Assignment 1: Web services

Fabian Denoodt, Benjamin Vandersmissen & prof. José Oramas

DS 2023-2024

1 Introduction

This document provides an overview of the requirements and background information for the first practical assignment of the Distributed Systems course. The goal of this assignment is to familiarize yourself with accessing public-facing REST APIs, implementing your own web service via a REST API and consuming the web service in an (additional) Python script. This assignment is **individual**.

Context:
After a cold winter, you are tired of this cold weather and instead, plan on making a trip to a warm location. However, your requirements are strict, and you want to travel to the country that is currently the warmest. As a result, you first develop an API to find out what location is currently the warmest.

2 Goal

The goal of the assignment is twofold:

- Firstly, you should create a **RESTful** API which by consuming three existing APIs (more on them later) provides a service to the user.
- Secondly, you should implement a Python script which consumes your API.

3 Requirements

3.1 Web service

The web service needs to offer the functionalities listed below. You should choose a good URI design for the web service that incorporates the functionalities, but also also adheres to the RESTful principles. This means that you may need to implement additional endpoints (and functionalities), depending on your URI scheme implementation.

The web service should be able to:

- Given the name of a continent, return the names of the countries in that continent. In addition, when no continent is given, all countries should be returned.
- Given the name of a country, return some details of the country. The details include the longitude and latitude of the capital city, the population size, and the country's area.
- It should also be possible to obtain the current temperature (in °C) of a country given its name. To achieve this, you can use the temperature of the capital city as the representative value for the country.
- You should be able to 'favorite' and 'unfavorite' a country, as well as generate a list of all favorite countries. (You can assume that the list of favorite countries is global, rather than specific for each user session.)
- Finally, given the name of a country and an integer $1 \leq n \leq 5$, a line graph should be generated displaying the temperature in that country for the coming n days. The measurements should be three-hourly based.

To implement these requirements, your API will need to consume the following three existing APIs:

- REST Countries (`https://restcountries.com/`) for retrieving any geographical information. Note that the service does not offer any information related to continents. Instead, you need to provide this functionality yourself for the following continents: Asia, Africa, North America, South America, Europe.
- OpenWeather (`https://openweathermap.org/`) for any information related to the weather.
- QuickChart (`https://quickchart.io/`) for consuming any graph-generation related services.

The following resources may be useful in your implementation:

REST API in Python: Implementing a REST API in Python is actually pretty easy using a couple of libraries. The main libraries are Flask (`https://flask.palletsprojects.com/en/2.0.x/quickstart/`) and Flask-RESTful (`https://flask-restful.readthedocs.io/en/latest/`). You should already have some familiarity with Flask from Programming Project Databases.

3.2 Python Script

The script (without GUI) should consume your API and demonstrate that all functionalities listed in 3.1 are present. Additionally, running the script should answer the query of which country in South America is currently the warmest. You should favorite that country and display a line graph containing the temperature forecast for the following four days.

4 Deliverables

The following are the minimum requirements and deliverables for a passing grade.

- You should include a way for us to easily run your solution via the files `run_api.sh` and `run_script.sh` that will automatically start your web service and run the Python script. Ensure that any information, such as API keys, is injected into the web service when it is launched. One way to achieve this is by passing the keys through command line arguments, e.g., `python main.py --key XXXX`. For more information, refer to the documentation on argparse: `https://docs.python.org/3/library/argparse.html`.
- You need to include a manual with your code that details the API endpoints you implemented along with the arguments they require, their return values and a small description.(for inspiration you can look at the documentation of the given APIs, or you can use specialized packages such as flask-restful-swagger)
- In addition, you should write a report (max 2 pages) which discusses the following aspects:
  - Motivate your design decisions and explain how the API is RESTFUL. Are there any designs you have considered but ended up not implementing and why?

  - Do you take care of efficiency? If so, how? If not, how would you deal with efficiency?
  - Can your API deal with faulty requests? If so, what kind of faulty requests and how do (or would) you handle them?
  - How many hours did you spend into this assignment?

- Make sure that your submitted solution is complete and has no missing files. If you use additional libraries, be sure to mention this clearly in the manual and either include them with the source or provide us with an easy way to install them. (for python this is with a `requirements.txt`). This also means that you need to provide the API keys you use as well, and need to make sure there is enough quota for us to test your solution.

5 Deadline and Submission

The deadline for this assignment is the **16th of April** at **23:59**. Late submissions will automatically have points deducted for tardiness.

Make sure that your submission has the following format and upload it via Blackboard!

`DS-Assignment1-`*`Snumber-Lastname`*`.zip`.