universidade de aveiro
deti departamento de eletrónica, telecomunicações e informática

TQS: Product specification report

Inês Leite [92928], Orlando Macedo [94521], Óscar Fernandez [101631], Pedro Marques [92926], Rui Fernandes [92952]

1 Introduction

1.1 Overview of the project

The main objective of this project is to apply all the knowledge acquired in this course to develop a small system but rigorously following a more realistic and professional workflow with a CI and CD. Additionality integrating the test development alongside the functionality development as we go through with the project.

As for the product itself, the main focus is on a generic delivery application called WeDeliver, and a smaller example app that subscribes to the WeDeliver service, DrinkUp, a alcoholic beverage distributor.

1.2 Limitations

For now, when it comes to the mobile App, it is not possible for a rider to change/update its details, as it was originally intended. Furthermore, we also originally expected to have a statistics section in the app, which was also not implemented.
Regarding the WeDeliver backend, we intended to use reviews as a method of rating the riders with the Comment entity. Unfortunately, due to lack of time, we were not able to make this implementation.
Also, on the Admin page we didn't implement the statistics section on the dashboard.

2 Product concept

2.1 Vision statement

The WeDeliver application is a platform with the main objective of providing a fleet of riders ready to deliver any kind of package to desired locations. It is a generic service so it is meant to be subscribed to by other companies that need a delivery system but don't have a fleet of their own or means of delivery.

As an example of a subscriber to the WeDeliver application we developed DrinkUp, a system that gathers alcoholic drinks from many stores and “garrafeiras” and makes them available on a web platform for purchase. Having no built in fleet or delivery system, they subscribe and integrate their system with WeDeliver, so their drinks can be brought straight to the customer’s hands.

2.2 Personas

João is a 22 year old student of Biotechnology at the University of Aveiro. He likes going to the movies and exercising with his friends. He is originally from Coimbra so he rents a room in the central area of the city during the school year.

He owns a motorbike and rides it both for necessity to move around town, and for leisure when on his free time.

He is very proactive, hardworking and ambitious, and so he sees no problem in working in multiple fronts to achieve his goals.

Being a university student in a city away from home, he has many expenses, the tuition fees, the house rent, and all the extras for himself. Given this he often finds it necessary to work a partime job when he is not studying.

When he found WeDeliver, he noticed it fit his needs perfectly, as he could earn some money on the side, and make use of his motorbike enjoying the deliveries as leisure time too.

Manel is a 32 year old entrepreneur and businessman, he has a bachelor’s and master’s in Marketing and is married with 1 son. He is from Porto and enjoys a good cup of wine from time to time.

He has contacts with many liquor stores and in his latest business opportunity, he has conceptualized a service that compiles all the alcoholic drinks from the stores he deals with in a single platform: DrinkUp.

While developing and managing an interface could be done relatively easily, the major problem sat with delivery, because Manel intends to have these drinks delivered to their clients directly to offer convenience with his service.

Hiring enough riders to make the deliveries manageable would be very expensive, and he would also need to buy vehicles for riders' locomotion. And so he turned to WeDeliver, as they themselves offered a fleet of riders and system of delivery as a service.

Ana is 20 year old and studies Psychology at the University of Aveiro. She loves music, partying and being with her friends. She often goes to discos and bars at night, but also enjoys house parties with friends and colleagues.

She is confident, open and outgoing, and never turns down a challenge, often getting herself into drinking games.

When she, or a friend organizes house parties, it can be hard to know how much drink to buy, since many times more people than expected end up showing up.

And when the alcohol does end, it’s too late to go buy at a supermarket, so having a simple place where she could choose any bottle and have it come quickly to her door, would be the ideal situation.

2.3 Main scenarios

Scenario 1 - João registers and account and logs in

Booting up the mobile app, João goes for the registration page and inserts his credentials and details like phone number and vehicle plate. Shorty after, he can login to the app using the same credentials and start accepting orders.

Scenario 2 - João receives and accepts an order assigned to him

Using the mobile app, João can check if any new order has been assigned to him. By selecting the displayed order, he is shown some details about it and can choose to either accept his assignment or decline it. By declining, the order goes back into the assignment queue and is assigned to a new rider. By accepting, João is shown a google maps widget which provides useful directions to the store and the customer.

