Backend Assignment

This document describes the evaluation assignment for the position of a Backend Engineer.

Required technologies to use are:

- PHP

Preferable, but not required, technologies to use are:

- Symfony framework (3.4+, 4.0+)

Techniques

- Object Oriented
- Code should follow SOLID principles
- 3rd party libraries should be installed with a package manager, like Composer

What are we assessing

In general architectural choices are important to us.

- Application architecture +++
- Programming patterns +++
- Coding style & quality overall +++
- REST API Design skills +

On theoretical level, we are looking into:

- Communication skills +++
- Application performance design skills ++

What are we NOT assessing

We do not expect beautiful pages. You should consider that a frontend developer will be giving a hand to make things pretty in an actual team.

Time

We understand that you are busy with more things, so the time you can afford to put on any parts of this assignment depends on you.

However we would like to agree upfront on a date when you will come back to us with your solution. See it as a mini project. In a real-life scenario, you would be giving an estimate and commit to a milestone, right?

The Exercise

A bit of context...

> Let's assume we are a startup trying to give users the best weather predictions on Netherlands (That's difficult!).
>
> Our app collects that from several weather channel partners (Weather.com, BBC, IAmsterdam, etc) around the world and internally it finds the accurate result based on the data collected from our partners.
>
> Since the Netherlands is a really international country we show these predictions in Celsius and Fahrenheit scales.
>
> We also know the importance of showing the predictions for the next upcoming days so our app provides a detailed report showing the predictions for the next 10 days.

The (actual) assignment

The goal is to create a REST API where users can search for a city and find the weather predictions for that city for the current day, and for any day in the upcoming 10 days.

- The data we use is provided via api integration with our partners. The data is provided in different formats (XML, JSON, CSV...)
- *The data samples provided are attached to this message*. You don't need to create a real server in order to fetch this data. You should mock it.
- The temperature that is shown to users is always the an average we collect from our partners:
    - TempPartner1 = 2, TempPartner2 = 2, TempPartner3 = 5 **=>** Average = **3**
- It should be easy to add new integrations with new partners.
- Should be easy to add new temperature scales like Réaumur, Newton and Romer.
- The data provided via our partners changes all the time. We've set a threshold of 1 minute to invalidate this data.
- Users of the API should always get the latest version of the prediction, as provided by our partners.
- It should not be possible to search dates greater than the current day + 10 days.
    - Today is 01/01/2018. Searching for data from 11/01/2018 is ok.
    - Today is 01/01/2018. Searching for data from 12/01/2018 is not ok.
- It should not be possible to search for dates in the past.

To UI or not UI?

Give us a way to see your API in action, and if you like, optionally, you can surprise us with any front-end engineering skills. No worries, we won't be shocked if the UI looks ugly or if we have to fall back to the command line.

What we need is a super simple interface to handle the following functionality:

- See the predictions for the current day.
- See the predictions for any day in the next 10 days.
- Predictions must be shown on Celsius and Fahrenheit scales.
- You should show data in a human readable format.

Deliverables

In your submission, we would like you to include:

- All code (frontend, backend, any QA assets) developed to create this functionality
- A working app hosted anywhere you like (amazon free tier?) OR installation instructions

Please use a code repository like Github or Bitbucket to share the deliverables with us.

Bonus Questions

- It's a big plus if you add (some) tests to your code (let us see your QA skills)
- Looking at the description of the context, please provide architecture diagram(s) on how you would technically design the functionality described for this application in a scalable way. You can decide yourself what types of diagram(s) will help you communicate to us your thoughts about building such application.
- Which parts of the application would you have implemented differently and why, should you have more time?
- If we would have asked you to unit test your code with 80% code coverage, which parts would you cover, which not and why. How would you be able to verify and prove to us that you achieve such coverage?

Thank you upfront for your time and effort, and for allowing us to see you in action!