🚌

Dev Challenge (backend) - Transport 2024

Interview process information

- Below is the task statement. Please read it carefully and reach out if you have any questions!
- The estimated workload is up to **6 hours** - don't feel pressured to spend more. It's possible that you won't have the time to make everything perfect, so make sure to prioritise areas which showcase your skills the most.
- You can **spend the time in any way you want** (1 burst, 1h/day, ...), **no need to track** exactly how much/on what you spent your time on.
- We would expect you to **complete the task within 5-7 days** - once you are done, please reach out (+send the source/github link to us), and we'll follow up with you shortly.
  If you need more time (holiday, etc.), let us know and we'll figure it out!

Task statement:

> 👉 Implement a service with persistent storage and REST API endpoints, which computes statistics over real-time Helsinki public transport data. No form of a frontend is required, if you can demonstrate the functionality without it.

Data source information

In Helsinki, there is a public API available for various public transport data over at Digitransit:

In particular, we will be interested in the high-frequency positioning data (which uses the MQTT protocol), details about which can be found at:

The idea is that the service will retrieve real-time positions of public transport vehicles in Helsinki (= the messages with `event_type == vp` - see the linked docs above), which will be parsed and stored in some storage/database (exact storage structure is up to you).

> ⚡ *Note: This task relies on data provided by a third party. While we believe it is reasonably stable, it might happen that some larger disruption is preventing you from properly working on the task. In that case, please reach out asap!*

> 🔑 *Note2: Recently, DigiTransit APIs started to require authentication. While we believe that the specific real-time positioning API we mention above does not require any form of authentication to function (tested on 28.5.2024), we will also provide you with a valid DigiTransit API key, which you should be able to plug-in easily.*
> *(We requested the key clearly describing that we're using the data for interviewing purposes, and we believe it is in line with the provider's terms of use.)*

Required REST API endpoints

The REST API of your service will serve as a means to query statistics about the data captured via the MQTT connection. Below are three endpoints that you should implement:

1. Given a latitude and longitude, show the "three closest vehicles at the moment" (for each vehicle you may use the latest position at which you observed it, but feel free to do a better approximation of their current position if you want to). In the response, return some basic information about the vehicles, such as their route numbers, vehicle numbers, locations, speeds, distances to the given position, ...
2. Using all data observed so far, for each route number, return the average speed across all vehicles observed on that route so far (so the response will be some array/object). The response time of this endpoint should *not* depend on the total amount of ingested messages so far.
3. As a company, we have a hypothesis, that every time a nearby metro goes too quickly, we feel the ground move! To investigate, let's define for every metro vehicle `v_max` = the maximum

speed it was observed driving, while being *close* to our office. This endpoint should return the list of metro vehicles with the following information:

- route & vehicle number
- `v_max` = maximum speed while *close* to the office (in *km/h*)
- how many milliseconds ago `v_max` was (last) achieved
- how far away from the office it was at the time of `v_max`

Relevant details:

1. There are two metro lines in Helsinki currently, denoted by route number `M1` and `M2`.
2. Let `close=400m` in this case.
3. Our office location can be approximated as:

```
lat: 60.18408281913011
lon: 24.960090696148473
```

Useful notes & tips:

- The definitions of paths, parameters, etc. of the endpoints are vague on purpose - in your implementation, finish them in the way you think makes most sense.
- Beware that the data returned by the positioning system is not always reliable - for some lines, some data (eg. position) might be missing. The app should not crash, or stop working in case incomplete data is received - some best effort behaviour is expected + you can discard the incoming messages which do not make sense.
- If you are stuck on the implementation of one of the endpoints, or you feel like their implementation is out of the challenge's timeframe - try to write down at least some basic structure and let's talk more about it on the review!

Final recap

Required features of the service

- Connects to the DigiTransit high-frequency positioning API using MQTT and saves the relevant data to a *persistent* storage.
- Allows querying the three REST API endpoints, which return the data specified above.
- Can run long-term without crashing, or corrupting the data.

Things to keep in mind

- The endpoints should be reasonably fast with data spanning multiple days. We leave it up to you what exactly "reasonably fast" means.
- Be sure about the *correctness* of your implementation - rather implement fewer features, which you really believe work correctly.

What we'll be looking for:

- Clean and reasonable choices/decisions and the ability to argument about them.
- Ability to prioritise and recognise what is valuable - focus on those aspects of this task where you can showcase your skills.

Recommended tech stack

- Backend: Node/TS
- Storage: any relational database
- MQTT client: https://github.com/mqttjs/MQTT.js
  (we tested this one and are sure that it works fine with the HFP API, exactly the way they show it in the docs)
- Other libraries (web server, ...): up to your preference

Still unclear? More questions? Don’t hesitate to ask!

❤ Good luck, have fun. We're looking forward to your solution!