**Testes e Deployment (TD)**

|  |  |
| --- | --- |
| **Disciplina:** | TD |
| **Data:** | 2018.10.02 |
| **Coordenador:** | Eduardo Guerra |
| **Status:** | *baselined* |
| **Versão:** | 2F |

**Lista de membros da unidade de testes:**

|  |  |  |  |
| --- | --- | --- | --- |
| Eduardo Guerra | LEI | Coordenador | eduardo.vilasboas.guerra@gmail.com |
| André Correia | LEI | Colaborador | andrefcorreia88@gmail.com |
| Ricardo Gonçalves | LDM | Colaborador | rgoncalvestlm@gmail.com |
| Tomás Lopes | LEI | Vice-coordenador | tomasgabriel1998@gmail.com |
| João Campos | LDM | Colaborador | jgmmcampos@gmail.com |

**Objetivos da unidade de TST:**

● Garantir que os requisitos do cliente são implementados e que o código desenvolvido é bem aplicado, tendo em conta os diagramas elaborados pela unidade de requisitos.

● Garantir a qualidade e o bom funcionamento do código implementado.

● Fazer deployment do programa.

**Processos utilizados por esta unidade:**

|  |
| --- |
| Processo para averiguar se requisitos estão a ser cumpridos |
| Processo para testar código |
| Processo para deployment |

**Processo para escrita de testes**

|  |  |
| --- | --- |
| **Disciplina:** | TST |
| **Data:** | 2018.09.26 |
| **Coordenador:** | Eduardo Guerra |
| **Status:** | *baselined* |
| **Versão:** | 2 |

**Descrição do processo:**

Consiste na escrita do procedimento a ser realizado por um *tester* (pessoa que testa código), em particular as pré-condições, os *inputs* (texto a ser preenchido e botões a *clickar*) e o resultado esperado. As pré-condições contém o estado do sistema antes do teste ser realizado (dados da aplicação e página em que o utilizador se encontra), os *inputs* referem-se aos dados inseridos pelo tester e que pretendem alterar o sistema (exemplo, criar uma nova entidade) e o *output* diz respeito ao estado final do sistema e resultados obtidos.

Para os requisitos funcionais, existe o documento “**Testes funcionais**”. Netse encontra-se uma lista de testes, cada um baseado num requisito do Documento de requisitos. Cada teste tem:

1. Objetivo do teste
2. Página de pré-condição
3. Valores de pré-condição
4. Ação a realizar
5. Resultado esperado
6. Resultado obtido

Um requisito é aprovado se o resultado esperado coincidir com o resultado pretendido. Caso contrário, o teste recebe uma reprovação. Depois de todos os testes serem concluídos, o documento é submetido com o número de testes aprovados/reprovados no inicio do documento e o nome do autor no nome do documento.

Para os requisitos não-funcionais, existe o documento “**Testes não-funcionais**”. Estes testes servem para averiguar se os requisitos não-funcionais estão a ser cumpridos, podendo os testes ser de natureza diversa.

Existem testes de carga e testes de usabilidade.

Cada teste de carga tem:

1. Constituição do teste
2. Reação do sistema (a que ponto começa a mostrar alterações)

Se as métricas o indicarem, pode ter de haver correção do código.

Cada teste de usabilidade tem:

1. Constituição do teste
2. Número de cliques ideal
3. Número de cliques efetuado
4. Tempo ideal
5. Tempo efetivo
6. Tarefa concluída? (o utilizador pode não conseguir completar a tarefa, sendo um forte argumento a favor da dificuldade da utilização do programa)
7. Autor e horas despendidas

Caso as métricas desejáveis e as métricas obtidas sejam muito diferentes, pode ter de haver um redesign do layout.

**Inputs:**

Lista de requisitos

Lista de *mockups*

**Critério de entrada/ativação:**

O coordenador de requisitos envia uma mensagem no Discord a informar que existe uma nova versão do documento de requisitos. A partir deste documento, é elaborado uma nova versão do documento de “Testes funcionais” e “Teste não-funcionais”.

**Descrição das tarefas:**

Elaboração do documento de requisitos por parte do coordenador da unidade. Os restantes elementos poderão ajudar corrigindo eventuais lapsos/falhas.

**Critério de saída:**

Assim que cada requisito contenha um test case para testar, então o processo dá-se como concluído, sendo que será mais tarde utilizado por esta unidade para testar código implementado. Apesar de concluído e em uso num

outro processo, este documento pode ser novamente reaberto, para adição/edição de casos de uso.

**Métricas de avaliação deste processo:**

Número de testes criados por requisito;

Qualidade das pré-condições e do output;

Horas de trabalho semanais por elemento;

**Outputs gerados (artefactos):**

Gera-se uma tabela de testes com as etapas a serem realizadas aquando de um teste. Esta tabela será preenchida na verificação de cumprimento de requisitos e de testes de código.

**Monitorização deste processo pela equipa de Qualidade:**

A unidade de qualidade faz uma comparação entre a tabela de requisitos e a tabela de testes para saber se os testes criados testam efetivamente o código

implementado, se vão de acordo com os requisitos do cliente.

**Processo para averiguar se requisitos estão a ser cumpridos**

|  |  |
| --- | --- |
| **Disciplina:** | TST |
| **Data:** | 2018.10.02 |
| **Coordenador:** | Eduardo Guerra |
| **Status:** | *baselined* |
| **Versão:** | 2 |

**Descrição do processo**

É feito uma série de testes a partir da lista de requisitos.

No caso dos requisitos funcionais, estes são testados para averiguar se forma implementados ou não.

No caso dos requisitos não-funcionais, estes são testados de acordo com os testes e métricas definidas no documento “**Testes não-funcionais”.** Os requisitos não funcionais devem ser testados de preferência com pessoas não familiares ao projeto.

