Report

Assignment 1: Web services

Orfeo Terkuçi

Design choices

I chose the following two as my resources: Country and Favorite

As 4 of the 5 requirements of the API, noted in the assignment, required me to return something related to a country, whether it be its name, information about the capital, temperature, etc…, it just felt natural, and logical for `country` to be a resource of my API.
As for the routes:

- `GET /country` returns information about all countries. I could have labeled this `/countries`, but decided against it as the rest of the routes revolve over a single country. There is a `continent` filter parameter, which is a query parameter
- `GET /country/{country_name}` returns information over a single country.
  You can be more specific and request the temperature with `GET /country/{country_name}/temperature` or the forecast for a specified number of days with `GET /country/{country_name}/forecast/{days}`.

I could have made `country_name` and `days` query parameters, but as they are both required for the special routes that they represent, I decided on making them path parameters instead. The routes are readable and you can figure out the meaning easily by looking at each route.
As for `favorite`, I found the idea of having them as a separate resource better, even though I could just as well have added them as a special route to the `country` resource, as technically, being “favorited” and “unfavorited” can be an action you can execute on a country.

My API is RESTful because:

- **Interface uniformity**: My API has a consistent and limited set of well-defined methods. In my case, I am using HTTP-based methods (GET, POST, DELETE).
- **Client-Server architecture**: My API is separated from the client. The client is not concerned with data storage, the latter remaining internal to the server
- **Stateless communication**: Each HTTP request from the client to server contains all of the information needed to understand and process the request. The continent filter is a query parameter for the request to get all continents
- **Layering**: My API gives no indication that there is a direct connection to a server, or that the client is connected to a proxy of the server instead.
- **Resource-based**: All the routes are based on resources, and different actions you can execute on these resources. They are also all nouns, short, hackable up the tree, and query arguments are only for parameters (such as the continent filter in my case). There are no file extensions in the paths

Are there any designs you have considered but ended up not implementing and why?

I considered making `temperature` and `forecast` a separate resource, but I decided against it because I only had one route for each of them, and they were both tied to a single country, so I decided on making them subroutes for the country resource, to request a specific “resource” for a specific country.

Do you take care of efficiency? If so, how? If not, how would you deal with efficiency?

There was not much to take care of regarding efficiency. I send requests to the respective APIs, and wait for the response. I have not modified the timeout (should be 10s), and such each response from my API should, in principle, not last longer than that timeout. The resources I get back are not large enough to require any optimisations, but I still only return a small part of the resources back with my response (namely, only the capital information, or country name).
If I had to process more data regarding the countries, depending on the quantity of the data I would consider processing it in parallel.

Can your API deal with faulty requests? If so, what kind of faulty requests and how do (or would) you handle them?

My API can deal with faulty requests. These faulty requests are:

1. API key is missing for the forecast: `Error 400: The API key is not correct`
2. Number of days is < 1 or >5: `Error 400: The number of days must be between 1 and 5`
3. Trying to unfavorite a country that has not been favorited: `Error 404: Country not found`

Every other error that can return from the API calls that I make, or in the inner-workings of my API returns an error `500: Internal Server Error`

- Error parsing the response
- Error getting the temperature forecast
- Error getting the country information
- Error getting the countries

There is also the possibility of favoriting a country with a wrong or partial name, but I always double check with the REST Countries API to check what country the user meant, as they seem to have a feature such that they try to find the best match for the query name you give them. For the removal of the country, however, no such check is made, and the blame is left solely on the user.

How many hours did you spend on this assignment?

I have spent 10 hours on this project. Around 7 hours on implementing the API and 3 hours on documentation, writing the testing scripts, refactoring and optimisations.