HW1: Mid-term assignment report

João Pedro Ferreira Morais [103730], v2025-04-09

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 developed system will manage meals booking in the Moliceiro University campus.
The core functionality allows users to:

- View meals planned for a selected restaurant for the upcoming days, including the weather forecast, retrieved via OpenWeatherMap.
- Book a reservation in advance (and get a code).
- Cancel an active reservation
- Verify the reservation and mark the code as used if the user is a food service worker.

The backend is developed using Spring Boot, with a REST API, and the frontend is a minimalist HTML/JavaScript interface. It also includes a caching mechanism to avoid repeated calls to the weather API.

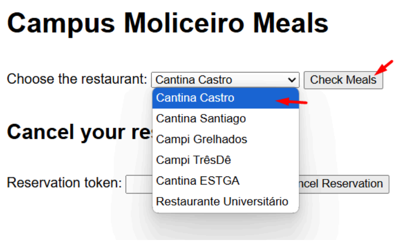
1.2 Current limitations

- Users cannot check reservation details, only workers do.
- The frontend is intentionally minimalist.
- No user authentication is implemented.

2 Product specification

2.1 Functional scope and supported interactions

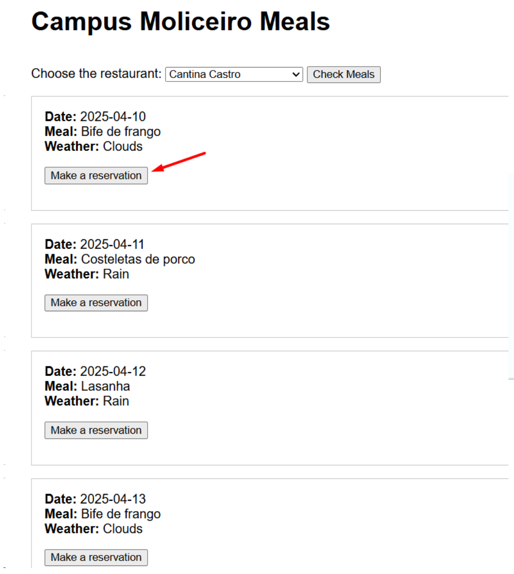
Students can select their preferred restaurant, view the available meals along with the weather forecast for each day, and proceed to book a reservation. Upon booking, they receive a unique reservation token required for check-in.


Campus Moliceiro Meals

Choose the restaurant: Cantina Castro | Check Meals

Cancel your re

Reservation token: | ncel Reservation


Campus Moliceiro Meals

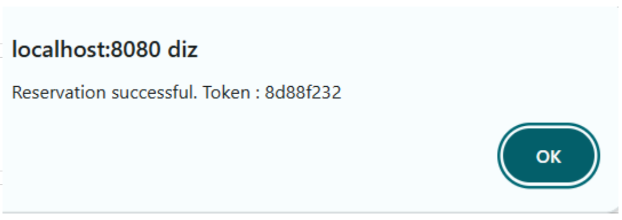
Choose the restaurant: Cantina Castro | Check Meals

Date: 2025-04-10
Meal: Bife de frango
Weather: Clouds
Make a reservation

Date: 2025-04-11
Meal: Costeletas de porco
Weather: Rain
Make a reservation

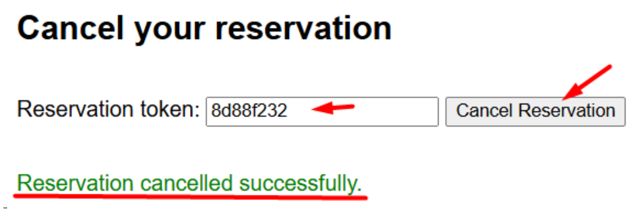
Date: 2025-04-12
Meal: Lasanha
Weather: Rain
Make a reservation

