Jason Liu

Webservices Country

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?
I follow the Representational State Transfer (REST) architectural style. My endpoints are designed to be clear, descriptive, and follow a consistent pattern to make it easy for users to understand the resources they are requesting and the actions they are performing. Here's a breakdown of some endpoints:

1. */country/{country}* GET: This endpoint is RESTful because it uses a clear and descriptive URI, /country/{country}, to identify the resource being requested, which is the information about a specific country. I use the HTTP method GET to retrieve this information.
2. */country* GET: This endpoint is RESTful because it uses the same clear and descriptive URI pattern, /country, to identify the resource being requested, which is the information about all countries. Again, I use the HTTP method GET to retrieve this information.
3. */favorite* GET: This endpoint is RESTful because it uses a clear and descriptive URI, /favorite, to identify the resource being requested, which is the user's favorite countries. I use the HTTP method GET to retrieve this information.
4. */favorite/{country}* POST: This endpoint is RESTful because it uses a clear and descriptive URI, /favorite/{country}, to identify the resource being requested, which is the action of adding a country to the user's favorites. I use the HTTP method POST to send data to the server to create a new resource.
5. */favorite/{country}* DELETE: This endpoint is RESTful because it uses a clear and descriptive URI, /favorite/{country}, to identify the resource being requested, which is the action of removing a country from the user's favorites. I use the HTTP method DELETE to send a request to the server to delete the specified resource.

For the last two endpoints I just mentioned, it makes use of the same endpoint but just different call methods, so it is still based on the same resources but different functionality.

Yes I did consider moving everything under one endpoint instead of having country and favorite as separate resources, but because they had different uses I rather had divided them. This would have resulted in two less endpoints, if I did combine them but I preferred simplicity.

Do you take care of efficiency? If so, how? If not, how would you deal with efficiency?
Yes and no, like I mentioned before I split the resources up in two, so this is less efficient, but yes because the favorite endpoints can be differentiated with the call method.

Also I make use of FastAPI and not Flask, so I can use asynchronous API calls, what makes it also more efficient.

In the sense of an efficient way of scaling, it is also more efficient because just adding functionality after the {country} tag, and then adding the necessary resource is the only thing you need to do (REST principle).

Jason Liu

Can your API deal with faulty requests? If so, what kind of faulty requests and how do (or would) you handle them?
In case of a faulty request (I assume when we make a call to a different API, like openweathermap), I check if the status code is 200, if so then all good, if not we send back a different status code to the user, indicating something went wrong.

In case a user makes a faulty request in our API, depending on what mistake he made we send a status code back, if for instance a user adds a country to the favorite list and it is already in there, we just send “400 countries already added”. Similar to other mistakes we respond with different status codes.

How many hours did you spend on this assignment?
I started with Flask (pain 😭), which took me 6 hours just for the setup. I then switched over to FastAPI (I was familiar with both but because it was recommended to use Flask I tried Flask first) and that took me 2 hours for the project itself, bash script 30 minutes and this report 30 min. So overall 3 hours minus the bad decision of using Flask.

How to use my project?
I included a video in case it is not self explainable, but first make a virtual environment and then start the scripts in the main project folder.

First run the api script and then the bash to run the python script.