UP42 Backend Coding Challenge

Thanks for trying our coding challenge! We are excited that you made it to this stage, and we are looking forward to reviewing your solution.

For this challenge, we want you to extend a simple backend application that is capable of working with imagery metadata in the GeoJSON format (it is widely used in the geo-special industry as well as on the UP42 platform). The goal is to expose several endpoints to allow clients to access features representing an actual image with its associated metadata.

State of Affairs

We are sharing an existing project with you. Following the tech stack of choice at UP42, the project is built with the Gradle Wrapper and contains a Spring Boot application written in Kotlin.

The data set that has to be used to extract data from is already placed in the project under `/src/main/resources/static/source-data.json`.

There is also a rough implementation of listing features endpoint - it returns a list of all features in the following format (this example data matches the first item in the static feature collection):

```
{
    "id": "39c2f29e-c0f8-4a39-a98b-deed547d6aea",
    "timestamp": 1554831167697,
    "beginViewingDate": 1554831167697,
    "endViewingDate": 1554831202043,
    "missionName": "Sentinel-1B"
}
```

Your Task

1. The ‘listing of all features’ endpoint was implemented as a PoC and is currently in a poor state. Your first task is to refactor the code and bring it to a state when you find it good enough for production usage. Ideally, your refactoring result should help you with the second task.

2. Please add a new endpoint to get an image for the given feature id. It should be a GET endpoint with the path `/features/{featureId}/quicklook` and returning the `image/png` as a content type. You can find the image as a base64 encoded string in the data set under the `quicklook` field.

Requirements and Submission Guidelines for Your Solution

We expect you to send us the source code of a running application for the proposed challenge. Please make sure to meet the following requirements for your submission:

- We highly encourage you to write your solution in Kotlin, even admitting that you might have never had a chance to work with Kotlin in your professional career. However, if you do not feel comfortable with this, please provide your solution in Java - we will review it regardless of which option you choose!
- Please take some time to think about testing and demonstrate your approach to this aspect.
- When submitting, please provide a ZIP file with your project including the local `.git` folder.
- Code and commit messages should be treated as you would on a real-world task. It means we want your code to be well-factored, without needless duplication, and follow good practices.

We estimate that this challenge might take around 2 hours to complete, but this is not a hard limit. Feel free to take the time you need to ensure that it reflects your skills. Please also feel free to document your assumptions, trade-offs and every decision worth mentioning in the README file.

Challenge Assessment

We assess the challenge according to the following points:

- Clean code and general code structure (separation of concerns),
- Code quality (testing and maintainability),
- API design and error handling,
- Data management,
- Bonus requirement: Kotlin idiomacy.

We hope that you will find this challenge enjoyable & we are looking forward to your solution!

– The Engineering Team at UP42 🚀 –