Date: 2025-04-13
Meal: Bife de frango
Weather: Clouds
Make a reservation


localhost:8080 diz

Reservation successful. Token : 8d88f232

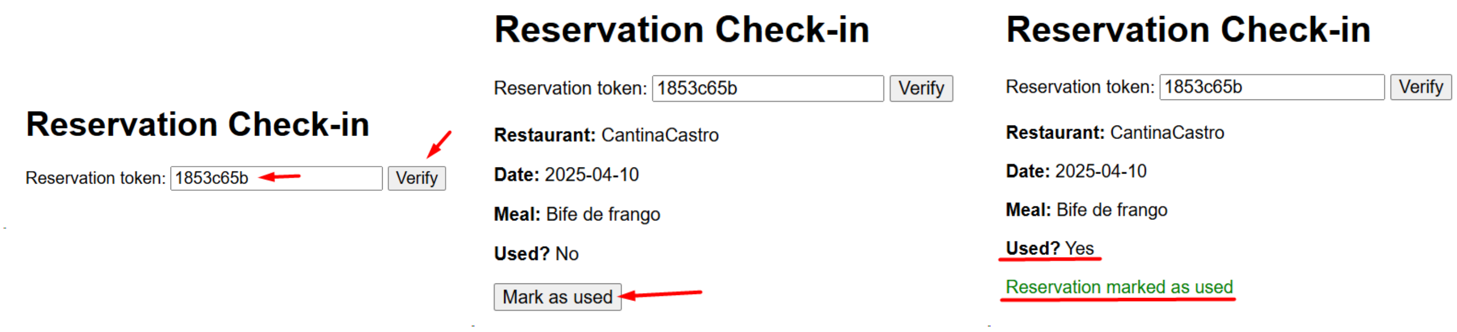
Students can also cancel their reservation by entering their token and clicking the "Cancel Reservation" button.


Cancel your reservation

Reservation token: 8d88f232 | Cancel Reservation

Reservation cancelled successfully.

Food service workers receive the reservation token and, upon verifying it, can view the booking details and check whether it has already been used. If not, they have the option to mark it as used.



Reservation Check-in

Reservation token: 1853c65b Verify

Reservation Check-in

Reservation token: 1853c65b Verify

Restaurant: CantinaCastro

Date: 2025-04-10

Meal: Bife de frango

Used? No

Mark as used

Reservation Check-in

Reservation token: 1853c65b Verify

Restaurant: CantinaCastro

Date: 2025-04-10

Meal: Bife de frango

Used? Yes

Reservation marked as used

2.2 System implementation architecture

- Backend: Java with Spring Boot
- Frontend: HTML/CSS and JavaScript
- Persistence: Spring Data JPA with entities ‘Meal’ and ‘Reservation’
- DTOs: ‘MealWithWeatherDTO’ used to encapsulate combined meal and weather data
- Services: ‘MealService’, ‘ReservationService’, ‘WeatherService’ encapsulate business logic
- Cache: Spring Cache with TTL and custom stats via ‘CacheStats’
- External API: OpenWeatherMap for weather data, consumed via ‘RestTemplate’
- Testing: JUnit, Mockito, MockMvc and Selenium WebDriver
- Data Seeder: ‘MealSeeder’ for generating test meals on startup
- Quality Analysis: SonarQube Cloud

2.3 API for developers

The project exposes a RESTful API that allows clients to interact with the core meal booking system and monitor the weather forecast caching mechanism.

Meals - MealController

- GET /api/meals?restaurant={restaurant}
    - Returns a list of upcoming meals for the selected restaurant, including the weather forecast for each day.

Reservations - ReservationController

- POST /api/reservations
    - Creates a new reservation for a given date and restaurant.

- GET /api/reservations/{token}
    - Retrieves a reservation using the given token.
- DELETE /api/reservations/{token}
    - Cancels an existing reservation by token.
- POST /api/reservations/{token}/checkin
    - Marks a reservation as used (check-in) if it hasn't already been used.

Cache Statistics - CacheController

- GET /api/cache/status
    - Returns usage statistics for the weather forecast cache.

3 Quality assurance

3.1 Overall strategy for testing

Different testing levels were combined:

- Unite tests using JUnit and Mockito.
- Service/integration tests using Spring Boot and MockMvc.
- Functional UI tests with Selenium WebDriver.
- Performance testing simulating load with a custom script.

3.2 Unit and integration testing

Unit tests were implemented for:

- MealService: tested meal/weather merging logic using mocked WeatherService.
- WeatherService: tested handling of API responses and cache logic.
- CacheStats: validated increment logic.
- DTO: tested constructors and getters/setters.

Integration tests used @SpringBootTest and MockMvc to test REST endpoints:

- MealControllerTest: tested /api/meals.
- ReservationControllerTest: tested all flows (create, get, delete, check-in).
- CacheControllerTest: tested /api/cache/status

3.3 Functional testing

Functional UI tests were implemented using Selenium WebDriver. The covered user scenarios were:

- View meals and book a reservation
- Cancel an existing reservation via token
- Verify the reservation and mark it as used

