UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FELIPE MOTA DOS SANTOS

UMA APLICAÇÃO WEB PARA ORGANIZAR PROGRAMAS DE PESQUISA EM ENGENHARIA DE SOFTWARE SEGUINDO MODELO ABC

CAMPINA GRANDE - PB

2021

FELIPE MOTA DOS SANTOS

UMA APLICAÇÃO WEB PARA ORGANIZAR PROGRAMAS DE PESQUISA EM ENGENHARIA DE SOFTWARE SEGUINDO MODELO ABC

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professora Tiago Lima Massoni

CAMPINA GRANDE - PB

2021

S237a Santos, Felipe Mota dos.
Uma aplicação web para organizar programas de pesquisa em Engenharia de Software seguindo modelo ABC. / Felipe Mota dos Santos. – 2021.

10 f.

Orientador: Prof. Dr. Tiago Lima Massoni.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Desenvolvimento de software. 2. Aplicação web. 3. Pesquisa em Engenharia de Software. 3. Engenharia de Software. 4. Programas de pesquisa. 5. Metodologia de pesquisa. 6. Modelo ABC. 7. Framework ABC. I. Massoni, Tiago Lima. II. Título.

CDU:004.41(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

FELIPE MOTA DOS SANTOS

UMA APLICAÇÃO WEB PARA ORGANIZAR PROGRAMAS DE PESQUISA EM ENGENHARIA DE SOFTWARE SEGUINDO MODELO ABC

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Tiago Lima Massoni

Orientador – UASC/CEEI/UFCG

Professora Wilkerson de Lucena Andrade

Examinador – UASC/CEEI/UFCG

Professor Cláudio Elizio Calazans Campelo

Examinador extra – UASC/CEEI/UFCG

Trabalho aprovado em: 25 de Maio de 2021

CAMPINA GRANDE - PB

ABSTRACT

During the realization of a research program there are several problems, among them that of organizing the studies carried out, as well as choosing which methods and methodologies will be used. Even with the increase in the number of researches in the area of software engineering, there is still a lack of a tool to facilitate the process of conducting research programs in this area. With this in mind, we propose a tool in the web application format following the ABC model, which was devised by Klaas-Jan Stol and Brian Fitzgerald. With the tool, the process of organizing studies will be facilitated and the way of visualizing the studies carried out will be improved.

Uma aplicação web para organizar programas de pesquisa em engenharia de software seguindo modelo ABC

Felipe Mota dos Santos
Federal University of Campina Grande
felipe.santos@ccc.ufcg.edu.br

Tiago Massoni
Federal University of Campina Grande
massoni@dsc.ufcg.edu.br

RESUMO

Durante a realização de um programa de pesquisa encontram-se diversos problemas, entre eles o de organizar os estudos realizados, como também escolher quais métodos e metodologias serão utilizados. Mesmo com o aumento no número de pesquisas na área da engenharia de software, ainda se vê a falta de uma ferramenta que facilite o processo da realização de programas de pesquisa nessa área. Pensando nisso, propõe-se uma ferramenta no formato de aplicação web seguindo o modelo ABC, que foi idealizado por Klaas-Jan Stol e Brian Fitzgerald. Com a ferramenta será facilitado o processo de organização de estudos e melhorado a forma de visualização dos estudos realizados.

Palavras-chave

Aplicação web, programas de pesquisa, metodologia de pesquisa.

1. INTRODUÇÃO

O número de pesquisas dentro da área da Engenharia de Software tem crescido nos últimos anos [19], tendo em vista esse crescimento, é interessante destacar a falta de uma organização referente a metodologia de pesquisas que podem ser utilizadas e quais as suas vantagens e desvantagens. Os resultados de uma pesquisa apresentam características diferentes dependendo da metodologia que é aplicada, por exemplo a metodologia estudo delphi permite uma reflexão dos participantes sobre as perguntas devido às diversas etapas necessárias, em contrapartida, uma pesquisa por amostragem utilizando questionário obtêm resultados com um grau menor de reflexão dos participantes, em torno das perguntas, no entanto, os resultados podem ser utilizados para a realização de extrapolações.

