Reestruturação do frontend do SIEC-UPE para Angular 9

Trabalho de Conclusão de Curso

Engenharia de Computação

Fernando Ferreira Ribeiro

Orientador: Profº. Joabe Bezerra de Jesus Júnior

Fernando Ferreira Ribeiro

Reestruturação do frontend do SIEC-UPE para Angular 9

Artigo apresentado como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Engenharia de Computação

Escola Politécnica de Pernambuco

Universidade de Pernambuco

Orientador: Profº. Joabe Bezerra de Jesus Júnior

Recife - PE, Brasil

Abril de 2021

Fernando Ferreira Ribeiro
Reestruturação do frontend do SIEC-UPE para Angular 9/ Fernando Ferreira Ribeiro. – Recife - PE, Brasil, Abril de 2021-
34 p.
Orientador: Profº. Joabe Bezerra de Jesus Júnior

Trabalho de Conclusão de Curso – Engenharia de Computação
Escola Politécnica de Pernambuco
Universidade de Pernambuco, Abril de 2021.

MONOGRAFIA DE FINAL DE CURSO

Avaliação Final (para o presidente da banca)*

No dia 11/5/2021, às 16h00min, reuniu-se para deliberar sobre a defesa da monografia de conclusão de curso do(a) discente **FERNANDO FERREIRA RIBEIRO**, orientado(a) pelo(a) professor(a) **JOABE BEZERRA DE JESUS JÚNIOR**, sob título Evolução do FrontEnd do SIEC-UPE para Angular 9, a banca composta pelos professores:

LARISSA TENÓRIO FALCÃO ARRUDA (PRESIDENTE)
JOABE BEZERRA DE JESUS JÚNIOR (ORIENTADOR)

Após a apresentação da monografia e discussão entre os membros da Banca, a mesma foi considerada:

[X] Aprovada [ ] Aprovada com Restrições* [ ] Reprovada

e foi-lhe atribuída nota: 8,5 (oito e meio)

*(Obrigatório o preenchimento do campo abaixo com comentários para o autor)

O(A) discente terá 7 dias para entrega da versão final da monografia a contar da data deste documento.

Larissa Falcão

AVALIADOR 1: Prof (a) **LARISSA TENÓRIO FALCÃO ARRUDA**

AVALIADOR 2: Prof (a) **JOABE BEZERRA DE JESUS JÚNIOR**

AVALIADOR 3: Prof (a)

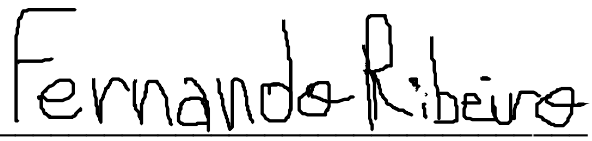
* Este documento deverá ser encadernado juntamente com a monografia em versão final.

ESCOLA POLITÉCNICA
DE PERNAMBUCO

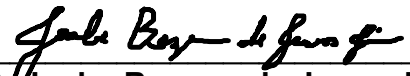
Autorização de publicação de PFC

Eu, **Fernando Ferreira Ribeiro** autor(a) do projeto de final de curso intitulado: **Evolução do FrontEnd do SIEC-UPE para Angular 9;** autorizo a publicação de seu conteúdo na internet nos portais da Escola Politécnica de Pernambuco e Universidade de Pernambuco.

O conteúdo do projeto de final de curso é de responsabilidade do autor.



Fernando Ferreira Ribeiro



Orientador(a): **Joabe Bezerra de Jesus Júnior**

Coorientador(a):

Prof, de TCC: **Daniel Augusto Ribeiro Chaves**

Data: 11/5/2021

Resumo

O setor de Extensão e Cultura é muito importante no âmbito universitário. Junto com ensino e pesquisa, constituem os 3 pilares básicos de uma universidade e devem receber igualdade de tratamento por parte das instituições de ensino. As atividades de ensino e pesquisa são as mais conhecidas, já que ocorrem dentro do ambiente acadêmico, com todos os alunos tendo acesso a elas. Isso significa que existe um desafio para perpetuar atividades de extensão dentre os discentes. A Universidade de Pernambuco possui uma ferramenta para contribuir com a resolução de tal desafio. O SIEC-UPE (Sistema de Extensão e Cultura da UPE) é uma aplicação *web* que tem o objetivo de ajudar gestores e professores da universidade a gerenciar as atividades desse ramo. O sistema, que ainda encontra-se em desenvolvimento, utiliza tecnologias modernas e de rápida evolução. O presente trabalho propõe a reestruturação do *software*, trazendo a nova versão do *framework* utilizado e conceitos de programação mais modernos, com intuito do projeto não precisar ser desenvolvido em código legado. A reestruturação de *software*, de fato, busca realizar atualizações e melhorias que vão aumentar a facilidade de manutenção do *software*, reduzindo custos e incrementando a vida útil do mesmo.

Palavras-chave: Desenvolvimento de Software, Evolução de Software, Reestruturação de Software, Cultura e Extensão.

Abstract

The Extension and Culture sector is very important at the university level. Along with teaching and research, they constitute the 3 basic pillars of a university and should receive equal treatment from educational institutions. Teaching and research activities are the best known, as they take place within the academic environment, with all students having access to them. This means that there is a challenge to perpetuate extension activities among students. The University of Pernambuco has a tool to contribute to solving this challenge. SIEC-UPE (Extension and Culture System of UPE) is a web application that aims to help university managers and professors to manage the activities of this branch. The system, which is still under development, uses modern and rapidly evolving technologies. The present work proposes the restructuring of the software, bringing the new version of the framework used and more modern programming concepts, in order that the project does not need to be developed in legacy code. Software restructuring, in fact, seeks to carry out updates and improvements that will increase the ease of maintenance of the software, reducing costs and increasing its useful life.

Keywords: Software Development. Software Evolution. Software Restructuring, Culture and Extension.

Lista de ilustrações

Lista de abreviaturas e siglas

HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
API	*Application Programming Interface*
JSON	*Javascript Object Notation*
HTML	*HyperText Markup Language*
CSS	*Cascade Style Sheets*
REST	*Representational State Transfer*

Sumário

1 Introdução

Atividades de extensão universitária estão presentes em todos os núcleos de ensino superior [1], não só do Brasil mas também do mundo. Além de serem requisitos obrigatórios para o cumprimento da carga horária do curso, são importantes para que os estudantes vejam na prática o ofício que estão aprendendo e o impacto que tal ofício pode causar na sociedade.

Em particular, a Universidade de Pernambuco (UPE) não possui um sistema apropriado, em operação, para registro, consulta e divulgação de informações sobre estas atividades. O projeto de extensão POLI-móvel, aprovado no edital de extensão 2016 da UPE (PIAEXT 2016), iniciou o desenvolvimento de um sistema de informação de extensão e cultura (SIEC) para as atividades da Escola Politécnica de Pernambuco (POLI), unidade mais antiga da UPE. Este projeto teve seu escopo alterado no final de 2016 para abranger todas as atividades de extensão e cultura da UPE, mas foi construído com recursos recebidos apenas em 2018. Assim, o sistema proposto, chamado SIEC-UPE (Sistema de Informação de Extensão e Cultura da UPE) foi redefinido como uma aplicação *web* e construído com linguagens modernas e de presença em mercado, sendo o seu *backend* (camada de negócio e acesso a dados) em .Net Core 2.1 e o seu *frontend* (camada de apresentação) no framework *Angular* 6, tecnologias e lançadas em novembro de 2017 e maio de 2018, respectivamente.

Contudo, com um mundo cada vez mais demandando por sistemas mais complexos e em maiores escalas [2], e com o desenvolvimento de software tendo um grande impacto na vida das pessoas, os *frameworks* utilizados precisam se adaptar rapidamente, e isso não foi diferente para as tecnologias supracitadas. O *framework* Angular, criado na linguagem JavaScript e depois TypeScript, evolui com velocidade, seguindo as tendências do mercado, sendo estas quaisquer mudanças que ocorram no ecossistema, como atualizações em protocolos de segurança, boas práticas de programação e implementações mais performáticas. Sua versão utilizada no SIEC-UPE possui uma atualização significativa. Mantido pela Google, cada versão majoritária possui 6 meses de suporte ativo, nos quais atualizações e correções são regularmente liberadas, e 12 meses de suporte de longo prazo (*LTS*, *Long Term Support*), nos quais apenas correções críticas são liberadas [3].

Como dito anteriormente, o SIEC-UPE utiliza o Angular 6, o qual já não está em *LTS* e não é mais desenvolvido e evoluído pela mantenedora e nem pela comunidade. É notável que evolução de software, embora seja uma prática custosa [4], se faz necessária. O objetivo deste trabalho é a reestruturação do *frontend* da aplicação do SIEC, com o propósito de atualizar a versão do Angular utilizada para a versão 9, que ainda está em

LTS até 2022. Essa atualização busca mitigar riscos, trazer robustez e confiabilidade, e evitar que o sistema entre em operação com uma tecnologia legada. Além disso, o trabalho também visa definir uma arquitetura para este *frontend*, de modo que os módulos Angular possam ter um *layout* base, ser mais facilmente integrados e novos módulos tenham uma estrutura baseada em padrões de projeto e serviços.

O restante deste documento segue a seguinte estrutura: a seção 2 apresenta a metodologia usada; a seção 3 traz o referencial teórico; a seção 4 detalha o sistema SIEC; por fim, a seção 5 apresenta as conclusões e trabalhos futuros.

2 Referencial Teórico

O referencial teórico deste trabalho foi particionado em cinco tópicos, são eles: a importância da reestruturação de código, aplicações web, utilização de *Javascript* no *frontend*; definição de *Typescript*; conceitos de *Node.js*; e conceitos de *Angular*.

