INSTITUTO INFNET

ESCOLA SUPERIOR DE TECNOLOGIA

GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

ENGENHARIA DISCIPLINADA DE SOFTWARE:

PROJETO DE BLOCO - TP9

Luiza Mendes

Índice

1. Projeto

1.1 VenturaHR

VenturaHR é um sistema para auxiliar o recrutamento de talentos, onde os recrutadores das empresas cadastram suas vagas em aberto, os candidatos podem se inscrever nelas e o sistema gradua os candidatos de acordo com suas informações e os requisitos da vaga, processando essas informações em um relatório para a empresa contratante. O sistema também gera relatórios e faz gestão de vagas expiradas. Link para github: https://github.com/luizamendes/VenturaHR

1.2 Atores

1.2.1 Recrutador

É o funcionário da empresa cliente responsável por cadastrar as vagas no sistema, por visualizar as candidaturas recebidas e fechar ou estender a vaga.

1.2.2 Candidato

É a pessoa que irá visualizar as vagas no sistema e se candidatar.

1.2.3 Sistema

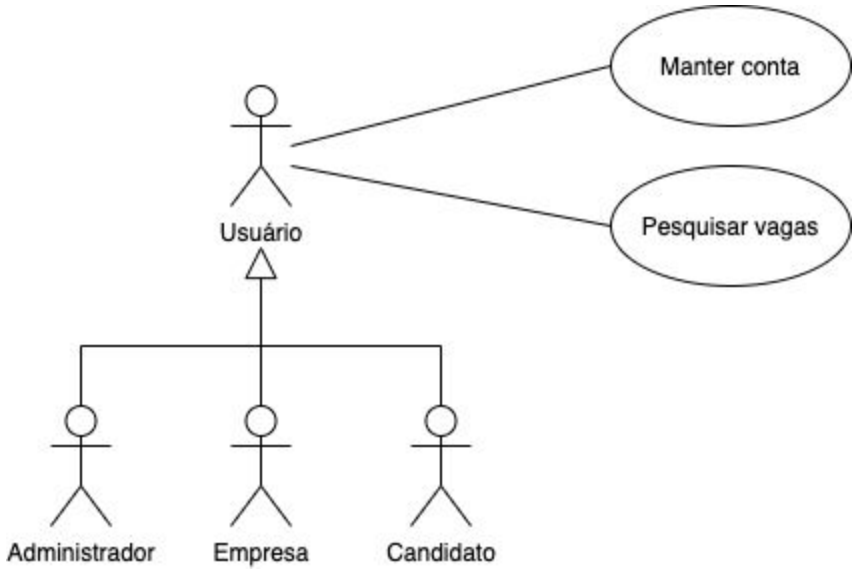
É o sistema onde serão cadastradas as vagas, responsável por controlar o cadastro e acesso de usuários (candidatos e empresas), fazer a manutenção automática das ofertas vencidas, e gerar relatórios básicos de acesso por usuário, empresa e número de ofertas.

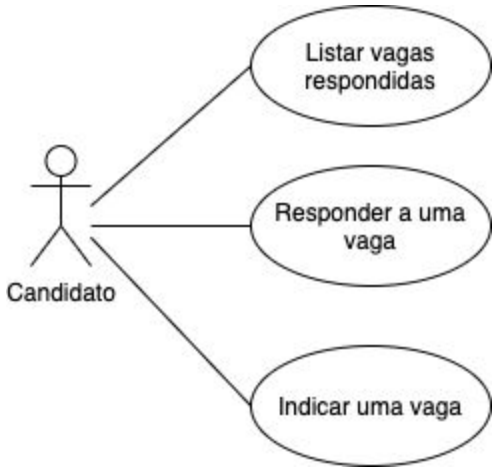
1.2.4 Administrador

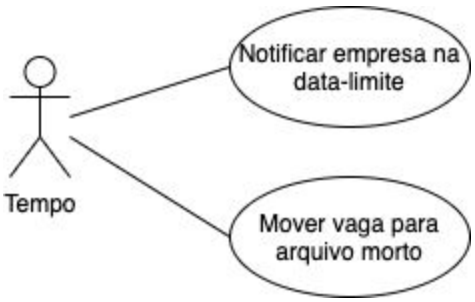
É a pessoa que tem acesso aos dados de acesso por usuário, por empresa e o número de ofertas