Apesar de ser possível fazer uma listagem de metodologias existentes que podem ser utilizadas, uma simples listagem não permitiria uma visualização de como as diferentes metodologias se completam e como seria possível enriquecer um programa de pesquisa adicionando um novo estudo por meio de outra metodologia. Diante desse cenário, Klaas-Jan Stol e Brian Fitzgerald propõem no Guia para condução de pesquisas de Engenharia de Software [1] o modelo ABC, que é um modelo que permite classificar e organizar pesquisas realizadas com base na metodologia que foi utilizada. O modelo se estrutura a partir da definição de oito estratégias de pesquisa; cada estratégia engloba metodologias de pesquisas que possuem um alto nível de similaridade em termos de benefícios e execução. Além disso, as estratégias também se distinguem em termos de intrusividade no cenário que foi pesquisado, quão genérico o resultado obtido é, além das configurações do ambiente em que a pesquisa foi realizada. Essas estratégias são então dispostas num gráfico de 2 dimensões, sendo os eixos referentes a intrusividade e generalização, cada estratégia sendo um setor de um círculo, todas tendo o mesmo tamanho, como é representado pela Fig 1. Com esta figura gráfica é possível então definir uma posição para cada estudo e com isso ter uma compreensão de como os resultados obtidos podem ser utilizados e de qual impacto um novo estudo pode ter.

No entanto, a utilização do *framework ABC* acaba por ser dificultada devido a falta de uma forma simplificada, centralizada e de fácil acesso para organizar os programas de pesquisa e estudos, a partir disso esse artigo propõe *uma implementação do framework ABC como uma aplicação web,* que é aberta para qualquer usuário e permitindo que este consiga ter uma visualização sobre o seu programa de pesquisa e como cada estudo impacta no resultado, além de se beneficiar de um sistema que facilita o gerenciamento dos estudos.

2. SOLUÇÃO

O objetivo deste artigo é o desenvolvimento de uma ferramenta, no formato de aplicação web. Este formato foi escolhido pelo seu nível de acessibilidade para os usuários além de simplicidade de uso. A aplicação permite que o usuário registre seus programas de pesquisa e estudos, consiga visualizar de forma gráfica seguindo o modelo ABC a distribuição dos estudos realizados e seja capaz de compreender o que pode ser feito para aprimorar os resultados da pesquisa.

2.1 Fundamentação teórica

O *framework ABC* [1] propõe formas de classificar pesquisas e estudos empíricos em Engenharia de Software.Nesta área, é possível dividir as pesquisas em 2 tipos: pesquisas que buscam uma solução e pesquisas que buscam conhecimento, no primeiro é esperado que seja gerado algo em um formato de artefato para resolução de um problema, já no segundo se busca gerar novo conhecimento que irá compor a base teórica para uso futuro, tanto

para pesquisas em busca de solução quanto pesquisas em busca de conhecimento.

O *framework ABC* se limita para pesquisas que buscam conhecimento, no entanto, também é possível ser incrementado de forma que se torne um *framework* mais genérico. Para classificar as pesquisas o *framework* utiliza 3 características presentes em qualquer tipo de pesquisa, sendo intrusão no tópico, possibilidade de generalização e configuração do ambiente da pesquisa. Com essas 3 características são gerados 8 segmentos diferentes de pesquisa, denominados de estratégias de pesquisa, cada estratégia é composta por diferentes métodos de pesquisa, sendo que os métodos de uma mesma estratégia tem vantagens e desvantagens similares.

A Figura 1 é uma demonstração visual do posicionamento das estratégias de pesquisa. Em relação ao eixo x quanto mais para a direita mais específico o método utilizado é, em oposição, quanto mais para esquerda mais genérico, onde ser genérico é um indicativo de ser possível generalizar os resultados para outros contextos. Em relação ao eixo y, um posicionamento na parte superior indica um nível maior de intrusão, enquanto na parte inferior um nível menor. Além disso, estão presentes 4 quadrantes que dividem a figura igualmente com um deslocamento de 45 graus a partir do centro.

Os quadrantes são:

- Q1: representa uma configuração de ambiente natural onde existe um baixo ou inexistente nível de intervenção do pesquisador no ambiente;
- Q2: representa uma configuração de ambiente artificial, onde o pesquisador exerce um nível de controle maior sobre o ambiente de pesquisa;
- Q3: representa uma configuração de ambiente neutra, onde o ambiente não tem tanto impacto em relação ao resultado da pesquisa;
- Q4: representa um configuração de ambiente não empírica, onde não há nenhum nível de intrusão do pesquisador em relação a pontos empíricos.

