Pega a Visão: Um sistema de divulgação de oportunidades para universitários

Bruno de Paiva[1], **Eduardo Guedes**[1], **Pedro Kuchpil**[1], **Rodrigo Palmeira**[1], **Tales Coutinho**[2], **Thiago Guimarães**[1]

[1]Escola Politécnica – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

[2]Departamento de Ciência da Computação – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ – Brasil

{bruno.dantas,eduardogds,pedrokuchpil,talesmoreira,thiagoguima}@poli.ufrj.br

Abstract. *This paper describes Pega a Visão, a web system made to advertise internship, scientific initiation and extension opportunities to students of the Federal University of Rio de Janeiro, made for alumns who have difficulties in finding opportunities and recruiters that can not find students with the intended profile due to lack of intelligence in the current process.*

Resumo. *Este artigo descreve o Pega a Visão, um sistema web para a divulgação de vagas de estágio, iniciação científica e extensão para a Universidade Federal do Rio de Janeiro, voltado para alunos que possuem dificuldades de encontrar oportunidades e recrutadores que não conseguem encontrar estudantes com o perfil desejado pela falta de inteligência no processo atual.*

1. Introdução

É bem estabelecido entre os alunos da Universidade Federal do Rio de Janeiro (UFRJ) que as vagas de estágio, iniciação científica e extensão ofertadas não são bem divulgadas. Entre papéis grampeados nos incontáveis murais nos corredores dos campi, que podem muito bem datar de anos, e e-mails que ficam perdidos na caixa de entrada em meio a diversas oportunidades, um estudante que está procurando uma oportunidade do tipo se sente perdido e desamparado pela universidade.

Os recrutadores também são prejudicados com esse processo confuso. Muitas vezes, dependem da boa vontade de professores para encaminhar suas vagas para os alunos de perfil desejado, ou então acabam na caixa de entrada de estudantes de cursos que fogem totalmente do escopo da vaga proposta. Um sistema que os ajude será de grande valor para a empresa, laboratório ou projeto em que trabalham.

Dessa forma, foi identificada a necessidade de um sistema que seja um agregador de oportunidades de estágio, iniciação científica e extensão para alunos da universidade pelos membros da CorchoForce. Foi desenvolvido então um Mínimo Produto Viável (MVP) dentro do escopo da disciplina de Programação Avançada (EEL418), da Escola Politécnica da UFRJ, e orientado pelo professor Cláudio Miceli de Farias, do Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ.

O relatório aborda a pesquisa de mercado realizada pela equipe na Seção 2, descreve o funcionamento do sistema na Seção 3, detalha a arquitetura do mesmo na Seção 4, aponta os passos futuros do projeto na Seção 5 e apresenta as conclusões na Seção 6.

2. Pesquisa de Mercado

Para dar suporte às impressões da equipe, foi realizada uma pesquisa de mercado com 10 alunos da universidade de diversos centros especificamente sobre as oportunidades de iniciação científica, que são abrangidas pelo Pega a Visão. Durante as entrevistas, podemos destacar três pontos:

- Dificuldades com forma de divulgação científica: Cerca de 83% dos entrevistados acordaram que existem problemas em como a divulgação científica é feita na UFRJ. Muitos apresentaram que a única forma de divulgação científica era a JIC-TAC e outros apresentaram também que esta não divulga ciência da forma que deve ser feita (acesso ao público e fácil entendimento para pessoas que desconhecem a temática).
- Dificuldades com relação à divulgação de vagas de iniciação científica: 100% dos entrevistados afirmaram que a UFRJ deveria ter algum sistema de divulgação das vagas ofertada pelos professores, enquanto uns afirmaram que tinha um roteiro seguido por alguns professores, que se baseiam inicialmente em buscar por recomendações dentro do laboratório e só num processo final divulgar esta vaga.
- Os requisitos deveriam ser definidos pelo próprio professor: Grande parte dos alunos afirmaram que os professores deveriam ser inteiramente responsáveis pelo critério de inclusão do aluno à vaga de iniciação científica.

