HW1: Mid-term assignment

I. Oliveira, v2025-10-22

Objectives

Develop a simple **full-stack web application** and a supporting test automation strategy.
This is an **individual** assignment.

Required elements

Project scope

The MonosClean provides garbage collection services for multiple municipalities and is now implementing a system to **allow citizens to book the collection of large bulky waste items** (such as mattress, old appliances,…). For better customer satisfaction, the company wants a web portal and a mobile app to allow citizens' self-booking.
The solution should support the following use cases:

- A citizen can request the collecting of items for a specific date and approximate time slot. A basic description of the item(s) should be provided.
- For correct assignment of work lists, the citizen should pick its municipality from a closed list. This should be obtained from an official resource.
- Upon successful booking, the user gets a token to enable further inquiries (or even cancelation) of the service booking. Check details for an active reservation (and optionally cancel it).
- Each service request goes through different states (received, assigned, in progress,…); the evolution of states is timestamped and displayed in the web page
- The system should not accept bookings that go beyond reasonable service level limits or resource capabilities.
- The Staff can see the service requests and update their state.

You should implement a backend (exposing a REST API) and a simplified web app to demonstrate the core user stories.
Variations on the subject 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 service requests, create and check service bookings.
2. A minimalist **web app/page** which allows users to enter new garbage collection requests.
3. The location (municipality) is mandatory and should be selected from a list. You should **get the valid municipalities from an on-line resource/API** (programmatically).
4. Use a service ticket code (=token) to identify the reservation to facilitate later queries (check the reservation status given the access token).
5. No user authentication is required for this project (web and API). You may opt to add (or "stub") user management services, but it is not mandatory. The "citizen facing" and "staff facing" services can be available from different pages.

6. 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[1].
Keep in mind that, for this assignment, it is much more important to have a robust backend than investing (too much effort…) in the frontend.

Tests to implement

The project should include different types of tests:

A) Unit tests as applicable [suggestion: booking logic (Is there service in that day? Are there resources available?...)].
B) Service 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.
E) Performance tests.

Quality metrics

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

F) Integration of SonarQube code analysis. Note: If the Git project is public, it can be analyzed in SonarCloud.

Extra points (optional)

For extra credits:

G) Add more entities to your data model (e.g.: manage work lists and assign work to employees).
H) Implement an "operations dashboard" for the Staff, enabling a visual monitoring of requests and operations for each municipality.
I) Implement Quality Gate enforcement in a continuous integration (CI) workflow.

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 tests 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 will be available.]
2. The **code project**, committed to your TQS personal Git repository (under folder <my repo>/**HW1**).

[1] For web applications, Spring Boot supports the integration with templating systems (e.g.: thymeleaf). However, this was not explored in classes and your are not required to use.

Besides the code, be sure to include in the repository a short video (screencast) with a demonstration of your solution.

3. **Oral presentation**. To be scheduled → book a slot.