2.1 Reestruturação de Software

Sistemas legados são de complexo entendimento, o que restringe possíveis mudanças. A reestruturação de *software* é responsável por reimplementar sistemas legados, com o objetivo de facilitar sua manutenção, reorganizando a estrutura do sistema. Usualmente não se alteram as funcionalidades do sistema nem a arquitetura, mas isso não é regra, principalmente para a arquitetura [5].

Um *software* existente pode variar, mudanças durante sua vida útil são prováveis de acontecer, e essas mudanças contínuas geralmente fazem com que ele tenda a ficar menos estruturado, devido à fatores como tempo, recursos, escopo. Isso manifesta-se em documentação incorreta, código sem padrão e tempo maior para que os programadores possam entendê-lo, elevando custos com a manutenção de *software*. Reestruturação é o processo de transformação de um software de uma forma estruturada preservando sua funcionalidade. Possui a finalidade de fazer o software mais fácil de se entender e mudar, tornando-o menos suscetível a erros quando mudanças futuras precisarem ser realizadas. O conceito de reestruturação, na teoria, pode ser usado como prova de que qualquer programa pode ser escrito em uma forma estruturada, usando apenas 3 construções: sequência, condição e repetição. Em 1971, Ashcroff e Manna [6] demonstraram que programas que utilizam estruturas primitivas de código, como o comando GOTO, poderiam ser traduzidos facilmente em uma forma estruturada equivalente [7].

Para *frameworks* como o *Angular*, as necessidades de mudança acontecem rápido. Sua versão *LTS* possui um ano, o que significa que uma nova versão é liberada, e esta pode sinalizar que recursos já utilizados tornaram-se *deprecated*, seja por razões de performance ou correção de vulnerabilidades.

2.2 Aplicações Web

Uma aplicação *web*, em termos gerais, pode ser definida como um sistema computacional para utilização através de um *browser*, construído com as tecnologias que este tem a capacidade de interpretar, que são HTML, *Javascript* e CSS. Pode ser executado a partir

de um servidor HTTP ou localmente, no dispositivo do usuário. Quanto à estrutura para esse tipo de aplicação, está dividida em duas: cliente e servidor. O cliente é responsável pela apresentação, interação com o usuário e comunicação com o servidor, por meio de requisições HTTP. O servidor é uma REST API que dispõe os *endpoints* para o cliente acessar. Ele é responsável pelo processamento dos dados, devolvendo ao cliente uma resposta de acordo com o que foi pedido pela requisição.

2.3 Javascript

Javascript é uma ferramenta poderosa que pode ser de grande utilidade em um site. De maneira geral, ele é responsável pelo comportamento e pelas interações. *Javascript* torna possível a construção de componentes e comportamentos visuais mais complexos, como um carrossel de imagens, menu de navegação interativo, validação de formulários, dentre outros.

Outra importante funcionalidade nativa do *Javascript* é o AJAX (*Asynchronous Javascript and XML*). Consiste em uma ferramenta que possibilita interação com *web servers* através de requisições HTTP. Indo mais além, tais requisições HTTP não exigem que a página seja totalmente recarregada, o que traz melhor experiência para o usuário e consequentemente faz do AJAX um importante aliado dos *frameworks* modernos de *frontend*, como *Angular*, *React* e *Vue*.

2.4 Typescript

Embora *Javascript* esteja consolidado e possua funcionalidades importantes para o desenvolvimento, sua natureza interpretada ainda serve como barreira para profissionais que estão habituados com linguagens de programação orientadas à objetos e fortemente tipadas. É claro que ele conta com recursos de inferência de tipos, mas nem sempre tal recurso consegue livrar o desenvolvedor de problemas, sobretudo aqueles em tempo de execução. Foi dessa necessidade que nasceu o *Typescript*.

Typescript surge como um superset do *Javascript*, adicionando a este funcionalidades que nativamente não estão disponíveis ou requerem grande esforço para utilização [8]. Baseado nas especificações do ES 6 (EcmaScript 2015, especificação de linguagens de script) [9], o *Typescript* tem como principal recurso a tipagem estática é capaz de oferecer outros como interfaces e *generics* [8], características já presentes em outras linguagens de programação orientadas a objetos, como C♯ e Java.

Contudo, é importante dizer que *Typescript* não é uma linguagem que os *browsers* entendem, esta ainda se limita a *Javascript* puro. O que acontece para os *browsers* conseguirem entender código *Typescript* é uma transpilação para *Javascript*. O processo de

transpilação é uma especialização da compilação, a diferença é apenas que no compilador tradicional o alvo é um código de mais baixo nível, provavelmente alguma forma de *Assembly* ou código de máquina, enquanto que o transpilador tem como alvo um código fonte de uma linguagem de alto nível diferente ou a mesma escrita de outra forma.

2.5 Node.js

O *Node.js* é uma plataforma de execução de *Javascript* assíncrono e orientado a eventos. Comumente conhecida como uma plataforma *server-side* por executar no lado do servidor de aplicação web e por ser independente de navegadores (*web browsers*). Essa abordagem livre de *threads* faz com que aplicações que utilizam *Node* sejam mais eficientes e fáceis de usar. Além disso, quase nenhuma função no *Node* realiza diretamente operações de E/S, fazendo com que *deadlocks* não sejam um problema e facilitando o desenvolvimento de sistemas escaláveis.

No *Node*, o *HTTP* é tratado como prioridade, projetado para ter baixa latência e alta taxa de fluxo [10]. Isso torna essa plataforma uma boa opção para servir como base de uma biblioteca *web* ou até mesmo um *framework* [10].

O *Node.js* a partir da sua versão 0.6.3 passou a incluir de forma nativa o seu instalador de dependências, o *NPM* (*Node Package Manager*), um sistema auxiliar que ao ler o arquivo *package.json*, que contém as referências do projeto devidamente listadas, realiza o *download* de cada uma das dependências [11].

2.6 Framework Angular

Angular, em sua versão atual, é um *framework* de código aberto que utiliza a linguagem *Typescript*. É baseado em componentes, o que o faz ideal para aplicações escaláveis. Também traz consigo uma coleção de bibliotecas que cobrem uma grande variedade de funcionalidades, incluindo roteamento, gerenciamento de formulários e comunicação com servidor.

De acordo com dados de uma pesquisa de 2019 realizada pela plataforma *Stack Overflow*, 30.7% dos engenheiros de software estão utilizando *Angular* para construir interfaces gráficas interativas [12]. Esse alto percentual deve-se as funcionalidades mais importantes do *Angular*. Além das supracitadas, existem também dois tópicos relacionados ao *framework* que contribuem pro seu sucesso. Primeiramente, ele possui uma *CLI* (*Command Line Interface*) de uso intuitivo e fácil de configurar, tornando o cotidiano dos usuários mais produtivo [12]. A *CLI* tem comandos configuráveis para criação de componentes, módulos, serviços dentre outras elementos de uma aplicação de larga escala.

Por último, temos o gerenciamento de pacotes e dependências do *NPM*, trazendo total controle ao desenvolvedor sobre quais pacotes e quais versões destes devem ser utilizadas.

3 Metodologia

Este trabalho propõe a reestruturação da aplicação *Web* SIEC-UPE para cadastro, visualização e gerenciamento de projetos de extensão e cultura da Universidade de Pernambuco (UPE). A aplicação visa suprir uma demanda do corpo de gestores, professores e funcionários técnico-administrativos da UPE. O trabalho tomou como base uma versão existente do código do *frontend* do SIEC hospedado na ferramenta de controle de versão *GitHub*[1].

A realização das atividades foi dividida em três fases:

1 Planejamento: referente ao estudo e leitura do código existente, assim como a investigação das telas a serem reimplementadas;

2 Desenvolvimento e execução: com a definição do contexto, a definição do modelo de requisições, arquitetura, implementação e testagem da ferramenta;

3 Redacional: referente à formulação do texto final do estudo.

Para a realização deste trabalho, foi escolhida a ferramenta *Visual Studio Code*[2], pelo fato desta ser otimizada para desenvolvimento com tecnologias *web*. Ela conta com terminal embutido, depuração, *auto-complete*, integração com ferramentas de versionamento, dentre diversos *plugins* que podem ser instalados a escolha do desenvolvedor [13].

O planejamento teve início no momento em que houve acesso ao código. Nesta fase foram decididos tópicos como a estratégia de atualização do *framework*, modularização de áreas, componentização do *core* e arquitetura de serviços e rotas.

Na fase de desenvolvimento, as atividades em foco foram a implementação das decisões do planejamento e das outras funcionalidades e restrições a serem atendidas. Para isso, foram utilizados protótipos de tela e *wireframes* que representavam a visualização do sistema em ambos tipos de dispositivos, *desktop* e *mobile*. Para a reestruturação do sistema atualizou-se o *framework Angular* para sua versão 9, *framework* de código aberto que utiliza a linguagem *Typescript*. A comunicação com o servidor seguiu padrão de requisições *JSON-API* [14]. Como uma aplicação cliente, e sem *backend* disponível, foi utilizada a biblioteca *Mirage.Js* [15] para "mockar" os dados e tornar possível o desenvolvimento e a demonstração do uso da aplicação. Em particular, a ausência do *backend* reflete a independência na evolução do *frontend* e relação ao *backend*, além do fato da existir uma reestruturação do *backend* iniciada paralelamente a este trabalho.

[1] https://www.github.com
[2] https://www.code.visualstudio.com

O desenvolvimento seguiu uma abordagem de construção iterativa, garantindo a continuidade de pequenas entregas ao longo do processo de desenvolvimento e incorporação de novas melhorias. Por fim, ocorreu a construção deste texto final do estudo.

