TQS: Quality Assurance Manual

Hugo Ribeiro [113402], João Neto [113482], Rodrigo Abreu [113626], Eduardo Lopes [103070]

v2025-04-22

Project Management **2**
Team and Roles 2
Agile backlog management and work assignment 2
Code Quality Management **3**
Coding style guidelines 3
Code Quality Metrics 3
Continuous Integration/Delivery Pipeline (CI/CD) **3**
Software Testing **3**
Overall Testing Strategy 3

Project Management

Team and Roles

Role	**Member**
Team Coordinator	Hugo Ribeiro
Product Owner	João Neto
QA Engineer	Rodrigo Abreu
DevOps Master	Eduardo Lopes
Developer	Everyone

Agile backlog management and work assignment

For the development of our project our group adopted the Agile methodology and decided to use Jira as the project managing tool. Our Team Coordinator is responsible for assigning tasks to each team member and defining the respective story points, priority and corresponding deadlines.
Our Product Owner defined the core user stories and respective acceptance. We aim for implementing the respective user stories based on its priority, so the tasks assigned are accordingly to this.
The team adapted the following flow for the development:

- At the start of each iteration the tasks are divided by each team member based on each role and priority.
- To effectively complete a task, a member must verify if the task definition of done is satisfied. Then creates a pull request that must be reviewed for at least 2 members to be validated. Each member must carefully analyse the content before accepting/rejecting the pull request.
- All pull requests must target the ‘dev’ branch. This branch, each iteration release, is merged into the main branch.
- When a pull request is accepted the respective issue is marked as done, and it's closed.

Story points have the following meaning:

- 1 story point: 0-2h of estimate work
- 2 story points: 2-4h of estimate work
- 3 story points: 4-6h of estimate work
- 4 story points: 6-8h of estimate work
- 5 story points: 8+h of estimate work

[Falar do X-ray quando implementado]

Code Quality Management

Coding style guidelines

To ensure code consistency, readability and maintainability since the beginning, the following code guidelines were defined. For the the Maven Spring Boot Backend (in Java) the group adopted the Google Java Style Guide. For the frontend in React JS we adopted the Airbnb React/JSX Style Guide guidelines.
A github-actions workflow was set up using Google Java Format, to automatically format the code to comply with the Google Java Style automatically and ensure the guidelines are met.
SonarQube was also integrated from the beginning to detect bad smells and other issues on the code. To simplify detection, each developer installed the SonarQube IDE plugin to detect these issues before running the analysis.

[Conventional Commits??]

Code Quality Metrics

For static code analysis, we implemented a github-actions workflow that runs every time a commit is done to the main and dev branches and in every pull request.

We defined a quality gate from the beginning and ensured the project met the objectives on each iteration.

Continuous Integration/Delivery Pipeline (CI/CD)

Software Testing

Overall Testing Strategy

The QA Engineer in conformity with the rest of the members defined the following testing strategy. When it is reasonable, developers should implement Test Driven Development (TDD) with a top-down approach, starting from the controller, then the service, then the repository, and ending with integration tests. This should be implemented in a flexible way, depending on the available time and the complexity of tasks, always prioritizing the quality of the final product.
To perform functional tests the QA Engineer defined that the group would adopt a Behaviour Driven Development (BDD) approach using Cucumber and the Selenium Web Driver, to simulate the user's perspective interacting with the web app.