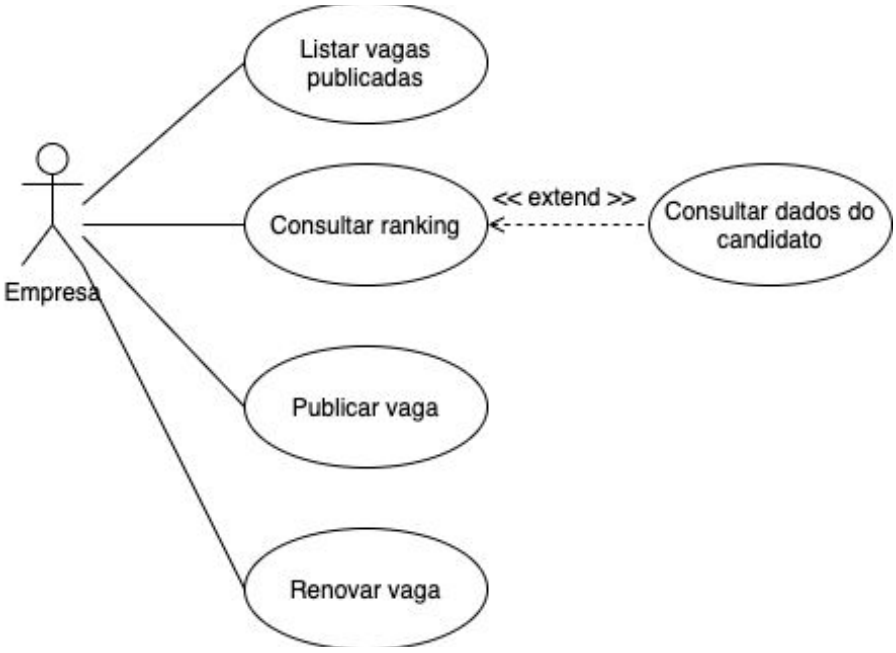
1.3 Casos de Uso

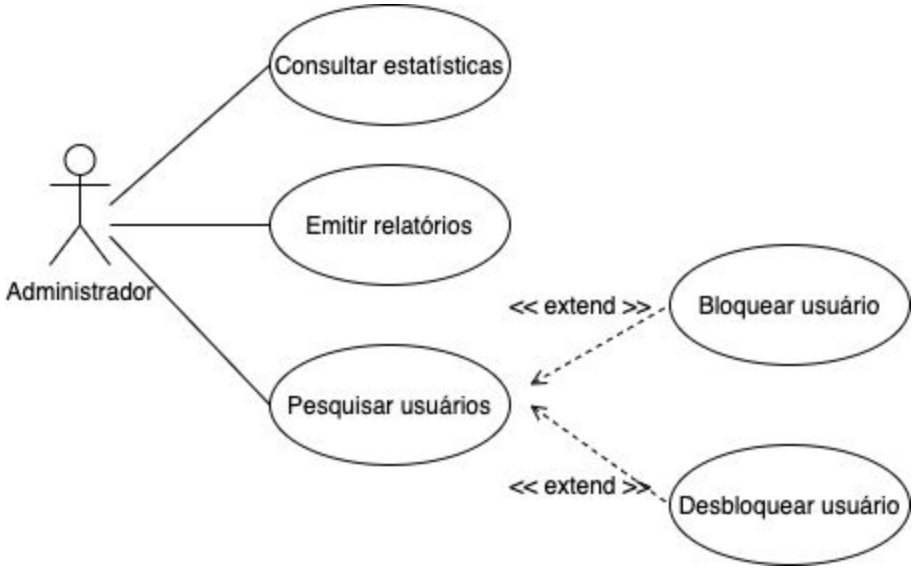
1.3.1 Diagrama de caso de uso








Listar vagas publicadas
Consultar ranking
<< extend >>
Consultar dados do candidato
Empresa
Publicar vaga
Renovar vaga