Scenario 3 - Ana buys an alcoholic beverage through DrinkUp and awaits delivery

Using the DrinkUp Web Application, Ana selects a bottle of vodka to put in her cart and submits her order, hoping the drink will come in time for the party she's going to. The order is sent to the WeDeliver system and any driver nearby can accept it for delivery.

Scenario 4 - Ana checks delivery state of her order

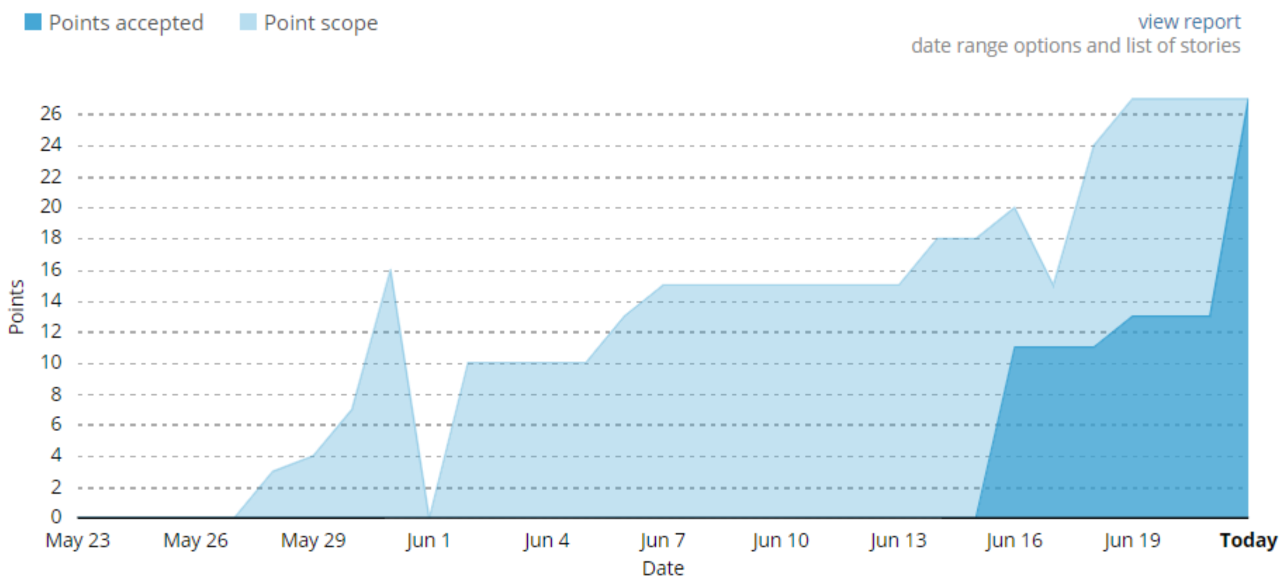
While waiting for the order, Ana checks up on the order state, to make sure a rider has already accepted it, and then when it arrives at her location.

2.4 Project epics and priorities

As our main objectives center around the WeDeliver application, most of our initial focus went into developing it's features and functionalities. To support all operations we first handled registration and login of 2 kinds of users, the riders and the customers (subscribers like DrinkUp). Subsequently we developed the main functionalities like placing orders, receiving them as riders and setting them as delivered. Finally we focused on the DrinkUp application's functionalities.

