Projeto de Bloco - Engenharia Disciplinada de Softwares

TESTE DE PERFORMANCE - TP9

Rodrigo de Moraes Filomeno

002.628.957-14

Sumário

Introdução

Trata-se de um teste de performance onde o aluno deve demonstrar as competências adquiridas durante as etapas lecionadas

Objetivo

Este trabalho tem o objetivo de documentar as competências que o aluno adquiriu durante as etapas.

Justificativa

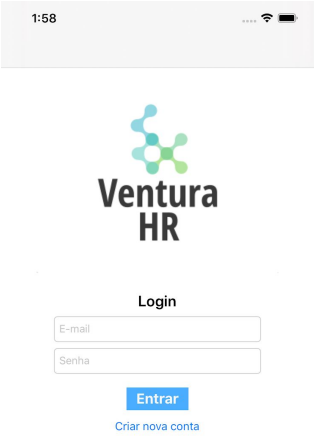
Este trabalho se justifica na necessidade da Instituição verificar o desenvolvimento dos alunos durante o curso.

Descrição da tarefa

1- Protótipo funcional do sistema:

Código Fonte no github:
https://github.com/Rfilomeno/VenturaHR

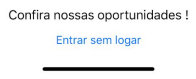
Breve resumo do Protótipo:

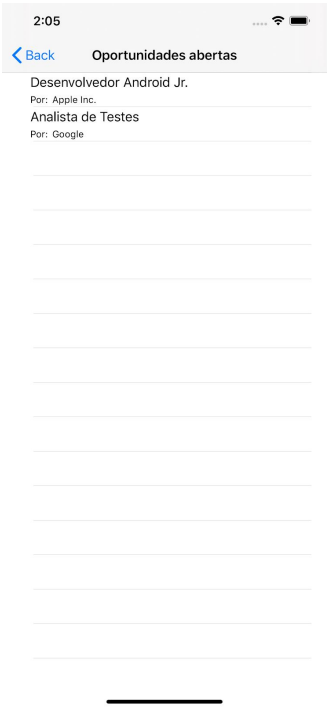


O sistema já se inicia com a Tela de Login, dando ao usuário a possibilidade de visualizar as oportunidades publicadas sem a necessidade de se Logar clicando no botão "Entrar sem Logar".

Damos também a possibilidade de criar uma nova conta de Candidato ou Empresa, ou logar no sistema com uma conta já criada, onde o sistema verifica no autentication do Firebase se o usuário já está cadastrado e se as informações foram inseridas de forma correta.

Ao entrar com dados de uma conta cadastrada, o sistema utiliza o padrão strategy verificando se é uma conta Empresa ou Candidato para mostrar a tela Home apropriada para cada tipo de conta.

