

# HW1: Mid-term assignment report

*Tiago Filipe Gonçalves Pereira [108546], 2024-04-10*

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Introduction.....</b>                         | <b>1</b> |
| 1.1      | Overview of the work.....                        | 1        |
| 1.2      | Current limitations.....                         | 1        |
| <b>2</b> | <b>Product specification.....</b>                | <b>2</b> |
| 2.1      | Functional scope and supported interactions..... | 2        |
| 2.2      | System architecture.....                         | 2        |
| 2.3      | API for developers.....                          | 2        |
| <b>3</b> | <b>Quality assurance.....</b>                    | <b>2</b> |
| 3.1      | Overall strategy for testing.....                | 2        |
| 3.2      | Unit and integration testing.....                | 2        |
| 3.3      | Functional testing.....                          | 3        |
| 3.4      | Code quality analysis.....                       | 3        |
| 3.5      | Continuous integration pipeline [optional].....  | 3        |
| <b>4</b> | <b>References &amp; resources.....</b>           | <b>3</b> |

## Introduction

### 1.1 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. In this sense, I've developed a full stack application that allows for reservations of trips between european cities. Measures have been taken in order to boost comodity, such as expedition of return trips reservation, and currency exchange rates at user disposal.

### 1.2 Current limitations

All implemented features are working fine. However, due to time limitations, the seating arrangement is calculated, at the backend, meaning, no 2 reservations can choose the same seat in the same connection, but that decision is made by the backend, not the user, as there was no time to develop the needed front-end.

# Product specification

## 1.3 Functional scope and supported interactions

The ticket reservation application facilitates the booking of tickets for buses. The main actors who will use the application are:

1. **Customers:** Users who want to book tickets for their travel needs.

### Main Usage Scenario:

1. **Customer Reservation:** Customers can search for available transportation services based on origin, destination, and date. They can then select the desired service and proceed to book tickets with the expedition of their return trip, if needed.

## 1.4 System architecture

The system architecture follows a typical client-server model, with the application split into frontend and backend components.

### Frontend:

- The frontend is developed using Thymeleaf, a server-side Java template engine for web and standalone environments.
- Thymeleaf templates are rendered on the server-side and served to the client browser.

### Backend:

- The backend is built using Spring Boot, a popular Java framework for creating standalone, production-grade applications.
- It consists of a web controller responsible for rendering views using Thymeleaf and communicating with the REST API.
- The REST API handles requests from the frontend and interacts with a MySQL database to fetch or update ticket reservation information.

### Technologies/Frameworks Used:

#### 1. Frontend:

- Thymeleaf: For server-side HTML rendering.
- HTML/CSS/JavaScript: For frontend development and user interface interaction.

## 2. Backend:

- Spring Boot: For creating RESTful APIs and handling HTTP requests.
- Spring Data JPA: For interacting with the MySQL database.
- Hibernate: As the ORM (Object-Relational Mapping) tool for database operations.
- Java: As the primary programming language for backend development.

## 3. Database:

- MySQL: Relational database management system for storing ticket reservation data running on docker.



## 1.5 API for developers

### 1.5.1 API Documentation

#### 1.5.1.1 *Origin Cities*

- **Description:** Retrieves a list of available origin cities.
- **URL:** /api/v1/origins
- **Method:** GET
- **Response:**
  - Status Code: 200 OK
  - Content Type: application/json

- **Body:** List of origin cities in JSON format.

## Destination Cities

- **Description:** Retrieves a list of available destination cities.
- **URL:** /api/v1/destinations
- **Method:** GET
- **Response:**
  - Status Code: 200 OK
  - Content Type: application/json
  - Body: List of destination cities in JSON format.

## Terminals

- **Description:** Retrieves a list of terminals.
- **URL:** /api/v1/terminals
- **Method:** GET
- **Response:**
  - Status Code: 200 OK
  - Content Type: application/json
  - Body: List of terminals in JSON format.

## Connections

- **Description:** Retrieves connections between origin and destination cities.
- **URL:** /api/v1/connections
- **Method:** GET
- **Query Parameters:**
  - origin: Origin city name.
  - destination: Destination city name.
  - fromDate (optional): Start date for filtering connections.
  - toDate (optional): End date for filtering connections.
- **Response:**
  - Status Code: 200 OK
  - Content Type: application/json
  - Body: List of connections in JSON format.