As oito estratégias são:

- **Experimento de campo**: utilizado para realizar a análise dos efeitos de manipulações em um cenário real, por exemplo uma pesquisa-ação sobre problemas de comunicação entre desenvolvedores de projetos de uma empresa;
- **Simulação Experimental**: utilizada para buscar balancear a medida de comportamentos emergentes de cenários reais e conseguir obter um grau de realismo, por exemplo a simulação de cenários da indústria em ambientes acadêmicos, como o desenvolvimento de aplicações de software;
- **Experimento de laboratório**: utilizado quando se requer um alto nível de precisão sobre o que está sendo pesquisado e analisado, por exemplo avaliação da performance de algoritmos em cenários reais via experimentos;
- **Estudos de julgamento**: utilizada quando se busca obter um grau de generalização sobre respostas de participantes em uma determinada pesquisa, por exemplo uma execução de um estudo delphi investigando aspectos presentes nas escolhas de ferramentas por engenheiros de software;
- **Estudos de amostra**: utilizada quando se busca obter o maior grau de generalizabilidade sobre um população, pode ser essa população desenvolvedores, gerentes, artefatos de software e projetos, como no caso da execução de questionários;
- **Teoria formal**: utilizada quando a área de interesse possui um grande número de estudos empíricos, mas sem um estudo que integre as descobertas, por exemplo a junção de várias observações para criar uma nova teoria sobre um determinado algoritmo;
- **Simulação por computador**: utilizada quando se realiza criação de um modelo para representar e simular um sistema ou fenômeno do mundo real, que em geral seria complicado e custoso de ser observado se não fosse por simulação, como no caso de simulações por computador para prever o comportamento da chuva e mudanças climáticas;
- **Pesquisa de campo**: utilizada para realizar o estudo de fenômenos em ambientes naturais com intuito de determinar o que acontece e como acontece, como no caso de um estudo de caso em determinados projetos ou organizações de software.

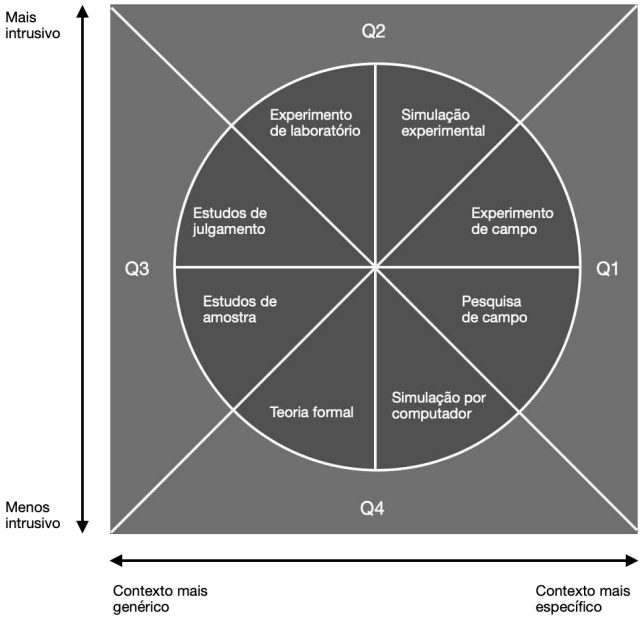

Figura 1 - Visualização gráfica do modelo ABC, tendo como eixos nível de generalização e nível de intrusividade. Dividindo um círculo em 8 setores de mesmo tamanho.

2.2 Funcionalidades

Na primeira página da aplicação web, o usuário pode utilizar o signup/login na aplicação apenas por meio de Google Oauth [2], assim todas as outras páginas requerem que o usuário esteja autenticado devido a serem relacionadas aos dados de um usuário. Após a realização do login, o usuário poderá cadastrar um programa de pesquisa, após o qual será possível realizar o

cadastro de estudo dentro do programa de pesquisa. Todas as informações que o usuário cadastrar poderão ser alteradas ou deletadas por meio da aplicação. A exibição dos estudos cadastrados é feita por meio do gráfico de estratégias do modelo ABC.

2.3 Arquitetura

O projeto foi idealizado como uma aplicação web de acesso aberto, tendo como base uma arquitetura cliente-servidor [3]. De forma abstrata, na arquitetura cliente-servidor, existe um servidor que provê os recursos aos clientes, assim basta o cliente requisitar o recurso que é desejado. No contexto de desenvolvimento de aplicações web, é muito comum considerar duas camadas, *front end* e *back end*, o *front end* é responsável pela estrutura e definição das interações com o usuário, nas páginas a serem exibidas, enquanto o *back end* é responsável pela lógica do negócio e tratamento dos dados.

