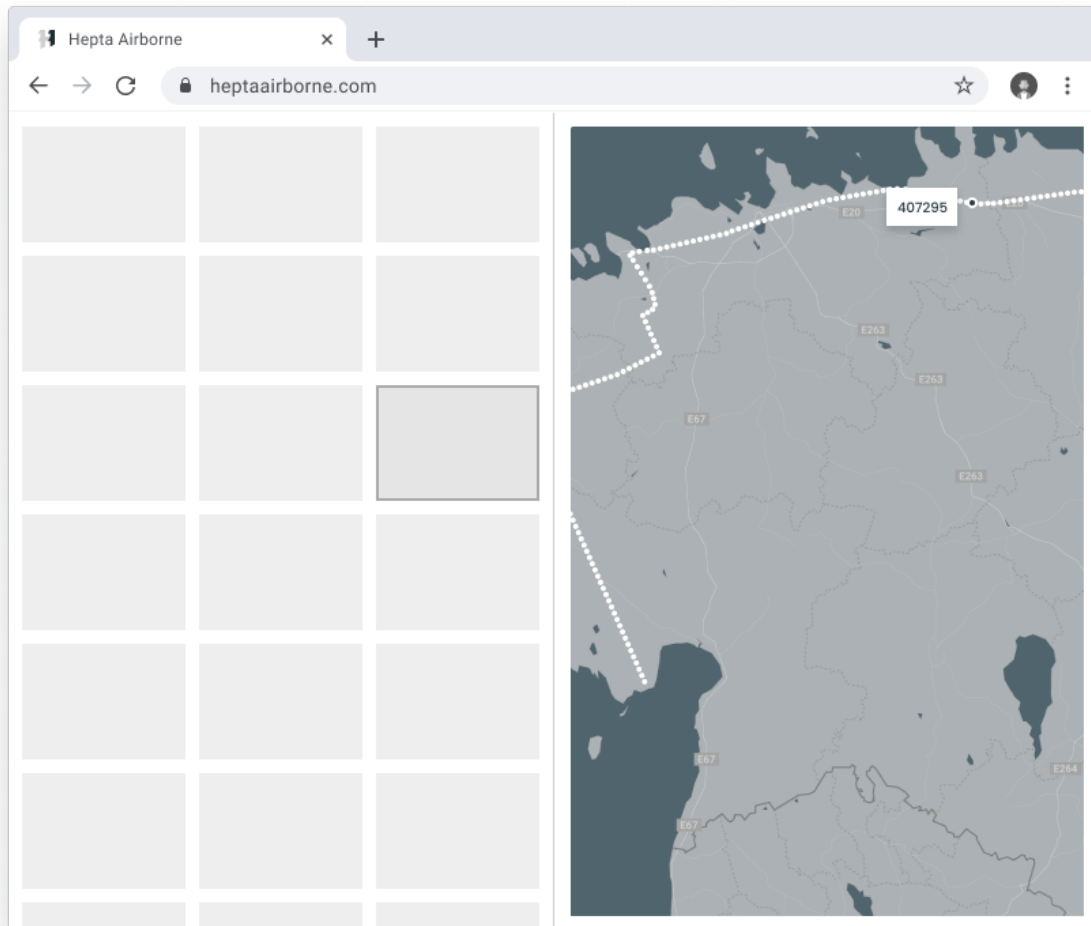


Software Engineer Technical Exercise

Introduction

Hepta builds tools for working with large amounts of data. One of the main tools used by grid experts daily is the browser-based infrastructure viewer, which allows to quickly find a location and browse through images captured of the object.



The application has to be extremely snappy and optimized for rapid viewing as it is used by many internal and external parties in situations where operational speed is of utmost importance. This means that the developer creating the UI has to be clever to allow for rapid scrolling around to sift through hundreds of thousands of images without a performance or usability penalty.


Task

Your task is to create **a simple app to show a grid of images and an API endpoint to get the data**. The resulting application needs to be performant and easy to use.

Create a Spring Boot application with:

- Spring Data JPA
- Relational database (e.g) Postgres

- Liquibase
- Swagger (OpenApi)

Import a list of 15000 images to the database. You can use  [Lorem Picsum](#) as a source for the images or any other images you prefer. You can use raw SQL in a changeset for the importing or any other way you prefer doing it.

Create one API endpoint where you can get the list images as a JSON with optional pagination url parameters like pageNumber, pageSize.

Example return JSON format for the list of images:

```
1 [
2   {"id": 0, "url": "https://picsum.photos/id/602/240/170.jpg"},
3   {"id": 1, "url": "https://picsum.photos/id/486/240/170.jpg"},
4   {"id": 2, "url": "https://picsum.photos/id/666/240/170.jpg"},
5   ...
6 ]
```

The resulting JSON should also include currentPage, totalPages, pageSize and totalResultCount. When Return some default values for page and pageSize values if paging parameters aren't provided in the request.

Create the API endpoint so it would be possible to test it using Swagger.

The application needs to request the list of images through the API endpoint that you created earlier.. The images have to be displayed in the grid.

Main objectives are

1. Instantaneous feel – time to wait for anything to load (as perceived by the user) should be minimized, UI shouldn't jump when results are loaded, the content should be loaded quickly even if it's of lower quality at first etc.
2. 60 fps scrolling – the UI should remain responsive even when looking at hundreds of thousands of images
3. Rapid browsing – UI should enable quick and responsive browsing in both ways, i.e. if the user opens the page in the middle of the grid, the UI should respond quickly when scrolling up or down
4. Image loading trigger - add an UI element that will trigger loading of the images to the grid (button, clicking on the left column, etc.). So, the user could load new images by clicking on that element instead of reloading the page.

You don't have to implement the map on the mock-up. To get the look and feel of a real application please do create a two-column layout. Feel free to fill the right half with non-interactive mock content or however you see fit.

In the field, internet speed is anything but excellent. If you feel that you'd need to have photos at a different resolution, apply effects or something else, feel free to use the full potential of Lorem Picsum to make the app fast.

Technology-wise, we use React and TypeScript so we would appreciate if you'd use the same setup.

Results

- Please implement the task above and **share it privately** on Github with users [german-hepta](#) and [kuido85](#)
- Make sure to provide the instructions how to run the app locally for the person reviewing the task results.
- The application has to be able to run without contaminating the reviewer's global environment.