SDE project report:
Flight tracker

Davide Zanella

February 2020

1 Introduction

The project, following described, is a microservice architecture implementing a Telegram bot capable of search flights, track them and help the user to arrive at the destination airport in time for the landing of an airplane.

To achieve the objective two external API are used: one for obtaining the flights' information and the other for calculating the time needed for moving between two locations.

The whole project can be seen as a standalone service, except for the "telegram wrapper" service, which allows the users to interact with the microservice architecture using Telegram.

2 The architecture

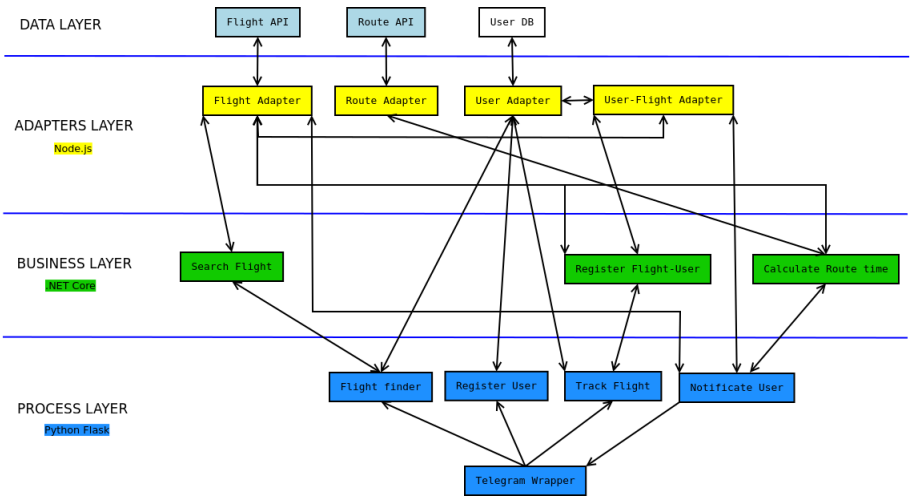

Figure 1: Scheme of the architecture

As an adding value, for every layer of the architecture, its services are written using a different programming language from the other layers. Specifically, the Adapters layer's services are written in Node.js using the Express framework, the Business layer's services in .Net Core and the Process layer's services in Python3 using Flask.

2.1 The Data layer

In the Data layer, there are, mostly, external API services. In particular, the Flight API is a service provided by Lufthansa which allows, using an API key to search for flights and airports and retrieve information about them.

The Route API is provided by "openrouteservice" and allows, among other things, to calculate metrics about a trip between two points.

The User DB is a PostgreSQL database server that stores information about all the Telegram users that are registered by the bot, such as first name, last name, ID of the Telegram chat and Telegram username.

2.2 The Adapters layer

In this layer, two over four services, Flight and Route adapters, interact with an external API and get information from them returning to the caller only the information needed to achieve the objective of this Telegram bot.

The User adapter executes queries on the User DB after doing checks on the inputs asserting no wrong data are inserted.

The User-Flight adapter stores in memory the associations between users and flights. It stores the flights that are being tracked by the users.

2.3 The Business layer

The Search flight service has the role to check if the destination airport, given as parameter, is valid and then to call the Flight adapter to retrieve all the flights arriving at that airport. In the future, it would be useful to allow this service to accept the name of the airport as a parameter, instead of its IATA code.

The Register flight-user service aims to check if a given flight number is correct and then forward the association request to the User-flight adapter. This service was designed to check if the destination airport of the flight is achievable by the user, so at least located in the same continent.

The Calculate route time service first checks if the airport code is correct, then retrieves the location of the airport using the Flight adapter and then demands on the Route adapter the calculation of the distance and the time needed to go between the two points.

2.4 The Process layer

The Flight finder and the Track flight services implement both a sort of authentication middleware, getting the Telegram ID of the user in the header of

the request and checking if the user is registered. After that, they will check the presence of the parameters and forward the request to the specific business layer's service. Thanks to the authentication control implemented in a high layer like the process one, every unauthorized user is blocked at the beginning of the request flow.

The Register user service is the only one, except for the telegram wrapper, which doesn't want an authenticated user. Its role consists of checking if a user is already present in the DB. If this is the case, it will update the user's information otherwise it will create a new user, calling, in any case, the User adapter.

The Notify user service has two different objectives. The first one is to check, at every specific interval of time, if there is any news about the state of a flight tracked by a user. For doing this it uses the User-flight adapter and the Flight adapter. The second objective consists in serving an endpoint that, using the Calculate route time service, returns the duration of the trip to the airport and the suggested time to depart.

The Telegram wrapper service, as said in the Introduction, is a support service that interacts with Telegram using a python library. It gets the messages sent by the user, parses them, calls the proper process service and then formats the result as a human-readable string.

3 Technologies used

The entire architecture can be run using Docker and in particular with the docker-compose tool. Each service has a swagger file where its API is documented and its Dockerfile making each service independent from the others. A docker-compose file in the main directory allows us to build and run each service with just one command. Some services, like the Flight adapter and the Route adapter, need an API key to run. They are provided as environment variables and declared in the proper service's Dockerfile.

To develop this architecture it was used a Github repository that can be found at `https://github.com/davidezanella/SDE_flight_tracker_bot`.