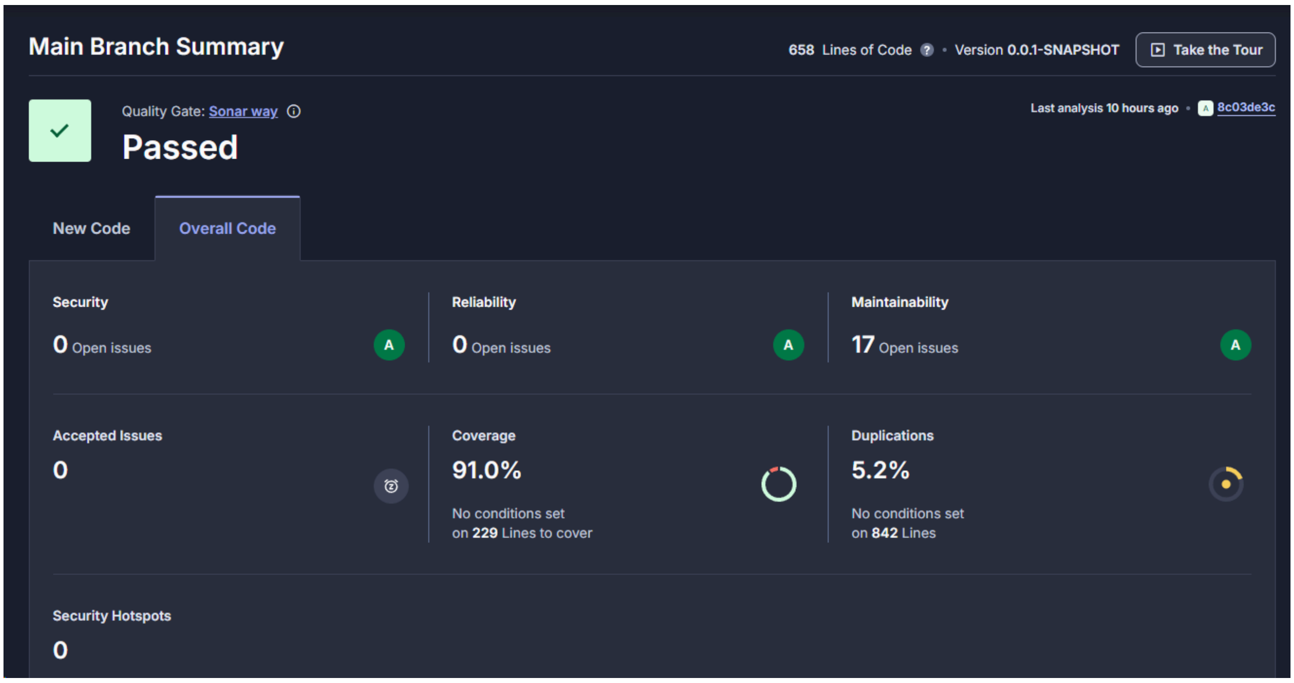
3.4 Non functional testing

In PerformanceTest.java, 100 consecutive calls to the /api/meals endpoint were simulated. The results were as following:

- 100 requests completed in under 5 seconds
- No errors or timeouts
- Response times were consistently low

3.5 Code quality analysis

Code quality was monitored using SonarQube Cloud, integrated with the project's GitHub repository. The final analysis shows that the codebase passed the quality gate with top grades in all key areas.


Main Branch Summary

658 Lines of Code · Version 0.0.1-SNAPSHOT

Quality Gate: Sonar way

Passed

Last analysis 10 hours ago · 8c03de3c

Security: 0 Open issues — A

Reliability: 0 Open issues — A

Maintainability: 17 Open issues — A

Accepted Issues: 0

Coverage: 91.0% — No conditions set on 229 Lines to cover

Duplications: 5.2% — No conditions set on 842 Lines

Security Hotspots: 0

Integrating SonarQube helped identify impactful issues, such as a potential security hotspot caused by exposing the external weather API key directly in the source code. This allowed me to apply a fix and safely externalize the key using configuration properties.

4 References & resources

Project resources

Resource:	URL/location:
Git repository	https://github.com/Bruhlin/TQS_103730/tree/main/hw1
Video demo	https://drive.google.com/file/d/1qob8eYGM2m5YwwxtZNIEQhIJWMYVCJPQ/view?usp=sharing
QA dashboard (online)	https://sonarcloud.io/summary/overall?id=Bruhlin_TQS_103730&branch=main
CI/CD pipeline	Not used
Deployment ready to use	Not used

Reference materials

https://spring.io/projects/spring-boot
https://docs.spring.io/spring-framework/reference/integration/cache.html
https://openweathermap.org/api
https://www.selenium.dev/
https://junit.org/junit5/docs/current/user-guide/#overview
https://www.baeldung.com/mockito-series
https://docs.spring.io/spring-framework/reference/testing/webtestclient.html