Consultar estatísticas
Emitir relatórios
Administrador
<< extend >>
Bloquear usuário
Pesquisar usuários
<< extend >>
Desbloquear usuário

1.3.2 Lista e classificação

Casos de uso em ordem dos mais interessantes para os menos interessantes classificados baseado na complexidade de implementação.

Legenda de classificacão:

1 - Muito fácil, 2 - Fácil , 3 - Médio, 4 - Complexo, 5 - Muito complexo.

- **Publicar vagas** - Complexidade 2
- **Listar vagas publicadas** - Complexidade 2
- **Responder a uma vaga** - Complexidade 3
- **Listas vagas respondidas** - Complexidade 2
- **Pesquisar vagas** - Complexidade 2
- **Manter conta** - Complexidade 3
- **Consultar ranking** - Complexidade 2
- **Consultar dados do candidato** - Complexidade 3
- **Renovar vaga** - Complexidade 3
- **Indicar uma vaga** - Complexidade 1
- **Pesquisar usuários** - Complexidade 2
- **Bloquear usuário** - Complexidade 3
- **Desbloquear usuário** - Complexidade 3
- **Mover vaga para arquivo morto** - Complexidade 3
- **Notificar empresa na data limite** - Complexidade 3
- **Consultar estatísticas** - Complexidade 3
- **Emitir relatórios** - Complexidade 3