## Tickets

- **Description:** Retrieves a ticket reservation by token.
- **URL:** /api/v1/tickets
- **Method:** GET

- **Query Parameters:**

- **token:** Unique token for the ticket reservation.

- **Response:**

- **Status Code:** 200 OK
- **Content Type:** application/json
- **Body:** Ticket reservation details in JSON format if found, or empty response if not found.

- **Description:** Creates a new ticket reservation.

- **URL:** /api/v1/tickets

- **Method:** POST

- **Request Body:** JSON object containing purchase form details.

- **Response:**

- **Status Code:** 200 OK if successful, 500 Internal Server Error if failed.
- **Content Type:** application/json
- **Body:** Ticket reservation details in JSON format if created successfully, or null if failed.

### Currency

- **Description:** Changes the currency used for ticket reservation.

- **URL:** /api/v1/currency

- **Method:** POST

- **Request Body:** String containing the new currency code.

- **Response:**

- **Status Code:** 200 OK
- **Content Type:** application/json
- **Body:** Exchange rate for the new currency in JSON format.

## Quality assurance

### 1.6 Overall strategy for testing

Initially did TDD. Then, due to time reasons, switched to TAD. Used Rest Assured to test the API, and have a commented test using cucumber. Didn't use it in the

final submission as it was having some problems at testing time that I didn't had time to resolve.

## 1.7 Unit and integration testing

Tested the cache behaviour, Util functions, such as calculate distance between two cities, and tested multi-services functions. This was to make sure the main parts of the code worked just as expected.

To implement this, I mocked the necessary dependencies with mockito.

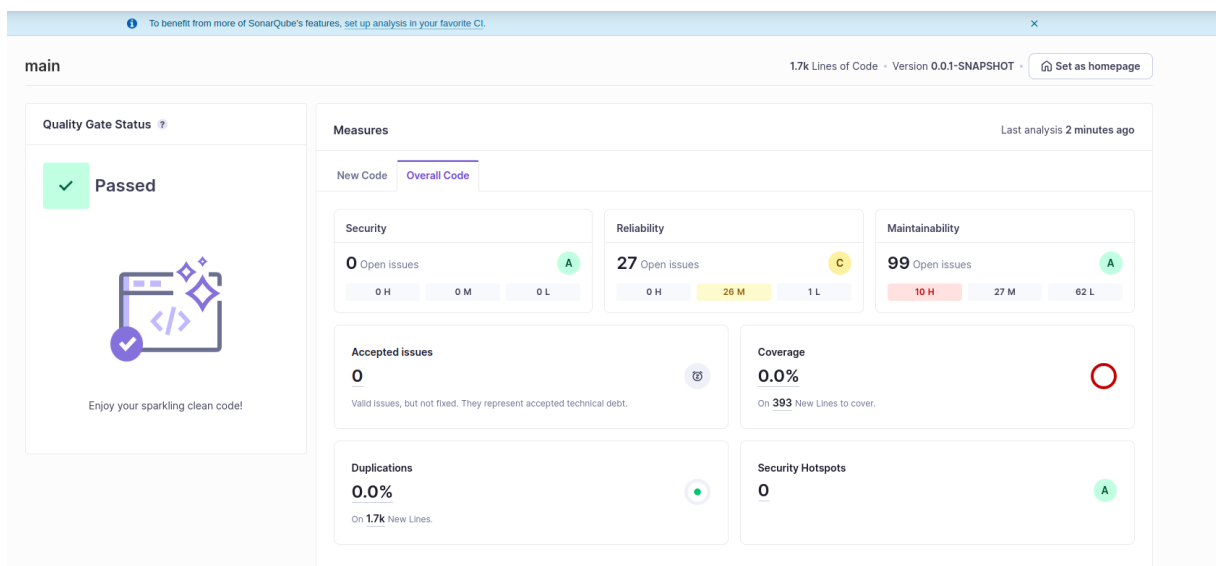
## 1.8 Functional testing

Used Testing with PageObjects and WebDrivers. Tested the main useCases: List Terminals, Buy Oneway Trip, Buy Two Way Trip. Only was left to test the check ticket after reservation use case.

