POLITECNICO
MILANO 1863

Software Engineering 2
Politecnico di Milano, A.A. 2017/2018

TravIendar+

Implementation and Test Deliverable Document

Version: 1.0
Release date: 7/1/2018

Riccardo Novic (mat. 829184)
Lorenzo Pratissoli (mat. 893980)

Software: https://github.com/EsotericVoid/NovicPratissoli/tree/master/Implementation

1 Introduction

1.1 Scope

This document describes the code developed that runs Travlendar+. It provides detailed information on the adopted frameworks and on how these frameworks are used the system. It also provides an insight on how the tests have been performed.
In the section 6 are provided installation instructions for running the code.

2 Functionalities implemented

We list here what we've implemented, what not and some slightly difference with respect to the DD and the RASD we had to insert the implementation code.
What has been implemented:

- Related the the user resource: account creation, account deletion, login, logout, account modify, token authentication access.
- Related to the calendar resource: calendar creation, deletion and modify, global preferences modify constraining the calendar ones.
- Related to the event resource: event creation, deletion and modify. Generation of recurrences according to daily, weekly, monthly and yearly recurrence rule with the possibility of deleting single recurrences.
- Related to the instructions resource: instructions creation, delete and modify according to the user schedule changes with caching to improve overall response speed.

What has not been implemented:

- We've not taken into account weather consideration, because it was not considered a core functionality to be released with the first version of our app.
- We've not taken into account extra personal vehicles besides the implemented car, bike travel mode since we were not able to find a reliable and fast to way to predict the user's personal vehicles positions according to the schedule possible changes.
- We've not taken into account the boolean setting "active" in calendars: all calendars are considered active, regardless of the choice set inside the client app.
- We've not taken into account the carbon footprint option, because it was not considered

What are some of the differences with respect to what written in the other documents:

- Since we have not found public APIs for Mobike we treated it as bicycling.
- Since we have not found public APIs for Enjoy we treated it as driving.
- Because of the impossibility to have access to the APIs listed above we do not show client side the location of Mobike bikes and Enjoy cars.
- Handling data consistency and atomicity turned out to be very difficult and we had to tackle this by because during development issues arised regarding the event sourcing protocol and we were forced to find new solutions that were different in nature from the ones presented in the Design Document.
- The "Return to base" option when creating an event is intended to work only for events at the end of the day. Thus, a return instruction will be provided only if it's set on the last event of the day.

3 Adopted development frameworks

3.1 Programming languages

Regarding the implementation of the backend logic we've written every microservices using Python.

- Pros: it really boosted our production with its built-in functions. Great for math and when working with dates. It has a lot of external libraries that fits really well in the development of microservices.
- Cons: some run-time errors due to type checking because of Python dynamic typing nature.

The Android app has been developed with Java.

- Pros: safer thanks to being statically typed.
- Cons: more verbose than python, really bad when working with Dates without external libraries.

3.2 Middlewares

- As an application server in our microservices we used uwsgi.
    - https://uwsgi-docs.readthedocs.io/en/latest/
    - Pros: versatile and performant,
    - Cons: in a production ready environment you have to set some low level config and you may want to work at an higher level.
- For the database and the event store we used PostgreSQL.
    - https://www.postgresql.org/docs/
    - Pros: provides support for custom columns like JSON or Arrays.
- As the API Gateway we used Kong.
    - https://getkong.org/about/
    - Pros: built on top of nginx.
- Redis as an in-memory database for caching
    - https://redis.io/
    - Pros: supports for a variety of data structures
- Rabbitmq as the message broker
    - https://www.rabbitmq.com/
    - Pros: by setting it correctly it guarantees at least once delivery of message.
    - Cons: we have found the documentation to be not so well precise.

3.3 Other libraries

We list the python libraries we used for writing the backend.

- Flask for writing the http server
    - http://flask.pocoo.org/
    - Pros: micro-framework that fits really well with microservices development
    - Cons: not so much configurable
- SQLAlchemy as an object relational mapper
    - https://www.sqlalchemy.org/
    - Pros: boosts production by abstracting database tables into objects

4 Structure of the source code

In the source code folder there is the Travlendar+ folder which contains the client(the Android application) and the folders of the five different microservices that runs in the backed. In the source folder there are also two bash scripts to run the Travlendar+ system. The install script basically runs all the docker_cmd.sh scripts inside each microservice folder.

5 Tests performed

Every microservice has been first developed and tested independently from the other ones. For each microservices both unit tests and integration tests are provided. Regarding the integration ones we checked the correct behaviour when interfacing to eventual databases, event stores, in-memory database and message broker also taking into account the eventual consistency the Command Query Responsibility Segregation pattern can bring.
After each microservice was ready we then moved to system testing. We first added the API gateway and checked the correct request forwarding to the various microservices.
Once the REST API was ready and the commands were received correctly from the client side we tested the correct dispatch of the events by the message broker to keep data consistency through event sourcing between the microservices. We focused a lot on this phase by assuring that every event was correctly sent and acked with an atleast once delivery. It has also been checked the correct behaviour in case of a failure both in the events senders and consumers or in the message broker.

6 Installation instructions

For running the Travlendar+ system you need to have Docker installed:
https://docs.docker.com/engine/installation/

- Open a terminal with bash and path where you downloaded the software.
- Change directory and go inside the Implementation folder: cd /Implementation
- Run the install_script.sh: sh install_script.sh
- (This will setup also the containers we used for the tests, so you can run the tests)
- After the install script sets up everything run the kong_setup.sh script to add the API endpoints to the API gateway: sh kong_setup.sh
- You can check if everything has been setup correctly by trying to send a get request to http://localhost:8000/users/1. It should return an unauthorized message.
- You can now hit the travlendar endpoint and send commands using curl, for example.
- To connect the android app to Travlendar+ you need to know the IP of your machine in your net. After you have it you need to modify the ApiCreator class that you can find inside this folder by adding your ip as the value of the private static final String BASE_URL variable:
- cd /Travlendar+/app/src/main/java/com/polimi/travlendar/api/
- You need now to build the app and launch it on your phone. For doing this you can download android studio and open as a new project the Travlendar+ folder you find in the Implementation folder. You can now connect the phone to the computer with an usb and launch the app by Running the project.
- You need to be connected with the phone on the same net on which you've find the ip of your computer.
- You should now be able to start using the app.