1.4 Requisitos

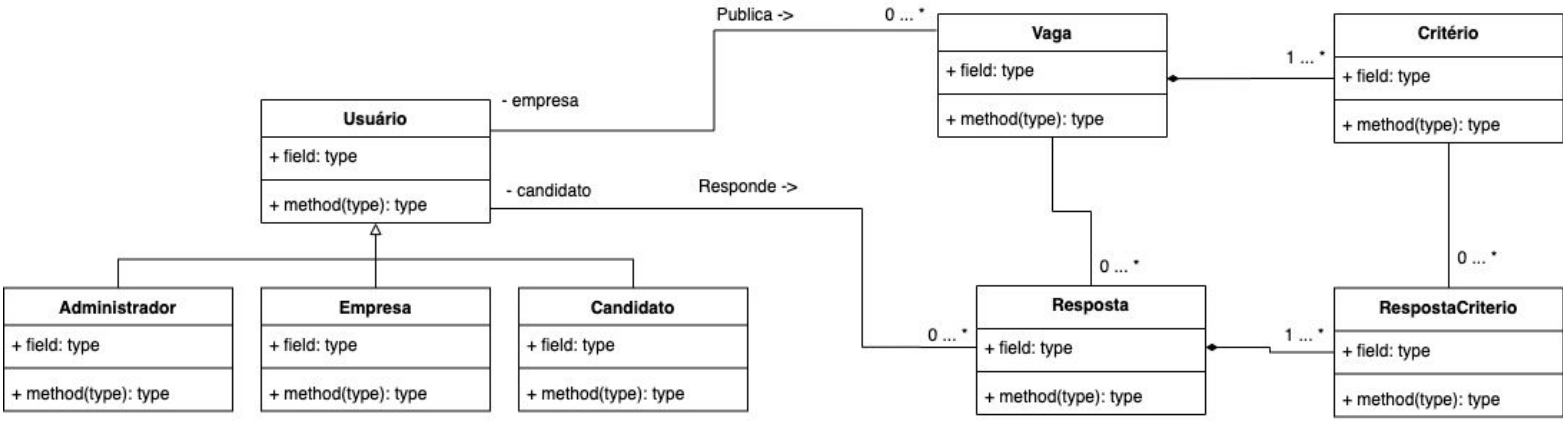
1.4.1 Matriz de requisitos

ID	Nome	Descricão	Prioridade	Complexidade
RF-01	Publicar vagas	Os usuários do tipo recrutador devem poder publicar vagas	Essencial	2
RF-02	Listar vaga publicadas	Os usuários do tipo recrutador devem poder listar suas vagas publicadas	Essencial	2
RF-03	Responder a uma vaga	Os usuários do tipo candidato devem poder responder a vagas	Importante	3
RF-04	Listas vagas respondidas	Os usuários do tipo candidato devem poder visualizar as vagas para as quais se candidatou	Essencial	2
RF-05	Pesquisar vagas	Qualquer usuário do sistema deve poder pesquisar vagas	Essencial	2
RF-06	Manter conta	Qualquer usuário do sistema deve poder se cadastrar e possuir uma conta	Desejável	3
RF-07	Consultar ranking	Os usuários do tipo recrutador devem poder consultar o ranking dos candidatos para cada uma das vagas	Essencial	2
RF-08	Consultar dados do candidato	Os usuários do tipo recrutador devem poder visualizar os dados dos candidatos do ranking	Essencial	3

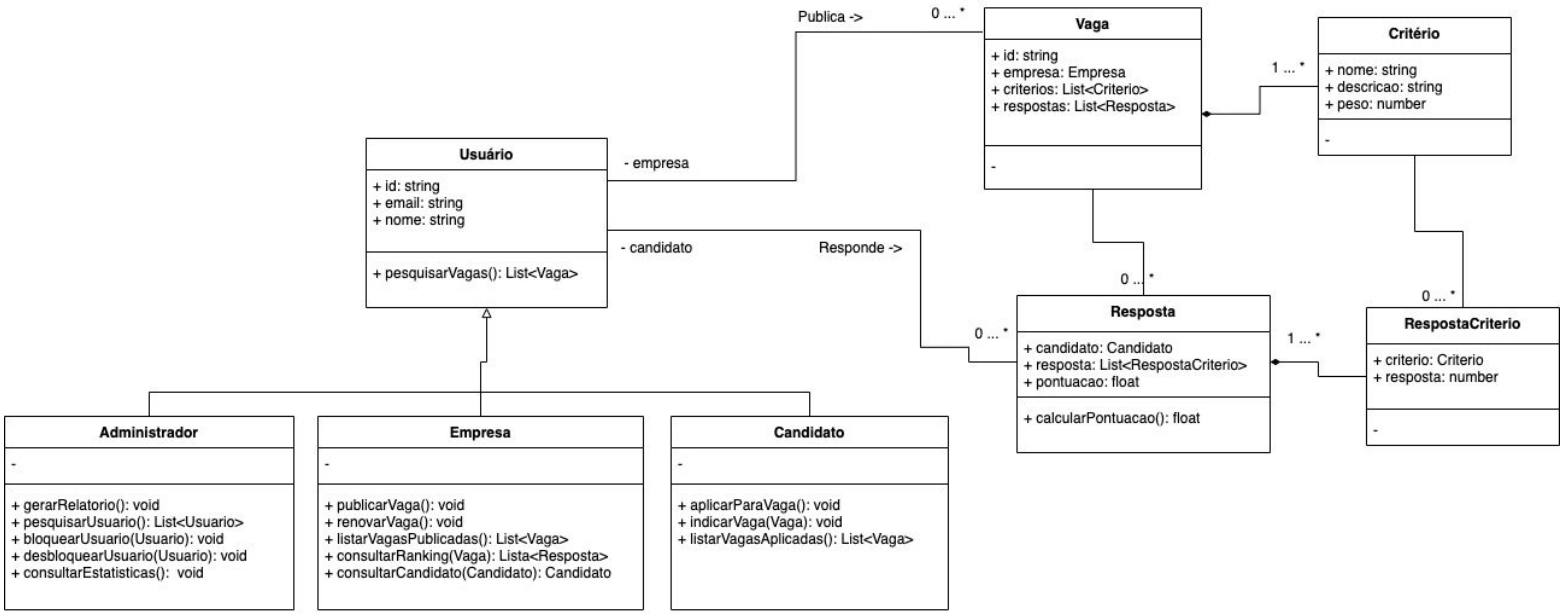
RF-09	Renovar vaga	Os usuários do tipo recrutador devem poder renovar uma vaga para que ela fique mais tempo ativa no sistema	Importante	3
RF-10	Indicar uma vaga	Os usuários do tipo candidato devem poder indicar vagas publicadas para terceiros	Desejável	1
RF-11	Pesquisar usuários	Os usuários do tipo administrador devem poder pesquisar usuários do sistema	Desejável	2
RF-12	Bloquear usuários	Os usuários do tipo administrador devem poder bloquear um usuário do sistema	Desejável	3
RF-13	Desbloquear usuários	Os usuários do tipo administrador devem poder desbloquear um usuário bloqueado do sistema	Desejável	3
RF-14	Mover vaga para arquivo morto	A entidade tempo deve mover as vagas para arquivo morto caso expire	Importante	3
RF-15	Notificar empresa na data limite	A entidade tempo deve mover as vagas para arquivo morto caso expire	Importante	3
RF-16	Consultar estatísticas	Os usuários do tipo administrador devem poder consultar as estatísticas de uso dos usuários	Importante	3
RF-17	Emitir relatórios	Os usuários do tipo administrador devem poder emitir relatórios	Essencial	3

