Distributed System
Assignment 2 : MicroServices

Benjamin Vandersmissen and José Oramas
Benjamin.Vandersmissen@uantwerpen.be
University of Antwerp

2022-2023

1 Introduction

This document provides an overview of the requirements and background information for the second practical assignment of the Distributed Systems course. The goal of this assignment is to familiarise yourself with the microservice design pattern, decompose a problem in microservices and implement microservices using Docker/PodMan. You should solve the assignment **individually**.

2 Goal

The goal of this assignment is twofold:

- First, you should decompose a problem in microservices, i.e., you should provide a description of the functionality that each microservice has, which data storage it uses and the communications between microservices.
- Secondly, you should implement a part of the problem using the Docker/PodMan toolbox.

3 Requirements

3.1 Decomposing in Microservices

Given the following scenario, decompose in microservices in an intelligent way. This means grouping related functionality together in a single microservice and making graceful failure possible. Provide an overview of the total microservice architecture; this means which features each microservice implements, which databases a microservice has (if any) and which communication channels they have with the other microservices. In addition, you should provide an explanation for the decomposition. (i.e., why do you group those Features together in a MicroService?)

Scenario: Suppose your best friend has a bright new idea for a revolutionary website, that is guaranteed to generate billions of dollars. The goal is to fuse music streaming and social features into a single experience. Everything is prepared, even the name — Spotibook — and a logo. He only needs the following small set of features implemented.

1. A user can create a profile (Register).
2. The site can verify username & password combinations (Login)
3. A user has the ability to add other users as friends.
4. The user can view a list of all its friends.
5. The user can create playlists.
6. The user can add songs to a playlist.
7. The user can view all songs in a playlist.
8. The user can share a playlist with another user.
9. Each user has a feed that lists the last N activities of its friends. (sorted by time) Activities are : creating a playlist, adding a song to a playlist, making a friend and sharing a playlist with a friend.

3.2 Implementing Microservices

Using the microservice decomposition you made in the previous section, implement the features using Docker images. Furthermore, you should implement the databases for the microservices you use in this part.

To make things easier for you, you can use separate Docker containers for the databases, however you still need to group databases together that are meant to be together.

For example, if you decomposed the problem in two microservices *microA* and *microB*, you can implement two additional Docker containers *microA-persistence* and *microB-persistence*, but they should only contain databases associated with *microA*, *microB* respectively.

Internal (from *microA* to *microA-db*) communications *with a database* can be done using python libraries (e.g. psycopg), while **external** (from *microA* to *microB*) communications between microservices need to be done via the REST protocol.

You will be provided with a script that implements the UI and boilerplate code, so you can focus on the implementation of the microservices.

4 Tools

The following tools and links might be helpful while implementing your solution.

- REST API in python: Implementing a REST API in python is actually pretty easy using a couple of libraries. The main libraries are Flask (https://flask.palletsprojects.com/en/2.0.x/quickstart/) and Flask-RESTful (https://flask-restful.readthedocs.io/en/latest/). You should already have some familiarity with Flask from Programming Project Databases.
- Docker: a tutorial that goes over the basic functionality of Docker can be found at https://docs.docker.com/language/python/build-images/. To run multiple microservices in a simple fashion, you can use docker-compose (https://docs.docker.com/compose/). Additionally, a introductory presentation is provided on BlackBoard.
- Database: for the databases included in your microservices, you can use any database dialect you want as long as it works. However, most of you will have experience with psql (https://www.postgresql.org/) from Programming Project Databases, so feel free to use that one. Interacting with a database from Python can be done with packages such as psycopg2 (https://psycopg.org) in the case of PostgreSQL.

5 Good to know

Here you will find some general tips that can save you some time and headaches.

- Be sure to take a look at the python files accompanying the assignment, as there you can see the endpoints implemented and where in the code Features will be accessed. The only things you should change in those files are REST calls to your microservices.
- As the main focus of this project is on making you familiar with some tools and practices related to microservices, **you should not** focus on real-world problems out-of-scope. So **don't worry** about SQL injections, storing passwords securely, latency and more.
- There are Docker images online that are pre-packaged with all kinds of tools. You can always take a look at `https://hub.docker.com` to see if you find anything useful.
- There is one microservice already provided in the provided files, you can base your other microservices based on that one.
- You can either use Docker or PodMan to run your solution, as both should work with the same files. However, PodMan has the advantage that it does not require root access, and as such your solution will always be tested using PodMan!

6 Requirements and Deliverables

The following are the minimum requirements and deliverables for a passing grade:

- You should include a way for us to easily run your solution with the inclusion of a `run.sh` that will automatically start your microservices and the user interface.
- You need to include a report for part 1 of the assignment. As a reminder, this needs to include the decomposed microservices, which features each microservice implements, which data it stores and which connections it has to other microservices.
- The report also needs to a small section for part 2 of the assignment, where for each required Feature, you list the API endpoint that implemented it and document it, similar to Assignment 1.

- Make sure that your submitted solution is complete and has no missing files. Be sure that the Docker images you make will function on another computer.

7 Deadline and Submission

The deadline for this assignment is the **16th of May** at **23:59**. Late submissions will automatically have points deducted for tardiness. In addition to the submission, you will also be required to give a small demo of around 5 minutes on the **24th of May**. (details will follow later)

Make sure that your submission has the following format and upload via Blackboard!

`DS-Assignment2-`*`Snumber-Lastname`*`.zip`