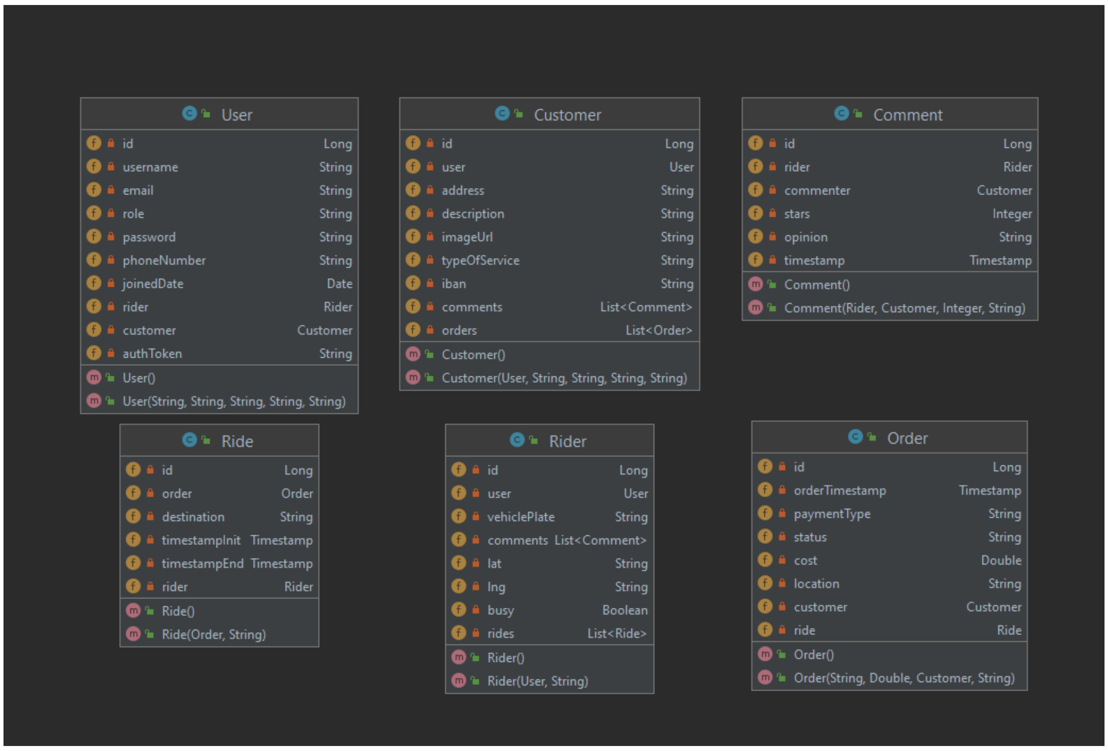
Sprint	**Epic**
1	Rider and customer registration and login, Place an order
2	Update order state, Rider delivery map, Track all deliveries
3	List items, Place a delivery order, get all orders and state from WeDeliver

Included is a burnup chart considering our story points in PivotalTracker:



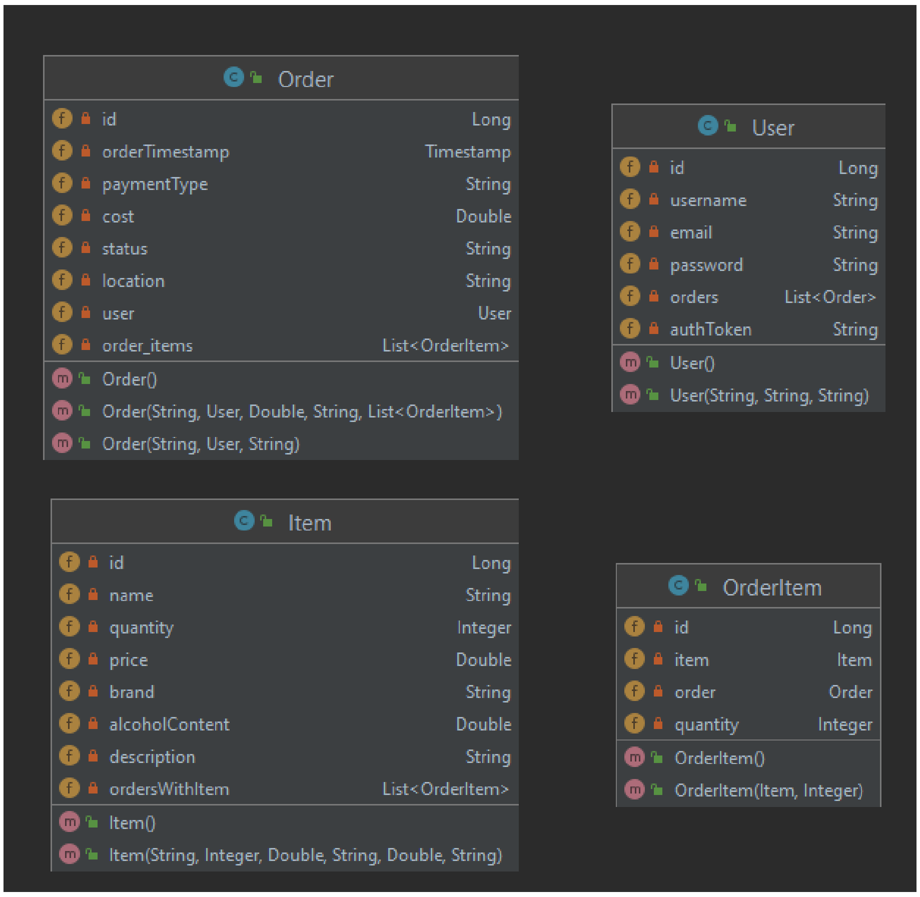
3 Domain model

WeDeliver Domain Model:



Our WeDeliver domain model was composed of 6 main entities. Both Customer and Rider had an attribute User. The User entity was developed in order to maintain general information about a specific registered app user. The Ride entity corresponds to the data about a certain delivery whereas Order adds relevant info about the customers order. Although not implemented, there was supposed to be a rating system based on the Comment entity.

