HW1: Mid-term assignment report

*Joao Santos[110555]*, v09

[1 Introduction 1](#_Toc130550537)

[1.1 Overview of the work 1](#_Toc130550538)

[1.2 Current limitations 1](#_Toc130550539)

[2 Product specification 2](#_Toc130550540)

[2.1 Functional scope and supported interactions 2](#_Toc130550541)

[2.2 System architecture 2](#_Toc130550542)

[2.3 API for developers 2](#_Toc130550543)

[3 Quality assurance 2](#_Toc130550544)

[3.1 Overall strategy for testing 2](#_Toc130550545)

[3.2 Unit and integration testing 2](#_Toc130550546)

[3.3 Functional testing 3](#_Toc130550547)

[3.4 Code quality analysis 3](#_Toc130550548)

[3.5 Continuous integration pipeline [optional] 3](#_Toc130550549)

[4 References & resources 3](#_Toc130550550)

# Introduction

## Overview of the work

This report presents the midterm individual project required for TQS, covering both the software product features and the adopted quality assurance strategy.

The main purpose of BusConnection Application is to provide users with the ability to

Query bus connections between specific origin and/or destination locations, along with

filtering options such as departure date and currency. They can also book reservations choosing what seat they want and check them with the token generated.

## Current limitations

 <explain the known limitations 🡪 unimplemented or faulty (but expected) features>

About limitations, I tried to implement a persistence database, but I couldn´t, I´ve tried with SQL, with PostgreSQL, with hibernate but my application never connected to the database server, so I gave up and use spring hibernate non persistent database. So, whenever I shutdown the Spring Application I lose all progress on the database.

Also, I’m using spring cache to store the API information, so the cache is emptied every time Spring Application is shut downed.

# Product specification

## Functional scope and supported interactions

<functional description of the application: who (actors) will use the application and for what? Briefly explain the main **usage scenario.** >

The main usage scenario is a regular person that has interest in travelling from Lisbon to Porto by bus and wants to book a ticker online. For that the person will fill the from with the origin and destination and see the available trips. Then the person will book the reservation and reserve a seat. A token will be generated, and the person must save it. Then if the person wants to check their reservation, he must use the token previously generated.

## System architecture

<briefly present the software architecture. Include one or more diagrams.>

<detail the specific technologies/frameworks that were used>

The application can be divided into 2 big parts, front-end and back-end.  
  
The front-end was made via HTML, CSS and JS.

The backend was developed with Spring boot, and it has a controller layer to communicate with the front end. The service layer to handle the service logic and functions. The repository layer to have functions to interact with the database. The entity layer to create the entities and provide a definition for the database table. The component layer that has some data initialization and some functions to help the internal system.

Uma imagem com texto, captura de ecrã, Retângulo, design

Descrição gerada automaticamente

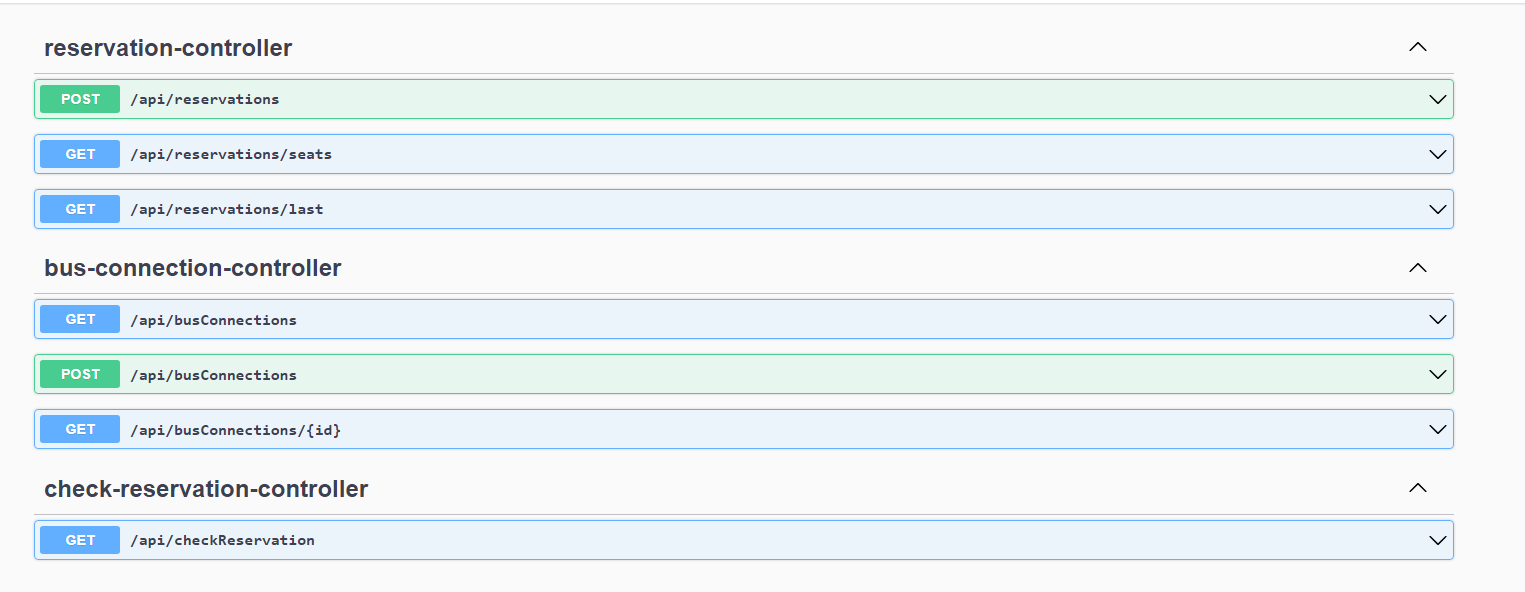
To launch the application run mvn spring-boot:run and access the port http://localhost:8080.

