Python Assignment: serving a weather API as a microservice

Background

In order to generate forecasts for individual customers, we use a number of features, including: day of the week, maintenance calendar, holiday calendar.

A feature that is conspicuously missing from our stack at the moment is *weather information*. One of the reasons is that B2B gas consumption is (almost) independent from weather conditions, and another reason is that obtaining high-quality weather information is not straightforward.

The assignment consists of providing imperfect weather information using the API of http://www.7timer.info/ through a dockerized microservice. This can be seen as “wrapping 7Timer API in your own API”. While the information available is probably not good enough, the output of this exercise might be a base we will iterate upon.

Assignment

Using the API available in 7Timer and docker, produce a microservice which, given a point in space, returns for the next 48 hours (or at least the next full day) a time series with the following data per time point:

- `start_period_utc`: start of time point interval.
- `end_period_utc`: end of time point interval.
- `cloud_cover`: This is what I mean by “imperfect information”: ideally we would have “solar radiation”, but I think it is not available in the API. So let’s use `cloud_cover` as a proxy.
- `temperature`

The result should be a container that upon run locally, allows to access a REST API from the local browser: the *input* to a GET request should be a URL with `longitude` and `latitude` as query parameters, and the *return* an JSON response with the information described above.

It is important that `start_period_utc` and `end_period_utc` are returned in unambiguous UTC timezone. The time resolution of the return will be determined by the one of the raw data available.

For the code inside the container use FastAPI unless you believe there is a better choice.

The result can be shared either in a public git repo, or through a zip folder including all the files.

It must be reproducible on local using a simple `docker` or `docker compose` command.

Expectations

The result is not intended to be *perfect*, but rather show the tradeoffs taken at working on the assignment from scratch during (at most) one day.

For instance, code testing and documentation is overkill for the scope of the assignment; it sounds more reasonable to focus on producing solid, succinct code that is clear in its intent and easy to iterate over.

Regarding your decisions and thought process, we will have a conversation about your design choices, what corners were cut given the time available, and how subsequent iterations could improve the assignment.

Bonus points

The following points are avenues for investigation if you end up with spare time. They are out of scope because they seem unreasonably time-consuming. Still, it would be interesting to hear at least your ideas about how to implement them:

- Instead of `longitude` and `latitude`, use as input a Spanish postcode (for instance, `04006`, `28014` or `47012`).
- Somehow estimate the solar radiation from `cloud_cover` and/or other readily available information.