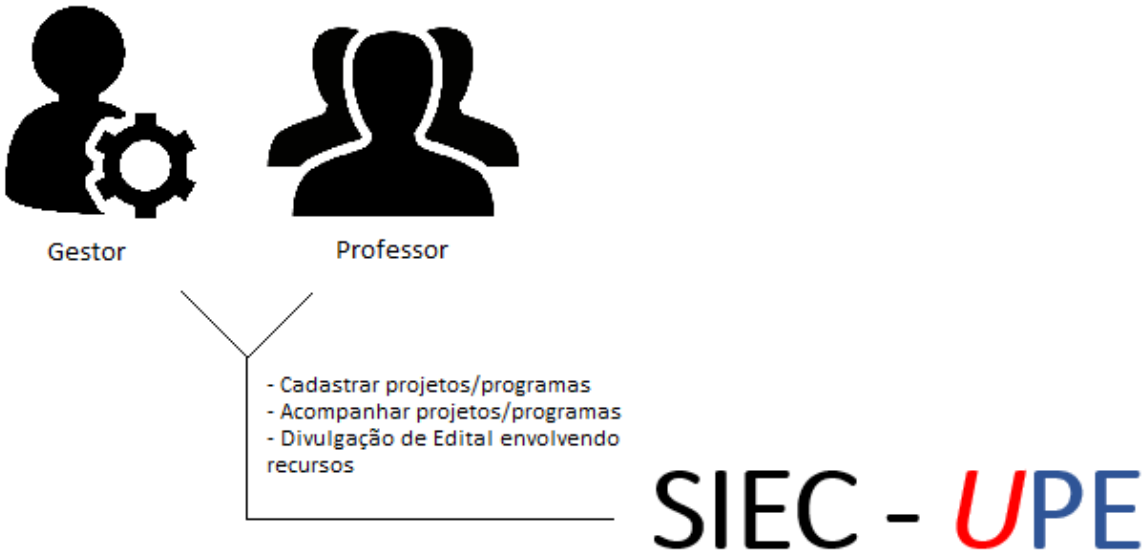
4 SIEC-UPE - Sistema de Informação de Extensão e Cultura da UPE

Nesta seção são apresentados a visão e contexto do SIEC, suas principais funcionalidades propostas e a reestruturação do *frontend* do sistema para o*framework* Angular versão 9.

4.1 Visão e Contexto do Sistema

O SIEC tem como usuários não apenas professores e gestores da UPE que possuem necessidade em registrar e consultar os projetos de extensão e cultura, mas também alunos e funcionários técnico-administrativos da universidade, uma vez que estes últimos também são habilitados a criar e coordenar projetos de extensão e cultura. A figura 1 exibe os usuários e principais funções que podem ser realizadas pelo sistema.

Figura 1 – Visão geral do SIEC e funcionalidades dos usuários



Fonte: Autor.

O SIEC foi proposto para ter funcionalidades como: Cadastro e visualização de programas e projetos, gerenciamento e visualização de galerias de fotos dos projetos, visualização de mensagens, acompanhamento completo dos projetos que conta com: aprovação, fomento financeiro, relatórios acadêmicos, prestação de contas, divulgação e publicação e ainda certificação de equipe. Por último, o destaque vai para editais de fomento da

pró-reitoria de extensão e cultura (PROEC). Dessa maneira, podemos descrever o SIEC como uma ferramenta de suporte às atribuições dos professores, alunos, funcionários técnico-administrativos e gestores, que visa permitir explorar com facilidade as atividades referentes aos projetos desse pilar da universidade que é a extensão, engajando todos os envolvidos, inclusive a sociedade.

4.1.1 Requisitos do SIEC

Nesse contexto, diferentemente das formas de registros e acompanhamento de projetos e programas existentes, como o uso de formulários do Google[1] ou através de correio eletrônico, o SIEC automatiza o processo, gerando e disponibilizando uma base de consulta desses projetos. Assim, o sistema visa colaborar para atender à política nacional de extensão universitária, definida na Resolução Nº 7, de 18 de dezembro de 2018[2].

Os requisitos funcionais do SIEC estão listados abaixo:

- Gestão das linhas de extensão;
- Gestão dos editais de registro ou fomento de atividades de extensão;
- Contestação de editais de extensão;
- Gestão dos programas e projetos de extensão, incluindo a gestão das áreas temáticas;
- Gestão dos atividades extensão: cursos de extensão, eventos de extensão, prestação de serviços e produtos resultantes de projetos/programas de extensão;
- Controle do processo de avaliação/aprovação de projetos e programas de extensão registrados: registro de avaliadores, distribuição de solicitações de avaliação, avaliação, compilação de resultados avaliativos, divulgação de resultados parciais e finais e tramitação de recursos;
- Gestão dos relatórios projetos e programas de extensão;
- Avaliação/aprovação de relatórios projetos e programas de extensão;
- Gestão da equipe (discentes e membros externos) e participantes dos programas/-projetos de extensão;
- Envio dos certificados das equipes dos projetos e programas de extensão;
- Divulgações de projetos e programas de extensão.

[1] https://www.google.com/intl/pt-BR/forms/about/

[2] Resolução Nº 7, de 18 de dezembro de 2018 - https://www.in.gov.br/materia/-/asset_publisher/Kujrw0TZC2Mb/content/id/55877808

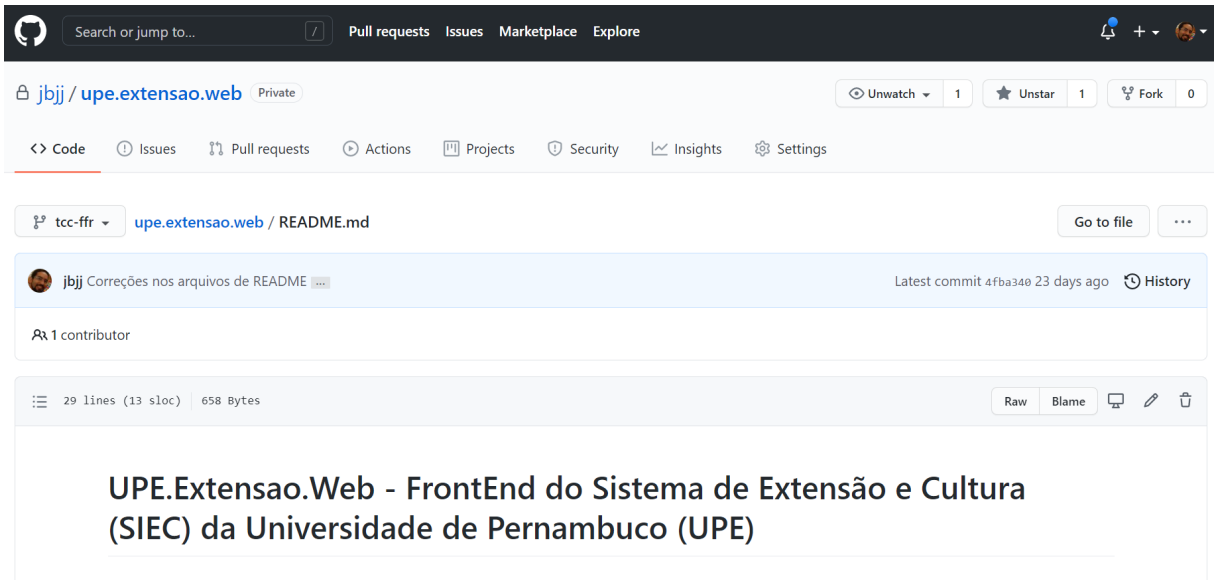
Em particular, usamos o termo gestão como o agrupamento das funções de cadastrar, buscar, atualizar e remover a entidade em questão. Além disso, as atividades de extensão são: cursos, eventos, produtos e serviços que podem ser agrupados em projetos de extensão, os quais podem por sua vez serem agrupados em programas de extensão. Assim, é importante destacar que a gestão de projetos/programas é o ponto chave para as demais funções do sistema.

Além disso, os coordenadores de projetos/programas, sejam eles docentes ou técnicos administrativos da UPE, podem ter vários programas/projetos geridos num painel de bordo (*dashboard*) disponibilizado pelo SIEC. Eles coordenam equipes de discentes que realizam um papel ativo nas ações de extensão. Finalmente, também destacamos a importância da divulgação dos programas/projetos para um maior envolvimento da comunidade acadêmica e da sociedade nessas atividades. O SIEC auxilia essa divulgação com uma galeria de imagens dos programas/projetos, como apresentaremos na seção 4.1.3.

4.1.2 Repositório do SIEC

O código e a documentação do projeto SIEC estão armazenados num repositório *Git*[3] na plataforma *GitHub*[4]. Esse repositório é controlado pelo criador/coordenador do projeto SIEC, Prof. Joabe Jesus, sendo um repositório privado, ver figura 2.

Figura 2 – Repositório GitHub do Frontend do SIEC



Fonte: Autor.