Como observado, todo o processo de divulgação de iniciação científica é alvo de críticas dos alunos. O Pega a Visão almeja dar suporte tanto a estudantes quanto a professores da universidade, sendo um facilitador para as partes envolvidas na hora de cadastrar e achar oportunidades do tipo.

3. Funcionamento do Pega a Visão

O sistema disponibiliza oportunidades enviadas e as mostra para os alunos interessados. Atualmente, no MVP construído, é necessário contactar o time de desenvolvimento para cadastrar uma nova vaga, o que será alterado no produto final. Cada oportunidade tem seu tipo (estágio, iniciação científica, extensão ou outro), descrição da vaga, requisitos, valor da bolsa, local de trabalho, carga horária e contato do recrutador. Elas são demonstradas na página inicial com um card para cada e, afim de manter uma experiência simplificada e intuitiva com o usuário, apenas as informações principais da vaga estão contidas em seu card. Caso o usuário se interesse ou queira saber mais detalhes sobre a oportunidade, ao clicar na vaga desejada o usuário é redirecionado à página da oportunidade que contém todas as informações disponíveis sobre ela. A Figura 1 mostra a tela inicial da aplicação, com todas as oportunidades disponíveis, enquanto a Figura 2 apresenta o resultado de abrir uma oportunidade para ver mais detalhes da mesma.

4. Arquitetura do Sistema

Para o sistema, decidimos utilizar os princípios SOLID, em detrimento do IDEALS e do DDD. Como o primeiro é voltado para microsserviços, ele não é adequado para o projeto no momento. Decidimos também por não utilizar o segundo pois sua grande vantagem é estabelecer uma comunicação produtiva entre os detentores do domínio e os engenheiros de software. No entanto, esses papéis estão misturados no Pega a Visão.

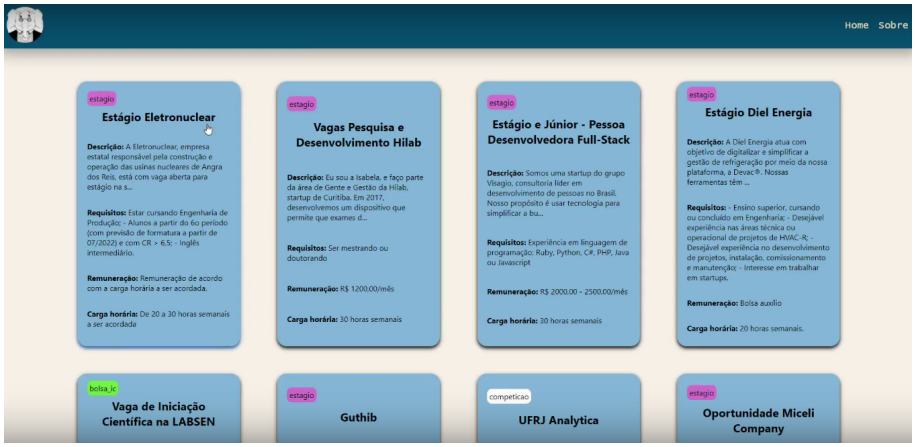

Figura 1. Lista de Oportunidades no /Home

Figura 2. Oportunidade após o usuário clicar nela

4.1. Front-end

O esquema de cores do Pega a Visão foi pensado para demonstrar suavidade e apartar-se de outros modelos utilizados da UFRJ, mostrando a identidade de uma ferramenta por alunos para alunos. Os *cards* são dispostos de uma forma lúdica, permitindo o usuário utilizar a plataforma pelo tempo necessário, sem sentir uma sobrecarga de informações e estímulos visuais. Sobre os elementos que compõem a página, todos foram desenvolvidos para serem responsivos ao dispositivo e resolução do usuário.

