WEB AND CLOUD COMPUTING 2017

Project: Uber for Bikes

GROUP 18

Authors
THOM CARRETERO SEINHORST
ALEXANDRU TATAROV
ANASTASIA SEREBRYANNIKOVA

November 3, 2017

1 Introduction

The application will represent a map on which the positions of bikes will be shown. These bikes are used by people for transportation. Apart from the map, there is a table with the information about the bikes (the address). When a bike is taken, it's corresponding point on the map disappears (because it is no longer available). When the trip is finished the bike appears back on the map.

2 Architecture

In Figure 1, a broad overview of the architecture of the application is given.

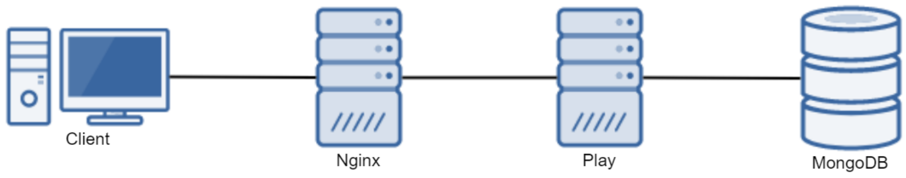

Figure 1: Overview of the architecture of Uber for Bikes

An instance of the application has several virtual machines running so that each could be placed on different hardware.

For our webserver we have written the main component in Scala. Scala is functional programming language supporting high scalability. To write not everything from scratch, we used the Play Framework.

For our project we used for our database MongoDB with ReactiveMongo. ReactiveMongo is a Scala driver that provides fully non-blocking and asynchronous I/O operations. ReactiveMongo is designed to avoid any kind of blocking request. Every operation return immediately, freeing the running thread and resuming execution when it is over.

In the frond-end AngularJS and Bootstrap is used for providing a nice user interface. The UI serves a simple purpose: Renting and unrenting bikes. The main view of the UI is a Google Maps view. It is currently set to the location of Groningen, but could eventually be used to show the region where the user resides. In our project we use an external third party service. The service of Google is needed to convert a textural representation of a location into the corresponding geographical representation of the GPS system. This conversion is done via the Google Maps API.

2.1 Docker swarm

Explain the docker swarm

3 Design decisions

3.1 Database

We use MongoDB for our database. We chose to use this database as it is easy to implement and easy to use. We have a MongoDB database with 1 collection, Bikes. The Bikes collection stores all bike locations and their availability. There should be a second database collection, Users, for storing the name, passwords and a list of rented bikes. Due to lack of time, we weren't able to implement this correctly.

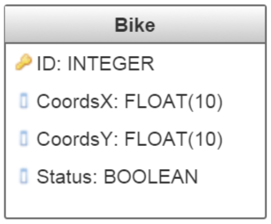

Figure 2: Overview of the database

3.2 Webserver

Why Play Framework with Scala

3.3 Client

Angular is a framework for building Web single-page applications. This is the main reason why we chose this framework. Other advantage of using Angular is its highly readable and comprehensive code. It was an important factor too, because we wanted something that will take as little time to learn as possible. We created the project using Angular Cli and used components to create the navigation bar, map component, and two (login and registration) forms (unfortunately not seen in the final layout). Even if it is not linked to framework itself, we found useful the large community of developers using Angular as online answers helped us solve most encountered problems.

4 Discussion

In this chapter we review and reflect on the project.

In the beginning of the project we started with developing an UI. We used a nginx webserver, but we struggled to put it in a docker container en connect it to the back-end. It took us some time to figure it out how exactly to connect these parts. We managed to retrieve data from the database and to present it in the UI. We chose to use AngularJS as a framework for our UI.

It took us some time to set up the back-end with Play and Scala. Since the programming language Scala was new to all of us, we had to put some effort into it to understand how Scala works.

4.1 Scala

Discuss difficulties of Scala

4.2 Docker

Explain difficulties of docker and/or swarm

4.3 Future work

One important part of the project that is missing is the login and register part for users of the application. Users should be able to create an account and to login. When a user is logged in, they can rent a bike and see their history of rented bikes. Due to the difficulties we encountered there was not enough time to implement this correctly.