HW1: Mid-term assignment report

Ana Alexandra Antunes [876543], v2024-04-09

1 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.
The application under review is named TicketBus and is a relatively simple full-stack (webpage and API) for the search and reservation of bus trips.

1.2 Current limitations

The project's current limitations are:

- No ability to see/cancel/modify a user's previous reservations.
- The webpage does not display already occupied seats.
- No facilitation for return trips/regular trips.
- Small sample size.
- No administrative interface/endpoints.
- Barebones webpage design.
- No prices/exchange rates.

- Lack of logging.

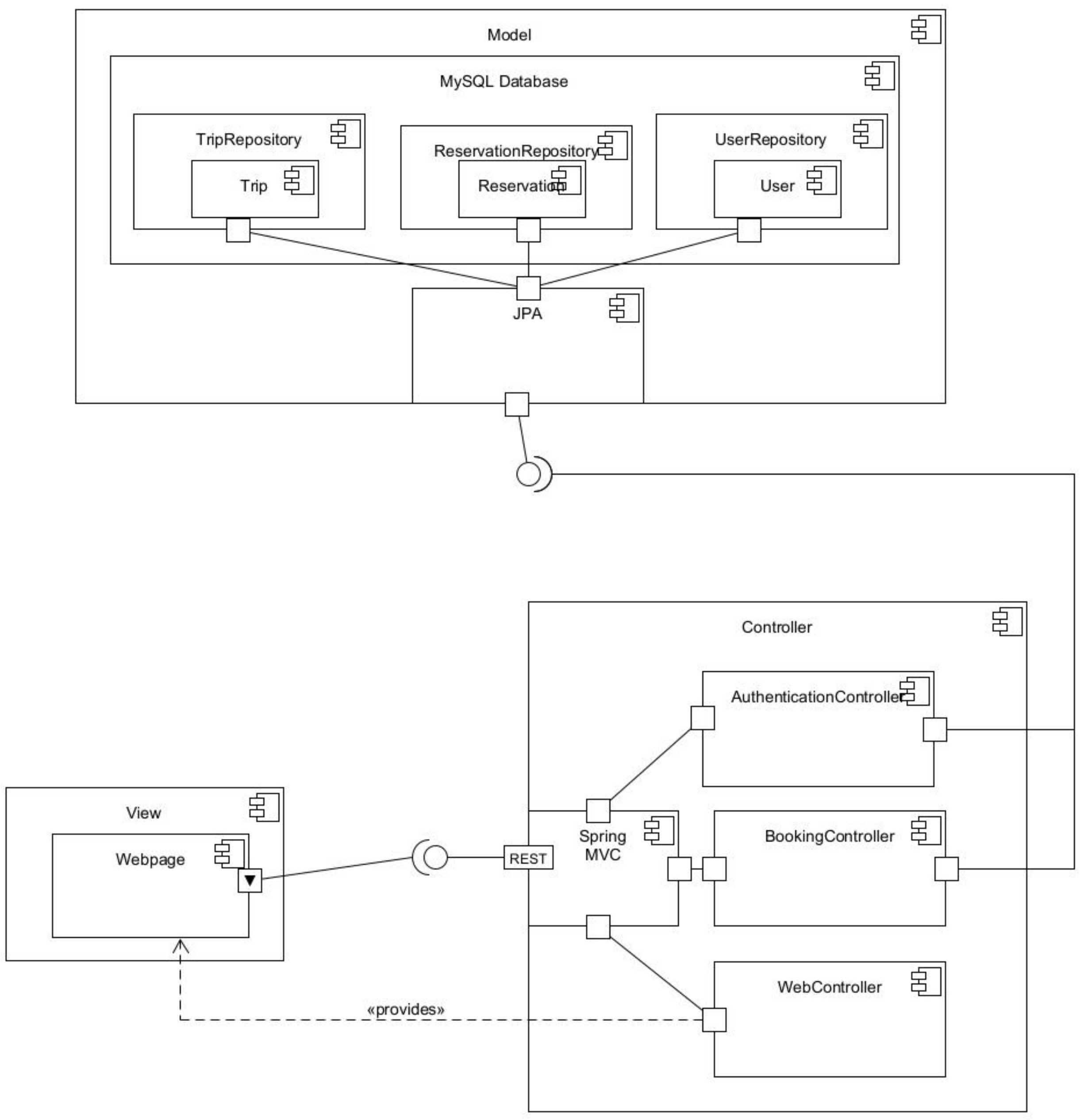
2 Product specification

2.1 Functional scope and supported interactions

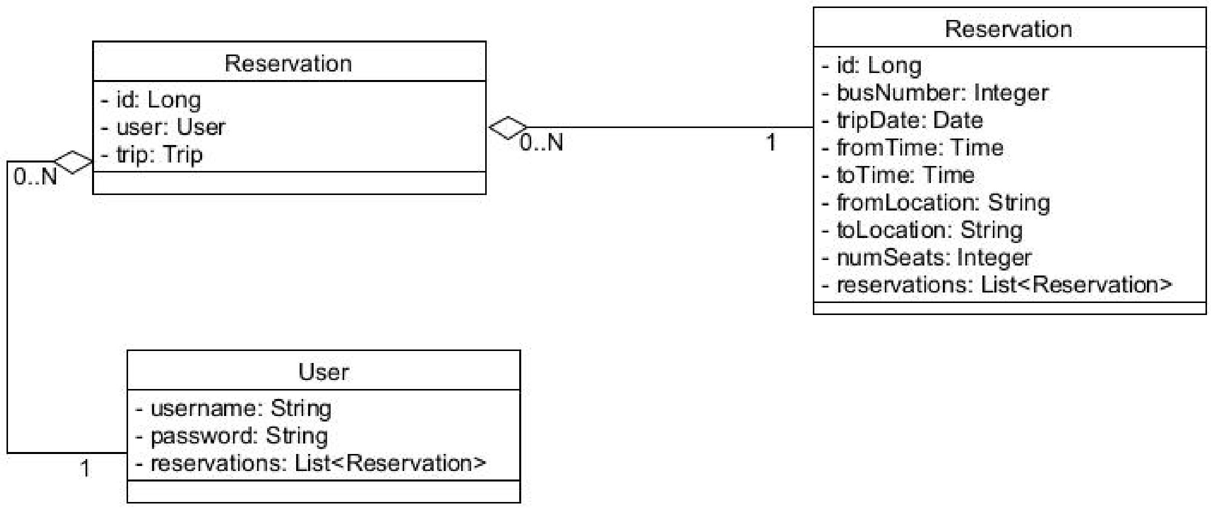
The application's webpage is mostly meant to be used by the "common person", while still providing an interface for more advanced users who wish to make automated tools or make their own webpage.
In both cases, the main usage scenario is to search for bus trips and reserve a seat in one.

2.2 System architecture

Architecture Diagram:



Entity Class Diagram:



Technologies used:

- Runtime:
  - Spring boot: main engine behind the website.
  - JPA: abstraction to access the database.
  - MySql: databased used for storage.
  - Spring Caching: allows the use and abstracts the caching of results
  - Guava: framework used for hashing passwords
  - Lombok: used to reduced the need to write boilerplate code.
  - Flyway: allows easy versioning and setup for databases
  - Thymeleaf: used to generate html templates
- Testing:
  - RestAssured: simplifies testing controllers.
  - H2: in-memory database for testing
  - TestContainers: allows to use a mysql container for integration testing
  - Selenium: driver for website testing
  - Junit: main testing framework

2.3 API for developers

The developer API is available here.

3 Quality assurance

3.1 Overall strategy for testing

