HW1: Mid-term assignment report

*Vasco Miguel Fernandes Faria [107323]*, v2024-04-09

[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 application BusPeak is crafted to streamline the process of purchasing tickets for trips between major cities on the Iberian Peninsula. Its primary purpose is to facilitate customers in conveniently acquiring tickets for their travel needs.

## Current limitations

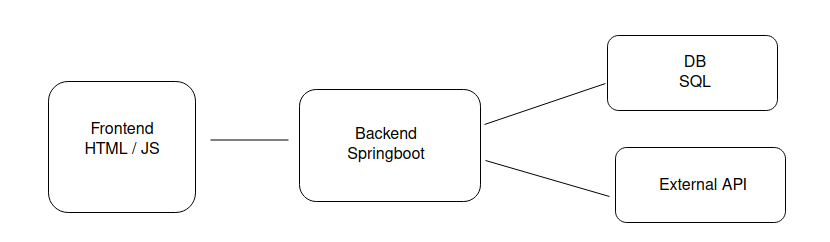
The application relies solely on one API as a data source for currency exchange rates, which poses a risk in case of failure of the API. Although these values are cached, the exchange rate values have a limited duration of validity.

# Product specification

## Functional scope and supported interactions

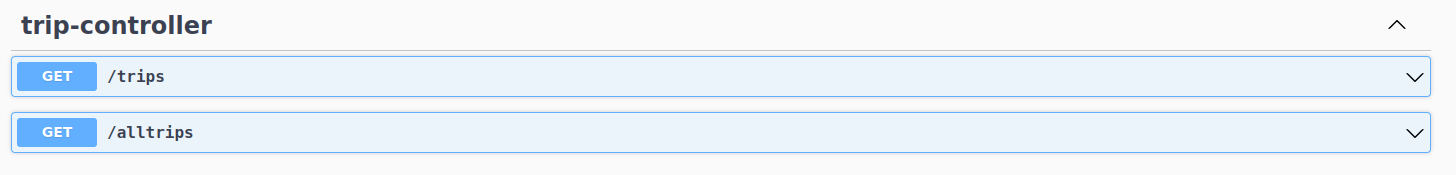
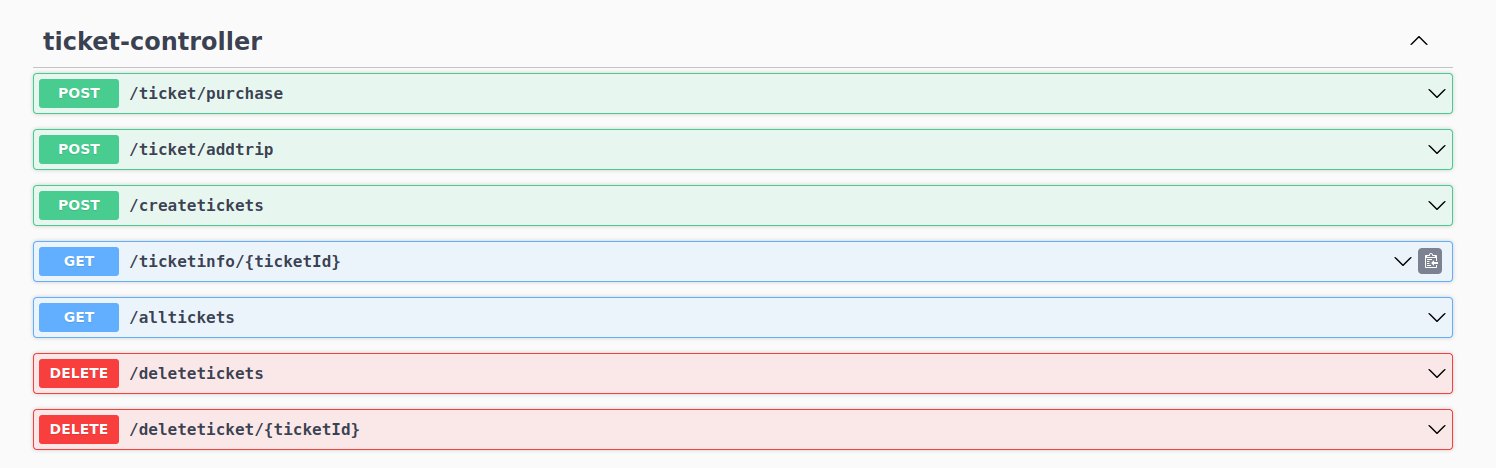
Users can purchase tickets to travel between 2 cities by bus, needing to choose the cities and date, the most comfortable bus for the trip, and finally provide the information to make the ticket purchase.