		para os recrutadores		
NF-01	Deve ser acessível via browser	O sistema deve ser uma solução web, disponível na rede para acesso de todos por meio de um navegador	Essencial	1
NF-02	Os dados devem ser trafegados de forma segura	O sistema deve realizar as requisições HTTP de forma segura, utilizando HTTPS	Desejável	2

1.5 Modelo de domínio



1.6 Diagrama de classes



1.6.1 Responsabilidade dos objetos

Administrador: É responsável pela gestão da aplicação e dos seus usuários, por gerar relatórios, fazer bloqueio e desbloqueio de usuários, consultas de estatísticas.

Empresa: Possui a responsabilidade de gerir as vagas: criação e renovação. Padrão *creator* do GRASP.

Candidato: Possui a responsabilidade de fazer as aplicações para as vagas e também podem indicar vagas a terceiros.

Vaga: É responsável por ter as informações da vaga, possui uma lista de critérios e uma lista de respostas. Está atrelada a um objeto do tipo Empresa.

Critério: Objeto independente, auxiliar à Vaga.

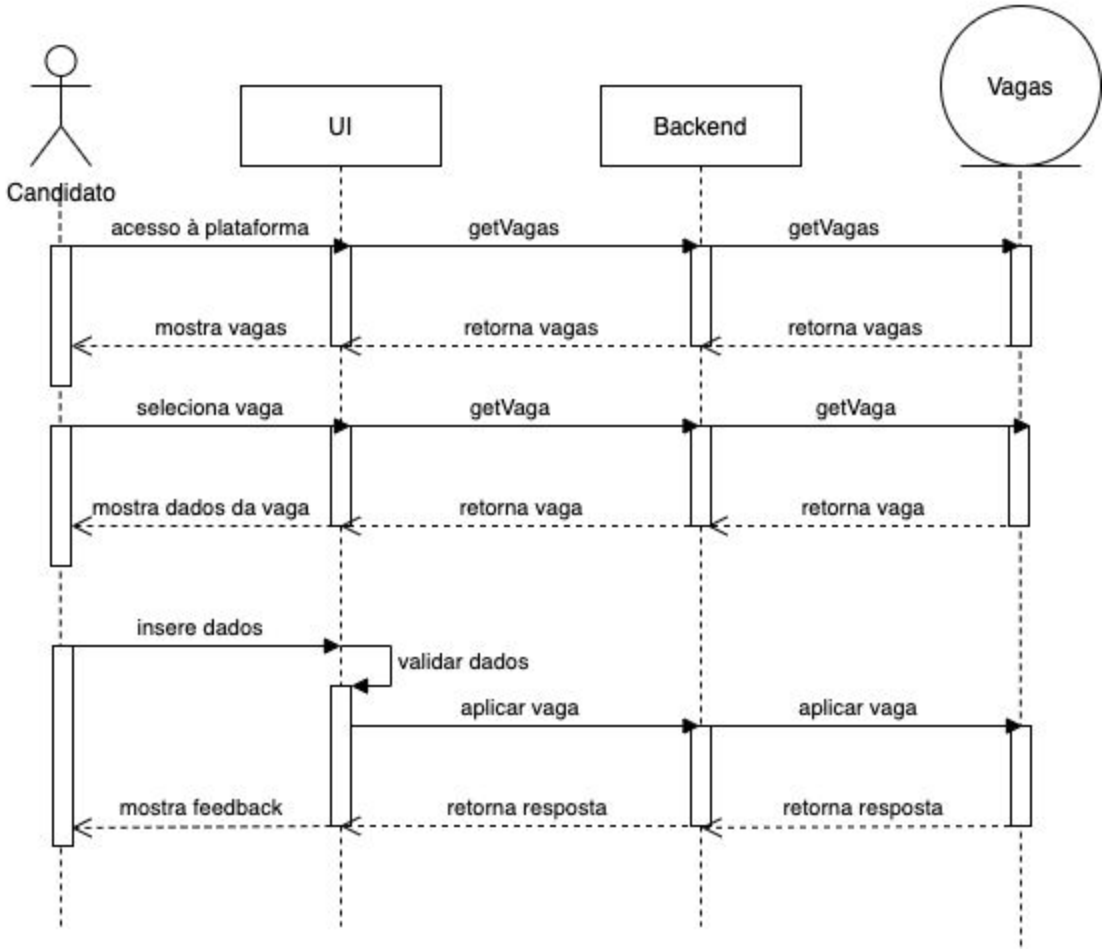
Resposta: É responsável por ter informações da resposta do usuário a uma determinada vaga e por fazer o cálculo da pontuação daquela resposta.

RespostaCritério: Objeto independente, auxiliar à Resposta.

1.6 Diagramas Sequenciais

1.6.1 Aplicar para vaga

Aplicar para vaga



2. Processo de Desenvolvimento de Software

2.1 TP1

A. Considerando o que aprendeu sobre DAD, qual seria o ciclo de vida de software mais adequado para o desenvolvimento desse projeto? Justifique a sua escolha.

O ciclo de vida **ágil**, porque é de simples entendimento e execução e as entregas são planejadas, baseado em três fases, que são a iniciação, construção e transição, que resultam em pequenas entregas de valor a cada ciclo.

B. Considerando que iremos utilizar o DAD no desenvolvimento desse projeto, quais elementos do Processo Unificado serão importados para esse projeto?

Modelo de casos de uso e modelo de diagrama de classes.

C. Quais seriam as diferentes etapas do desenvolvimento desse projeto?

Etapas de concepção, construção e transição.

D. Que elementos de desenvolvimento ágil devem ser utilizados nesse projeto?

Sprint, planning, retrospectiva e weekly.

2.2 TP9

A. Protótipo funcional do sistema:

a. Código fonte no Github;

https://github.com/luizamendes/VenturaHR

b. Respeito às melhores práticas segundo o que foi ministrado na disciplina Melhores Práticas em Codificação (nomenclatura de variáveis e classes, escopo e duração de vida das variáveis e tratamento de erros);

Trabalhei com as convenções de Javascript moderno (ES6) e também com o Typescript no mobile, além de, claro, utilizar as melhores

convenções e práticas para as tecnologias utilizadas: React, React Native e NodeJs.

