TQS: Quality Assurance manual

***Marcel de Araújo Santos Souza - 101043***

v2023-04-18

[1 Project management 1](#_Toc132723568)

[1.1 Team and roles 1](#_Toc132723569)

[1.2 Agile backlog management and work assignment 1](#_Toc132723570)

[2 Code quality management 2](#_Toc132723571)

[2.1 Guidelines for contributors (coding style) 2](#_Toc132723572)

[2.2 Code quality metrics 2](#_Toc132723573)

[3 Continuous delivery pipeline (CI/CD) 2](#_Toc132723574)

[3.1 Development workflow 2](#_Toc132723575)

[3.2 CI/CD pipeline and tools 2](#_Toc132723576)

[3.3 System observability 2](#_Toc132723577)

[3.4 Artifacts repository [Optional] 2](#_Toc132723578)

[4 Software testing 2](#_Toc132723579)

[4.1 Overall strategy for testing 2](#_Toc132723580)

[4.2 Functional testing/acceptance 3](#_Toc132723581)

[4.3 Unit tests 3](#_Toc132723582)

[4.4 System and integration testing 3](#_Toc132723583)

[4.5 Performance testing [Optional] 3](#_Toc132723584)

# Project management

## Team and roles

Todo o projeto foi realizado pelo aluno Marcel Santos Souza.

## Agile backlog management and work assignment

Como foi um projeto de realização individual, o BackLog não foi implementado e os princípios de spaced-delivery do AGILE não puderam ser implementados também, uma vez que são necessários pelo menos 2 integrantes para tal.

# Code quality management

## Guidelines for contributors (coding style)

O projeto utiliza-se de linters para a organização das files que foram rodados antes de serem commitados na branch. Como foi realizado por apenas um estudante, não foi necessário realizar o trabalho com “branch model” como prevê a metodologia AGILE.

# Continuous delivery pipeline (CI/CD)

## Development workflow

Mais uma vez, infelizmente, como o projeto foi desenvolvido por apenas uma pessoa, n]ao julguei, como desenvolvedor do projeto,necessário criar mais do que uma branch para o porpósito. Sendo assim o workflow constituido apenas da branch main com commits que utilizam mais de uma linha para explicar o que foi desenvolvido na branch.

A definição de finalizado para o projeto foi: uma vez que a tarefa foi completada, revisada, passou pelo “lint” e por todos os testes, foi adicionado ao pipeline de CI e CD. Ou seja, uma vez que toda a tarefa foi finalizada e pode ser acessada pelo utilizador final.

## CI/CD pipeline and tools

A ferramenta utilizada para CI foi o Github Actions. A partir deste, é possível isolar o projeto ReactJS e o SpringBoot. Estes rodam um build todas as vezes que são enviados para a branch online do Github por meio do git. O resultado dos testes em SpringBoot, tais como quaisquer erros de utilização e instalação de bibliotecas tanto no ReactJS quanto no SpringBoot são enviados ao email do utilizador responsável pelo commit.

A ferramenta para CD utilizada foi o Docker (Compose). Por meio deste, o deployment é facilitado e pode-se trabalhar com isolamento dos serviços. Assim, Database MySql, ReactJS e o SpringBoot rodam em containers distintos e autônamos, no entanto estão 100% integrados no que diz respeito a sua utilização como aplicação completa.

# 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.]