2.3.1 Persistência dos dados

Em conjunto à arquitetura cliente-servidor é necessário definir como os dados serão persistidos, existem diversas alternativas, uma delas é in-memory [4] onde os dados ficariam armazenados na memória principal do servidor ao invés de arquivos. Apesar de ser simples, esta alternativa ofereceria problemas no caso do servidor ficar fora do ar e em necessidade de alterações na aplicação que define o servidor. Para não correr riscos com perda de dados, buscando uma simplicidade na hora de desenvolvimento e devido à experiências passadas foi escolhido o MongoDB [5] como banco de dados, sendo utilizado o MongoDB Atlas [6] como serviço que provê o banco de dados. A aplicação utiliza o serviço gratuito do Atlas, que permite até 500MB de armazenamento gratuito.

2.3.2 Tecnologias do back-end

O servidor foi implementado utilizando Node.js [7] como ambiente de execução em conjunto com o *framework* Express [8] para a estruturação dos recursos a serem servidos, além disso foi utilizado a ferramenta Mongoose [9] para realizar a conexão com o banco de dados e modelagem dos dados que deveriam ser armazenados. Para realizar a validação dos dados enviados em requisições foi utilizado o pacote express-validator [10]. Tais escolhas foram feitas levando em conta a praticidade para iniciar o projeto e o desenvolvimento da aplicação, além de experiência prévia nas tecnologias escolhidas, assim agilizando o desenvolvimento.

2.3.3 Estrutura do back-end

A organização da estrutura do back-end foi feita tendo como base prévios projetos e boas práticas consideradas pela comunidade de desenvolvedores. Considerando isso, a estrutura foi dividida em 6 diretórios: *controllers*, *helpers*, *middlewares*, *models*, *routes* e *utils*, como apresentado na Figura 2.

Os *models* são um conjunto de modelos que definem as estruturas que serão armazenadas no banco de dados, seguindo o padrão do Mongoose e permitindo um acesso facilitado aos dados que já estão persistidos no banco de dados. A aplicação é composta por 3 modelos:

- *ResearchProgram*, para a representação de programas de pesquisa. Cada programa de pesquisa é composto por vários estudos (*researches)* e cada estudo tem sua respectiva estratégia que deve ser uma das 8 listadas na Figura 1.
- *User*, para a representação dos usuários cadastrados.
- *TutorialInfo*, para a representação das informações que compõem o tutorial.

A Figura 3 mostra o modelo de representação de um programa de pesquisa.

Os *helpers* são um conjunto de funções utilitárias que representam operações importantes e divididas dos outros arquivos para simplificar a leitura do código e organização. Na aplicação é utilizado um helper para conexão com o banco de dados e outro para permitir que o servidor comece a escutar por requisições.

Os *utils* são um conjunto de funções utilitárias para reduzir repetição de código e deixar o código mais legível.

Os *controllers* são um conjunto de módulos que possuem os métodos para resolução de chamadas mais complexas da API, para isso combinando lógica de negócio e comunicação com os *models.*

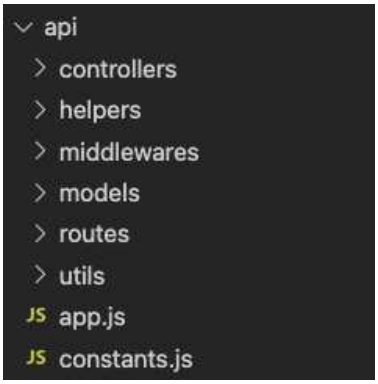

Figura 2 - Organização dos arquivos do back end

```
const ResearchProgram = mongoose.model('ResearchProgram', {
  title: String,
  description: String,
  ownerId: String,
  researches: [{
    title: String,
    description: String,
    strategy: {
      type: String,
      enum: strategies,
    },
    link: String,
    method: String,
  }],
});
```

Figura 3 - Modelo para representar um programa de pesquisa seguindo o padrão proposto pelo Mongoose.

Os *middlewares* são um conjunto de métodos utilizados como interceptores de requests que são executados antes da chegada da chamada ao seu respectivo *controller*. Na aplicação foram utilizados 3 middlewares. O *verifyToken* é um *middleware* para validar o token de autorização provido pela requisição do cliente, garantindo assim que requests não autenticados não sejam aceitos. O *verifyOwner* é um *middleware* para verificar que o usuário que fez uma requisição não está acessando dados que são relativos a outro usuário, garantindo assim que apenas o usuário de um dado possa acessar ele pela API. O *verifyBody* é um *middleware* para verificar se os requests enviados possuem um corpo com campos que possuem valores válidos para permitir a execução do request.