Além das melhores práticas e convenções ligadas às tecnologias específicas, também procurei trazer os padrões de projeto aprendidos na disciplina de melhores práticas, como o repository pattern, testes unitários, onion architecture, entre outros.

c. Testes unitários para os métodos desenvolvidos.

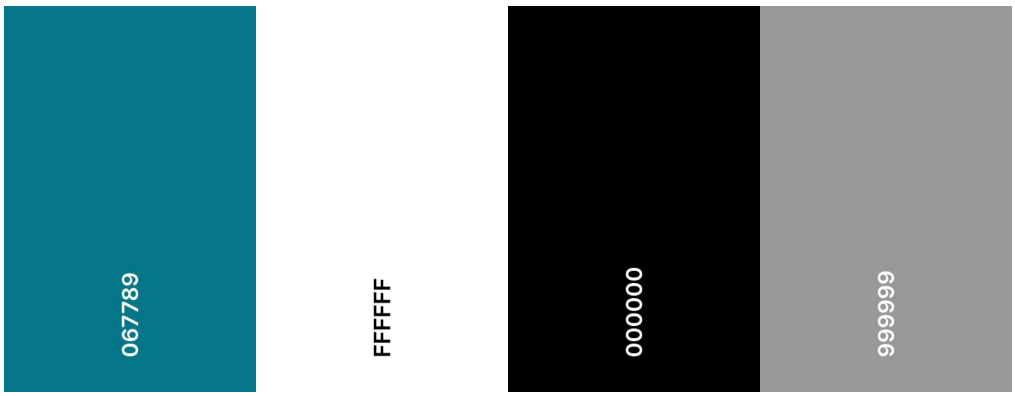
No que diz respeito a testes unitários, os que foram desenvolvidos para o projeto foram testes para o cálculo do PMD e do resultado da aplicação do usuário. Embora o desejável seja obter uma cobertura de testes acima de 85%, por conta da escassez de tempo decidi priorizar testes para esse cálculo, uma vez que o sucesso do recrutamento depende deste cálculo.

Escrevi, portanto, um teste que verifica se, dado uma lista de critérios, com seu peso e perfil desejado, o PMD foi calculado corretamente. O outro teste verifica se, dado uma resposta de candidato, o resultado daquela aplicação tinha sido calculada corretamente. São testes da classe **CriteriaCalculator**.

d. Ajuste do projeto às especificações globais: ajustes nos pontos de acesso da API e documentação, normas estéticas (cores, fontes, imagens).

Ao início do projeto, eu defini que construiria uma API Rest em NodeJs, que seria consumida por dois projetos client-side, um client web em React e um client mobile em React Native. Sendo assim, a comunicação com a API é feita através do protocolo HTTP e recebe e envia informações no formato JSON.

As normas estéticas foram definidas ao começar o projeto web, em que foi definida a seguinte paleta de cores a ser explorada:



A paleta definida é bastante simples e que tem como cor principal o verde azulado, que é uma cor que expressa confiança e segurança, características importantes para uma plataforma de recrutamento e seleção. Em relação a fonte utilizada no projeto, é uma fonte sem serifa que traz um ar de modernidade. Essa identidade visual foi seguida no projeto mobile também, para manter a unidade visual entre os dois clients.

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

Boa parte das refatorações que fiz no projeto foram relativas ao banco de dados. A maior delas foi alterar o banco para utilizar uma única tabela de usuário. A primeira versão utilizada tabelas separadas para usuários tipo Empresa e usuários tipo Candidato. Entretanto após certo tempo decidi que seria melhor unificar as tabelas, pois assim eu poderia ter usuários únicos baseado em seu e-mail e iria identificá-los a partir de uma coluna que aponta o tipo dele (empresa, candidato ou administrador). Isso facilitaria o login do usuário e também me garantiria somente um usuário por e-mail.