O projeto foi feito com o *Create React App*, pacote *Node* que cria a base de uma SPA React. O React renderiza a root, que no projeto está dentro da pasta de components. A root é a div raíz da aplicação, dentro da qual terá o conteúdo das páginas. Nela, temos os componentes de layout fixos (no nosso caso, a barra de navegação e o footer), e um switch, que, dependendo da rota do navegador, carrega um componente em específico. Ainda dentro desse diretório, temos várias subpastas, cada uma se referindo a um componente. Dentro delas existem dois arquivos, um .jsx e outro .module.css, sendo o primeiro o arquivo que definirá o comportamento de fato desse componente e o segundo que definirá a estilização do mesmo. Os diretórios do front-end estão presentes na Figura 3.

A ideia é que as páginas, que estão dentro dentro do diretório *pages* e que para

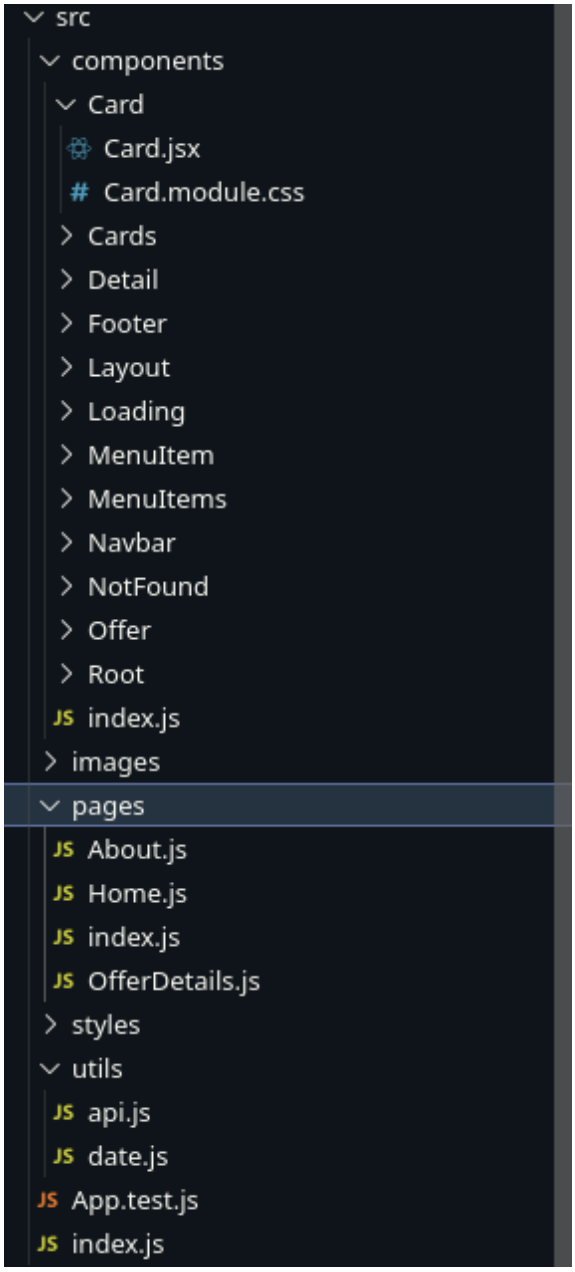

Figura 3. Estrutura do front-end do sistema

cada uma existe uma rota distinta, utilizem os componentes de acordo com sua demanda. Cada componente sempre terá uma única responsabilidade, seguindo o Single Responsability Principle. Dentro de *images* estão as imagens utilizadas pela aplicação, e em *styles*, se encontram um arquivo de estilização padrão do projeto, que define o padrão de fonte, cor do plano de fundo, entre outras características, e outro . Por enquanto, apenas a página ”About” necessitou dessa estilização própria. Os arquivos .js que exportam funções que serão úteis a longo do projeto, como o api.js, que retorna funções que possibilita o consumo de dados da API fornecida pelo back-end, estão no diretório *utils*. Por fim, o arquivo App.test.js realiza testes de renderezição.

Enquanto o usuário estiver navegando pela SPA, o front armazena os dados das

ofertas na memória de sessão do navegador, para que ele não tenha que consultar a API toda a vez que ele clicar numa oportunidade e voltar à página inicial, por exemplo.