Para o desenvolvimento deste trabalho, foi criado um *branch* (um braço de desenvolvimento) derivado do código Angular 6 do *frontend*, mas considerando sua versão inicial

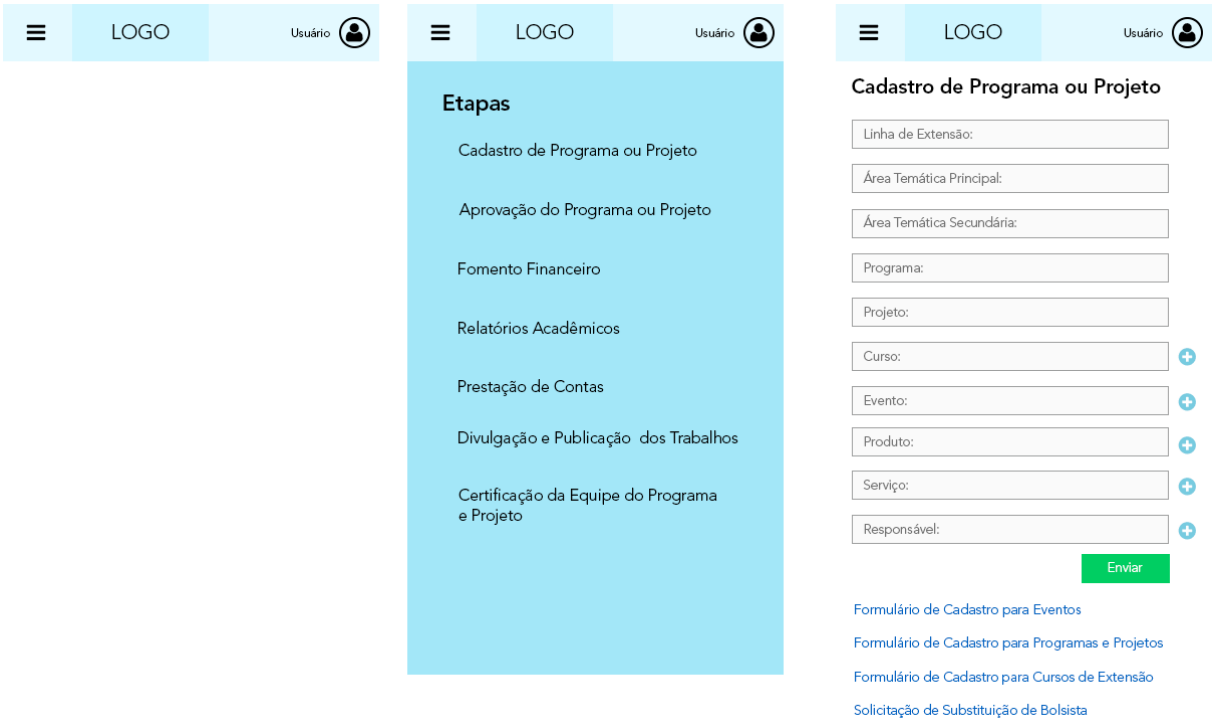
[3] https://git-scm.com
[4] https://github.com

em relação à proposta do *design* gráfico proposto para o projeto pela equipe de design da empresa Mídias Educativas. Este *branch* foi nomeado `tcc-ffr` e a reestruturação foi feita a partir do código inicial para que o trabalho não sofresse com problemas da arquitetura utilizada anteriormente no *frontend*.

4.1.3 Design Gráfico do frontend do SIEC

O *design* gráfico do *frontend* do SIEC seguiu uma abordagem *mobile first*, isto é, foi criada de forma a permitir a responsividade, uma característica fundamental nos sistemas atuais e que permite que um mesmo código possa ser utilizado para gerar as telas do sistema seja para um *smartphone* seja para um *tablet* ou para um navegador *web*. A figura 3 e 4 apresentam o esboço (*wireframe*) criado inicialmente para as telas do *frontend* pela empresa Mídias Educativas.

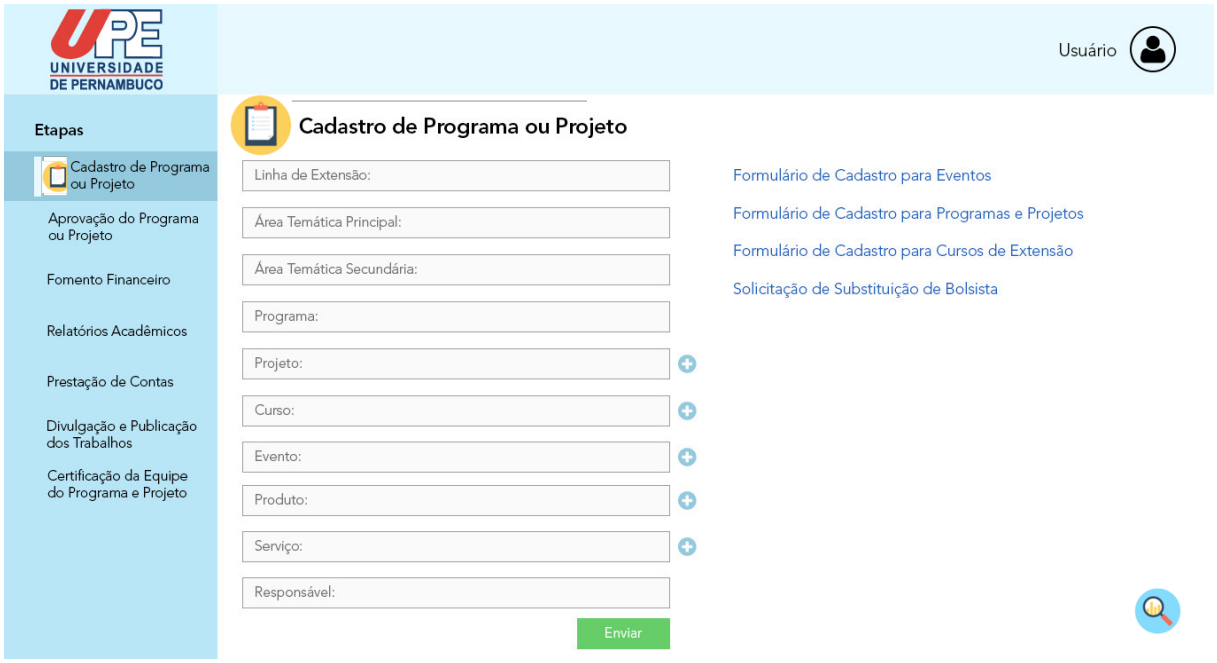
Figura 3 – Esboço (wireframe) inicial para smartphone



Fonte: Autor.

Como pode ser observado na figura 3, as várias etapas de cadastro à certificação da equipe do projeto foram objetivos do desenvolvimento do sistema SIEC-UPE. Além disso, a figura 4 destaca a necessidade do sistema possuir registros das linhas de extensão, áreas temáticas e até bolsistas dos projetos, pois estas informações são fundamentais para a gestão da extensão universitária.

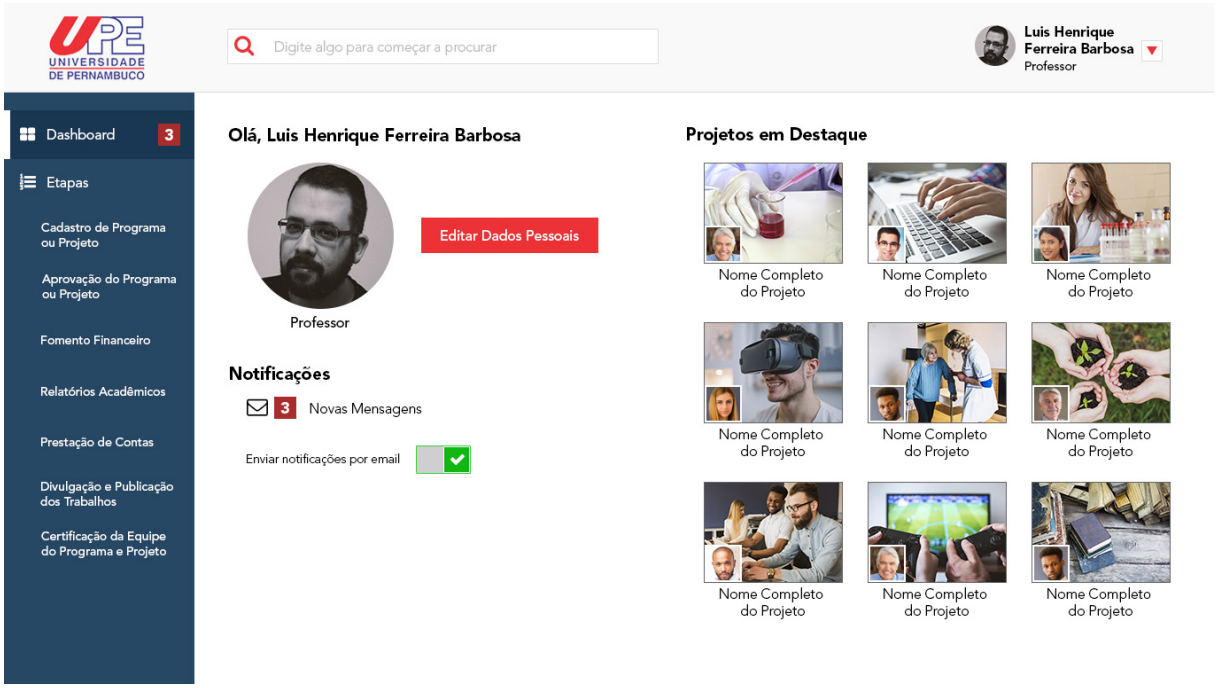
Figura 4 – Esboço (wireframe) inicial para navegador web



Fonte: Autor.

A partir desses esboços das telas do *frontend* foi desenvolvido pela empresa Mídias Educativas, em 2018, o design gráfico final (ver figura 5) e o código inicial em Angular 6 foi fornecido para este design gráfico.

Figura 5 – Design gráfico inicial para navegador web



Fonte: Autor.

5 Reestruturação do frontend do SIEC-UPE

A partir do entendimento de todas as informações apresentadas na seção anterior, o trabalho foi realizado com a definição da arquitetura e a migração do código em Angular versão 6 para Angular versão 9. Adicionalmente, foi realizada de maneira incremental a implementação de melhorias visuais e estruturais. O exemplo visual é percebido na componentização de estruturas estáticas como o cabeçalho e o menu interativo lateral. A adoção de padrões de projetos e práticas que modernizam o código, como o *Facade* [16] e os *Dumb Components* exemplifica a melhoria estrutural. Tais melhorias possibilitam que o desenvolvimento torne-se fluído e menos suscetível a problemas de acoplamento, facilitando a manutenção do código e atingindo o objetivo da reestruturação de *software*.