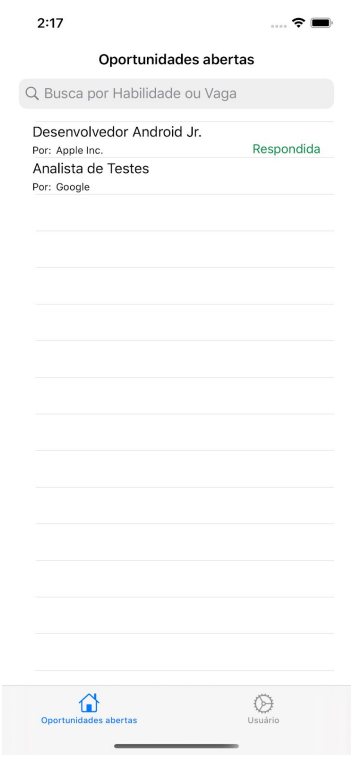


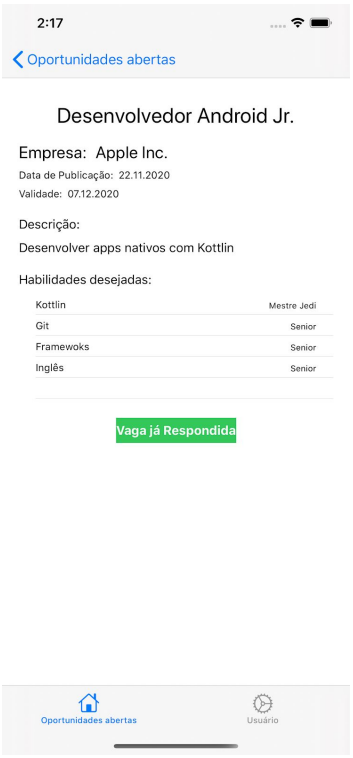


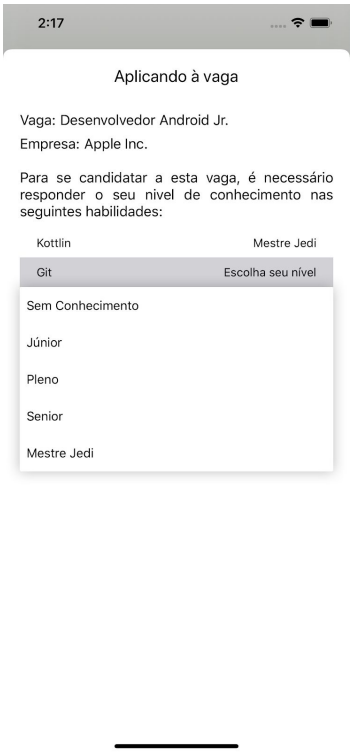
Caso o usuário entre sem se Logar, o sistema mostra as oportunidades publicadas por empresas cadastradas e permite ao usuário ver os Detalhes de cada Vaga, porém para se candidatar a uma vaga, é necessário ter uma conta cadastrada e estar logado no sistema.

As figuras acima demonstram o protótipo de tela de criação de novas contas, tanto de candidato quanto de Empresa, onde os campos estão com validação de preenchimento

obrigatório, bem como as devidas máscaras para telefone e CPF/CNPJ e mascara de ocultação de senha.

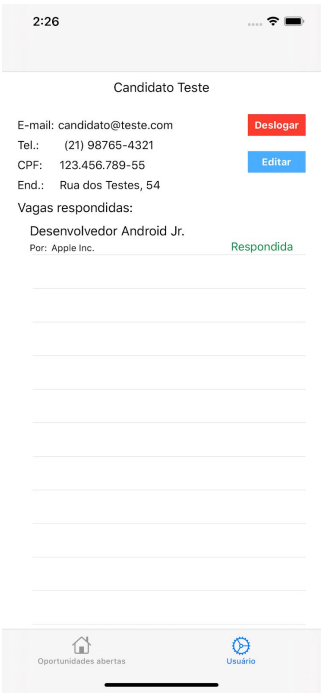


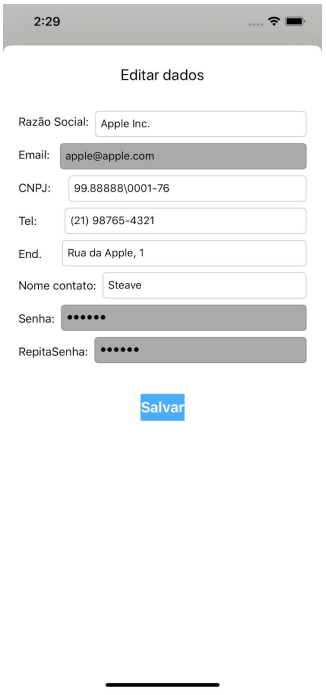




Ao Logar com uma conta de Usuário, o sistema apresenta as oportunidades disponíveis (dentro da data de validade da publicação), indicando se a vaga já foi respondida pelo candidato, dá a possibilidade de olhar os detalhes da vaga e se candidatar a ela (caso ainda não o tenha feito).

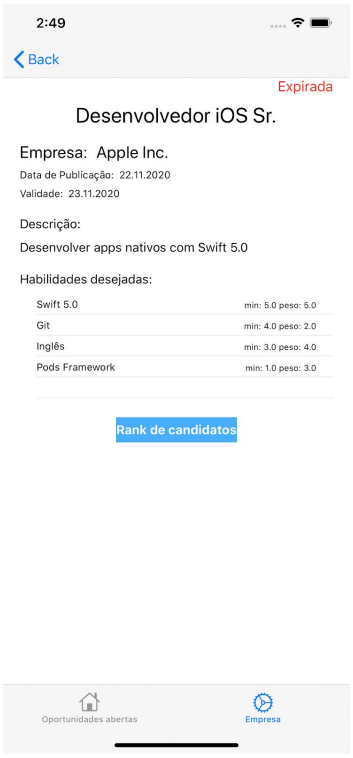
Para se candidatar, o usuário deverá responder as habilidades de acordo com seu perfil de senioridade.