4.2. Back-end

O back-end do sistema é uma API RESTful feita para ser consumida pelo front-end de forma eficiente e manejar o estado permanente da aplicação, comunicação que é feita utilizando JSON. Ele foi feito em JavaScript, utilizando-se da biblioteca Express, escolhida por ser uma das mais utilizadas e possuir uma boa documentação disponível. Também é bastante leve e não determina uma estrutura de código prévia, e assim é possível organizar o mesmo de tal forma que siga boas práticas, que em nosso caso, são os princípios SOLID. Para a comunicação com o banco de dados não-relacional MongoDB, escolhido por sua praticidade e orientação do professor, foi utilizado o ODM Mongoose. Esta é a tecnologia mais utilizada para isso pela comunidade *Node*, e assim, é amplamente testada e documentada. No entanto, também decidimos por não utilizar a validação do Mongoose, pois desta forma feriríamos os princípios SOLID, visto que faríamos a associação das regras de negócio a uma biblioteca, além de misturar validação com a comunicação com o banco de dados. A estrutura do sistema é mostrada na Figura 4.

No diretório *config*, são guardados os arquivos contendo configurações que não estão no .env (que fica fora do *src* e guarda variáveis de ambiente ou que precisam ser parseadas primeiro). Já os scripts ligados ao banco de dados são armazenados no diretório *db*. Atualmente, há um que inicia o banco de dados na memória para testes (tema que será aprofundado em outra seção) e um para *seedar* (inserir dados de teste) no banco de dados. Os arquivos do diretório *loaders* inicializam o necessário para iniciar o servidor que, no caso do Pega a Visão, são o Express e o Mongoose. O diretório *models* possui os modelos necessários para a utilização do ODM Mongoose. Os arquivos referentes às rotas e as "*controllers*" estão no diretório *routes*. No diretório de *services* se encontram as regras de negócio devidamente isoladas, como preconizado pelos princípios SOLID. Por enquanto, não há nenhuma, pois a API funciona apenas como uma CRUD. Porém, ao avançarmos no projeto, devemos adicionar autenticação e validação. No *utils*, se encontram os arquivos de utilidade que não fazem parte das regras de negócio. Os diretórios de logs e middlewares são autoexplicativos. Por fim, o arquivo app.js invoca tudo e inicializa o servidor do sistema.

Para os testes do sistema, teríamos que testar uma CRUD simples, cuja principal complexidade é a conexão com o banco de dados. Assim, decidimos por rodar o banco de dados MongoDB em memória para realizar os testes. Desta forma, conseguimos ter uma boa cobertura e a execução dos testes é rápida. Não temos testes para as regras de negócio pois, como já explicitado, elas ainda não se encontram no back-end. As bibliotecas utilizadas para testes foram o Mocha e o Chai, também amplamente usadas e documentadas pela comunidade.

Já para o CI/CD, criamos um docker composer de desenvolvimento para rodar o node e o banco de dados. Poucas alterações seriam necessárias para criar uma imagem de produção e, nesse caso, não teríamos uma imagem para o banco de dados, e sin utilizaríamos de um cluster externo para maior estabilidade, inclusive podendo subir vários conteiners do node. Criamos *GitHub actions* para testar o código antes de mergear com as branchs de interesse (stage e main). Assim, será fácil de integrar com o ambiente de produção, que testa e deploya automaticamente o código, criando uma *pipeline* CI/CD.