As *routes* são um conjunto de definições que indicam os endpoints da API, quais os middlewares que devem ser utilizados, qual o formato do corpo do request é esperado e qual o controller deve ser invocado para o respectivo endpoint. A Figura 4 demonstra a descrição da rota POST */programs*, onde é possível ver o uso dos *middlewares verifyToken* e *verifyBody*, a descrição de campos *title e description* do *body* utilizando o pacote *express-validator* e a delegação da chamada para o controller *createResearchProgram*.

```
router.use(verifyToken);
router.post(
  '/programs',
  [
    body('title')
      .trim()
      .notEmpty()
      .isLength(PROGRAM_TITLE_LENGTH),
    body('description')
      .trim()
      .notEmpty()
      .isLength(PROGRAM_DESCRIPTION_LENGTH),
  ],
  verifyBody,
  createResearchProgram
);
```

Figura 4 - Exemplo de descrição de *route POST /programs*, que permite a criação de novos programas de pesquisa.

2.3.4 *Autenticação*

A autenticação foi feita utilizando o serviço a parte Google OAuth, com este serviço é possível realizar o signup/login no sistema sem precisar definir um usuário e senha, apenas com o conta do autenticação do Google é possível obter o email do usuário do Google e a partir deste criar um novo usuário para a aplicação, tudo isto é feito sem que o usuário note, tornando a experiência de signup/login mais simples e fácil. Além disso, o sistema evita possíveis questões de seguranças que surgiriam caso o usuário tivesse que escolher definir sua senha, além de toda a necessidade de tratamento de casos como a perda de senha, tornando assim o sistema compacto sem perda de segurança. A Figura 4 demonstra o fluxo de autenticação da aplicação, colocando o Google OAuth como um serviço à parte e como as informações vão do cliente para o servidor por meio de um token.

2.3.5 *Tecnologias do front-end*

Para implementação do cliente foi utilizada a ferramenta de construção de visuais e design Plasmic [11], que permite gerar componentes seguindo o formato da biblioteca React [12] a partir de um design gráfico do que é necessário e de quais informações o componente requisita. Em conjunto foi utilizada a biblioteca Redux [13] para o gerenciamento do estado da aplicação. A escolha da ferramenta Plasmic se deu pela praticidade e ganho de tempo durante o período de desenvolvimento, enquanto as bibliotecas React e Redux foram escolhidas devido a experiências prévias.

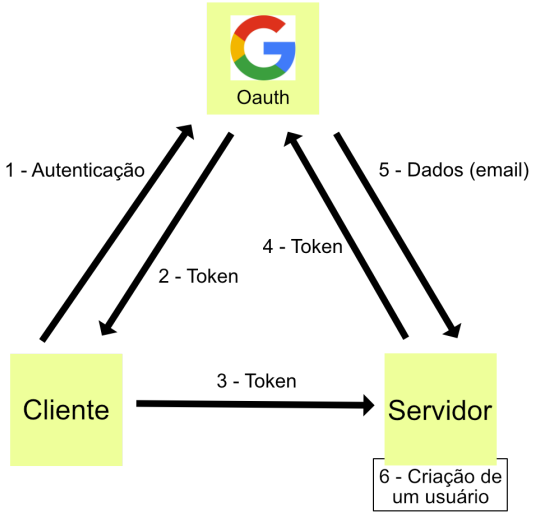

Figura 4 - Fluxo de autenticação

2.3.6 *Plasmic*

Toda a parte visual da aplicação foi feita por meio da ferramenta Plasmic, com exceção do gráfico de seleção de estratégias que não pode ser feito dentro da ferramenta devido a limitações da mesma na criação de componentes que tivessem como base elemento *html svg* ao invés do elemento *div*. Dentro da ferramenta foi definido as 5 páginas e diversos componentes que compõem a aplicação, as páginas sendo construídas utilizando os componentes definidos. Tendo um componente sido definido visualmente, era requisitado a geração do código fonte para o componente, esse era então adicionado e integrado ao código fonte da aplicação. Na Figura 5 é possível ver um exemplo de componente criado utilizando a ferramenta.

Figura 5 - Exemplo de componente criado utilizando a ferramenta Plasmic, componente de exibição de pesquisa cadastrada e possibilidade de edição.

