TQS: Quality Assurance manual

Diogo Costa[112714],Bruno Tavares[113372],Francisco Pinto[113763],André Alves[113962]
v2025-XX-XX

Contents

1 Project management

1.1 Assigned roles

Francisco Pinto- Team Leader/Developer
Diogo Costa- Product Owner/Developer

Bruno Tavares- QA Engineer/Developer
André Andre- DevOps/Developer

1.2 Backlog grooming and progress monitoring

What are the practices to organize the work in JIRA?
How is the progress tracked in a regular basis? E.g.: story points, burndown charts,…
Is there a proactive monitoring of requirements-level coverage? (with test management tools integrated in JIRA)

2 Code quality management

2.1 Team policy for the use of generative AI

Clarify the team position on the use of AI-assistants, for production and test code
Give practical advice for newcomers.
Be clear about "do"s and "Don't"s

2.2 Guidelines for contributors

Coding style
[Definition of coding style adopted. You don't need to be exhaustive; rather highlight some key concepts/options and refer a more comprehensive resource for details. → e.g.: AOS project]

Code reviewing
Instructions for effective code reviewing. When to do? Integrate AI tools?...

Feel free to add more section as needed.

2.3 Code quality metrics and dashboards

[Description of practices defined in the project for *static code analysis* and associated resources.]
[Which quality gates were defined? What was the rationale?]

3 Continuous delivery pipeline (CI/CD)

3.1 Development workflow

Coding workflow
[Explain, for a newcomer, what is the team coding workflow: how does a developer get a story to work on? Etc…
Clarify the workflow adopted [e.g.. gitflow workflow, github flow . How do they map to the user stories?]
[Description of the practices defined in the project for *code review* and associated resources.]

Definition of done
[What is your team "Definition of done" for a user story?]

3.2 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*]

3.3 System observability

What was prepared to ensure proactive monitoring of the system operational conditions? Which events/alarms are triggered? Which data is collected for assessment?...

3.4 Artifacts repository [Optional]

[Description of the practices defined in the project for local management of Maven *artifacts* and associated resources. E.g.: github]

4 Software testing

4.1 Overall testing strategy

[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?...]
[do not 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 CI process.]

4.2 Functional testing and ATDD

[Project policy for writing functional tests (closed box, user perspective) and associated resources. when does a developer need to develop these?

4.3 Developer facing tests (unit, integration)

[Project policy for writing unit tests (open box, developer perspective) and associated resources: when does a developer need to write unit test?
What are the most relevant unit tests used in the project?]

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

4.4 Exploratory testing

[strategy for non-scripted tests, if any]

4.5 Non-function and architecture attributes testing

[Project policy for writing performance tests and associated resources.]