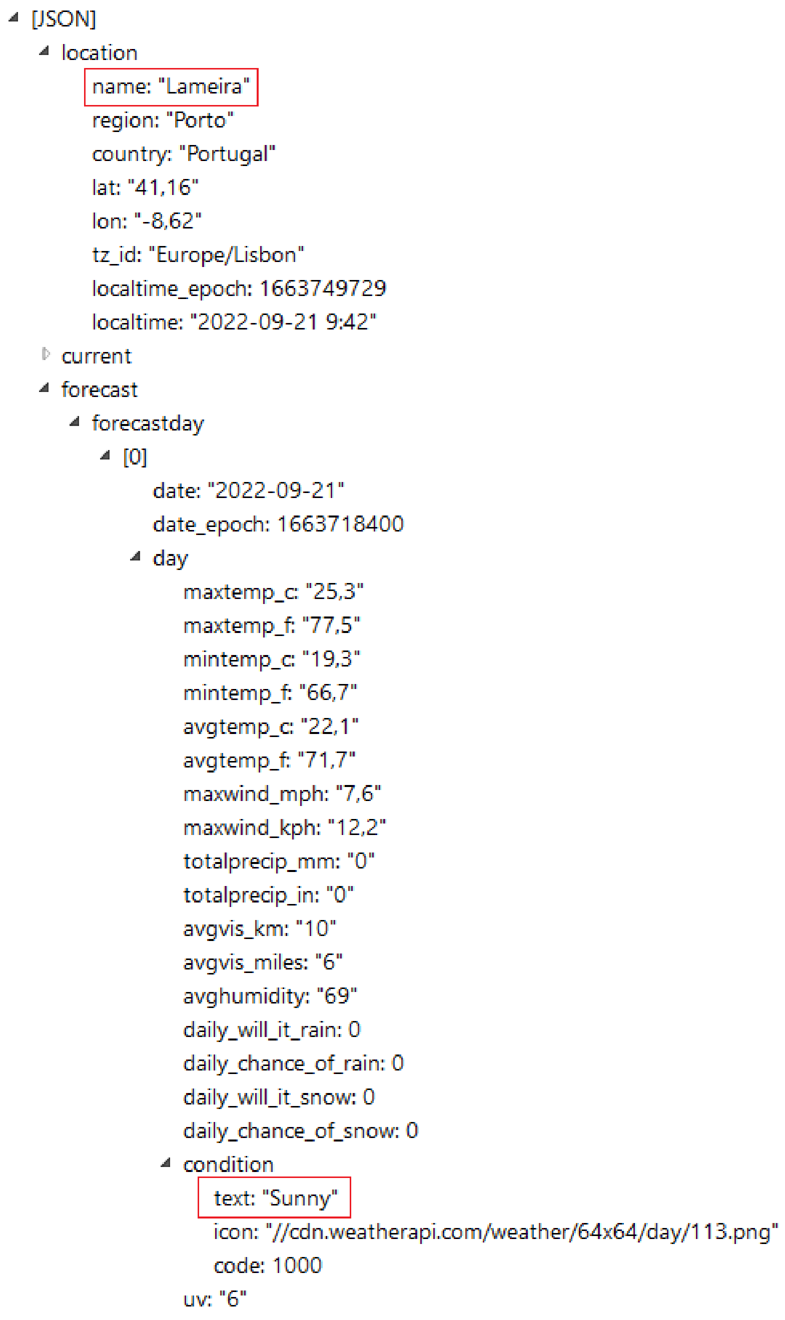


```
[JSON]
  [0]
    id: 72
    uuid: "416f62f0-3384-11ea-a8ce-06c1426e0cac"
    top: False
    name: "Porto"
    code: "porto"
    content: "Portugal's second city after Lisbon, Porto is known as the "Capital of the North" or the "u
    meta_description: "Discover Porto and book tickets to tours and attractions. Visit the Douro Valley
    meta_title: "Things to do in Porto: Attractions, tours, and museums"
    headline: "Things to do in Porto"
    weight: 100
    latitude: "41,162"
    longitude: "-8,623"
    country
    cover_image_url: "https://images.musement.com/cover/0002/69/porto-jpg_header-168797.jpeg"
    url: "https://www.musement.com/us/porto/"
    activities_count: 90
    time_zone: "Europe/Lisbon"
    list_count: 0
    venue_count: 7
    show_in_popular: False
  [1]
  [2]
```

Figure 1 – GET /api/v3/cities sample response content

2.4.2. *GetWeatherAsync(string url)* method

This method gets the weather forecast for a single city from "weatherapi's" provided url (as function parameter) with an *asynchronous GET* call, *deserialize* the JSON response into implemented class (Weather) and returns it.



```
[JSON]
  location
    name: "Lameira"
    region: "Porto"
    country: "Portugal"
    lat: "41,16"
    lon: "-8,62"
    tz_id: "Europe/Lisbon"
    localtime_epoch: 1663749729
    localtime: "2022-09-21 9:42"
  current
  forecast
    forecastday
      [0]
        date: "2022-09-21"
        date_epoch: 1663718400
        day
          maxtemp_c: "25,3"
          maxtemp_f: "77,5"
          mintemp_c: "19,3"
          mintemp_f: "66,7"
          avgtemp_c: "22,1"
          avgtemp_f: "71,7"
          maxwind_mph: "7,6"
          maxwind_kph: "12,2"
          totalprecip_mm: "0"
          totalprecip_in: "0"
          avgvis_km: "10"
          avgvis_miles: "6"
          avghumidity: "69"
          daily_will_it_rain: 0
          daily_chance_of_rain: 0
          daily_will_it_snow: 0
          daily_chance_of_snow: 0
          condition
            text: "Sunny"
            icon: "//cdn.weatherapi.com/weather/64x64/day/113.png"
            code: 1000
          uv: "6"
```

Figure 2 – "Weatherapi" sample response content

2.5. Application Logic

The application logic is implemented inside a method (RunAsync).

