HW1: Mid-term assignment

I. Oliveira, v2024-03-19

Objectives

Develop a simple **full-stack web application** supplied with automated tests.
This is an **individual** assignment.

Required elements

Project scope

Your application should provide a services API for bus tickets selling, implementing, at least, two use cases[1]:

- Search for bus connections (trips) between two cities.
- Book a Reservation for a passenger.

In addition to the backend, you should also create a simplified web app to demonstrate the core use cases.
Variations on the theme are acceptable and welcome, given you meet the mandatory requirements presented next.

The project must include**:**

1. Your own **API (REST)** that can be invoked by external clients. Your API should allow one to programmatically search the backend for trips (direct bus connections between two major cities), create and check reservations.
2. A minimalist **web app/page** which allows users to enter/select the origin, destination, and date to search for possible trips. Then the user should pick one and provide passenger details. The web app is expected to be simple: use a limited number of “cities”, no authentication, etc.
3. Prices should be given in the currency preferred by the user (reflecting current exchange rates). You should **get the current exchange rate from an on-line resource** (and *cache* them for optimization; the *cache* should implement a Time-to-Live logic).
4. No user authentication is required for this project (web and API). Consider adding a “token”/code to the Reservation to facilitate later queries.
5. Use some logger support (so that your solution produces a useful log of events).

Technologies stack

The solution should be based on **Spring Boot for the services/backend**. The web (presentation) layer can be implemented with any HTML/JavaScript framework or using a templating system that integrates with Spring Boot (e.g.: thymeleaf).

[1] For additional business context you may check, for example, FlixBus, though you are expected a simplified demonstrator (compare with BlazeDemo).

Tests to implement

The project should include different types of tests:

A) Unit tests as applicable (suggestion: booking logic (is bus complete?); *cache* behavior; are there utils for "validators"/"converters"?).

B) Service level tests, with dependency isolation using *mocks* (suggestion: test with isolation from external data provider).

C) Integration tests on your own API (suggestion Spring Boot MockMvc and/or REST-Assured).

D) Functional testing (on the web interface). Suggested technology: BDD with Selenium WebDriver.

Quality metrics

The project should include code quality metrics (e.g., from SonarQube). To do this, you should also implement:

E) Integration of analysis with Sonar Qube (or Codacy). Note: If the Git project is public, it can be analyzed in SonarCloud.

Extra points (optional)

For extra points:

F) Extend the logic for booking for a more realistic business case, eg: expedite the search/booking of return trips; consider different types of seats (priority…); work with specific seats numbers.

G) Continuous Integration pipeline with the automation of testing and static code analysis (e.g.: GitHub Actions, GitLab CI/CD).

Submission

The submission consists of:

1. A brief **Technical Report** should **explain the strategy** that was adopted (your options as a developer) and offer **evidence** of the results obtained (e.g.: which testes per testing level, screenshots of the representative steps, (small) code snippets of the key parts, screenshots with the test results, etc.). The results of the SonarQube dashboard should be included (and discussed). [A report template is available.]
2. The **code project**, committed to your TQS personal Git repository (folder /HW1).

   Besides the code, be sure to include in the repository a short video with a demonstration of your solution (or a link to the video, instead, if the video is a large file).
3. **Oral presentation**. To be scheduled → book a slot.