Backend case - AppTweak

Congratulations again for reaching this milestone in our recruitment process! The next step is a technical case. Regarding the technical content of this case, we decided to have something similar to the kind of challenges you would be facing in your day-to-day work at AppTweak. While working on this case, you can imagine that you are already part of the team so if you have any questions, don't hesitate to reach out to us!

The following instructions might seem a bit vague, but that is on purpose. We want you to come up with your own solution instead of just completing a list of bullet points. We have no requirements on what language to use but we do have a preference for Ruby as it's what we use here.

To finish this technical case, we give you **7 days** but no worries, you won't need all that time to accomplish it! We are expecting you to need around 6-8 hours to complete it. If you finish before the deadline, feel free to share your code with us and we'll be happy to review it sooner than expected.

Introduction

For this exercise, we will ask you to build two API endpoints that allow users to follow locations and get temperature forecasting data.

First, you need to implement an endpoint to create and store locations in the database. This endpoint should accept:

- Latitude: a floating point that specifies the latitude of the location
- Longitude: a floating point that specifies the longitude of the location
- Slug: a name for the location that's URL-safe. This slug is unique per location

After that, you need another endpoint that returns the temperatures forecasted for a previously saved location for the given date range. This endpoint should accept:

- Slug: a name for the location that's URL-safe. This slug is unique per location
- Start_date: a start date to query for temperatures, this is a string of the form YYYY-MM-DD
- End_date: end_date for the query

The response must have a list of the dates contained in the period requested and for each one of them, the minimum and maximum temperatures that were forecasted for that date, in JSON format.

For example, if a user asks for forecasted temperature data between Jan 1 2021 and Jan 3 2021, the API should return:

```
[
  {"date": "2021-01-01", "min-forecasted": 18, "max-forecasted": 26},
  {"date": "2021-01-02", "min-forecasted": 17, "max-forecasted": 24},
  {"date": "2021-01-03", "min-forecasted": 19, "max-forecasted": 25}
]
```


Data Origin

To fetch forecasted temperature values, we ask you to leverage: http://www.7timer.info/doc.php?lang=en

Use the “Machine-readable API”, which can return values in XML or JSON. This API does not require any key/registration. We ask that you only use the ASTRO endpoint that returns temperatures for this case.

In the sample call:
https://www.7timer.info/bin/astro.php?lon=113.2&lat=23.1&ac=0&unit=metric&output=json&tzshift=0

The API returns the forecasted temperature in Celsius for every 3-hour window, in the “temp2m” key (Temperature at 2 meters).

Additional Requirements

We are aware that the 7timer API does not allow querying on past forecasts, so you must think of a strategy to regularly get data for the locations stored in the database. Please only leverage that API as a source of data and ensure your system is built to ensure all required data are fetched as needed.

Important information

- This is purely a backend project, we don’t expect any visual interface.
- We do not require any kind of authentication/security for the API. Consider that this API would run in a secure network with trusted and cooperative clients. Design the API to be as easy and simple to use as possible.
- We also do not require any user management system. You should focus on the solution itself without worrying about user access to the stored data.
- You are free to use your favourite programming language/libraries / ..., as long as you don’t leverage any other source of data for the temperature values. If in doubt, don’t hesitate to ask.
- We do care about data correctness and the general resiliency of the architecture of your system.

Delivery

For the delivery of your solution, we ask you to share a private GitHub repository with us. We are looking forward to seeing what you can bring to us.

Good luck!