TQS: Manual de Garantia de Qualidade

[1 Project management 1](#_Toc39440117)

[1.1 Team and roles 1](#_Toc39440118)

[1.2 Agile backlog management and work assignment 1](#_Toc39440119)

[2 Code quality management 2](#_Toc39440120)

[2.1 Guidelines for contributors (coding style) 2](#_Toc39440121)

[2.2 Code quality metrics 2](#_Toc39440122)

[3 Continuous delivery pipeline (CI/CD) 2](#_Toc39440123)

[3.1 Development workflow 2](#_Toc39440124)

[3.2 CI/CD pipeline and tools 2](#_Toc39440125)

[3.3 Artifacts repository [Optional] 2](#_Toc39440126)

[4 Software testing 2](#_Toc39440127)

[1.1 Overall strategy for testing 2](#_Toc39440128)

[1. Functional testing/acceptance 2](#_Toc39440129)

[2. Unit tests 3](#_Toc39440130)

[3. System and integration testing 3](#_Toc39440131)

[4. Performance testing [Optional] 3](#_Toc39440132)

[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

A nossa equipa é constituída por quatro membros:

Dono do produto: [João Carvalho](https://github.com/joaocarvalho19) - 89059

DevOps: [Vinícius Ribeiro](https://github.com/viniciusbenite) - 82773

DevOps: Bernardo Rodrigues - 88835

Gestor da equipa: [Alina Yanchuk](https://github.com/alina-yanchuk02) – 89093

## Agile backlog management and work assignment

Para gerir o backlog do projeto, adotamos uma prática baseada em user stories, sendo que estão são descritas e colocadas, por ordem de prioridade, no Pivotal Tracker.

Link : <https://www.pivotaltracker.com/projects/2448629>

Iremos implementar as user stories com maior prioridade primeiro, seguindo uma metologia Agile, de planeamento, desenho, implementação, testes e deploy recorrentes.

# Code quality management

## Guidelines for contributors (coding style)

Iremos adotar um estilo de código universal para todos os contribuidores do projeto de modo a garantir a qualidade de leitura e compreensão do codigo.

Como guidelines para a escrita de codigo iremos usar as normas recomendadas pela Google:

<https://google.github.io/styleguide/javaguide.html>

<https://google.github.io/styleguide/htmlcssguide.html>

<https://source.android.com/setup/contribute/code-style>

## 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?]

Para analisar estaticamente a escrita do codigo optamos pelo uso do SonarQube, recusando o uso de qualquer acrescimo de codigo que obtenha uma classificação de qualidade pouco favoravel.

# 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?]

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

[What is your team “[Definition of done](https://medium.com/@anca_51481/user-story-definition-of-done-dod-in-agile-software-development-and-the-technical-debt-a3abf6821ef2)” for a user story?]

Durante o workflow do nosso projeto vamos proceder à criação de varios branchs com a intenção de cada um ter como objetivo a criação de uma feature distinta, esta feature vai ter uma ou mais user stories ao qual é relevante,

Quando a feature da branch for concretizada, proceguiremos para a realização de um pull request que passará por um processo de analise/discussão, após aprovação será então deployed e apos passar nos testes, será realizado o merge.

<https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>

<https://guides.github.com/introduction/flow/>

Haverá uma lista de critérios que serão necessários para declarar uma user story com realizada nomeadamente os critérios de aceitação cumpridos, o product owner aprovou-a, testes realizados e passados para a lista completa consultar ([Definition of done](https://medium.com/@anca_51481/user-story-definition-of-done-dod-in-agile-software-development-and-the-technical-debt-a3abf6821ef2))

## 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*]

## Artifacts repository [Optional]

[Description of the practices defined in the project for local management of Maven *artifacts* and associated resources. E.g.: a[rtifactory](https://www.jfrog.com/artifactory/)]

# 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

## Performance testing [Optional]

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