TQS: Quality Assurance manual

**Bruno Moura (97151), João Teles (104360), Marcus Peterson (), Victor Melo ()**

v2023-05-01

[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)

[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)

# Project management

## Team and roles

**Team Leader** ➝ João Teles

Ensure that there is a fair distribution of tasks and that members work according to the plan. Actively promote the best collaboration in the team and take the initiative to address problems that may arise. Ensure that the requested project outcomes are delivered in time.

**QA Engineer ➝** Marcus Peterson

Responsible, in articulation with other roles, to promote the quality assurance practices and put in practice instruments to measure que quality of the deployment. Monitors that team follows agreed QA practices.

**DevOps Master ➝** Bruno Moura

Responsible for the (development and production) infrastructure and required configurations. Ensures that the development framework works properly. Leads the preparing the deployment machine(s)/containers, git repository, cloud infrastructure, databases operations, etc.

**Product Owner** **➝** Victor Melo

Represents the interests of the stakeholders. Has a deep understand of the product and the application domain; the team will turn to the Product Owner to clarify the questions about expected product features. Should be involved in accepting the solution increments.

## Agile backlog management and work assignment

[Jira](https://amanacu.atlassian.net/jira/software/projects/AT/boards/1) is used alongside [GitHub](https://github.com/TQS-Project-Org/Main-Repo) to track issues, manage projects, automate workflows and assign tasks to team members.

# Code quality management

## Guidelines for contributors (coding style)

For Java development, the [Google Java Style](https://google.github.io/styleguide/javaguide.html) guidelines will be followed to maintain a cohesive coding style.

Similarly, [Google JavaScript Style](https://google.github.io/styleguide/jsguide.html) will be used during the development of the React applications.

## Code quality metrics

[Description of practices defined in the project for *static code analysis* and associated resources.]

[Which quality gates were defined? What was the r[ationale?]

SonarQube will be used for continuous inspection of code quality through static analysis.

The following quality gates are defined:

* Test coverage greater than 80%

# Continuous delivery pipeline (CI/CD)

## Development workflow

[Clarify the workflow adopted [e.g.. [gitflow](https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow) workflow, [github flow](https://guides.github.com/introduction/flow/) . How do they map to the user stories?]

Code review practices are integrated into the project to guarantee code quality and foster collaboration. Through peer reviews and the utilization of resources such as checklists and style guides, issues are identified, and coding standards are upheld. This approach facilitates the creation of reliable and maintainable software.

A user story is completed when the following DoD (Definition of Done) criteria are met:

* Story approved by the Product Owner.
* Unit tests passed.
* Functional tests passed.
* Code reviewed.
* Non-functional requirements met.
* Documentation written.

## CI/CD pipeline and tools

[Description of the practices defined in the project for the continuous integration of increments and associated resources. Provide details on the tools setup and config.]

[Description of practices for continuous delivery, likely to be based on *containers*]

We will utilize GitHub Actions for our CI/CD pipeline. We'll define stages like build, test, and deploy, and create YAML workflow files. Leveraging pre-built actions and customizing as needed, we'll automate tasks such as building, testing, and deploying our application. Regular monitoring and improvements will ensure efficient and reliable software delivery.

## System observability

# Software testing

## 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?...]

[it is not to write here the contents of the tests, but to explain the policies/practices adopted and generate evidence that the test results are being considered in the IC process.]

## Functional testing/acceptance

[Project policy for writing functional tests (closed box, user perspective) and associated resources.]

## Unit tests

[Project policy for writing unit tests (open box, developer perspective) and associated resources.]

## System and integration testing

[Project policy for writing integration tests (open or closed box, developer perspective) and associated resources.]

API testing