**Inputs**

O documento de requisitos enviado pela equipa de testes.

Versão do programa a testar.

**Output**

Lista de funcionalidades implementadas e por implementar.

Lista de testes elaborado por esta unidade de modo a testar requisitos não-funcionais juntamente com as métricas recolhidas pelos diferentes elementos da equipa.

Horas despendidas por cada elemento em cada tarefa.

**Critério de ativação:**

O coordenador da unidade de implementação envia uma mensagem pelo Discord à equipa de testes a avisar que há uma nova versão a ser testada.

**Descrição das tarefas:**

Cada elemento terá que averiguar se o sistema funciona corretamente e se está de acordo com os requisitos do cliente.

Também deve recolher métricas de usabilidade do programa e testar o programa com indivíduos fora do contexto de testes e implementação, de forma a obter dados realistas e demonstrativos do utilizador-alvo.

**Critério de saída:**

Assim que cada elemento tenha realizado preenchido os documentos de testes, este pode ser dado como concluído e o documento deve ser submetido no GitHub, com o nome e versão como título. O teste anterior da mesma pessoa para a mesma versão deve ser apagado.

**Métricas de avaliação deste processo:**

Tabela com requisitos aprovados ou reprovados.

Horas de trabalho semanais por elemento;

**Outputs gerados (artefactos):**

Gera-se uma tabela de requisitos, que conforme tenham sido cumpridos ou não, recebem uma denominação de “aprovado” ou “reprovado”.

**Processo para testar código**

|  |  |
| --- | --- |
| **Disciplina:** | TST |
| **Data:** | 2018.10.02 |
| **Coordenador:** | Eduardo Guerra |
| **Status:** | *Baselined* |
| **Versão:** | 2 |

**Descrição do processo:**

Este processo tem como objetivo averiguar se o trabalho desenvolvido pela equipa

de implementação foi bem realizado, estando de acordo com os requisitos e que satisfaça as normas de programação e interação com um utilizador (não permitindo a inserção de valores indesejados, por exemplo: uma expressão num campo numérico). Em suma, é um processo que testa o código implementado.

**Inputs:**

O código realizado pela equipa de implementação que terá sido disponibilizado no GitHub, pelo coordenador da unidade de implementação, assim como os documentos “**Testes funcionais**” e “**Testes não-funcionais**”.

**Critério de entrada/ativação:**

Uma mensagem gerada no *Discord,* enviada do coordenador de implementação, para esta unidade. Esta mensagem consiste em informar a unidade de testes de que uma tarefa já foi desenvolvida e que está pronta para ser testada.

**Descrição das tarefas:**

São divididas tarefas e realizados testes de códigos, com base nos documentos “**Testes funcionais**” e “**Testes não-funcionais**”.

Para os testes funcionais, o *tester* deve proceder como no documento respetivo, devendo alterar o estado do teste para *passed* ou *failed*, consoante o resultado do teste esteja de acordo com o *output* descrito no vetor de testes. Quando o documento estiver completo, este é submetido no GitHub, de forma a unidade de testes poder examiná-lo.

Para os testes não-funcionais, são realizados por elementos de fora do contexto de implementação e testes e supervisionados por um elemento da unidade de testes. Este deve indicar ao utilizador as ações a realizar e recolher métricas e comentários com base na experiência do utilizador. Caso os testes indiquem uma má experiência de utilizador, deve ser escrito um comentário na linha do teste de forma a equipa de implementação corrigir o problema de forma a melhorar a experiência do utilizador. Quando o documento estiver preenchido, este deve ser submetido no GitHub de forma a poder ser examinado pela equipa de implementação.

Para além destes testes, devem ser realizados testes gerais a fim de encontrar bus e outros comportamentos indesejados do software. Estes devem ser descritos e submetidos no Trello da Unidade de Implementação.

**Critério de saída:**

Elaborados documentos com resultados de testes funcionais e não-funcionais e realização de testes gerais.

**Métricas de avaliação deste processo:**

* Número e taxa de testes aprovados/reprovados por requisito funcional.
* Tempo e taxa de tarefas realizadas por requisito não-funcionais
* Bugs encontrados no programa
* Horas de trabalho semanais por elemento da unidade;

**Outputs gerados (artefactos):**

O artefacto gerado são documentos “**Testes funcionais**”, agora com o resultado dos testes (aprovado/reprovado) e com observações ao mesmo, “**Testes não-funcionais**”, com as métricas obtidas de cada utilizador e comentários de forma a melhorar a experiência de utilizador, e bugs encontrados

**Monitorização deste processo pela equipa de Qualidade:**

A unidade de qualidade faz uma leitura sobre o resultado de cada teste e averigua, em caso de falha, se a unidade de implementação foi notificada e se sabe o que deve editar.

**Deployment**

|  |  |
| --- | --- |
| **Disciplina:** | TD |
| **Data:** | 2018.09.26 |
| **Coordenador:** | Eduardo Guerra |
| **Status:** | *baselined* |
| **Versão:** |  |

**Descrição do processo:**

O deployment é feito na própria máquina do tester, estando a aplicação limitada a esse utilizador.

**Inputs:**Código desenvolvido pela unidade de implementação, disponível no GitHub. Versão da aplicação

**Output:** Aplicação deployed.

**Critério de entrada/ativação:**

Existência de uma versão do produto devidamente testada, para ser feito o deployment.

**Validação**:Operação realizada sem erros fatais.

**Métricas de avaliação deste processo:**Versão da aplicação;

**Outputs gerados (artefactos):**

Número de horas despendidas na tarefa.

Versão da aplicação.