5.1 Arquitetura

A arquitetura escolhida para este trabalho segue uma abordagem facilitada do padrão de projeto *Facade* [16]. Em resumo, esse padrão abstrai a complexidade de um módulo para demais que necessitem de tal módulo. No SIEC, a funcionalidade de acesso à camada de serviço e aos dados no estado do sistema estão abstraídos por uma fachada, fazendo com que os componentes *Angular* precisem conhecer apenas essa fachada e não conexões HTTP. Isso faz com que os componentes tenham código mais limpo e fiquem extremamente mais fáceis de manter. A figura 6 detalha a arquitetura. O *frontend* é o foco deste trabalho e está mais detalhado para poder ser explicado em partes. A explicação das camadas será por tópicos.

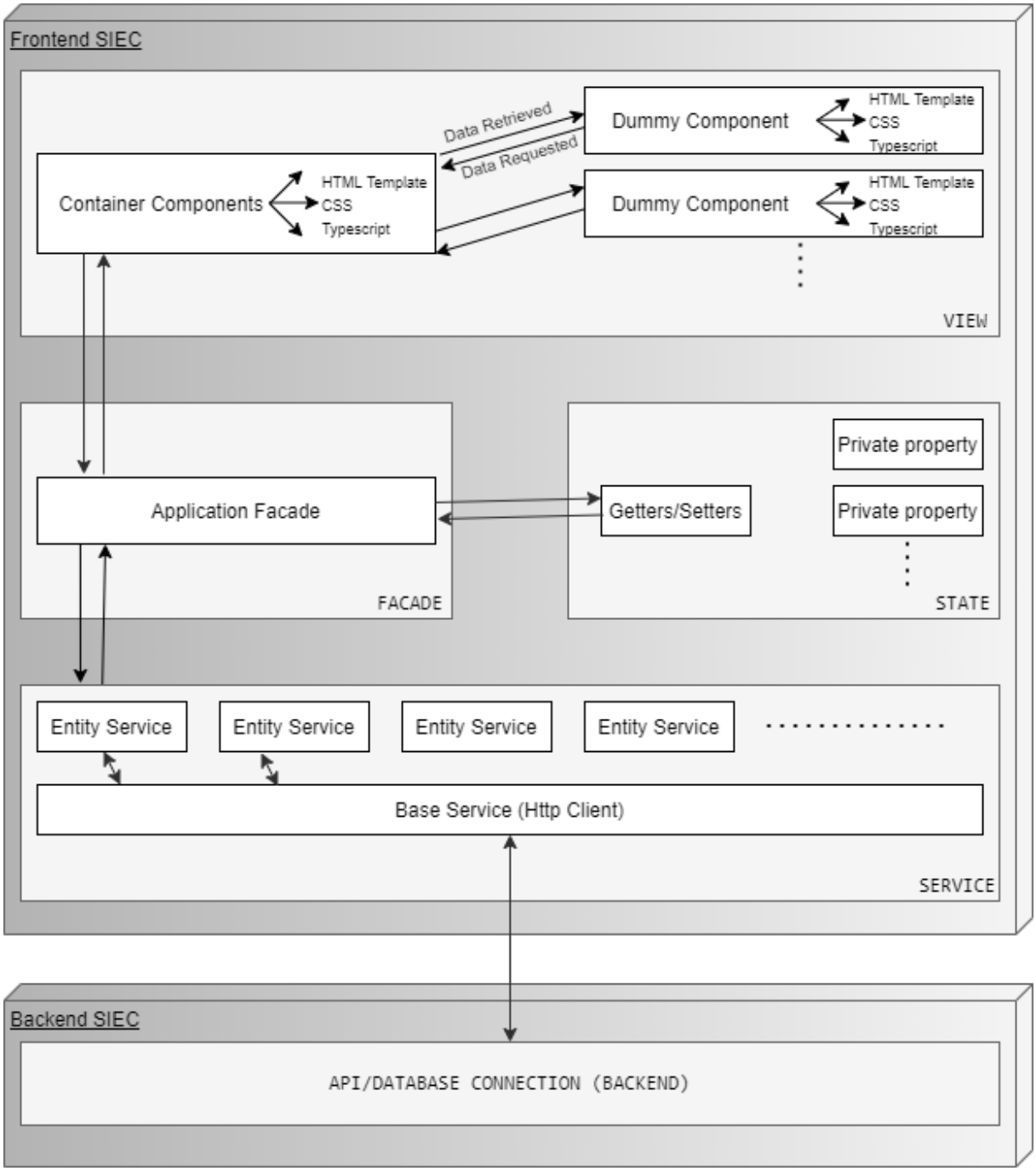
1 *VIEW*: Camada de visualização, exibição de dados, interação com o usuário. Nessa camada se encontram os componentes, estruturas do *Angular* que são exibidas por demanda. Tais componentes foram separados em dois tipos, *Containers* e *Dumbs*, também chamados *Smart* e *Presentational*. Essa abordagem independe do *Angular*, podendo ser utilizada com qualquer outor *framework* moderno, e permite o reuso facilitado de componentes *Dumbs*, visto que eles não possuem lógica dependente de outras partes do sistema. É definido por ter acesso à fachada apenas em componentes *Containers*, e estes por sua vez passam as informações para os *Dumbs* que resumem-se em exibir essas informações. No caso do *Angular*, esse fluxo de dados se dá por *@Input* e *@Output*, estruturas para comunicação inter-componentes [17];

2 *FACADE/STATE*: Camada onde ficam as chamadas ao *backend* e a quaisquer dados que estejam no estado da aplicação. Ela tem a função de fazer a comunicação entre os componentes *Containers* e a camada de serviço. É no *Facade* que estão definidas

as funções que externalizam os dados para o resto da aplicação. O *State*, por sua vez, encapsula os dados em variáveis privadas e os expõe para o *Facade* via *Getters* e *Setters*;

3 *SERVICE*: Camada de comunicação com o servidor. Cada *service* vai conter métodos com as APIs existentes. Para reutilizar código e reduzir injeção de dependências desnecessárias, essa camada também conta com um serviço base, abstrato e que é herdado por todos os outros serviços. É este serviço base que dispõe da biblioteca *Angular* de comunicação com servidor, o *HttpClient*. Este trabalho objetivou a reestruturação do *frontend* da aplicação, o qual é responsável pelo envio das requisições ao servidor, no entanto, como dito anteriormente, devido à ausência de *backend* disponível, a camada de serviço trabalha com um banco de dados em memória provido pela biblioteca *Mirage.Js* [15].

Figura 6 – Arquitetura do Frontend do SIEC



Fonte: Autor.

5.2 Migração do código Angular 6 para Angular 9

Para facilitar o processo de migração, a mesma foi realizada em duas etapas: primeiro com migração da versão 6 para a versão 8; e em seguida da versão 8 para a versão 9. Após a migração, alguns pontos do código precisaram ser alterados para ficarem de acordo com as regras atualizadas da versão 9. A figura 7 mostra um exemplo de alteração,

onde foi necessário atualizar a sintaxe de carregamento de módulos, que foi atualizada e está menos suscetível a erros a partir de agora.

Figura 7 – Sintaxe para Lazy Load de módulos no Angular 9

```
{
    path: 'dashboard',
    loadChildren: () => import('./modules/dashboard/dashboard.module').then(m => m.DashboardModule)
},
{ path: 'aprovacao', component: AprovacaoProjetoComponent },
{
    path: 'cadastro',
    loadChildren: () => import('./modules/cadastro-projeto/cadastro-projeto.module').then(m => m.CadastroProjetoModule)
},
{
    path: 'mensagens',
    loadChildren: () => import('./modules/mensagens/mensagens.module').then(m => m.MensagensModule)
},
```

Fonte: Autor.

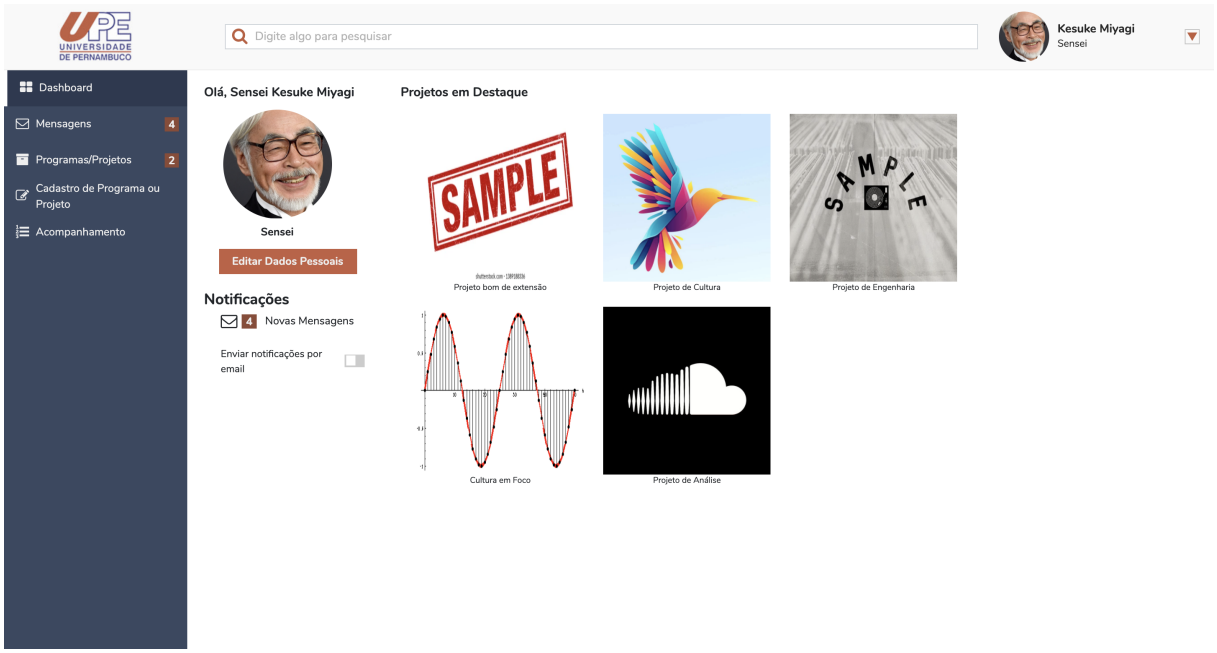
5.2.1 Telas do SIEC

O *layout* da aplicação reestruturada é simples. Consiste de um cabeçalho com menu de usuário e campo de pesquisa e uma barra lateral esquerda com um menu interativo que permite a navegação pelo sistema. Todas as telas que um usuário autenticado terá acesso vão possuir este *layout* aplicado, seguindo a facilidade do *Angular* de componentização, um importante benefício do *framework* que implementa conceitos de SPAs (*Single Page Applications*).

