UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE SISTEMAS DE COMPUTAÇÃO

Plano de Testes
UniJobs

1. Identificação

a. **Nome do projeto**
UniJobs

b. **Equipe**
Lorem Ipsum

c. **Data criação do documento**
13/05/2019

2. Introdução

a. **Escopo: descrição dos aspectos a serem testados, dos aspectos que não serão testados e dos critérios para aceitação, ou seja, o que será necessário para que o sistema seja aprovado.**

A solução de software proposta inclui dois componentes: aplicativo mobile (Front-end) e servidor (Back-end).

A avaliação da interface com o usuário inclui aspectos subjetivos que são difíceis de serem captados através de testes automatizados. Além disso, as ferramentas para a construção desse tipo de teste normalmente demandam um aprendizado suplementar. Visto que a maioria dos membros da equipe não está familiarizada com as ferramentas de desenvolvimento Front-end, escolhemos, à princípio, não criar testes automatizados para o aplicativo.

Desta forma, a avaliação desta parte do produto será feita junto com o cliente através do seu feedback após utilização do produto entregue ao final de cada sprint.

A linguagem utilizada para o desenvolvimento do Back-end (Golang), por sua vez, oferece ferramentas nativas para a construção de testes e verificação de cobertura de código. Embora elas exijam uma fase de adaptação, a curva de aprendizado é menor que a de ferramentas de teste de Front-end. Desta forma, a ferramenta "go test" será utilizada para

rodar testes unitários e o comando "go test -coverprofile" dará a cobertura do código atingida com os testes criados. Desta forma, será possível testar a manipulação e fluxo de dados presentes na plataforma, as quais são controladas, em sua maioria, pelo servidor.

Para aceitação das funcionalidades desenvolvidas no lado back-end, deverão ser testados pelo menos os casos indicados na seção 4 e os testes deverão cobrir ao menos 50% de todo código presente em cada *package*.

3. Planejamento para realização dos testes

a. Cronograma de atividades (descrição dos passos que serão realizados na condução dos testes)

O teste de validação do front será realizado ao final de cada sprint junto ao cliente. Será recolhido seu feedback e, caso necessário, modificações serão feitas ao longo da próxima sprint.

Em relação ao back-end, uma história de usuário só será considerada como feita se os testes associados a ela estiverem sido realizados e se a cobertura do código estiver acima de 50%.

b. Responsáveis pelos testes

A validação com o cliente ocorrerá com o Product Owner e ao menos um dos desenvolvedores responsáveis pelo Front-end. Os testes do back-end devem ser realizados pelo desenvolvedor responsável pela implementação da task associada a eles.

4. Projeto de casos de teste

5.

a. Casos de uso considerados

Em primeiro momento, somente serão planejados os testes para as histórias de usuário que correspondem ao MVP, ou seja, aos blocos Core 1, 2, e 3. A seguir, essas histórias são listadas, separadas por bloco. Para cada história, será apresentado o mínimo conjunto de testes que deve ser realizado para que sua validação seja completa.

I. Core 1

H1 - Como usuário, quero conectar-me utilizando minha conta criada durante o cadastro para ter acesso aos serviços oferecidos pelo UniJobs.

- Teste verificando login de um usuário válido
- Teste verificando login de um usuário inválido (não cadastrado)

H6 - Como usuário, quero poder criar uma solicitação para informar à comunidade sobre meu interesse em um serviço.

- Teste para inserção de uma nova solicitação na base de dados por um usuário. Verificar se ligação entre entidades solicitação - usuário está bem realizada e nova solicitação foi efetivamente inserida na base de dados.
- Um usuário não pode ser capaz de criar duas solicitações com mesmo título, descrição e categoria.
- Usuário não deve ser capaz de criar uma solicitação sem todos campos preenchidos (título, descrição, categoria, valor e contato).

H3 - Como usuário, quero ter acesso às últimas solicitações criadas para poder consultá-las.

- Teste para recuperar 20 últimas solicitações inseridas na base de dados.
- Solicitações retornadas devem estar em ordem decrescente de data de criação
- Opcional: habilitar paginação para recuperação de mais de 20 solicitações. Testar paginação.

H7 - Como universitário, quero poder criar uma oferta para oferecer um serviço que estou disposto a realizar.

- Teste para criação de uma oferta à partir de um usuário. Verificar que conexão entre oferta - usuário foi bem estabelecida e nova oferta foi inserida corretamente na base de dados.
- Um usuário não pode ser capaz de criar duas ofertas com mesmo título, descrição e categoria.
- Usuário deve ser capaz de criar uma oferta apenas com os campos obrigatórios (título, descrição, categoria, valor e contato) preenchidos.
- Usuário não deve ser capaz de criar uma oferta sem os campos obrigatórios preenchidos.

H2 - Como usuário, quero ter acesso às últimas ofertas criadas para poder consultá-las.

- Teste para recuperar 20 últimas ofertas inseridas na base de dados.
- Ofertas retornadas devem estar em ordem decrescente de data de criação
- Opcional: habilitar paginação para recuperação de mais de 20 ofertas. Testar paginação.