## API for developers

<what services/resources can a developer obtain from your project? document your API endpoints>

<note: for the homework, you are expected to expose two “groups” of endpoints:

* Problem domain: get the environmental data data by region/city, etc.
* Cache usage statistics: how many hits/misses,… >.



This API documentation can be accessed through <http://localhost:8080/swagger-ui/index.html.´>

Uma imagem com texto, captura de ecrã, file

Descrição gerada automaticamente

The schemas are also present.

# Quality assurance

## Overall strategy for testing

[what was the overall test development strategy? E.g.: did you do TDD? Did you choose to use Cucumber and BDD? Did you mix different testing tools, like REST-Assured and Cucumber?...]

Since the application was organized in layers, I used integration tests to test a single layer component to then start another one. For that I used mocks and simulate the behavior of the layers that were not implemented yet. I also used cucumber

## Unit and integration testing

As unit tests I tested some validate functions when initializing the entities such as valid phone number or email address.

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

After that I went on a multi-layer application test were I mock the behavior of some layers and test internal functions of a single layer. When I’ am assured that the layer is tested I start developing the next layer.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, software, Tipo de letra

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

## Functional testing

[which user-facing test cases did you considered? How were they implemented?]

[may add some screenshots/code snippets]

## Code quality analysis

[which tools/workflow did you use to for static code analysis? Show and interpret the results.]

[you may add some interesting lessons learned, e.g., some code smell reported by the tool that was difficult to spot and otherwise you wouldn’t address it]

For code quality analysis first used only Jacoco to see what part of the code was I covering with the tests and after most of the tests were done I used SonarCloud to see more detailed analysis on my tests results and I fixed a lot of things.

Uma imagem com texto, diagrama, número, Tipo de letra

Descrição gerada automaticamente

The security Hotspot caught on the SonarCloud is due to the exposing of the currency API key (which I don’t mind being exposed to).

The 56 maintainability issues are due to package names or public functions mostly and I don’t see that as a impactful issue.

Uma imagem com texto, número, Tipo de letra, file

Descrição gerada automaticamente

For the coverage I have a 93 % coverage which I consider being ok. When I looked up for the parts of the code that were not being covered, I didn’t noticed anything of big importance.

There were a total of 61 tests made on this application

## Continuous integration pipeline [optional]

[did you implement a CI pipeline? What was the setup? Illustrate with screenshots, if applicable]

# References & resources

Project resources

|  |  |
| --- | --- |
| **Resource:** | **URL/location:** |
| Git repository | https://github.com/JotaCLS/TQS\_110555 |
| Video demo | Video included in git repository |
| QA dashboard (online) | I runned sonar locally via docker |
| CI/CD pipeline | [**optional**; if you have th CI pipeline definition in a server, place the URL here] |
| Deployment ready to use | [**optional**; if you have the solution deployed and running in a server, place the URL here] |

Reference materials

<https://currencyapi.com> – For the currency API