Service Design and Engineering Project

Alex Zanetti
220225

Massimo Mengarda
214290

February 2021

1 Introduction

The project represents a dashboard for monitoring COVID data for different regions in Europe. For the implementation of the project, three regions have been chosen (Belgium, Italy, United Kingdom), but the architecture is easily scalable given the proper dataset and adapter.

The project has been divided in two parts: a back-end and a front-end. The back-end, which respects the RESTful design principles, is organized in multiple layers and provides API to interact with the COVID data. A small front-end has been developed to create the dashboard itself, gathering all the information in automatic and user-friendly way.

The back-end has been deployed on Heruku and it is reachable through this link.

2 Back-end and Architecture

The back-end has been developed in NodeJS and organized in different layers.

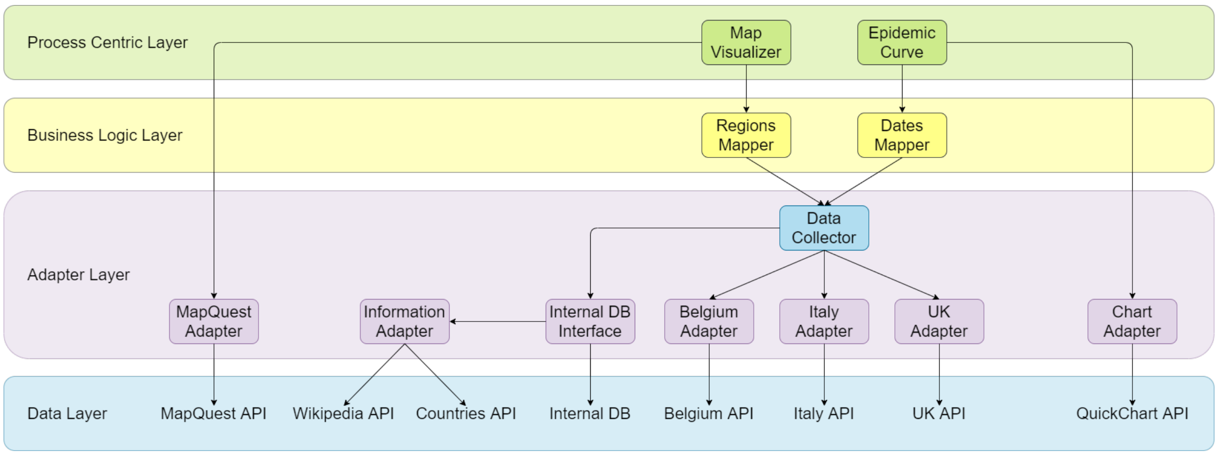

Figure 1: Graphical representation of the architecture of the API

In the data layer are listed all the external APIs (apart from the Internal DB) that have been integrated in the project: these interfaces provide raw data that needs to be elaborated in order to make it consistent and to better interact with the above layers.

2.1 Adapters

As mentioned before, a new region can be added to the system if an external dataset is provided and an apposite adapter is created. This is the case of Belgium, Italy and UK adapters, where the extracted data represents the number of cases in each province for every day. Since the fetch operation is costly (it can take many seconds), an internal database has been deployed. To orchestrate all these components, a data collector has been developed: in this way we were able to provide fine-grained end-point semantics and to avoid code duplication. As it can be seen in Figure 1, the internal database interface communicates with the information adapter: this interaction happens only once (at the creation of the database) and it is useful to gather static information about the regions in the system, such as country size, coordinates and population size. Finally, the MapQuest adapter and the chart adapter are respectively used to create a map of infection in a country and a chart representing the variation of new cases in a date interval.

2.2 Business Logic

These components are used to collect data in a well-defined way. In particular, the regions mapper service is in charge of mapping the new cases on a given date in a region in the following way: supposing that *Milan* and *Rome* are the two provinces of the requested country, the output will be in the following format:

```
[["Milan", 100], ["Rome", 200]]
```

This data will most likely be used by the map visualizer process to fetch the infection map. In a similar way, the dates mapper service has the task of mapping regional cases to dates, therefore its output will be in the following format:

```
[["2021-02-05", 100], ["2021-02-06", 200]]
```

Again, this data is useful for the goal of the chart visualizer process, which is in charge of fetching the chart mentioned above.

2.3 Processes

The processes chart visualizer and map visualizer are in charge of gathering data from the lower layers to provide a more user-friendly API. These, along with the raw data collected from the internal database, have been displayed in the front-end to provide a full and clear view.

3 Front-End

The idea behind the creation of a dashboard was to provide a simple and immediate to understand interface for the final user. To achieve this, the front-end of our project has been developed in Angular 11, using components belonging to the Angular Material library.

3.1 Structure

The interface is composed of three main parts. The first one is composed by three buttons (in our case representing Italy, Belgium and United Kingdom) that allow to choose which country to inspect in detail and report the latest data available.

The other components show full details about the chosen country. The second part indicates all cases divided by province and a graph showing the trend of new cases day by day. By default, the date interval is set to the last week, but it is possible to select different values.

The last part is in charge of showing the geographical distribution of the cases in the country through the use of a map. As before, by default the map is shown with the cases of the latest available day, but it is possible to choose another date thanks to the date picker.

3.2 Architecture

The dashboard queries the back-end through HTTP calls, receiving JSON files in the case of data, or images upon chart or map requests. All these procedures have been enclosed in specific services to allow code reusability and improve the project modularity. This logical division also allowed us to create a highly scalable architecture, since the deployed components are general enough to guarantee the possibility of adding new countries to the system in the future.

3.3 User Experience

Since data fetching operations can take a long time to be completed, to improve the user experience we decided to show a loading GIF while retrieving data from the back-end. Also, acknowledged the fact that there is the possibility of connection failures, the web-app is provided with an HTTP interceptor as provider. This component is in charge of showing custom errors, instead of showing the loading GIF indefinitely.

4 Documentation

The documentation of the API can be found at this link. It has been written using Swagger (OpenAPI version 3.0.0) and the Heroku server has been linked to it, meaning that it can be used directly to try out the project API.