TQS: Quality Assurance manual

António Fernandes [92880], David Morais [93147], Mariana Ladeiro [92964]
v2021-06-23

Project management **1**
Team and roles 1
Agile backlog management and work assignment 1
Code quality management **2**
Guidelines for contributors (coding style) 2
Code quality metrics 2
Continuous delivery pipeline (CI/CD) **2**
Development workflow 2
CI/CD pipeline and tools 2
Software testing **2**
Overall strategy for testing 2
Functional testing/acceptance 3
Unit tests 3
System and integration testing 3

1 Project management

1.1 Team and roles

The team is composed of 3 elements: António Fernandes (QA Engineer, Developer), David Morais (DevOps Master, Developer), Mariana Ladeiro (Team Manager, Product Owner, Developer). All members are contributors to the development of the project.

1.2 Agile backlog management and work assignment

The backlog management will be task oriented and each developer will develop the tasks that will give continuity to their previous work (i.e. a developer working on the engine API will preferably continue working on the API).
This backlog will be managed on github projects at https://github.com/davidgmorais/TQSProject/projects.

2 Code quality management

2.1 Guidelines for contributors (coding style)

When working as a group it's important that all developers follow a set of rules regarding their code so the whole project can have a more homogeneous look and make it easier for any developer to work with code that wasn't developed by himself. In order to accomplish this, the standard java code style was adopted and integration tests must follow a when...then naming policy.

2.2 Code quality metrics

The code analysis was performed by sonar cloud with support for java and custom quality gates were defined to better suit the group's work.The defined quality gate consisted of a minimum of 70% code coverage for Book2Door and 80% for Engine, up to 3% of code duplication and A for Maintainability, Security and Reliability. The code smells, however insignificant, must be fixed before a merge to the dev or main branches.
The differences between the code coverage relay on the fact that Book2Door has more code that can't really be covered by tests, and thus we lowered the coverage.

3 Continuous delivery pipeline (CI/CD)

3.1 Development workflow

The group adopted a feature-branch workflow. Each feature was developed on a branch with the feature name and as soon as the feature was completed the branch was merged into the development branch after a successful pull request. In order to merge the pull request into the dev branch, other group members had to review the changes being committed and accept them. Furthermore, the new code had to pass the sonarcloud quality gates and the maven verification.

3.2 CI/CD pipeline and tools

The CI pipeline was implemented with github actions and would take effect after a pull/push to the main/dev branch. The CI pipeline would build the project and run the sonarcould and maven analysis.
As for the CD pipeline, it was also implemented using github actions but would only run when a pull/push happened to the main branch. This pipeline creates the containers for the engine and deployed them.

4 Software testing

4.1 Overall strategy for testing

The strategy for test development did not follow the TDD approach completely. However, it was used in some cases in the later stages of the development. The tools used for tests consisted mostly of Junit, Mockito, multi layer testing with spring boot, assertJ.

4.2 Functional testing/acceptance

The group didn’t have time to test the frontend with Selenium, as originally planned, however, the frontend is tested while developing and after code development is done.

4.3 Unit tests

The testing method used was the white-box policy since the developers had knowledge of the system before creating tests. This allowed for more coverage with unit tests.

4.4 System and integration testing

The API testing was done using a white-box policy making use of the spring testing resources, testing the API endpoints for status and model attributes while using Userdetails beans to access endpoints that require authentication.