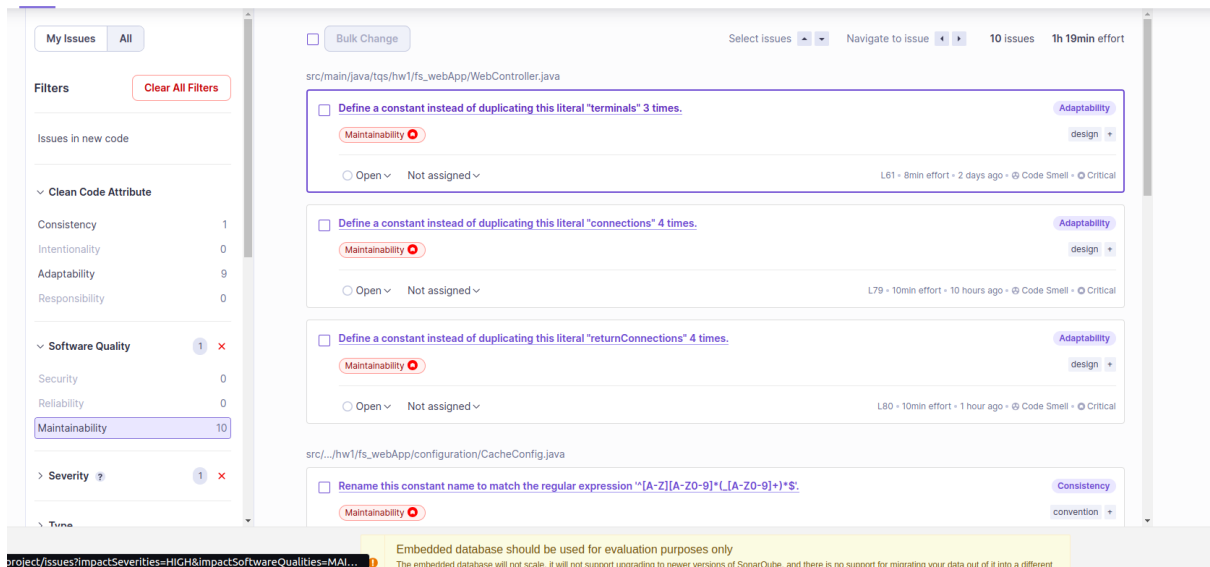
## 1.9 Code quality analysis

Tried to use sonar cloud, but due to tests not executing on Github Actions, had to resort to local scan.

Realised my coding style makes bad use of efficiently declaring variables... No other significant issues arose.



## 1.10 Continuous integration pipeline [optional]



My Issues All

Filters [Clear All Filters](#)

Issues in new code

▼ Clean Code Attribute

|                |   |
|----------------|---|
| Consistency    | 1 |
| Intentionality | 0 |
| Adaptability   | 9 |
| Responsibility | 0 |

▼ Software Quality 1 x

|                 |    |
|-----------------|----|
| Security        | 0  |
| Reliability     | 0  |
| Maintainability | 10 |

> Severity ? 1 x

▼ Tune

Bulk Change

Select issues [-](#) [+](#) Navigate to issue [1](#) [2](#) 10 issues 1h 19min effort

src/main/java/tqs/hw1/fs\_webApp/WebController.java

☐ Define a constant instead of duplicating this literal "terminals" 3 times. [Adaptability](#)

Maintainability [-](#) [+](#) design [-](#) [+](#)

☐ Open ☐ Not assigned ☐ Critical

L61 • 8min effort • 2 days ago • @ Code Smell • @ Critical

☐ Define a constant instead of duplicating this literal "connections" 4 times. [Adaptability](#)

Maintainability [-](#) [+](#) design [-](#) [+](#)

☐ Open ☐ Not assigned ☐ Critical

L79 • 10min effort • 10 hours ago • @ Code Smell • @ Critical

☐ Define a constant instead of duplicating this literal "returnConnections" 4 times. [Adaptability](#)

Maintainability [-](#) [+](#) design [-](#) [+](#)

☐ Open ☐ Not assigned ☐ Critical

L80 • 10min effort • 1 hour ago • @ Code Smell • @ Critical

src/.../hw1/fs\_webApp/configuration/CacheConfig.java

☐ Rename this constant name to match the regular expression "[A-Z][A-Z0-9]\*([A-Z0-9]+)\*\$". [Consistency](#)

Maintainability [-](#) [+](#) convention [-](#) [+](#)

project/issues?impactSeverities=HIGH&impactSoftwareQualities=MAI...