Faz sentido que os projetos ágeis, por serem projetos incrementais, em que o time do projeto vai entregando valor a cada ciclo de desenvolvimento, sofram refatorações ao longo do projeto, uma vez que à medida que o projeto vai crescendo os desenvolvedores vão tendo novas visões sobre pedaços antigos

de código, não só para melhoria do projeto em si, como melhoria de entendimento do código, performance e etc.

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

Padrão de repositório, para abstrair a camada de persistência da aplicação. As classes de repositório se comunicam com o repositório, de maneira a criar um baixo acoplamento e uma alta coesão.

Padrão controlador, em que classes controladoras recebem os eventos vindos do usuário, ou seja, é a borda da aplicação. Essas classes conversam com as requisições do usuário e as enviam para a próxima camada que é a de serviço.

Essa camada de serviço é uma camada para lidar com todas as regras de negócio da aplicação e pode ser considerada um padrão façade, não sendo necessário saber como ela implementa regras e acesso a dados, tendo assim como responsabilidade e objetivo de isolar as regras de negócio.

D. Justifique a importância dos testes unitários realizados até aqui.

Os testes unitários são muito importantes para qualquer base de código pois são os testes mais rápidos e fáceis de fazer e dão uma boa base de cobertura de testes para nosso código. Eu optei por fazer testes para o cálculo do PMD/resultado da candidatura pois, para mim, é um dos pontos mais importantes para o funcionamento da aplicação. Outros testes que gostaria de escrever e que são importantes para o projeto seria testar os endpoints da minha API, uma vez que são utilizados por duas pontas diferentes, o client web e o client mobile. Os testes unitários nos garantem mais segurança de que os pontos-chave estão funcionando e também mais segurança para realizar refatorações no código e ter certeza de que nada foi quebrado.

E. Identifique possíveis pontos de evolução e variação no projeto.

Alguns pontos de evolução deste projeto incluem completar todos os requisitos listados no documento de visão e que não tive tempo hábil para completar, como por exemplo, a expiração da vaga após o tempo de candidatura ter acabado, o poder de renovar uma vaga caso ela chegue ao momento de expirar e também

gerar um relatório de acessos. Outro ponto seria adicionar a função de editar ou remover uma vaga.

Do ponto de vista de interface, poderiam ser feitas melhorias de UI e UX nos frontends.

F. 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.

      O ciclo de vida utilizado foi o Disciplined Agile Delivery (DAD), um ciclo de vida **ágil**, e foi colocado em prática desde o início. Na fase de inception, fui desenvolvendo aos poucos os documentos necessários, como levantamento de requisitos e diagramas UML.

      Na fase de construção, utilizei o plano de projeto abaixo, que nos foi sugerido.

Ator - Caso de Uso		NC	Iteração	Plataforma
Usuário - Manter Conta	Ver o Fluxo do Login	3	1 - 28/09	WEB
Usuário - Logar no Site				
Empresa - Publicar Vaga	Ver o Fluxo da Empresa	4	2 - 12/10	WEB
Empresa - Listar Vagas Publicadas				
Usuário - Pesquisar Vagas		5	3 - 26/10	WEB / Mobile
Candidato - Responder Vaga	Ver o Fluxo do Candidato	4	4 - 09/11	Mobile
Candidato - Listar Vagas Respondidas				
Empresa - Consultar Ranking		3	5 - 23/11	WEB
Empresa - Consultar Dados do Candidato				

      Eu gostei bastante da sugestão, por isso segui o planejamento à risca. Procurei sempre montar a funcionalidade na API, testando o funcionamento dos endpoints por meio do Postman e após testados, a funcionalidade era montada no(s) frontend(s). Assim, o processo de

construção do projeto foi feito de forma ágil, com pequenas entregas de valor a cada iteração, que duraram 2 semanas cada.

A execução cumpriu o que foi proposto no início, de fazer um projeto ágil, com entregas incrementais de valor, dentro do prazo estipulado e com as entregas planejadas na fase da construção (vide quadro de datas/entregas acima).