DrinkUp Domain Model:



In the DrinkUp domain model, all entities are easy to understand. The User is the DrinkUp client, who places an Order. Furthermore, this Order is composed of a list of OrderItems, which in turn is composed of Items and its quantity.

4 Architecture notebook

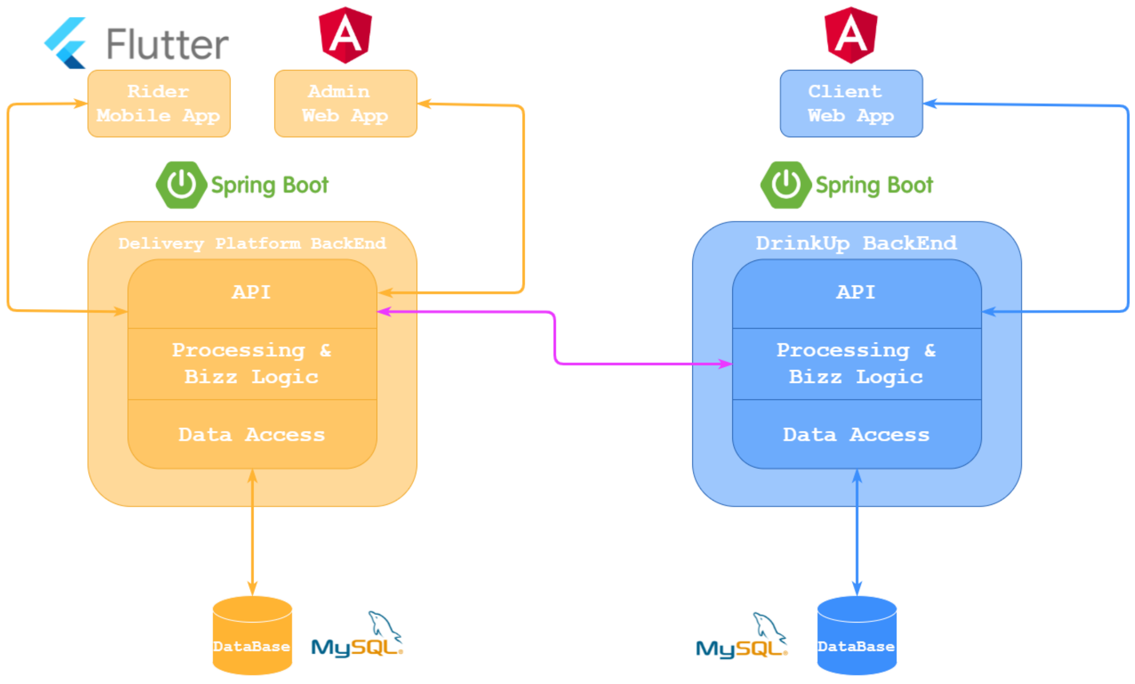
4.1 Key requirements and constraints

Our architecture choices were made setting on the following points:

➔ WeDeliver should be able to handle many customers/subscribers having an API that can receive many requests at once
➔ The system should have authentication for security reasons
➔ The system should have a specific platform for riders so they can be notified of orders
➔ The system should a platform dedicated for an administrator to check system statistics

4.2 Architectural View

When it comes to the final architecture of the project, both DrinkUp and WeDeliver backends use Spring Boot and each have their own MySQL database for storage. The DrinkUp client web app and the WeDeliver admin both use AngularJS whereas the WeDeliver rider mobile app is built with Flutter.