While initially a BDD approach using cucumber was attempted, due to difficulties in coordinating all the different dependencies, this idea was discarded in favor of a TDD approach.

3.2 Unit and integration testing

Unit tests were written for all 3 layers of the API (services, controllers and repositories), covering almost all decision paths. The controller tests were written before any code (except entities) to define the needed functionality, followed by the actual code and the remaining tests.
The main focus of the integration tests were to check the functionality of the web browser, testing the same test cases as the unit tests, but interacting through the browser instead.

3.3 Functional testing

The main test cases were:

- Logging in using both correct and incorrect credentials.
- Registering as a new/existing user.
- Listing all trips.
- Searching for trips with parameters.
- Reserving a free/occupied seat.

3.4 Code quality analysis

For static code analysis, the SonarLint VsCode extension, which analyses the code in real time and reports any bugs/code smells. As it follows the same rules as SonarCube/SonarCloud, it easily helped remove any issues as they were being written.
In terms of results, I have achieved a total of 86.7% coverage (and most of the uncovered code is constructors and other boilerplate code), and have no duplications, issues or security hotspots.

3.5 Continuous integration pipeline [optional]

A CI/CD pipeline was implemented using Github Actions, which automatically runs the tests (except for selenium integration tests, due to missing drivers), scans the project and publishes the results to SonarCloud. While it helped check the coverage, SonarLint already identified all issues before committing.

4 References & resources

Project resources

Resource:	**URL/location:**
Git repository	https://github.com/BrunoPascoa76/TQS_2023_2024/
Video demo	https://github.com/BrunoPascoa76/TQS_2023_2024/blob/main/HW1/docs/Demo.mp4
QA dashboard (online)	https://sonarcloud.io/project/overview?id=BrunoPascoa76_TQS_2023_2024
CI/CD pipeline	https://github.com/BrunoPascoa76/TQS_2023_2024/actions