TQS: Quality Assurance manual

***Dinis Cruz [93080], Duarte Mortágua [92963], José Sousa [93019], Tiago Oliveira [93456]***

v2021-06-06

[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

Team:

* Team Lider: Duarte Mortágua, assegurar que há uma boa distribuição das tarefas e que estas estão a decorrer como planeado. Promover uma boa colaboração entre a equipa e ter iniciativa para resolver problemas que possam existir. Assegurar que o trabalho é entregue quando necessário.
* Product owner: Tiago Oliveira, representa os interesses do cliente. Tem um conhecimento extenso daquilo que é pretendido do produto, a equipa recorrerá a este membro quando houver dúvidas sobre funcionalidades da aplicação. Deve estar presente na entrega de incrementos
* QA Engineer: Dinis Cruz, responsável de promoção quanto as praticas para assegurar bom software desenvolvido.
* DevOps master: José Sousa, responsável pela estrutura de desenvolvimento e de produção. Prepara os ambientes para serem deployed.
* Developer: Todos os membros vão ser desenvolvedores da plataforma.

## Agile backlog management and work assignment

Criar as user stories mediante aquilo que será pretendido da nossa aplicação e usar a Zen Board para estarmos cientes daquilo que já foi feito e aquilo que ainda não está completo.

Também faz sentido criar um link entre a Zen Board e aquilo que se passa no repositório do git hub para termos a certeza de que há coerência entre as 2 ferramentas que estamos a usar

O work flow normal não deverá fugir muito de:

**Criar User Story**

1. Entrar na Board do ZenHub.

2. Canto superior direito -> New Issue.

3. Template -> Story.

4. Preencher título da story, descrição, acceptance criteria, definition of done.

5. Atribuir assignees, sprint e story points (estimate) no painel da direita.

6. Submit new issue.

**Desenvolver Story**

1. No repo local -> git checkout develop. git pull. (git flow init se nao tiverem feito). git flow feature start <#\_user\_story>, onde # é o número da user story na board do projeto. Se forem precisas mais features fazer feature <#\_user\_story\_1>...

2. Desenvolver (add, commit, add commit...)... e ir preenchendo o acceptance criteria/definition of done no issue, no zenhub.

**Publicar Story**

1. No repo local -> git flow feature publish <#\_user\_story>.

2. No GitHub (não no ZenHub), fazer Compare & Pull Request da feature.

3. Mudar merge para o develop (e não para o master)

4. Comentar as mudanças.

5. Em baixo conectar o PR ao issue (aka user story) correspondente.

6. No painel da direita -> Team Workspace por ReviewQA ou Done (dependendo do caso), adicionar 2 reviewers.

7. Create pull request.

8. (Aqui vai entrar a CI pipeline)

9. (Pessoal aprova, da merge e delete do branch) e move a story para o Done na board do ZenHub.

[Description of agile practices defined in the project for backlog management (user stories oriented) and job assignment, and links to associated resources. cfr. [PivotalTracker workflow](https://www.pivotaltracker.com/help/articles/workflow_overview/) ]

# Code quality management

## Guidelines for contributors (coding style)

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

[Definition of coding style adopted. → e.g.: [AOS project](https://source.android.com/source/code-style.html)]

## Code quality metrics

Depois do sucesso que 3 dos 4 membros do grupo tiveram a usar sonar cloud decidimos usar de novo para analise estática de código.

Vamos usar a “quality gate” default visto que a nosso ver define boas praticas e para os nossos propósitos parece-nos que vamos conseguir criar software de qualidade com a ajuda desta métrica.

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

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

# Continuous delivery pipeline (CI/CD)

## Development workflow

O workflow complete já foi descrito acima.

Visto que somos um grupo pequeno achamos que 1 reviewer por “pull request” será o necessário para manter qualidade no código, esta decisão teve também em conta a confiança que já existe dentro do grupo depois de muitos projetos já desenvolvidos em conjunto.

[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?]

## CI/CD pipeline and tools

Pipeline desenvolvida no github actions com 2 tipos de workflows.

Feature workflow:

- Build e test para feaure branches

Sonar cloud com build e test:

- Build, test, e analise do sonar cloud com merge para develop e marter.

[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

A estratégia escolhida foi TDD e misturamos diferente ferramentes como REST-assured com mock mvc.

[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

Black box since we are writing the tests first we make these tests based on what we want the module to do

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

## System and integration testing

Closed box since we are mostly testing the results from what our application is expected to output.

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

## Performance testing [Optional]

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