II. Core 2

H4 - Como usuário, quero poder buscar por ofertas e solicitações criadas pela comunidade, filtrando-as por categoria, para achar aquelas de meu interesse.

- Busca somente por categoria deve retornar apenas 20 últimas ofertas/solicitações associadas a ela. Habilitar paginação para ter acesso a mais resultados.
- Busca deve retornar, primeiramente, ofertas/solicitações que contenham palavras-chave no título. Em seguida, ofertas/solicitações que contenham palavras-chave na descrição serão retornadas. Habilitar e testar paginação dos resultados.

H5 - Como usuário, quero declarar interesse por uma oferta para ter acesso ao contato do seu oferecedor.

- Testar se ligação entre usuário e oferta não está feita caso usuário não tenha declarado interesse por ela.
- Testar se ligação entre usuário e oferta foi realizada na base de dados após declaração de interesse.

H8 - Como universitário, quero ter acesso a todas minhas ofertas criadas para poder consultá-las.

- Teste para recuperar as ofertas inseridas na base de dados pelo usuário. Testar caso em que 0 oferta é retornada (usuário ainda não criou oferta) e caso em que mais de uma oferta é retornada.
- Apenas 20 últimas ofertas devem ser retornadas.
- Ofertas retornadas devem estar em ordem decrescente de data de criação.
- Opcional: habilitar paginação para recuperar mais de 20 ofertas. Testar paginação.

III. Core 3

H17 - Como universitário, quero ter acesso a todas minhas ofertas criadas para poder obter o contato das pessoas interessadas.

- Testes descritos para H8 já cobrem essa História de Usuário. Cabe ao Front-end mostrar as informações adequadas

H18 - Como usuário, quero ter acesso a um histórico das ofertas pelas quais tive interesse para poder recuperar informações de contato do oferecedor.

- Teste para um usuário que ainda não declarou interesse por oferta deve retornar 0 ofertas;
- Teste para um usuário com várias declarações de interesse deve retornar até 20 ofertas;
- Opcional: habilitar paginação para recuperar mais de 20 ofertas. Testar paginação.

H14 - Como usuário, quero ter acesso a todas minhas solicitações criadas para poder consultá-las.

- Teste para recuperar as solicitações inseridas na base de dados pelo usuário. Testar caso em que 0 solicitação é retornada (usuário ainda não criou solicitação) e caso em que mais de uma solicitação é retornada.
- Apenas 20 últimas solicitações devem ser retornadas.
- Solicitações retornadas devem estar em ordem decrescente de data de criação
- Opcional: habilitar paginação para recuperar mais de 20 solicitações. Testar paginação.

b. Geração de casos de teste (utilizar as técnicas de Particionamento em Classes de Equivalência e Análise do Valor Limite)

Os testes especificados na seção anterior tratam-se de Testes Funcionais, uma vez que não consideram a estrutura do código que será desenvolvido para implementar as funcionalidades às quais se referem. O único interesse é a saída esperada para cada entrada descrita.

Para alguns testes, foi utilizado o critério de Classes de Equivalência. Por exemplo, para acesso ao histórico de solicitações, o espaço das solicitações é particionado entre todos os usuários da plataforma e quer-se recuperar apenas aquelas relacionadas ao usuário que realiza a consulta.

Da mesma forma, o espaço das ofertas é particionado entre as datas em que foram criadas e colocando-as em data decrescente de criação, quer-se recuperar, no máximo, as 20 últimas ofertas criadas.

Testes para login de usuário também levam em conta esse critério: testa-se a classe de usuários que podem ter acesso ao sistema e a classe de usuários que *não* podem ter acesso ao sistema.

O Critério de Análise do Valor Limite também foi aplicado em alguns casos. Por exemplo, para a história de usuário H18 pede-se para testar a recuperação de ofertas de interesses

para um usuário com 0 interesses e para outro que tenha declarado ao menos um interesse por oferta.

Da mesma forma, ao recuperar o histórico de ofertas criadas por um usuário em H8, testa-se o caso em que o usuário não criou nenhuma oferta e o caso em que ofertas já foram criadas.

6. Conclusão

a. Recursos que serão utilizados (ferramentas de teste, etc)

Como foi especificado durante a seção *Escopo*, testes automatizados serão realizados apenas do lado Back-end. A ferramenta "go test", fornecida nativamente pela linguagem Golang, será utilizada para rodar os casos de teste criados. Para a análise da cobertura do código, o comando "go test -coverprofile" é capaz de gerar um arquivo que reporta a cobertura do código fonte, separada por *package*.

Um arquivo Makefile inserido no repositório do projeto deve conter uma target 'test' que dispara os testes unitários. Antes de realizar um merge na branch *dev* ou *master*, os desenvolvedores devem rodar todos os testes unitários utilizando o comando "make test" para garantir que suas modificações não introduzam erros no código desenvolvido anteriormente.