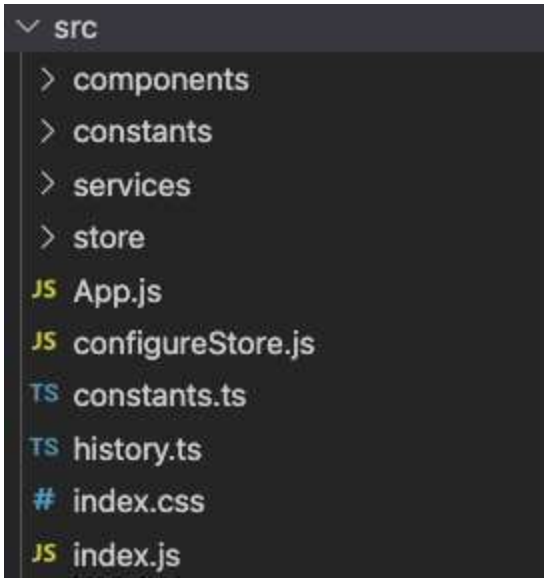

Figura 6 - Organização dos arquivos do front-end

2.3.7 Estrutura do front-end

A organização da estrutura do back-end foi feita tendo como base projetos passados, boas práticas consideradas pela comunidade de desenvolvedores [14] e a estruturação padrão que foi gerada pela ferramenta Plasmic, sendo então dividido em 3 grandes categorias *components*, *services* e *store* e tendo como uma divisão menor *constants*, como é demonstrado pela Figura 6.

Components contém todos os componentes da aplicação, este diretório foi automaticamente criado pela ferramenta Plasmic e contém como subdiretório *plasmic,* também foi criado automaticamente, neste está contido todo o código de componentes gerados automaticamente e seus respectivos estilos. Além disso, foi necessário desenvolver manualmente todo o componente referente a seleção de estratégias, buscando um resultado similar à da Figura 1, adicionando interatividade e cores com melhor contraste, toda essa lógica foi separada no subdiretório *StrategySelector* de *components*.

Services contém os serviços externos que a aplicação tem algum tipo de interação, nesse caso, apenas um módulo para realização de requisições para a API. O serviço de autenticação da Google não está presente como um serviço pois sua lógica ficou incorporada no pacote importado google-oauth.

Store contém toda lógica para gerenciamento do estado da aplicação sendo utilizado os serviços definidos em *services* para conexão com a API e seguindo o padrão sugerido pelo Redux dividindo o estado em várias partes cada uma contendo *state*, *actions, reducers* e *selectors*.

2.3.8 Deploy

Para o deploy da aplicação foi utilizado Github Actions [15] para automatizar o processo e reduzir erros que poderiam ser provenientes de realizar o processo manualmente toda vez. Para a o deploy do back-end foi utilizado o serviço Heroku [16] devido a simples integração com aplicações desenvolvidas em Node.js e o uso do certificado HTTPS e para o deploy do front-end foi utilizado o serviço Surge.sh [17] devido ao certificado HTTPS e experiências anteriores.

3. SISTEMA EM USO

A Figura 7 demonstra uma parte visual da aplicação, nesta página está presente a visualização dos estudos cadastrados de forma gráfica, sendo o gráfico bem similar ao da Figura 1, além disso é possível realizar o cadastro de novos estudos, como também alterar e deletar o que já foi cadastrado.

3.1 Avaliação

Disponibilizamos o acesso a aplicação para alunos de doutorado de Ciência da Computação da UFCG, que fazem pesquisas na área de engenharia de software. Foi preferível recrutar alunos de doutorado por estes terem maior contato com a realização de pesquisas, sendo assim possíveis usuários reais da aplicação.

Para coletar dados relacionados a satisfação dos usuários foi utilizado um formulário do Google, contendo uma matriz de concordância e de satisfação, para criar o formulário foi listado principais pontos que se deseja analisar, onde cada item poderia ser avaliado numa escala de 5 pontos (onde 1 = Concordo totalmente/Muito satisfeito e 5 Discordo totalmente/Muito insatisfeito). Os itens avaliados foram:

- Satisfação: indicar quão satisfeito com determinado aspecto da ferramenta.
    - Usabilidade da ferramenta.
    - Tempo para realização de uma atividade.

    - Funcionalidade da ferramenta.
    - Informações presentes na aplicação.
    - Interface da ferramenta.
    - Ferramenta no geral.
- Concordância: indicar quão de acordo se está com a afirmação.
    - Foi fácil aprender a usar a ferramenta.
    - Com a ferramenta é possível enriquecer o resultado de programas de pesquisa.
    - Gostei de utilizar a ferramenta.

Foram recrutados dois alunos, sendo obtido duas respostas. Tendo como exceção a avaliação de informações presentes na aplicação, não houve resultados negativos nas respostas, obtendo um índice de satisfação/concordância entre 1 e 3. Em todas as respostas foi destacado a dificuldade em obter informação sobre a aplicação diretamente pela interface, além disso algumas pequenas adições foram sugeridas:

- Melhorar o tutorial, adicionando um passo a passo da aplicação.
- Aumentar limite do tamanho da descrição.
- Aumentar o tamanho da fonte de texto das palavras no gráfico.

