



DEVGRID

Full Stack Code Assessment Challenge

Objective

Design and build an application that for a given city name, collects data from the [Open Weather API](#), caches it for some configurable time and returns it as a JSON object. Also returns a configurable number of last searched cities.

Challenge Description

Back-end

You're gonna need to implement a service API with the following endpoints:

PATH	METHOD	DESCRIPTION
/weather?max=<max_number>	GET	Get all the cached cities, up to the latest n entries (configurable) or max_number (if specified).
/weather/<city_name>	GET	Get the cache data for the specified city_name , otherwise fetches from the Open Weather API, caches and returns the results.

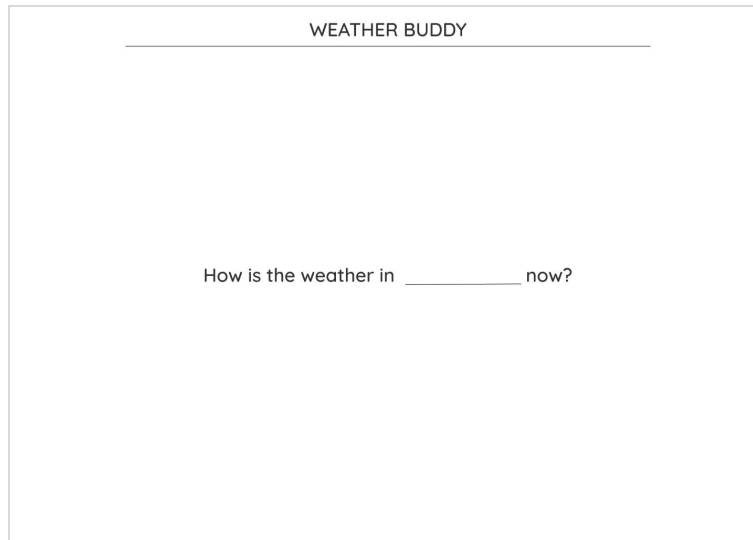
Specifications

- Use of Python 3 and Flask (for Python positions) or;
- Use of Node.js and Express (for Node.js positions);
- All information specific to the Open Weather API can be found in their [documentation](#);
- Cache must live for 5 min;
- Maximum of 5 entries by default.

Front-end

You're gonna need to implement a front-end app as specified in the attached mock-ups.

- When the page loads, the user needs to see an input for the city name with the label ***How is the weather in _____ now?*** The input goes on the empty space as shown below:



WEATHER BUDDY

How is the weather in _____ now?

It's important to have the focus on the input when the page loads.

- User types the city name and hits Enter (or just leaves the input field). Now the app will hit the endpoint to get the weather for that specific city:



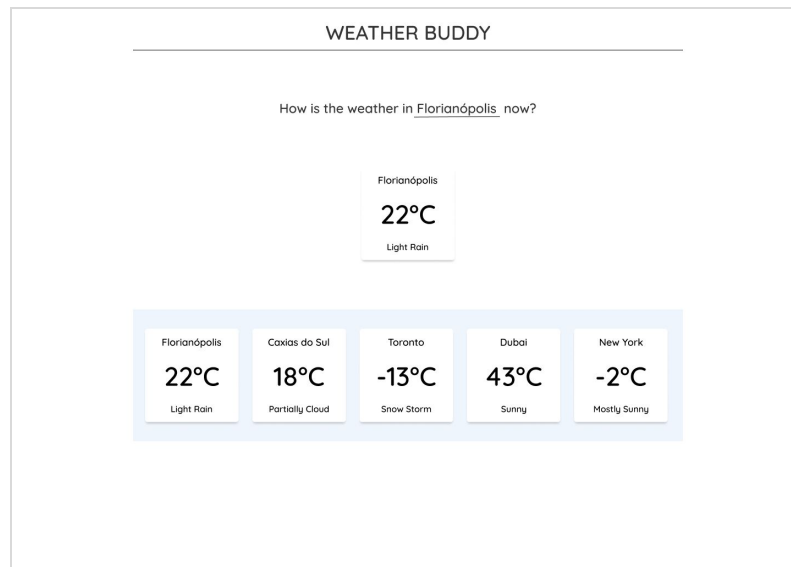
WEATHER BUDDY

How is the weather in Florianópolis now?

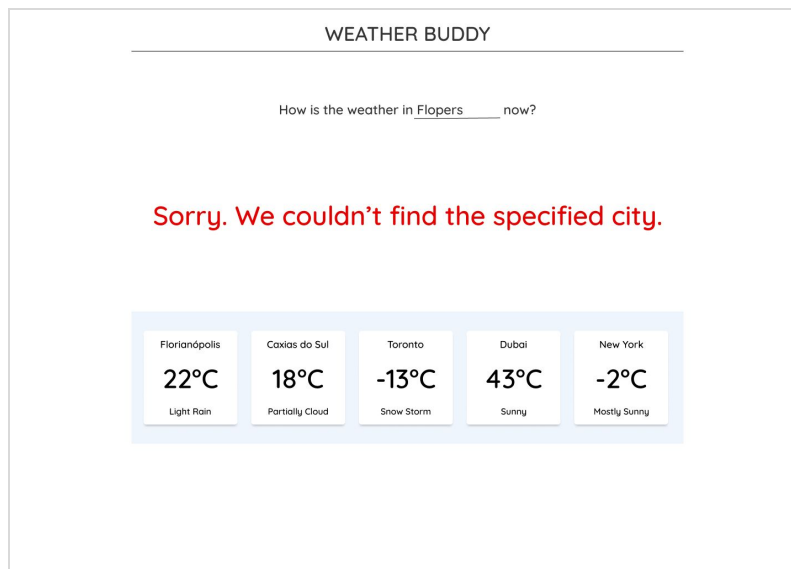


DEVGRID

- When the result loads, it should display the weather data as specified below. Also it is time to display the latest search results (up to 5 entries):



- If the entered name is invalid, it should display the message as follow:



- This is just an illustration of what is required and there's no need to be pixel perfect but you can use all of your CSS wizardry and creativity to take our breath away, if you wish so. :)

Specifications

- Use of React 16.3+ or;
- Use of Svelte 3+;
- Modern Javascript (ES6+);
- No TypeScript.

Overall Instructions

- Create a repository with your code on github / gitlab. Any one of your choice. We wanna see how you work your way through the code until you have the final solution so we don't wanna see everything in one single commit. Share the repository link with us when you feel you're done;
- Test as much as you can. We don't require 100% of test coverage but we want to see every possible decision branch tested. You can use any test strategy and tool set you want and are comfortable with;
- We need to run your application so a well written README file with instructions on how to run and how to test it is mandatory;
- Setting your environment in a Docker file (preferably compose) is better for all of us. We need to see your application running and also we don't want to break our own environments if something fails. Also, if you deploy your application in any deployment service is a plus but it is not mandatory. Don't forget to share with us the link, if that is the case;
- If you need clarification in some of the requirements, ask the person responsible for your application to contact us. We need to be sure everything is clear so you can deliver us the best solution ever!