HW1: Mid-term assignment report

*Ana Alexandra Antunes [876543]*, v2023-03-24

[1 Introduction 1](#_Toc130550537)

[1.1 Overview of the work 1](#_Toc130550538)

[1.2 Current limitations 1](#_Toc130550539)

[2 Product specification 2](#_Toc130550540)

[2.1 Functional scope and supported interactions 2](#_Toc130550541)

[2.2 System architecture 2](#_Toc130550542)

[2.3 API for developers 2](#_Toc130550543)

[3 Quality assurance 2](#_Toc130550544)

[3.1 Overall strategy for testing 2](#_Toc130550545)

[3.2 Unit and integration testing 2](#_Toc130550546)

[3.3 Functional testing 3](#_Toc130550547)

[3.4 Code quality analysis 3](#_Toc130550548)

[3.5 Continuous integration pipeline [optional] 3](#_Toc130550549)

[4 References & resources 3](#_Toc130550550)

<All remarks like this should be removed from the final document!

This a template for the expected **content/structure**. You may use any editing tool to prepare the report (LaTeX included).

Feel free to write in Portuguese or English, but do not mix languages between headings and body…>

# Introduction

## 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.

<briefly introduce your application: name the product, if applicable; what is its purpose?>

## Current limitations

 <explain the known limitations 🡪 unimplemented or faulty (but expected) features>

# Product specification

## Functional scope and supported interactions

<functional description of the application: who (actors) will use the application and for what? Briefly explain the main **usage scenario.** >

## System architecture

<briefly present the software architecture. Include one or more diagrams.>

<detail the specific technologies/frameworks that were used>

## API for developers

<what services/resources can a developer obtain from your project? document your API endpoints>

<note: for the homework, you are expected to expose two “groups” of endpoints:

* Problem domain: get the environmental data data by region/city, etc.
* Cache usage statistics: how many hits/misses,… >.

![Graphical user interface, text, application

Description automatically generated]()

# Quality assurance

## Overall strategy for testing

[what was the overall test development strategy? E.g.: did you do TDD? Did you choose to use Cucumber and BDD? Did you mix different testing tools, like REST-Assured and Cucumber?...]

## Unit and integration testing

[where did you use unit and integration test? for what? which was the implementation strategy?]

[may add some screenshots/code snippets for clarification]

## Functional testing

[which user-facing test cases did you considered? How were they implemented?]

[may add some screenshots/code snippets]

## Code quality analysis

[which tools/workflow did you use to for static code analysis? Show and interpret the results.]

[you may add some interesting lessons learned, e.g., some code smell reported by the tool that was difficult to spot and otherwise you wouldn’t address it]  
  
I used sonarqube with docker  
  
The only problems found were code smells.  
Initially I had 21, most of them considered major but then I fixed them one by one, while redoing the code analysis for problems in the new solutions I tried to employ.  
  
  
  
I learned about awaitility, that allows for asynchronous testing. Prevously, I was using Thread.sleep but I learned it is not always reliable and should not be used when testing.  
  
One code smell that was harder to fix, was one where fetchForecast method in the service had a cognitive complexity of 19. To fix it, was not easy, the code was confusing even for me (the person who wrote it). So, I learned it’s important to write more modular and clean code.

## Continuous integration pipeline [optional]

A screenshot of a computer screen

Description automatically generated with medium confidenceI implemented a CI pipeline using GitHub Actions. The setup involved defining a job named "build" that runs on the latest version of Ubuntu and sets up JDK 18. The pipeline checks out the code from the repository, builds the project using Maven, and runs the unit tests. The pipeline is triggered by push or pull request events and ensures that the code is automatically built and tested every time a change is made to the repository.

# References & resources

Project resources

|  |  |
| --- | --- |
| **Resource:** | **URL/location:** |
| Git repository | <https://github.com/DanielFerreira011102/TQS_102885/tree/main/HW1> |
| Video demo |  |
| CI/CD pipeline | <https://github.com/DanielFerreira011102/TQS_102885/blob/main/.github/workflows> |

Reference materials

<document the key components (e.g.: libraries, API) or key references (e.g.: blog post) that were helpful and certainly **would help other students pursuing a similar work**>