Distributed Systems Assignment 2: Microservices

Tom De Schepper, Glenn Daneels, and Steven Latré

2018-2019

1 Introduction

This document discusses the second assignment for the practical part of the Distributed Systems course. The goal of this assignment is to understand the principle of microservices and to successfully apply it in practice. The assignment will need to be completed individually. In this document we will provide you with a detailed list of the features and requirements that are needed.

2 Goal

The goal of the this assignment is the construction of an application that is built using a microservice architecture. The application itself allows a registered user to rate different aspects of the De Lijn infrastructure, i.e., vehicles (i.e., bus, trams) and stops (i.e., "haltes"). In addition, the application lets a user request the average rating and per-user rating of a vehicle or stop.

3 Requirements

This section lists the different features and components that will need to be implemented in the assignment.

3.1 Functionality

The application provides a webinterface that is the only way of communication between the client and the application. It allows a client to:

- register him-/herself to the application by specifying a username, email and password
- submit new vehicles by specifying the type (bus, tram) and unique number of the vehicle (this is necessary because the API of De Lijn does not provide information about vehicles)
- remove a vehicle based on its unique number
- submit a new rating for an already-added vehicle or available stop
- show the average and per-user rating of vehicle or stop

Forms to add/remove a new vehicle or rating also feature username/password fields as only registered users should be able to use add/remove new vehicles or give ratings. This means that login functionality is not necessary. Additionaly, removing a vehicle can only be done by the user who added the vehicle, but is not allowed anymore when another user already rated the vehicle. To show the ratings of a vehicle or stop, no username/password combination is necessary: this information should be publicly available.

Stops do not need to be added/removed manually by users as this information is available through the API of De Lijn. When adding a rating for stop, the application must offer the user 3 ways of specifying a list of stops out of which the user can rate a stop. The application allows to show the following lists of stops:

- a list of all available stops
- a list of all stops on a given line ("lijn")
- a list of all stops for a given town ("gemeente")

Subsequently, the user can rate a stop from the resulting list. This list is thus provided by the back-end system and not directly by the webinterface.

3.2 Microservices

The application must be based on microservice architecture, meaning that the application should be broken apart into different smaller parts that interact and communicate with each other.

Obligatory microservices that must present in your application (among other microservices) are:

- a *web UI* microservice that allows the client to use the functionality of your application
- a *users* microservice that deals with all user management, i.e., user registration, user checking, etc.
- a *persistence* microservice that stores all application data in a well-organized manner

Keeping in mind the microservices idea, this architecture should be extended appropriately to feature the full extent of the application's functionality. This architecture needs to be provided in the manual as well.

4 Tools

You will use a number of tools, technologies, and languages to successfully implement this application. While some of these are mandatory, others can be used based on your own preferences. Below you will find a list of tools and some information:

- Docker (`https://www.docker.com/`): the most popular open source container technology available today. Docker allows you to configure a microservice per container.
- Python and Flask (`http://flask.pocoo.org/`): the application consisting out of microservices will be created in Python using the Flask micro framework.

- DeLijn API (https://data.delijn.be/signup): this needs to be used to acquire the needed stops information.
- Postgress and Flask-SQLAlchemy (http://flask-sqlalchemy.pocoo.org/2.3/): the Postgress database is used to store all the application's data. (Flask-)SQLAlchemy will be used to access and edit that data.

Luckily, how to combine all of these tools in a nicely working microservice application is described in detail on http://testdriven.io. The Part 1 tutorial provides you with an elaborate tutorial on how to start. It also explains other useful tools such as Jinja Templates, Flask Blueprints, etc. to ease the implementation of your application.

5 Minimal requirements

There are minimum requirements that your solution should comply to, in order to pass for this part of the course. If these requirements are not fulfilled, it is not possible to pass:

- A manual needs to be present in your submission or it is not possible to evaluate your solution. This manual needs to describe the architecture of your application (by text and by diagram), how we can run and test your solution and an extensive explanation of the design choices you made and which tools were used. The installation of tools not mentioned in the http://testdriven.io should be explained in your manual.
- It needs to be able to run your solution (e.g., start up your Docker containers, open the web interface, ...). Therefore, make sure that your solution is complete upon submission and no configuration files and/or libraries are missing.
- Make sure to only use web services and web communication (e.g., HTTP calls) within this assignment and no other communication methods. The communication with the database may be the exception to this as it uses built-in Flask functionality.

Remark: if your solution complies to these minimum requirements, this does not mean that you will certainly pass this assignment.

6 Deadline and submission

The deadline for this is assignment is Wednesday 12th of December at 23:59. Your solution has to be submitted through Blackboard.