A tela de Perfil do Candidato, possibilita o mesmo a se deslogar do sistema, editar seus próprios dados, mostra a lista de todas as vagas as quais está concorrendo e ainda dá a possibilidade de excluir sua candidatura a uma uma vaga.

A tela de Perfil da Empresa, possibilita a mesma a se deslogar do sistema, editar seus próprios dados, mostra a lista de todas as vagas por ela publicadas, mostrando a quantidade de respostas (candidaturas) cada vaga publicada obteve, dá a possibilidade da empresa cancelar a publicação de uma vaga ou até mesmo publicar novas vagas(oportunidades).



Ao entrar nos detalhes das vagas que a própria empresa publicou, será possível verificar o Ranking de Candidatos que responderam à vaga, que estará ordenada por Score, mostrando em vermelho quando a pontuação do candidato não atinge a média ponderada do mínimo desejado pela empresa, ou quando um candidato não atingir a senioridade mínima requerida de habilidade definida como obrigatória.

Ao clicar no candidato, a empresa poderá visualizar os dados pessoais do mesmo e poderá verificar as respostas que o candidato deu a cada habilidade.

2- Aponte alguma refatoração realizada no código. Aponte o porquê realizar refatoração em projetos ágeis.

Foi realizada uma refatoração no ViewController da Tela de Login, para utilizar o padrão de projeto Strategy (HomeStrategy) retirando a responsabilidade de lidar com a apresentação da tela inicial (Home pós Login) de dentro do controller passando a responsabilidade de lidar com isso para a classe da estratégia que irá apresentar a tela correta dependendo do tipo de usuário.

A Refatoração de código é realizada para melhorar a legibilidade e a manutenibilidade do código que foi escrito, utilizando-se das melhores práticas de programação do mercado para que outros programadores consigam entender mais rapidamente e poder atualizar o código de maneira rápida.

A Refatoração é muito importante em projetos ágeis visto que trabalhamos sempre em equipe sendo fundamental que todos os integrantes consigam ler, identificar e corrigir o código de maneira rápida e eficaz.

3- Aponte todos os padrões de projeto utilizados na implementação e destaque as responsabilidades.

Singleton - Foi utilizado o padrão Singleton (GoF) nas classes de Repositório. Esse padrão tem como responsabilidade garantir que a própria classe seja responsável pela manutenção de uma instância única daquela classe no sistema por completo.

Esse padrão é muito importante para garantir que todos acessem a mesma instância daquela classe evitando por exemplo requisições conflitantes, acesso simultâneo aos mesmos dados, etc... Muito utilizado em classes de acesso a banco de dados.

Strategy - O padrão Strategy (GoF) também foi utilizado neste projeto, ele tira a responsabilidade do controller de decidir qual a estratégia de negócio se aplica ao caso concreto, é criado uma classe strategy que toma essa decisão e devolve por meio de interface a classe que corresponde a estratégia que deverá ser utilizada para o caso.

Delegate - Como o próprio nome sugere, esse padrão delega a responsabilidade da implementação de um método para outra classe. É um padrão relativamente novo e muito utilizado para métodos que devem ser rodados de forma assíncrona. a classe principal chama o método da classe delegada após o retorno da requisição (por exemplo) .

Creator - O Padrão Creator (GRASP), é muito utilizado na programação orientada a objetos, e traz a responsabilidade de da criação de um objeto para a classe correspondente, aumentando a coesão e mantendo o baixo acoplamento no código.

Controller - Esse padrão (GRASP) tem o objetivo de trazer a responsabilidade de manipular eventos do sistema para uma classe que não seja de interface do usuário representando um cenário global ou de um Caso de Uso.

Padrões Estruturais: MVC, Domain e repository

Esses padrões são definidos como estruturais por serem ligados a estrutura do projeto como um todo.
MVC - Model, View, Controller: Padrão que separa estruturalmente classes responsáveis pelo Modelo de Domínio (Model ou Domain), Classes Controladores e classes de View (classes responsáveis pela interface com usuário).

Repository - Também devem ser separadas estruturalmente classes responsáveis pelo acesso aos dados do sistema.

4- Justifique a importância dos testes unitários realizados até aqui.