Apesar do feedback negativo em relação ao nível de informação sobre a aplicação, se obteve unanimidade de resposta positivas em relação a como a ferramenta é capaz de enriquecer o resultado de programas de pesquisa, mesmo sendo apenas uma prova de conceito com poucos usuários.

4. EXPERIÊNCIA

4.1 Processo de desenvolvimento

Inicialmente foi feito um levantamento das funcionalidades necessárias para o sistema e então assinalada uma prioridade (Baixa, Média, Alta) para cada funcionalidade. Sendo as funcionalidades de alta prioridade: cadastrar programa de pesquisa, visualizar programas de pesquisa, cadastrar estudo em um programa de pesquisa, visualizar estudos de um programa de pesquisa de forma gráfica similar a Figura 1; todas sendo funcionalidades básicas da aplicação, mas que em conjunto permitiria a mínima usabilidade da ferramenta, por isso sendo consideradas de alta prioridade.

Com isso, criou-se um *backlog* e planejamento para realizar o processo de desenvolvimento em *sprints* bi-semanais. As funcionalidades foram distribuídas para as sprints levando em consideração a sua prioridade, funcionalidades com prioridade mais alta sendo realizadas nas sprints iniciais.

A partir disso, foram decididas as tecnologias a serem usadas e inicializado o repositório de desenvolvimento no Github, tendo como código base inicial *starter kits* das tecnologias selecionadas [18]. Mesmo sem o desenvolvimento de nenhuma funcionalidade, foi adicionado os *workflows* de Github Actions para automatizar o deploy, isso se provou uma boa decisão, pois permitiu utilizar a aplicação em produção após cada funcionalidade desenvolvida, sem ser necessário gastar tempo para realizar o deploy manualmente.

Com o uso da ferramenta Plasmic foi feita a prototipagem das principais telas da aplicação e integrado ao código fonte do *front-end*. As funcionalidades eram então desenvolvidas por meio de aprimoramento dos protótipos, integração do *front-end* com o