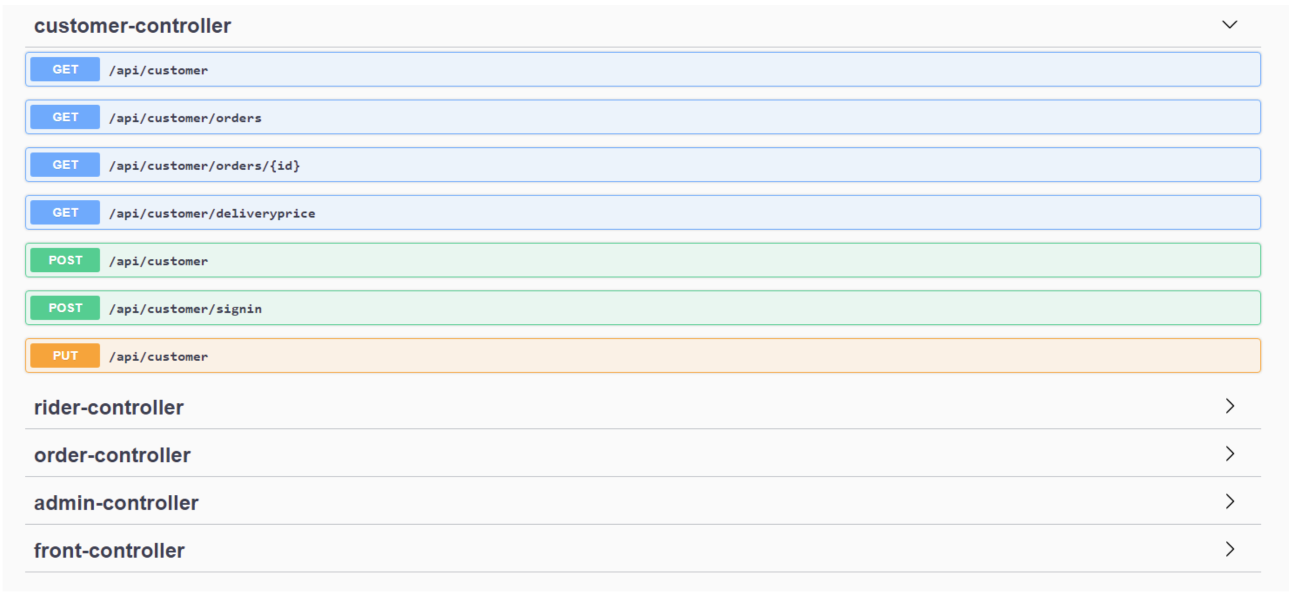


External API's were used in the development of this project. As such, we are using the Distance matrix API from Google in order to get the distance and estimated ETA of a rider to its destination as well as Google's Directions API and Maps SDK for both android and iOS in order to draw the route polylines in the rider's mobile App.
The DrinkUp client web app uses the DrinkUp backend api in order to place an order. Furthermore, this order is sent to the WeDeliver backend that generates a delivery forwarded to the riders. The riders then use the mobile app which communicates with the WeDeliver backend API in order to get their assignments and either accept or deny them. The WeDeliver admin web app also uses the API in order to display some useful information about the orders, riders and customers to the admin user.

5 API for developers

Every single endpoint is well documented in Swagger. These endpoints follow a simple structure. All API endpoints are identifiable by the **'/api/'** prefix, followed by the name of the controller.

Regarding the WeDeliver API endpoints:
The customer endpoints are mainly used by the *DrinkUp* application, whereas the admin endpoints are mostly used in the WeDeliver Admin Web App and lastly, rider endpoints are used in the *WeDeliver* Rider Mobile App.



customer-controller

GET /api/customer
GET /api/customer/orders
GET /api/customer/orders/{id}
GET /api/customer/deliveryprice
POST /api/customer
POST /api/customer/signin
PUT /api/customer