## System architecture

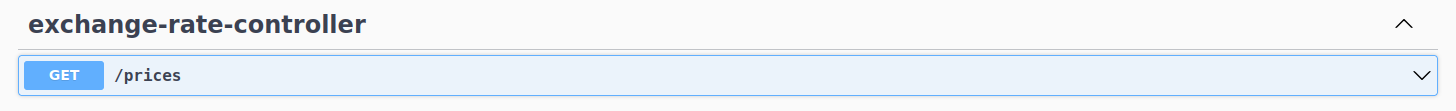


## API for developers

Problem Domain



Exchange Rate Values (Cache):



# Quality assurance

## Overall strategy for testing

Although the code development preceded the tests, the goal was always to use TDD (Test-Driven Development).

## Unit and integration testing

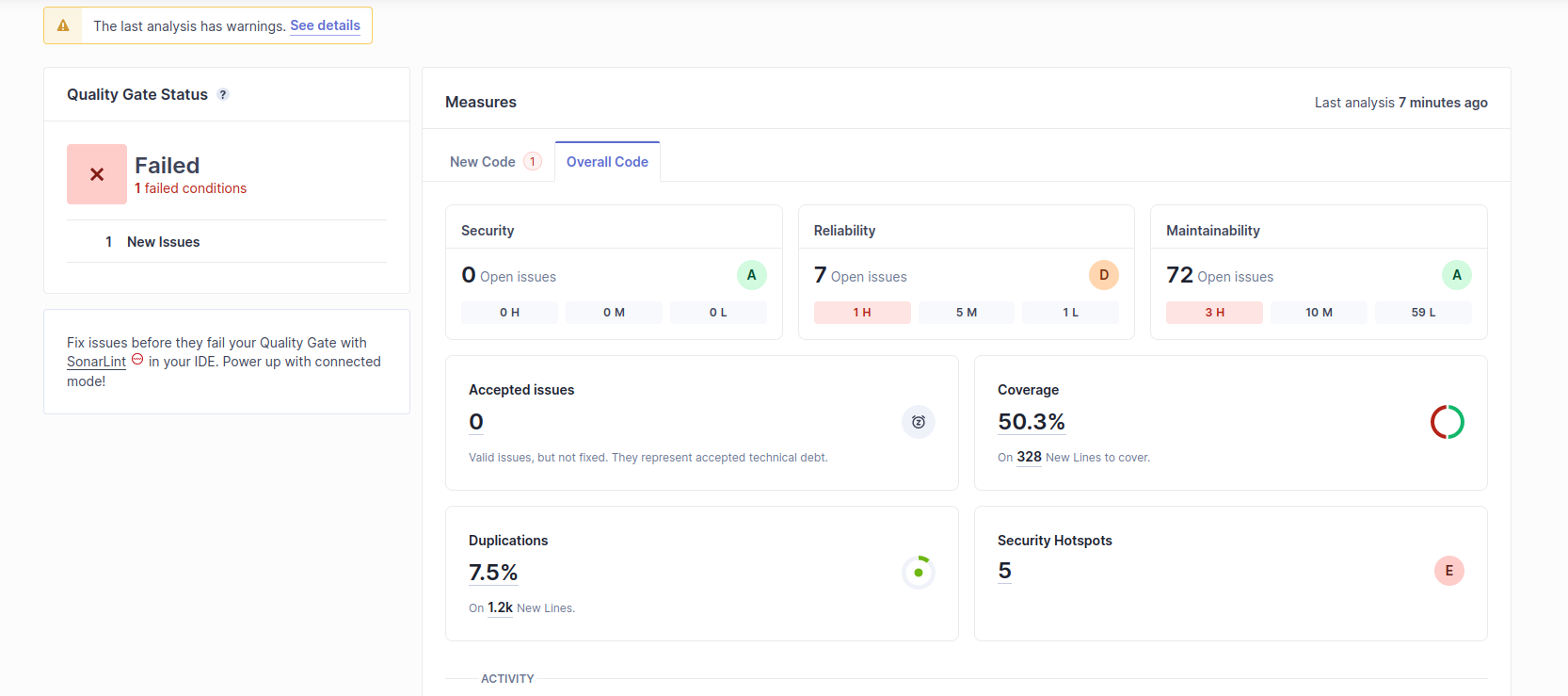
For unit and integration testing, we used JUnit 5, Mockito, and Spring Boot MockMVC.

## Functional testing

Selenium tests were used, using the Selenium WebDriver, with the Chrome Driver.

## Code quality analysis

SonarCube for code evaluation.



## Continuous integration pipeline [optional]

No

# References & resources

Project resources

|  |  |
| --- | --- |
| **Resource:** | **URL/location:** |
| Git repository | https://github.com/Vasco-Faria/TQS\_107323 |
| Video demo | < short video demonstration of your solution; consider including in the Git repository> |
|  |  |
|  |  |
|  |  |

Reference materials

https://www.exchangerate-api.com/