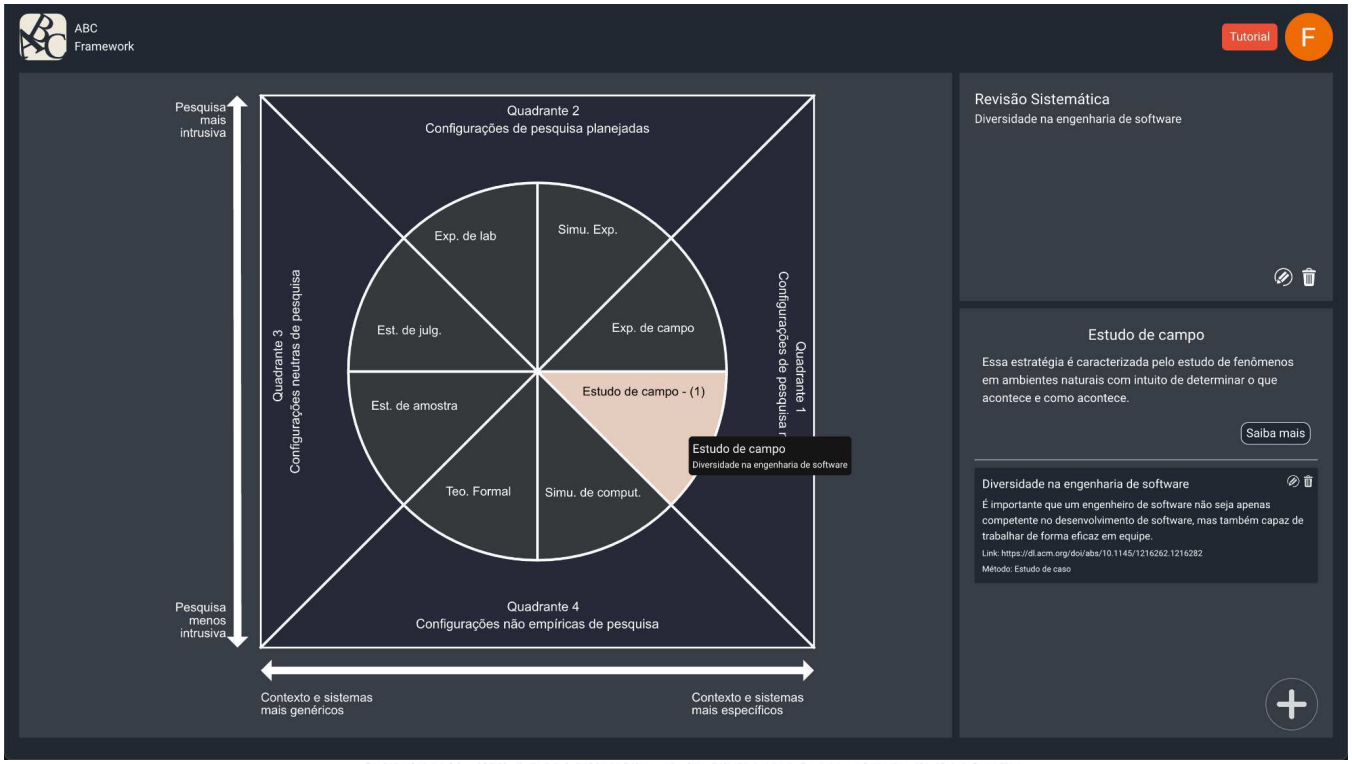

Figura 7 - Página de visualização de um programa de pesquisa

back-end e implementação dos requisitos no *back-end*, isto sendo feito em um formato de revezamento.

4.2 Desafios

O maior desafio encontrado foi a utilização da ferramenta Plasmic para gerar uma interface de usuário amigável e simples. Por ser a primeira experiência usando a ferramenta como método primário para desenvolver a interface visual, houve uma dificuldade inicial para se adaptar ao modelo de criação de páginas e utilização do código gerado mantendo sincronia entre *front-end* e *back-end*. No entanto, após a adaptação, o uso da ferramenta se tornou vantajoso por representar uma redução no tempo de desenvolvimento de novas funcionalidades.

5. CONCLUSÃO

Após o desenvolvimento e liberação da aplicação para alguns usuários, pôde-se concluir o potencial da ferramenta para ajudar pesquisadores no seu processo de trabalho, foi possível validar a possibilidade do uso focado no gerenciamento dos estudos. No entanto, os usuários demonstraram certa dificuldade no uso da aplicação, devido a falta de informações melhores detalhadas na própria interface.

Para trabalhos futuros, pretende-se adicionar novas funcionalidades à aplicação, como exemplo permitir que o usuário indique que um programa de pesquisa cadastrado fique público, possibilitando que outros usuários tenham acesso e possam usar como referência de uso da aplicação. Além disso, detalhar melhor as informações de uso da aplicação, como também realizar ajustes visuais para melhorar a experiência do usuário, por exemplo, aprimorar a visualização da aplicação em dispositivos mobile.

6. REFERÊNCIAS

[1] Guidelines for Conducting Software Engineering Research. https://link.springer.com/chapter/10.1007/978-3-030-32489-6_2

[2] Usando OAuth 2.0 para acessar APIs do Google. https://developers.google.com/identity/protocols/oauth2

[3] Client Server Architecture and HTTP Protocol. http://toolsqa.com/client-server/client-server-architecture-and-http-protocol/

[4] What is in-memory database. https://aws.amazon.com/pt/nosql/in-memory/#:~:text=An%20in%2Dmemory%20database%20is,the%20need%20to%20access%20disks

[5] MongoDB. https://www.mongodb.com/

[6] MongoDB Atlas. https://www.mongodb.com/cloud/atlas

[7] Node.js. https://nodejs.org

[8] Express - Node.js web application framework. https://expressjs.com

[9] Mongoose ODM - Object modeling for Node.js. https://mongoosejs.com/

[10] Express validator - Middlewares that wraps validator and sanitizer functions. https://express-validator.github.io/docs/

[11] Plasmic - visual builder and web design tool. https://www.plasmic.app/

[12] React - Uma biblioteca para criar interfaces de usuário. https://pt-br.reactjs.org/

[13] Redux - A Predictable State Container for JS Apps. https://redux.js.org/

[14] Redux Style Guide. https://redux.js.org/style-guide/style-guide

[15] Github Actions. https://docs.github.com/pt/actions

[16] Heroku - Cloud Application Platform. https://www.heroku.com/

[17] Surge.sh - Static web publishing for Front-End Developers. https://surge.sh/

[18] React Starter Kits. https://pt-br.reactjs.org/community/starter-kits.html

[19] Authorship trends in software engineering. https://www.researchgate.net/publication/271630732_Authorship_trends_in_software_engineering