2.5.1. *RunAsync()* method

This function contains the application logic and implemented using try-catch block to handle the exceptions properly. Firstly the “cities” List object is assigned to GetCityAsync() method. Then the “cities” List is being iterated using *foreach* loop to get the individual cities with “city” object. Inside the loop, GetWeatherAsync(string url) method is assigned to “weather” object to get the individual weather information from “weatherapi” for each city.

Url parameters for “weatherapi”:

- The “latitude”, “longitude” are extracted from “city” object (see Figure 1).
- The “API key” is obtained by registering to “weatherapi.com”
- “Days” parameter are defined globally (weatherDays) to provide versatility.

Followed by that, each “forecastday” condition is being captured inside a loop and the “resultString” is appended accordingly with the relevant data.

2.5.2. *Main(string[] args)* method

This method is the default entry point of the application where the application logic method “RunAsync()” is called.

```
RunAsync().GetAwaiter().GetResult();
```

Figure 3 – Application logic call inside main method

2.6. Static Code Analysis

The project is being implemented using Visual Studio 2015 which is a powerful IDE for creating web/windows based applications. Class, public/private class property, method, method parameter and variable naming conventions, code indentation and code statement implementation standards are considered precisely while doing the development. The code segments are supported with relevant comments to provide better understanding. Third party static code analysis tools such as PVS Studio and StyleCop is also used for measurement.

3. Web API Design

3.1. Description

In this section the API design for:

- endpoint/s to set the forecast for a specific city
- endpoint/s to read the forecast for a specific city

to/from TUI Musement API are defined.

The design is able to answer the questions such as:

- What's the weather in [city] today ?
- What will it be the weather in [city] tomorrow ?
- What was the weather in [city] on [day] ?

3.2. API Endpoints

The online API design editor, swagger, is used to design the API endpoints. (https://editor.swagger.io/). The raw swagger psudecode (weatherApiDesignSwagger.txt) and converted JSON description of the API (weatherApiDesign.json) are separately included inside deployment package.

There are 2 API endpoints designed:

- To set the forecast for a specific city (POST request)
  POST /cities/{cityid}/forecast
- To read the forecast for a specific city (GET request)
  GET /cities/{cityid}/forecast/{date}

3.2.1. POST */cities/{cityid}/forecast* endpoint

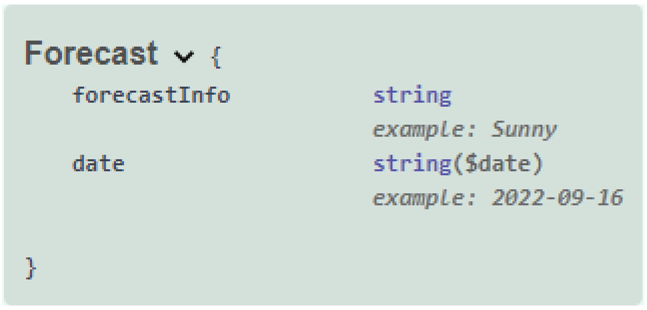
This request inserts weather forecast for a specific city.

- **Parameters**

cityid

- Desc: ID of the city to be inserted
- Type: Integer
- Example value: 1

- **Request body**
  - Type: JSON
  - Schema: Forecast



```
Forecast ⌄ {
    forecastInfo            string
                            example: Sunny
    date                    string($date)
                            example: 2022-09-16

}
```

- **Responses with codes**
  - **201**
    - Returned when successful creation.
  - **400**
    - Returned if sent data contains errors.
  - **404**
    - Returned when resource is not found.
  - **503**
    - Returned when the service is unavailable.

3.2.2. GET */cities/{cityid}/forecast/{date}* endpoint

This request retrieves weather forecast for a specific city for a given date.

- **Parameters**

cityid

- Desc: ID of the city to be returned
- Type: Integer
- Example value: 1

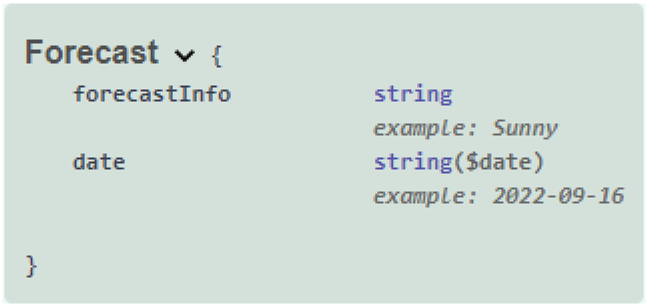
date

- Desc: The forecast date to be returned
- Type: Date
- Example: 16-09-2022

- **Request Body**

No payload for this request.

- **Responses with codes**
  - **200**
    - Desc: Returned when successful.
    - Response content type : JSON
    - Response schema: Forecast



```
Forecast ⌄ {
   forecastInfo           string
                          example: Sunny
   date                   string($date)
                          example: 2022-09-16
}
```

  - **400**
    - Desc: Returned if sent data contains errors.
  - **404**
    - Desc: Returned when resource is not found.
  - **503**
    - Desc: Returned when the service is unavailable.