Testes unitários são muito importantes para o projeto. São testes projetados para garantir que as funcionalidades do código continuam funcionando. Durante o desenvolvimento do projeto, várias alterações vão sendo incluídas no projeto para que ele responda da forma que desejamos, e como garantir que as funcionalidades continuam funcionando de maneira correta? Testes de unidade é a resposta, ele é rápido, simples e eficaz, evita teste morosos e manuais trazendo mais agilidade ao processo de desenvolvimento e garantindo a qualidade do software.

5- Identifique possíveis pontos de evolução e variação no projeto.

Até o momento, desenvolvemos um protótipo de sistema muito maior e mais complexo, existem vários pontos com possíveis melhorias de evolução e variação no projeto, alguns exemplos que podemos citar são:
- Notificação: O sistema deveria notificar a empresa quando uma de seus vagas publicadas fosse respondida por um usuário e quando alguma vaga chega a sua data de validade no sistema.

-Prorrogação de Vaga: deveria ser implementada uma feature que permitisse a empresa prorrogar a validade de uma vaga publicada caso suas expectativas, de encontrar um usuário dentro do perfil que procura, não sejam atingidas.

-Troca de mensagens dentro do sistema: Seria interessante ter a possibilidade da troca de mensagens entre candidato e empresa, para que ambas as partes possam solucionar dúvidas que restem sobre as vagas publicadas.

Favorito: Uma feature interessante de ser implementada é deixar o usuário colocar vagas que têm interesse como favoritas, onde o sistema poderia trazer informações sobre a evolução da vaga ao usuário como por exemplo a vaga está chegando ao fim de sua validade de publicação, ou houve uma alteração nos requisitos para a vaga.

6- Descreva brevemente o histórico da evolução do projeto destacando a composição dos documentos, a implementação do código e mostrando se foi aderente às metodologias ágeis propostas.

a) Relembre o ciclo de vida que foi utilizado, com a justificativa da escolha e compare com a execução.

Começamos o Projeto com um documento de Visão macro, dele conseguimos identificar e documentar os principais atores do sistema bem como os casos de uso.

Criamos um modelo de casos de uso do projeto e a partir deste criamos uma matriz de requisitos do sistema, identificando e documentando os requisitos funcionais e não funcionais e suas respectivas complexidades.

Criamos um modelo de classes de entidade identificando o relacionamento entre eles.

Após uma análise mais detalhada do projeto, refinamos os Casos de uso e os modelos de domínio, bem como documentamos os diagramas sequenciais do projeto.

Com toda a documentação prévia conseguimos documentar o projeto de objetos com um diagrama indicando Classes e métodos que deveriam ser implementados.

A partir desta iteração iniciamos o desenvolvimento do sistemas em conjunto com o desenvolvimento dos testes unitários necessários para garantir a qualidade do software, realizando pequenas entregas de acordo com o desenvolvimento e testes das features.

Conjuntamente com o desenvolvimento fomos aprimorando/refinando os casos de uso e requisitos do sistema, para que o mesmo se mantivesse sempre de acordo com as expectativas do cliente.

Para esse projeto escolhemos utilizar com base no DAD (Disciplined Agile Delivery) um **Ciclo de Vida Ágil** baseado no SCRUM, seguindo as etapas e realizando pequenos entregáveis durante o desenvolvimento do projeto, a execução seguiu, na medida do possível, de acordo com o planejado, mantendo-se dentro do ciclo de vida ágil ao qual foi adequado.

Conclusão

Concluímos este trabalho após uma longa jornada de especificações e implementações, onde foi construídas toda a documentação e diagramas para o desenvolvimento do sistema e consideradas as melhores práticas de Engenharia de Software, refatorando os pontos mais relevantes para a devida adequação às melhores práticas bem como aos padrões de projeto GoF e Grasp descritos.

Bibliografia

Infnet, Moodle, 2020,https://lms.infnet.edu.br/moodle/course/view.php?id=3387

Scott Ambler e Mark Lines, Disciplined Agile Delivery, IBM Press

Robert C. Martin, Código Limpo, Editora Alta Books

Craig Larman, Utilizando UML e Padrões, Editora Bookman

Erich Gamma, Richard Helm Ralph Johnson e John Vlissides. Padrões de Projeto, Editora Bookman

Martin Fowler, Refactoring: Improving the Design of Existing Code, Editora Addison-Wesley Professional