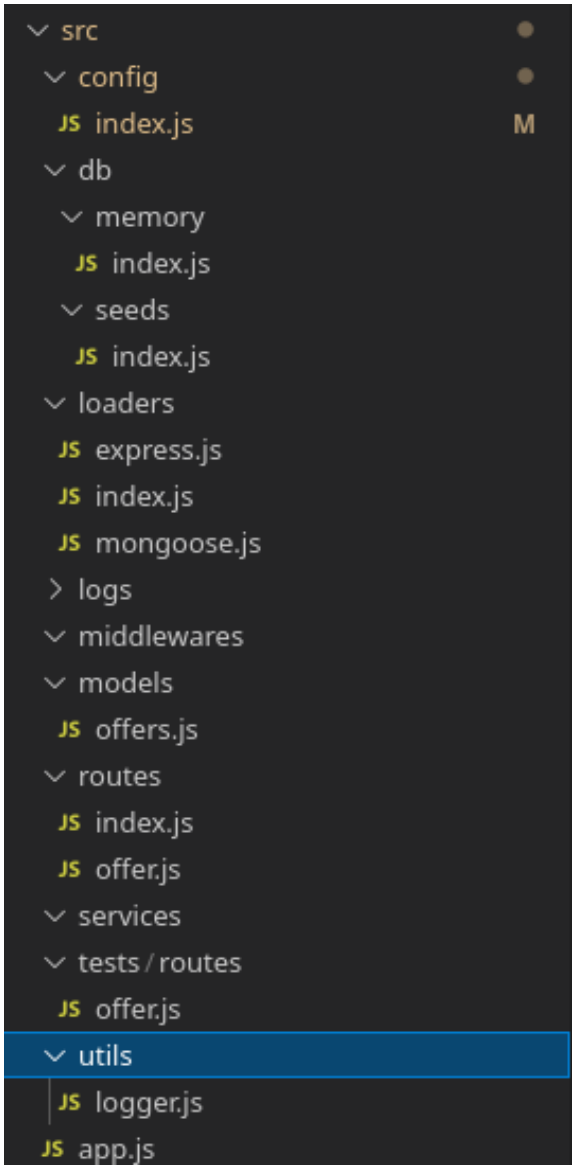

Figura 4. Estrutura do back-end do sistema

Por fim, para a padronização do código, é utilizada a biblioteca *Prettier*, que formata o código de forma uniforme sempre antes de um commit, o que facilita a leitura e o entendimento do mesmo.

5. Passos Futuros

Após tendo conhecido melhor sobre o projeto e entendido melhor sobre as demandas de usuário, definimos alguns próximos passos até que o produto possa ser utilizado por toda a comunidade da UFRJ e até mesmo ser expandida para outras universidades.

- Criação de uma barra de busca: É necessário que o usuário possa pesquisar dentre todas as ofertas existentes para que ele consiga encontrar facilmente a oportunidade que ele melhor se encaixa.
- Criação de uma interface de Login: Para que um ofertante (professor, empresa, coordenador, etc) possa ofertar sua vaga, é interessante que ele possua uma área

própria e consiga divulgar sua oferta. Além disso, para o aluno a possibilidade de logar e receber notificações a respeito das vagas que ele já se candidatou.
- Integração com os dados da universidade: É interessante, em nossa visão, que o aluno possa acessar o siga junto às ofertas, assim podemos sugerir vagas que se encaixem melhor com as habilidades dele ou requisitos da própria vaga.
- Melhoria na disponibilização das oportunidades: É notável que a forma que as ofertas estão aparecendo ao usuário toma bastante espaço e não é agradável aos olhos. Desta forma temos a ideia em mente de modificar a forma que essas oportunidades são disponibilizadas, a fim de demonstrar somente as vagas mais recentes na home e criar uma outra rota para as outras vagas.

6. Conclusão

Considerando a dificuldade dos alunos de encontrar oportunidades, sejam na academia ou em empresas, muitos destes tendem a encontrar dificuldades em se incluir no mercado de trabalho ou cumprir suas horas obrigatórias dentro da própria universidade. Neste trabalho, há uma proposição de uma plataforma web (SPA) onde os professores, empresas, coordenadores e alunos poderam ter contatos diretamente entre si, permitindo que essa dificuldade seja eliminada.

Neste MVP, grande parte da idealização deste produto foi implementada, onde o usuário poderá acessar as vagas disponibilizadas e por meio de email ou outra fonte de contato do próprio ofertante, este poderá se candidatar a vaga escolhida.

Todo o repositório de arquivos se encontram no GitHub da organização.