TQS: Quality Assurance manual

***Ana Alexandra Antunes [876543]*, *Bruno Bernardes [876543]***

v2023-04-18

[1 Project management 1](#_Toc132723568)

[1.1 Team and roles 1](#_Toc132723569)

[1.2 Agile backlog management and work assignment 1](#_Toc132723570)

[2 Code quality management 2](#_Toc132723571)

[2.1 Guidelines for contributors (coding style) 2](#_Toc132723572)

[2.2 Code quality metrics 2](#_Toc132723573)

[3 Continuous delivery pipeline (CI/CD) 2](#_Toc132723574)

[3.1 Development workflow 2](#_Toc132723575)

[3.2 CI/CD pipeline and tools 2](#_Toc132723576)

[3.3 System observability 2](#_Toc132723577)

[3.4 Artifacts repository [Optional] 2](#_Toc132723578)

[4 Software testing 2](#_Toc132723579)

[4.1 Overall strategy for testing 2](#_Toc132723580)

[4.2 Functional testing/acceptance 3](#_Toc132723581)

[4.3 Unit tests 3](#_Toc132723582)

[4.4 System and integration testing 3](#_Toc132723583)

[4.5 Performance testing [Optional] 3](#_Toc132723584)

[This report should be written for new members coming to the project and needing to learn what are the QA practices defined. Provide concise, but informative content, allowing other software engineers to understand the practices and quickly access the resources.

Tips on the expected content, along the document, are meant to be removed.

You may use English or Portuguese; do not mix.]

# Project management

## Team and roles

## Agile backlog management and work assignment

[Description of agile practices defined in the project for backlog management (user stories oriented) and job assignment, and links to associated resources. cfr. [PivotalTracker workflow](https://www.pivotaltracker.com/help/articles/workflow_overview/) ]

# Code quality management

## Guidelines for contributors (coding style)

Among others, the main coding style we adopted was the creation of Exceptions and not ignoring them. Since our API handles a lot of resources from database who might be non existent, or sometimes parsing might throw an exception, it’s important we don’t ignore these exceptions.

Besides that, we tried to follow naming conventions for our variables and ensured variable scopes were well defined.

## Code quality metrics

The metrics we used were the code coverage of our tests, the vulnerabilities found in our code, the security hotspots, and the bugs found.

# Continuous delivery pipeline (CI/CD)

## Development workflow

We adopted the github workflow, we had our main branch and each branch created from it corresponded to a user story we defined previously. A user story was considered closed when we were able to playout the story in the solution. After a user story was closed, someone would open a pull request to the main branch. Another person would review the pull request, although must of the times it was more than one person.

## CI/CD pipeline and tools

In our CI pipeline, each pull request or push made to the main branch was subject to a set of analysis. A group of tests were run to ensure the quality of the pull requests. These tests included unit test, integration tests, among others. Besides that, each pull request and push was subject to a static code analysis using Sonar and the tests were run using Jacoco.

Regarding the CD pipeline, we weren´t able to create one for this project, only deploying the solution in a Kubernetes cluster.

## System observability

# Software testing

## Overall strategy for testing

In the beginning we followed a test driven development approach to our solution. We draw some diagrams in classes and discussed how each entity, controller would be structured.

## Functional testing/acceptance

We weren’t able to implement functional testing.

## Unit tests

We used Junit to implement our unit tests. We opted for an white box approach to develop our tests, since we knew the internal structure of our components. We ensured the all methods in the entities were returning the right values.

## System and integration testing

We had to ensure how API and its endpoints were working as expected. So for that we implemented some integration tests to ensure our controllers and services returned the correct values.