# HW1: Mid-term assignment

*I. Oliveira*, v2025-03-26

## Objectives

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

## Required elements

### Project scope

The application will manage meals booking in the Moliceiro University campus. There are a few food facilities available, but they are somewhat apart, and students make their decisions also based on weather forecasts.
Your application should provide at least the following use cases:

- View meals planned for a selected restaurant unit for the upcoming days. (Also show the weather forecast for each day).
- Book a reservation in advance (and get a code).
- Check details for an active reservation (and optionally cancel it).
- Food services workers should be able to verify a reservation (and mark the code as used)

You should implement a backend, exposing a REST API, and a simplified web app to demonstrate the core use cases.
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 meal options, create and check reservations.
2. A minimalist **web app/page** which allows users to enter/select the restaurant, browse menus and book a meal. Meals/menus are expected to be saved in your database. Along with the menu, you should list the weather forecast for each day
3. Consider adding a “token”/ticket code to the reservation details to facilitate later queries (check the reservation status given its access token). Also implement a “check-in” support page for restaurants staff members to verify if the reservation is valid (and mark it as used).
4. You should **get the weather forecast from an on-line resource/API** (and *cache* them for optimization and save requests, using a reasonable Time-to-Live logic for its entries, after which the cached value expires).
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.
6. Use some logger support (so that your solution produces a useful log of events).
7. Include a resource in your API to monitor the behavior of the (weather requests) *cache*, e.g.: total requests, hits count, misses count.

### 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).
Keep in mind that, for this assignment, it is much more important to have a robust backend, then invest (too much…) in the frontend.

### Tests to implement

The project should include different types of tests:

A) Unit tests as applicable [suggestion: booking logic (is the facility full? Is there service in that day? Has the ticket already been used? etc); *cache* behavior; are there utils for "validators"/"converters"?].
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 (or Codacy) code analysis. Note: If the Git project is public, it can be analyzed in SonarCloud.

### Extra points (optional)

For extra credits:

G) Extend the logic of the booking service for a more realistic/worked business case, eg: book a specific menu option (which are limited), group booking,…
H) 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 (folder /HW1).

   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.