O fluxo principal do sistema será apresentado utilizando as telas reestruturadas neste trabalho, com visualização em resoluções *desktop* e *mobile*.

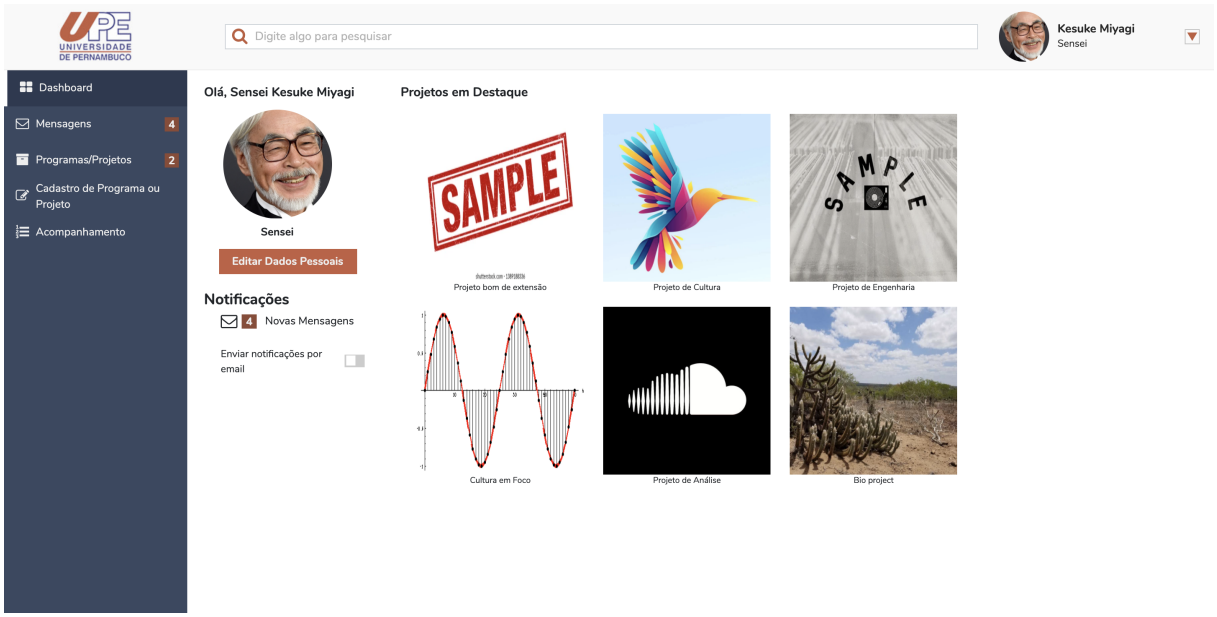
As figuras 8 e 9 mostram a tela de *dashboard*. Visualização *mobile* é visto nas figuras 10 e 11. É a tela inicial do sistema, para qual o usuário será redirecionado logo após se autenticar. Esse componente inteligente teve em sua codificação o acesso via fachada dos projetos e das mensagens, assim como dados do usuário.

Figura 8 – Dashboard Inicial



Fonte: Autor.

Figura 9 – Dashboard após cadastro de projeto



Fonte: Autor.

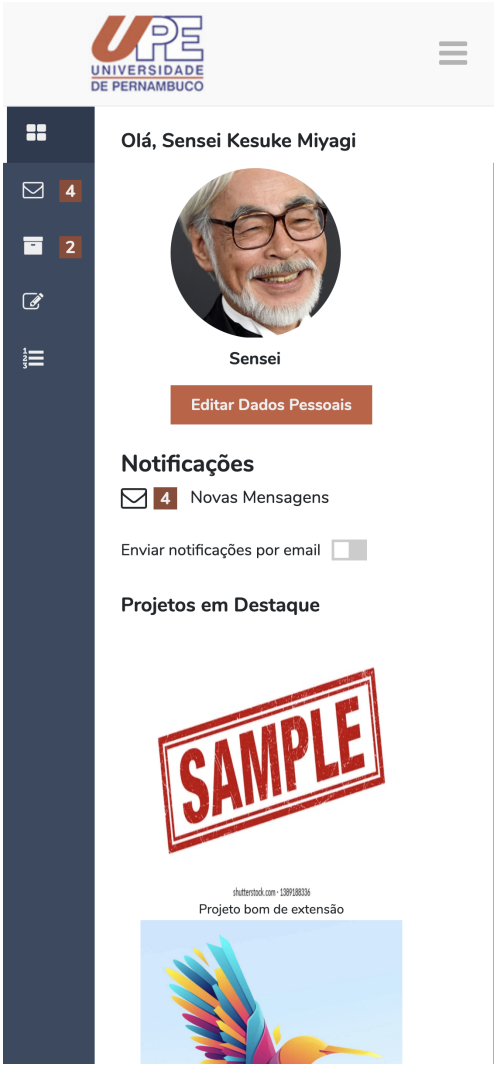


Figura 10 – Dashboard em resolução Mobile

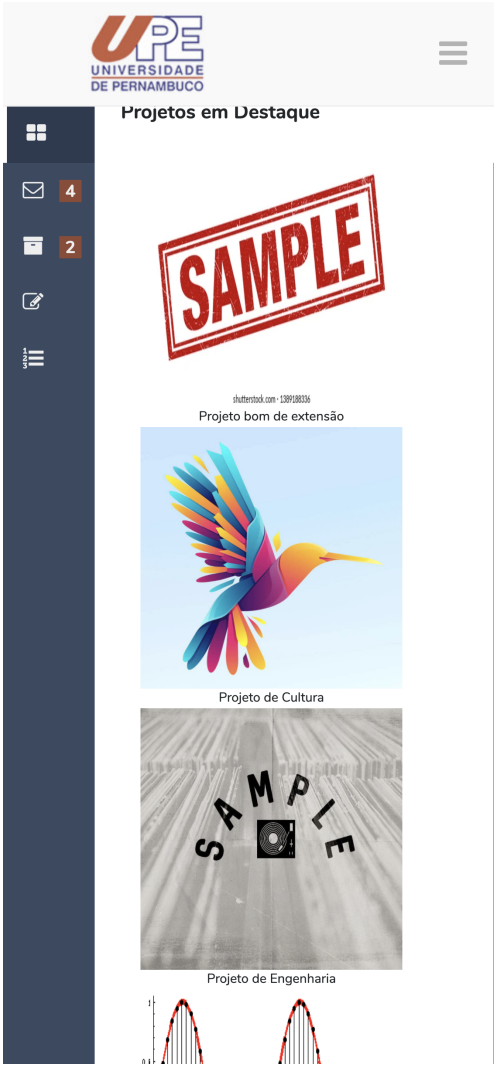
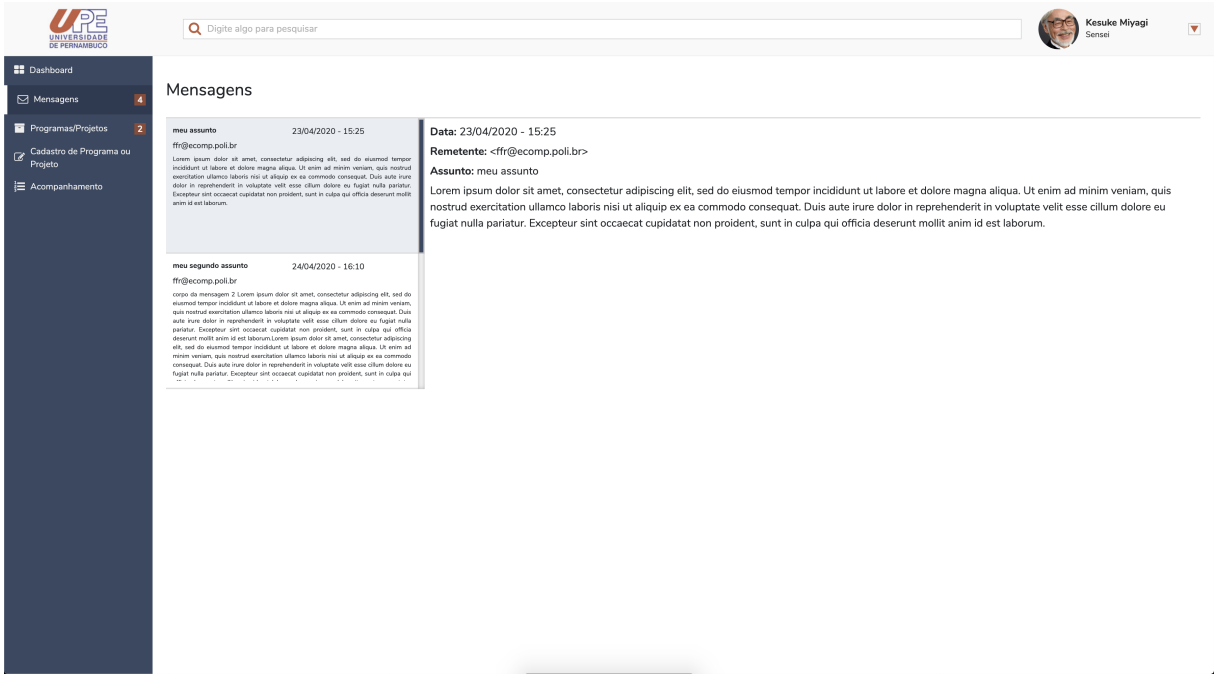


Figura 11 – Dashboard Mobile - Projetos Reorganizados

O *dashboard* conta com um sumário do usuário, contendo seu nome, foto e cargo. Ainda no sumário, conta também com atalho para desabilitar recebimento de mensagens e editar informações pessoais. Além disso, o *dashboard* possui a listagem de projetos daquele usuário, contendo foto e título de cada projeto. É uma visão geral dos artefatos que o usuário possui.

