Minervapply
Plano de Testes

Versão 1.0.0

Histórico de Revisão

Data	Versão	Descrição	Autor
2/12/2016	1.0.0	Versão inicial; serão documentadas apenas as estratégias de teste com as funcionalidades implementadas até o momento	De Lello, Ton

Sumário

Plano de Testes

1. Introdução

1.1. Finalidade

Este documento tem como finalidade descrever o Plano de Testes do software Minervapply, uma aplicação web desenvolvida em Python para gerenciamento de ofertas de bolsas e estágios na UFRJ.

1.2. Escopo

O escopo deste documento está intimamente relacionado à documentação de requisitos e às especificações de projeto. Leva em conta as funcionalidades já desenvolvidas do software e avalia o código já implementado.

1.3. Definições, Acrônimos, e Abreviações

O glossário está presente na Lista de Requisitos, que pode ser acessada utilizando-se a referência abaixo.

1.4. Referências

Título	Versão	Data	Onde pode ser obtido
Lista de Requisitos	2.1	3/12/2016	https://github.com/ESEGroup/Bolivia/blob/1c39efff3c11ee48cfc7951e4cbfefe7f0ddc6ef/DocumentaçãoProjeto/TemplateExemplo_LISTA-REQUISITOS_revisao-3-12.docx.pdf
Documentação de Casos de Uso	1.0	23/10/2016	https://github.com/ESEGroup/Bolivia/blob/novo-master/DocumentaçãoProjeto/Casos%20de%20Uso%20-%20Sistema%20de%20Gerenciamento%20de%20Bolsas%20UFRJ_atual.pdf

1.5. Visão geral

O restante do documento fornece com mais detalhamento os testes de software a serem realizados. São planejados especialmente os testes funcionais, de acesso e segurança e de tolerância a falhas.

2. Estratégia de Teste

Primeiramente, devem-se fazer os testes funcionais e de segurança do projeto, para as situações especificadas abaixo.

Os resultados devem ser documentados, analisados e, se necessário, correções devem feitas e, em seguidas, os testes devem ser novamente aplicados.

Serão também aplicados testes unitários e testes de integração, além dos testes de sistema. Os testes unitários serão feitos em paralelo com os testes funcionais e de segurança, pois eles são necessários para garantir o seu funcionamento correto. Serão usadas bibliotecas de Python para testes unitários, como a **unittest**. Para o teste de sistema, será utilizado o plugin Selenium IDE para Mozilla Firefox.

2.1. Teste Funcional

Este tipo tem por objetivo testar as funcionalidades do programa, de forma a verificar seu funcionamento correto e se atendem aos requisitos funcionais estabelecidos.

2.1.1. Prazo para realização

Este tipo de teste será realizado a cada acréscimo de uma nova funcionalidade ou após qualquer atualização que tenha impacto sobre quaisquer funcionalidades implementadas. Portanto, serão realizados periodicamente em paralelo com a codificação do programa para a Release 2. O prazo para o término dos testes funcionais é 19/12/2016.

2.1.2. Recursos necessários

Os recursos necessários para os testes funcionais são: a existência de um humano como usuário (que pode na maior parte das vezes ser representado por alguém da equipe, eventualmente ser qualquer indivíduo externo), conexão com a internet e um computador que possa utilizar a aplicação por meio de um browser.

2.1.3. Requisitos a serem testados

Os requisitos a serem testados são:

RF03: O sistema deve permitir que membros do corpo docente da UFRJ possam divulgar, dentro da própria aplicação, ofertas de vagas para bolsas e estágios dentro dos campi da UFRJ.

RF04: O sistema deve possibilitar a inclusão de informações sobre a vaga oferecida, como local do estágio/pesquisa, prazo de aplicação, valor da bolsa, entre outros.

RF05: O sistema deve permitir a visualização por parte de todos os indivíduos - inclusive sem cadastro - das ofertas de vagas divulgadas.

RF11: O sistema deve permitir que o fornecedor da vaga desative a mesma sempre que desejar.

