TQS 2019 | RELATÓRIO TÉCNICO

Manual de Qualidade e Entrega Contínua

|  |  |
| --- | --- |
| Projeto: | **Criação de um sistema capaz de gerir listas de compras, VLoja** |
| Preparado por: | João Paulo Marques Fidalgo, 62243 |
| Data: | 26/09/2019 |

**Conteúdo do relatório**

[1- Accesso rápido aos recursos (bookmarks do projeto)](#_d56xpcpzm824)

[2- Gestão do projeto](#_par1mnjwqynf)

[Equipa e papéis](#_md5rnv5xuu8)

[Gestão do backlog e atribuição de trabalho](#_r6fsb2up7tph)

[Issues tracking system](#_dpkzb29i9w7a)

[3- Gestão e qualidade do código](#_sclhyngo3qyn)

[Guia para os colaboradores (coding style)](#_mngurqdxamft)

[SCM workflow](#_am60dg50tabv)

[Revisão de código](#_abf4zzsx62ec)

[Análise estática](#_2b56b2jp0o22)

[Repositório de componentes [Opcional]](#_9uly2tsvzez9)

[3- Integração contínua & entrega contínua](#_nzjsi1tu722k)

[4- Testes](#_5ihctf27mtps)

[Testes funcionais/aceitação](#_lnkfvadi74s8)

[Testes unitários](#_pm03qsvye3p3)

[Testes de sistema e de integração](#_gfexy8q03vz9)

[Testes de desempenho [Opcional]](#_qeotwcp7tb9x)

# 

# 1- Accesso rápido aos recursos (*bookmarks* do projeto)

Sistematização dos links para os recursos desenvolvidos no projecto:

* Acesso ao(s) projecto(s) de código:
  1. [Módulo x](https://github.com/NYTimes/objective-c-style-guide)
  2. Módulo y
  3. Documentação da API de integração para programadores
* Ambiente SQA :
  1. Análise estática ([Sonarqube](http://192.168.1.1))
  2. Integração Contínua (Jenkins)
  3. xxx
* Coordenação da equipa:
  1. Projeto PivotalTracker

[podem ser acrescentadas mais entradas; a ideia é **sistematizar aqui os *bookmarks* para os vários resultados/ferramentas** do projeto]

# 3- Gestão e qualidade do código

## Guia para os colaboradores (*coding style)*

No projeto for a

[Definição do coding style adotado. Sugestão → conferir [AOS project](https://source.android.com/source/code-style.html)]

## SCM *workflow*

[Descrição das práticas definidas no projeto para o SCM git e recursos associados.

Clarificar o workflow adotado, cfr. [Bitbucket documentation](https://www.atlassian.com/git/tutorials/comparing-workflows)]

## Revisão de código

[Descrição das práticas definidas no projeto para *code review* e recursos associados.]

## Análise estática

[Descrição das práticas definidas no projeto para *static code analysis* e recursos associados.]

## Repositório de componentes [Opcional]

[Descrição das práticas definidas no projeto para gestão local de *artifacts* do Maven e recursos associados. E.g.: [artifactory](https://www.jfrog.com/artifactory/)]

# 3- Integração contínua & entrega contínua (CI/CD)

[Descrição das práticas definidas no projeto (estratégia) para a integração contínua de incrementos e recursos associados.]

[Descrição das práticas para entrega contínua, baseadas em *containers* (ou Kubernetes)]

# 4- Testes

[não é para escrever aqui o conteúdos dos testes, mas explicar as políticas/práticas adotadas e gerar evidência que os resultados dos testes estão a ser considerados no processo de CI.]

## Testes funcionais/aceitação

[Política do projeto para escrita de testes de funcionais (caixa fechada, perspetiva do utilizador) e recursos associados.]

## Testes unitários

[Política do projeto para escrita de testes unitários (caixa aberta, perspetiva do developer) e recursos associados.]

## Testes de sistema e de integração

[Política do projeto para escrita de testes de integração (caixa aberta ou fechada, perspetiva do developer) e recursos associados.]

## Testes de desempenho [Opcional]

[Política do projeto para escrita de testes de desempenho e recursos associados.]