As figuras 12, 13 e 14 mostram as tela de mensagens. Nela, o usuário tem uma interface semelhante a qualquer caixa de correio eletrônico, com listagem lateral e área de visualização trocando o conteúdo dinamicamente. Na resolução *mobile*, pela falta de espaço horizontal, foi implementada uma mecânica para esconder a listagem de mensagens durante a visualização. O *container* busca as mensagens via fachada e as disponibiliza para os dois componentes de apresentação, a lista de mensagens e o visualizador de mensagens.

Figura 12 – Tela de Mensagens



Fonte: Autor.

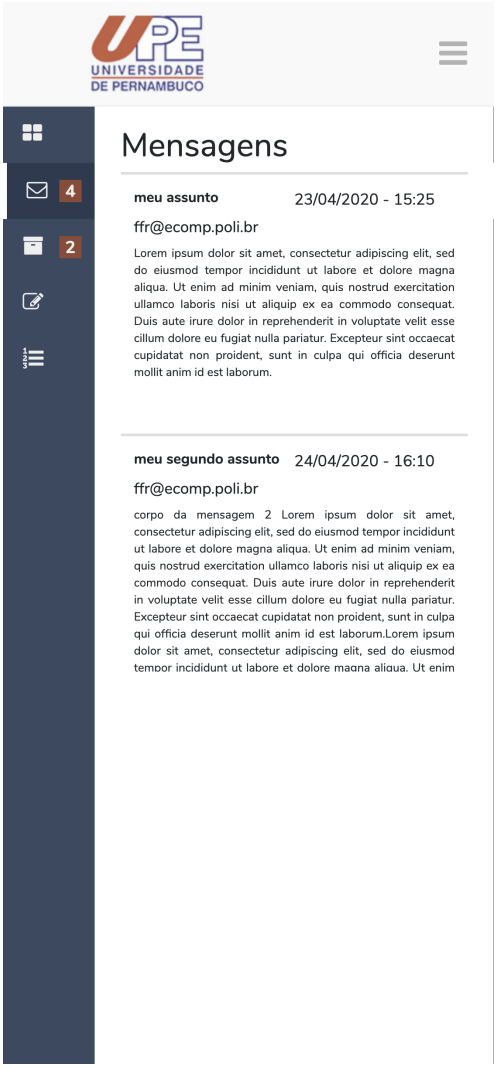


Figura 13 – Lista de Mensagens

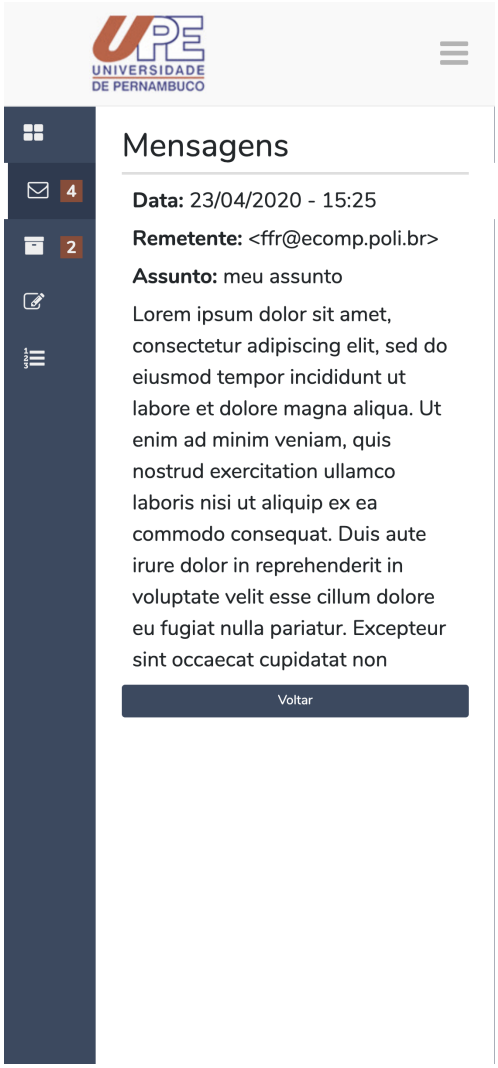
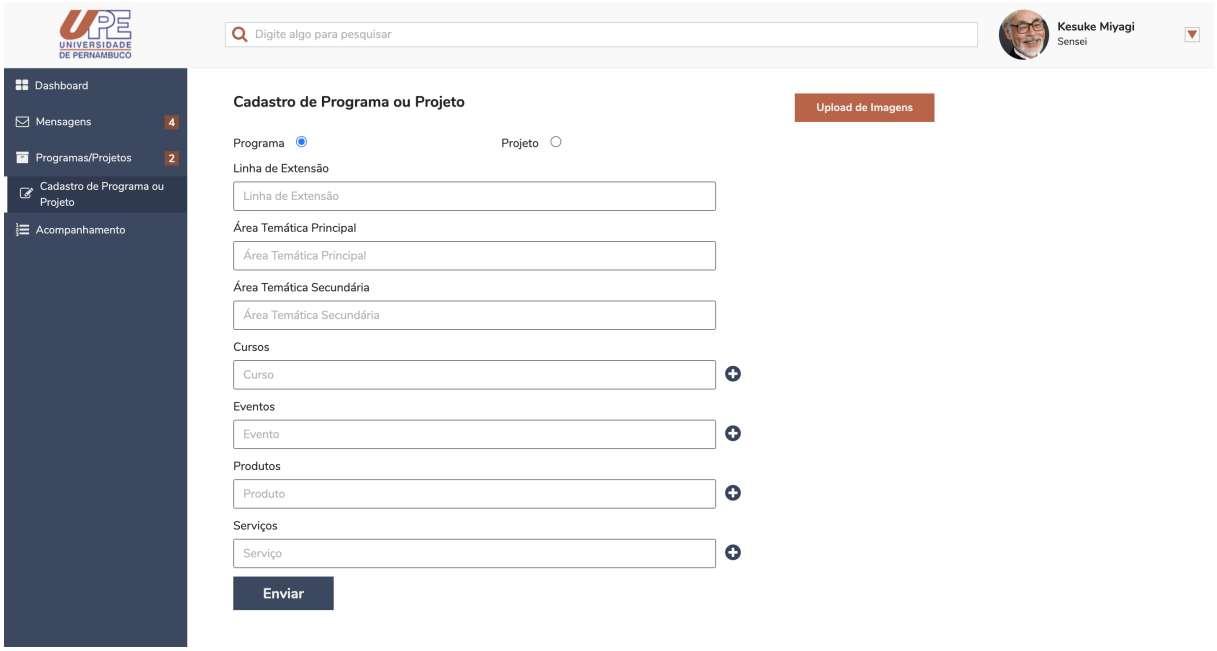


Figura 14 – Visualização de Mensagem

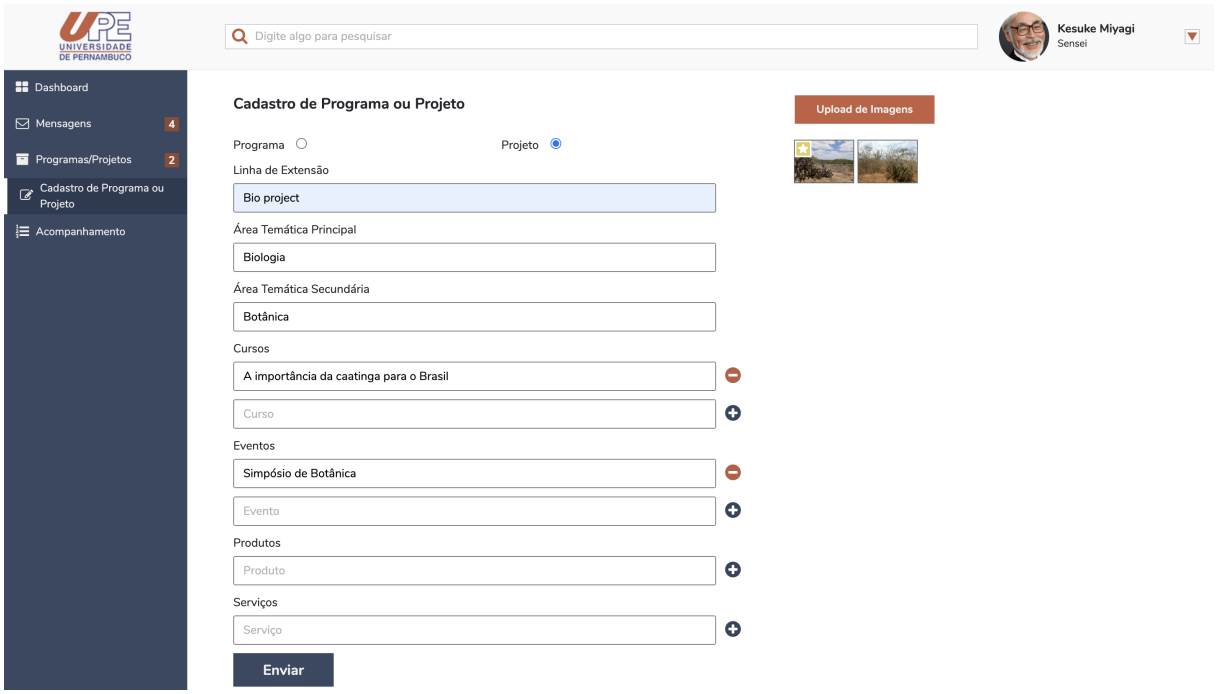
A funcionalidade de cadastro de projetos e programas pode ser vista nas figuras 15, 16, 17 e 18. O usuário dispõe de um formulário com alguns campos com dinâmica adicionável e removível, alguns campos obrigatórios e *upload* de imagens, seguindo o comportamento esperado. Todos os campos do formulário estão inseridos em um *form-group*, estrutura do Angular para trabalhar com formulários que facilita a implementação de validações e regras mais complexas.

