Digital Travellers

An application for recurrent travellers

Miquel de Domingo i Giralt

TBD

Contents

List of Figures

List of Tables

1. Introduction, motivation, purpose and project goals

2. Frame of work

3. Project structure

4. Theoretical background

5. Project structure

5.1 Team

Esto no sé como preferís explicarlo. A tener en cuenta:

- Sprints cada dos semanas.
- Proyecto en GitHub con las tareas.
- Anything else?

5.2 Backend

5.3 Frontend

Regarding the frontend, the application will be stored in a monorepository. The idea of keeping the code in a single repository is to simplify the workflow of the developer or developers. By having a single repository, changes in the domain layer can be made at the same time that the presentation layer is updated. Nonetheless, this small architecture decisions will be explained more in depth in further sections.

5.3.1 Planning

When planning the work for the frontend project, our main goal was to avoid being blocked by backend progress. It is common for frontend and native developers to fall behind schedule because they are unable to progress until the backend is complete. One solution to reduce this lag between frontends and backends is to use GraphQL. However, in our case, we opted to apply GraphQL ideas in a RESTful API, resulting in the following steps:

- Defining the expected request body of a specific endpoint, in order to let the frontend know *what the service expects to receive*.
- Defining the response, if any, that an endpoint would return for a given call.

This approach established a clear contract between the backend and the frontend, with separated implementations that both parties agreed to respect.
To facilitate development, we used a service worker to mock every API call made in the local environment of a frontend developer. The service worker respected the specified contract, allowing us to develop the frontend without being blocked by the backend. Once the app was deployed, the service worker was disabled, and the API calls were made to the *actual* RESTful API. This approach allowed for fast and non-blocking development for both frontend and backend.

The second thing to take into account before starting to structure and develop the frontend are the designs. Since there was not enough time to create a system design and then enough designs that would cover all use cases of our app, we opted for simple designs, which are shown in later sections. As a first sprint, or sprint zero, I was

in charge of prototyping the frontend application, which would simplify the process of developing the frontend app.

Next steps are more involved in the development and architecture of the application. As explained in previous sections, we got together every two weeks, as well as keeping an asynchronous communication. We considered each meeting to be a deadline, and in the meeting we would discuss the next steps to take or changes if any. However, since there are always unexpected tasks, it has not been easy to strictly follow the devised roadmap. The following diagram illustrates an aproximate planning of the sprints.
TODO: Roadmap picture

5.3.1.1 Sprint 0 and 1

5.3.1.2 Sprint 2

5.3.1.3 Sprint 3

5.3.1.4 Sprint 4

5.3.1.5 Sprint 5