Embedded database should be used for evaluation purposes only  
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different

Used Github Actions. Had some problems with all test succeeding on the actions environment which didn't allow for me to use sonar cloud.

| 47 workflow runs |   | Event ▾   | Status ▾             | Branch ▾                 | Actor ▾ |
|------------------|---|---|----------------------|--------------------------|---------|
| ❌                | hw1 - fix verify script                             | Java CI with Maven #48: Commit <a href="#">3717c0e</a> pushed by uTigas | <a href="#">main</a> | 16 minutes ago<br>1m 11s | ...     |
| ❌                | hw1 - fix verify script                             | SonarCloud #6: Commit <a href="#">3717c0e</a> pushed by uTigas          | <a href="#">main</a> | 16 minutes ago<br>1m 7s  | ...     |
| 🔄                | Merge branch 'main' of github.com:uTigas/TQS_108546 | Java CI with Maven #47: Commit <a href="#">1a567f5</a> pushed by uTigas | <a href="#">main</a> | 18 minutes ago<br>3s     | ...     |
| ❌                | Merge branch 'main' of github.com:uTigas/TQS_108546 | SonarCloud #5: Commit <a href="#">1a567f5</a> pushed by uTigas          | <a href="#">main</a> | 18 minutes ago<br>1m 22s | ...     |
| ❌                | Update build.yml                                    | SonarCloud #4: Commit <a href="#">259119f</a> pushed by uTigas          | <a href="#">main</a> | 24 minutes ago<br>51s    | ...     |
| 🔄                | Update build.yml                                    | Java CI with Maven #46: Commit <a href="#">259119f</a> pushed by uTigas | <a href="#">main</a> | 24 minutes ago<br>2s     | ...     |
| ❌                | Merge branch 'main' of github.com:uTigas/TQS_108546 | SonarCloud #3: Commit <a href="#">8323acc</a> pushed by uTigas          | <a href="#">main</a> | 24 minutes ago<br>47s    | ...     |
| 🔄                | Merge branch 'main' of github.com:uTigas/TQS_108546 | Java CI with Maven #45: Commit <a href="#">8323acc</a> pushed by uTigas | <a href="#">main</a> | 24 minutes ago<br>2s     | ...     |
| 🔄                | Update build.yml                                    | Java CI with Maven #44: Commit <a href="#">f128ef5</a> pushed by uTigas | <a href="#">main</a> | 40 minutes ago<br>2s     | ...     |

Code

Blame

34 lines (32 loc) · 899 Bytes

Code 55% faster with GitHub Copilot

```

1  name: Java CI with Maven
2
3  on:
4    push:
5      branches: [ "main" ]
6    pull_request:
7      branches: [ "main" ]
8
9  jobs:
10   build:
11     if: "contains(github.event.head_commit.message, 'hw1')"
12     defaults:
13       run:
14         working-directory: hw1
15     runs-on: ubuntu-latest
16     steps:
17       - uses: actions/checkout@v3
18       - name: Set up JDK 17
19         uses: actions/setup-java@v3
20         with:
21           java-version: '17'
22           distribution: 'temurin'
23           cache: maven
24       - name: Install Google Chrome
25         run: |
26           wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
27           sudo dpkg -i google-chrome-stable_current_amd64.deb
28           sudo apt-get install -f
29       - name: List Directory Files for Debug
30         run: ls -a
31       - name: Set executable permissions for compile.sh
32         run: chmod +x test.sh
33       - name: Run Application with testing
34         run: ./test.sh mysql hw1

```

# References & resources

## Project resources

| Resource: | URL/location: |
|-----------|---------------|
|-----------|---------------|



|                         |  |
|-------------------------|--|
| Git repository          | <a href="https://github.com/uTigas/TQS_108546">https://github.com/uTigas/TQS_108546</a>                      |
| Video demo              | <a href="https://youtu.be/aijHXUw5FDg">https://youtu.be/aijHXUw5FDg</a>                                      |
| QA dashboard (online)   | <b>[optional]</b> ; if you have a quality dashboard available online (e.g.: sonarcloud), place the URL here] |
| CI/CD pipeline          | <a href="https://github.com/uTigas/TQS_108546/actions">https://github.com/uTigas/TQS_108546/actions</a>      |
| Deployment ready to use | <b>[optional]</b> ; if you have the solution deployed and running in a server, place the URL here]           |

## Reference materials