Figura 15 – Tela de Cadastro



Fonte: Autor.

Figura 16 – Cadastro Preenchido



Fonte: Autor.

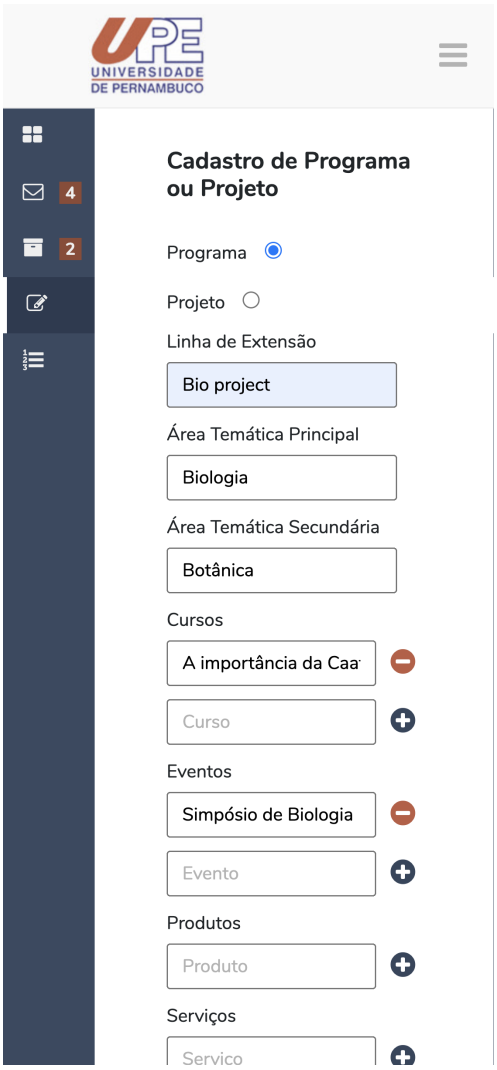


Figura 17 – Cadastro Mobile Preenchido

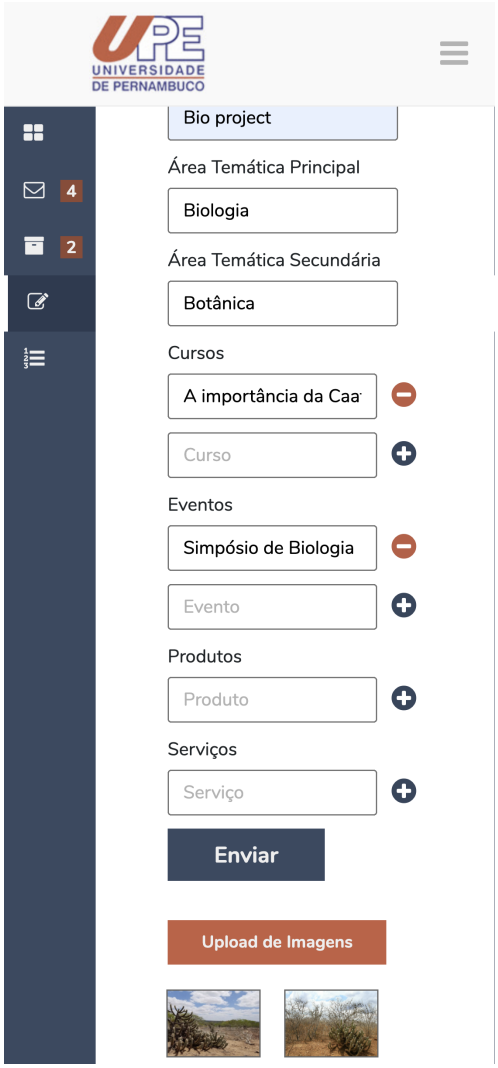


Figura 18 – Cadastro Mobile Preenchido e Reorganizado

6 Conclusões e Trabalhos Futuros

As universidades cumprem um grande papel no âmbito social. Isso, inclusive, está previsto na Constituição brasileira: o artigo 207 diz que "As universidades gozam, na forma da lei, de autonomia didático-científica, administrativa e de gestão financeira e patrimonial e obedecerão ao princípio da indissociabilidade entre ensino, pesquisa e extensão"[18]. O setor de Extensão é um pilar das universidades, contribui com a sociedade e valoriza o ensino superior. A Universidade de Pernambuco conta com uma aplicação ainda em desenvolvimento para atender as demandas gestoriais desse setor universitário. O SIEC, atualmente, tem seu *backend* com *.Net Core 2.1* e *frontend* com *Angular 6*. Contudo, o *framework Angular* evoluiu rapidamente, e sua versão atual já é bem superior à 6 originalmente usada. Essas atualizações trouxeram novas funcionalidades, corrigiram problemas e vulnerabilidades.

Este trabalho apresenta uma reestruturação do *frontend* do SIEC. A execução se deu a partir da atualização da versão do *framework* para a 9, e seguiu com implementações que utilizam conceitos modernos no que diz respeito a *frameworks* para aplicações *web*, tais como o padrão de projeto *Facade* [16] e a especificação de APIs JSON-API [14].

O trabalho restringiu-se a implementar as telas mais importantes, que são o *dashboard*, a visualização de mensagens e o cadastro de projetos, não contemplando demais funcionalidades, mas definindo parte da API do *backend* no padrão JSON-API.

Como trabalhos futuros podem ser realizadas melhorias no SIEC, no que diz a respeito às folhas de estilo (CSS) e a finalização da evolução (reestruturação) das demais funcionalidades, como:

1 Edição de projeto;

2 Acompanhamento de projeto;

3 Prestação de contas;

4 Geração de relatórios.

Assim, acreditamos que este trabalho foi capaz de contribuir com a evolução do SIEC-UPE ao realizar uma parte da reestruturação do *frontend* previamente construído.

Referências

[1] ALMEIDA, L. Pinho de. A extensão universitária no brasil: processos de aprendizagem a partir da experiência e do sentido. *DIversité REcherches et terrains*, 2015. Disponível em: <http://dx.doi.org/10.25965/dire.692>. Citado na página 9.

[2] DYBå T.; DINGSøYR, T. Empirical studies of agile software development: A systematic review. *Information and software Technology*, p. 833–859, 2008. Citado na página 9.

[3] ANGULAR Releases. 2021. Disponível em: <https://angular.io/guide/releases>. Citado na página 9.

[4] BENNETT K.; RAJLICH, V. Software maintainance and evolution: A roadmap. *Information and software Technology*, p. 73–87, 2000. Citado na página 9.

[5] FORNO, M. H. D. *Evolução de Software através de Reengenharia: um processo didático.* 118 p. Tese (Trabalho de Conclusão de Curso) — Universidade Federal do Pampa, 2014. Citado na página 11.

[6] ASHCROFT, E. A. Proving assertions about parallel program. *JOURNAL OF COMPUTER AND SYSTEM SCIENCE*, p. 110–135, 1975. Citado na página 11.

[7] REESTRUTURAçãO de Software. 2011. Disponível em: <http://tudoqueeugostoeoutrosassuntos.blogspot.com/2011/07/reestruturacao-de-software.html>. Citado na página 11.

[8] ARAúJO, D. *Do Javascript ao Typescript, Why?* 2021. Disponível em: <https://blog.vulpi.com.br/javascript-typescript/>. Citado na página 12.

[9] EDSON. *Introdução ao TypeScript.* 2021. Disponível em: <https://www.devmedia.com.br/introducao-ao-typescript/36729>. Citado na página 12.

[10] ABOUT Node.js. 2021. Disponível em: <https://nodejs.org/en/about/>. Citado na página 13.

[11] CRIANDO serviço de microblog com Node.js. 2021. Disponível em: <https://www.devmedia.com.br/criando-servico-de-microblog-com-node-js/31036>. Citado na página 13.

[12] THE Good and the Bad of Angular Development. 2021. Disponível em: <https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/>. Citado na página 13.

[13] VISUAL Studio Code. 2021. Disponível em: <https://code.visualstudio.com>. Citado na página 15.

[14] JSON-API. 2021. Disponível em: <https://jsonapi.org>. Citado 2 vezes nas páginas 15 e 32.

[15] MIRAGEJS. Build complete frontend features, even if your API doesn't exist. 2021. Disponível em: <https://miragejs.com>. Citado 2 vezes nas páginas 15 e 23.

[16] GAMMA, E. *Design patterns : elements of reusable object-oriented software*. 31. ed. Boston: Bookman, 1995. 395 p. Citado 2 vezes nas páginas 22 e 32.

[17] SHARING data between child and parent directives and components. 2021. Disponível em: <https://angular.io/guide/inputs-outputs>. Citado na página 22.

[18] BRASIL. *[Constituição (1988)]. Constituição da República Federativa do Brasil: promulgada em 5 de outubro de 1988*. 4. ed. [S.l.]: Saraiva, 1990. Citado na página 32.