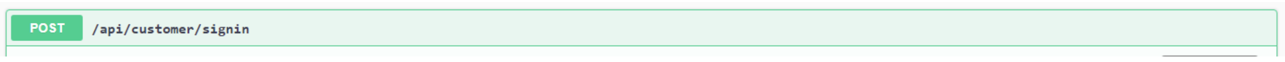
rider-controller

order-controller

admin-controller

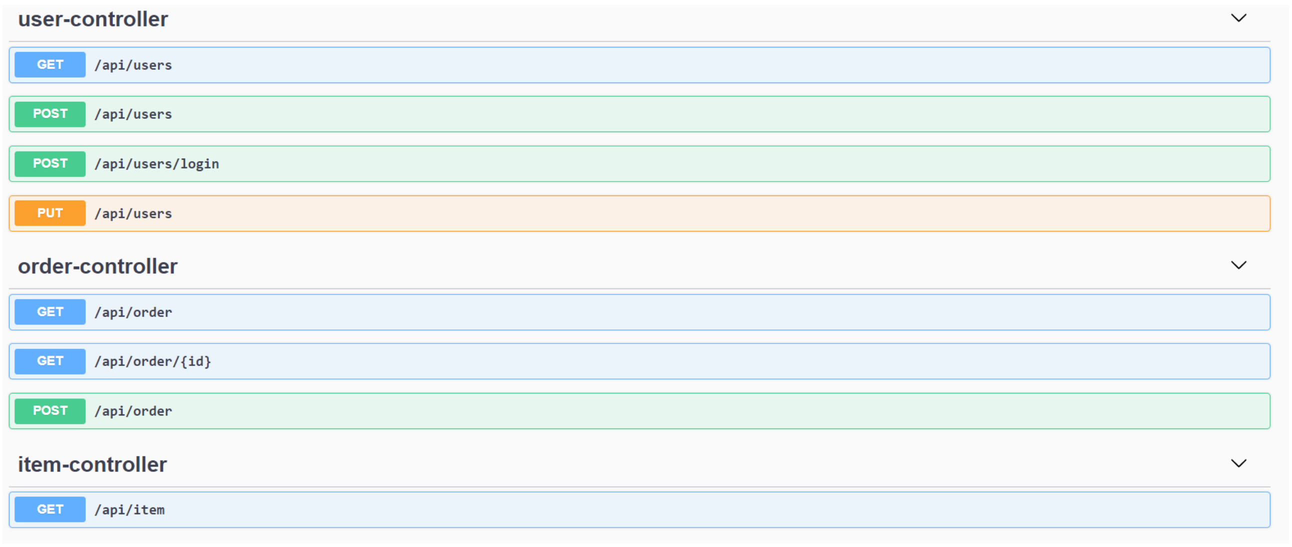
front-controller

Most endpoints are not public and in fact require user credentials for access. As such, a *login* endpoint is present and accessible to all riders, customers and admin users. Login returns either an authentication token that must be sent alongside the username for further requests or an error in case of incorrect credentials or nonexistent user.



POST /api/customer/signin

Regarding the *DrinkUp* API endpoints:
All of the endpoints are used by its Client Web App.



user-controller

GET /api/users
POST /api/users
POST /api/users/login
PUT /api/users

order-controller

GET /api/order
GET /api/order/{id}
POST /api/order

item-controller

GET /api/item

6 References and resources

- Documentation sonarcloud (static code analysis)
    - https://sonarcloud.io
- Documentation pivotal tracker (backlog management)
    - https://www.pivotaltracker.com
- Testing flutter applications
    - https://flutter.dev/docs/cookbook/testing/widget/introduction
- Dart library for mockito

    - https://pub.dev/documentation/mockito/latest/mockito/mockito-library.html
- Good practices for Spring Boot development
    - https://medium.com/the-resonant-web/spring-boot-2-0-project-structure-and-best-practices-part-2-7137bdcba7d3
- Understanding Environment Variables for Github Actions use
    - https://docs.github.com/en/github-ae@latest/actions/reference/environment-variables
- Documentation relative to Spring Boot Jpa
    - https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#reference
    - https://www.baeldung.com/spring-data-jpa-query
- Configure separate Spring DataSource for tests
    - https://www.baeldung.com/spring-testing-separate-data-source
- Separate clearly the unit test phase and the integration phase, using Jacoco Maven Plugin
    - https://www.petrikainulainen.net/programming/maven/creating-code-coverage-reports-for-unit-and-integration-tests-with-the-jacoco-maven-plugin/
- Secure a API
    - https://spring.io/guides/tutorials/spring-security-and-angular-js/
- Learn how to make deployments to Google Cloud
    - https://cloud.google.com/blog/topics/developers-practitioners/deploying-serverless-platforms-github-actions
- Learn how to test services of angular applications
    - https://angular.io/guide/testing-services
- Testing and deployment (using github actions) of angular applications
    - https://focisolutions.com/2020/04/github-actions-deploying-an-angular-app/
- Tell jacoco to ignore lombok code
    - https://www.rainerhahnekamp.com/en/ignoring-lombok-code-in-jacoco/
- Documentation jmeter
    - https://jmeter.apache.org/
- Github action to allow jmeter automation
    - https://github.com/marketplace/actions/apache-jmeter