2.1.4. Casos de Teste

1. Caso de Teste “Divulgar Vaga”

Dados de entrada: título da vaga + remuneração + local + prazo de aplicação + tipo (Estágio externo/Iniciação Científica/Estágio interno/Projeto)

Pré-condições: estar autenticado no sistema como usuário-professor

Resultado esperado: a vaga é divulgada na aplicação e é exibida nas listas de vagas

2. Caso de Teste “Editar Vaga”

Dados de entrada: campo(s) da vaga a ser(em) editado(s)

Pré-condições: estar autenticado no sistema como usuáeio-professor e possuir uma vaga divulgada

Resultado esperado: as alterações são feitas e exibidas junto com os outros dados da vaga nas listas de vagas

3. Caso de Teste “Apagar Vaga”

Dados de entrada: comando “Editar Vaga” > comando “Apagar vaga”

Pré-condições: estar autenticado no sistema como usuário-professor e ter uma vaga divulgada

Resultado esperado: a vaga é apagada e retirada das listas de vagas

4. Caso de Teste “Visualizar vagas”

Dados de entrada: —

Resultado esperado: página inicial exibe lista de vagas

2.2. Teste De Controle de Acesso e Segurança

Este tipo tem por objetivo testar as formas de acesso dos usuários ao programa, assim como a segurança dessas informações no mesmo. O prazo estimado para que os testes de segurança sejam integralmente realizados é até 19/12/2016.

2.2.1. Prazo para realização

Este tipo de teste será realizado periodicamente enquanto as funcionalidades de login e autenticação de usuários estiverem sendo implementadas no sistema. Portanto, serão realizados em paralelo com a codificação do programa para a Release 2.

2.2.2. Recursos necessários

Os recursos necessários para os testes funcionais são: a existência de um humano como usuário (que pode na maior parte das vezes ser representado por alguém da equipe, eventualmente ser qualquer indivíduo externo), conexão com a internet e um computador que possa utilizar a aplicação por meio de um browser.

2.2.3. Requisitos a serem testados

Os requisitos a serem testados são:

RF12: O sistema deve possibilitar a validação de cadastro de usuários. RN1, RN2.

RF13: Todos que executam login devem poder ver as informações básicas de todos os alunos.

RN1: O cadastro de usuários-alunos deve ser validado automaticamente pelo próprio sistema, pela análise do DRE.

RN2: O cadastro de usuários-professor deve ser validado pelo chefe de seu departamento ou pelo superadministrador. Coordenadores de departamento devem poder validar o cadastro somente de usuários-professores de seu próprio departamento. O superadministrador deve poder validar o cadastro de qualquer usuário-professor.

RNF05: O sistema só pode permitir a divulgação de qualquer oferta de vaga por usuários-professores autenticados.

RNF06: O sistema só pode permitir a candidatura a qualquer vaga por usuários-alunos autenticados.

RNF07: O sistema deve vetar a candidatura a uma vaga ou a divulgação de uma vaga a qualquer indivíduo não autenticado.

RNF08: O sistema deve fazer a autenticação de usuários cadastrados por meio de seu nome de usuário e senha.

2.2.4. Casos de Teste

1. Caso de Teste “Fazer Login”

Dados de entrada: nome de usuário e senha

Resultado esperado: O usuário consegue logar na sua conta dentro do sistema.

2. Caso de Teste “Cadastrar Aluno”

Dados de entrada: Campos dos dados do usuário a ser criado (Nome Completo, DRE/Identificação, Curso, Email, Senha)

Resultado esperado: O usuário é criado e faz login no sistema.

3. Caso de Teste “Cadastrar Professor”

Dados de entrada: Campos dos dados do usuário a ser criado (Nome Completo, DRE/Identificação, Curso, Email, Senha)

Resultado esperado: O usuário recebe uma informação de que a criação de sua conta está pendente e será validada por um professor chefe de departamento em breve.

2.3. Testes Unitários

Os testes unitários serão feitos usando bibliotecas do Python para testes unitários (*unittest*, *unittest.mock*). Serão realizados em paralelo aos testes funcionais e de segurança.

2.4 Testes de Integração

Serão realizados após todos os testes unitários terem sido feitos e os erros internos às unidades corrigidos.

2.5 Testes de Sistema

Entendemos os testes de sistema como um tipo de teste que nos permite avaliar o comportamento das funcionalidades criadas. Portanto, os testes funcionais serão realizados também como testes de sistema. Os testes de sistema serão repetidos muitas vezes, conforme correções, acréscimos e modificações forem feitos, e também em paralelo e após os testes de integração, para garantir o funcionamento correto do sistema como um todo.

3. Resultados dos Testes