UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

INSTITUTO DE COMPUTAÇÃO

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO PEREIRA MIRANDA SILVA

YAGO DE ARAUJO SERPA

GeProFi Web - GErenciador de PROjetos FInais

RIO DE JANEIRO

2022

GUSTAVO PEREIRA MIRANDA SILVA

YAGO DE ARAUJO SERPA

GEPROFI-WEB - GERENCIADOR DE PROJETOS FINAIS

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Valeria Menezes Bastos, D.Sc.

RIO DE JANEIRO

2022

Silva, Gustavo Pereira Miranda
S586g GeProFi Web: GErenciador de PROjetos Finais / Gustavo Pereira Miranda Silva, Yago de Araujo Serpa. – Rio de Janeiro, 2022.

59 f.

Orientador: Valeria Menezes Bastos.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2022.

1. Sistemas de informação. 2. Java. 3. JavaScript. 4. React. 5. SpringBoot. I. Serpa, Yago de Araujo. II. Bastos, Valeria Menezes (Orient.). III. Universidade Federal do Rio de Janeiro, Instituto de Computação. IV. Título.

GUSTAVO PEREIRA MIRANDA SILVA

YAGO DE ARAUJO SERPA

GEPROFI WEB - GERENCIADOR DE PROJETOS FINAIS

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 08 de Fevereiro de 2022.

BANCA EXAMINADORA:

gov.br Documento assinado digitalmente
VALERIA MENEZES BASTOS
Data: 09/02/2022 17:25:32-0300
Verifique em https://verificador.iti.br

Valeria Menezes Bastos, D.Sc (IC/UFRJ)

gov.br Documento assinado digitalmente
ADRIANA SANTAROSA VIVACQUA
Data: 09/02/2022 18:48:59-0300
Verifique em https://verificador.iti.br

Adriana Santarosa Vivacqua, D.Sc (IC/UFRJ)

gov.br Documento assinado digitalmente
GISELI RABELLO LOPES
Data: 23/03/2022 19:47:30-0300
Verifique em https://verificador.iti.br

Giseli Rabello Lopes, D.Sc (IC/UFRJ)

AGRADECIMENTOS

GUSTAVO PEREIRA MIRANDA SILVA

Agradeço, em primeiro lugar, à minha família, por todo o suporte dado nesses anos de vida, sempre me apoiando nas dificuldades e comemorando junto as alegrias.

Aos meus amigos, minha família escolhida, por toda a companhia, ajuda, conversas e suporte, desde os tempos de colégio, passando pelos anos de UFRJ, até o resto da vida.

À Profa. Dra. Valéria, por toda a ajuda, não só em relação a este projeto final, mas também pela orientação durante a vida acadêmica.

Aos amigos que fiz durante o curso, sem os quais teria sido muito mais difícil chegar no fim dessa jornada.

E em especial ao Yago, meu companheiro de projeto, por todo nosso empenho para concluir o mesmo.

AGRADECIMENTOS

YAGO DE ARAUJO SERPA

Quero agradecer, em primeiro lugar, aos meus avós Alfredo e Maria da Penha, pela criação e apoio durante toda minha vida.

Agradeço também à minha mãe, Jussara, por se esforçar para me dar o melhor que podia e ao meu pai Alexandre. Aos respectivos cônjuges, Antônio Romar, que infelizmente não verá isto, mas estará sempre conosco e Adriana, que em todos os momentos me trataram de forma similar aos seus filhos.

Aos meus irmãos Alana, Alessandra e Igor. As duas mais distantes, contudo, com a mesma importância. E meu irmão, que me faz trabalhar muito forte para que eu seja sempre um exemplo dentro de casa e a ser responsável para ajudar a guiá-lo à uma trilha para o sucesso.

Dedico esta, bem como todas as minhas demais conquistas, a eles e à minha namorada Isabella, que teve paciência, me deu forças, carinho e incentivo. É só mais uma das muitas vitórias que teremos pela frente!

Tive vizinhos que foram como pais para mim, e sou muito afortunado por isso, gostaria também de agradecer a Paulo Jorge, que me ajudou a iniciar no mundo da Computação, e Nadia pelo tratamento que me deram todos esses anos.

À orientadora, Profa. Dra. Valeria por todo o suporte para comigo, não só em relação ao projeto, como conversávamos sobre a vida acadêmica, pessoal e profissional em geral.

Não poderia esquecer das pessoas que começaram comigo no Bacharelado em Ciências Matemáticas e da Terra, e me ajudaram a seguir o meu caminho à conclusão em Ciência da Computação, bem como os amigos que fiz neste.

E claro, ao meu companheiro de trabalho, Gustavo, pelo companheirismo e todas as horas depositadas neste trabalho com o objetivo comum da conclusão do mesmo.

Às demais pessoas que passaram pela minha vida ou continuam, porém não estão enunciadas diretamente acima, de toda forma, o meu muito obrigado.

RESUMO

Dado o contexto da pandemia de coronavírus que vem afetando o mundo todo desde o começo de 2020, percebeu-se a falta de estrutura para muitas atividades presenciais no modelo remoto, e um deles é o trabalho de conclusão de curso. Já existia um trabalho realizado em 2015, pensando na automatização das tarefas que envolvem a construção e a finalização de um projeto final, desde sua concepção, passando pelo acompanhamento do(s) orientador(es) até a sua apresentação e entrega final, contudo, este se tornou obsoleto devido às ferramentas utilizadas durante o seu desenvolvimento em relação às ferramentas atuais e melhores práticas, adequadas a um ambiente disponível 24/7 e de alta escalabilidade.

O presente trabalho aprimora a ideia inicial e busca facilitar todos os passos de um projeto final e torná-lo informatizado, indo da criação de um novo projeto até a conclusão do mesmo, com ata assinada, convites por e-mail para a realização da defesa, entre outros recursos. Possui uma área pública com todos os projetos finalizados, os seus respectivos participantes e o *link* para o arquivo do texto do projeto, as áreas de interesse relacionadas aos projetos e aos docentes.

Palavras-chave: Sistemas de Informação; Java; JavaScript; React; SpringBoot; Gerenciamento de Trabalho de Conclusão de Curso; PostgreSQL; Heroku; Arquitetura Cloud.

ABSTRACT

Given the covid pandemic that is affecting the whole world since early 2020, it was notorious that there was lack of structure to many activities that formerly were on site, one of them being the final thesis to get the Bachelor's Degree. There was previous work done in 2015 automating the tasks from start to end of a thesis, however, it became obsolete when compared to the latest cutting edge technology regarding tools, best practices and a highly available, 24/7 online environment.

The following work improves that initial idea, with the goal to facilitate all steps of a final thesis, going from the concept and creation of the project to its conclusion, with signed documents, e-mail invites on the platform for the presentation, among other features. It also has a public area with all finalized projects, the link to the file, fields of interests and information of each teacher registered to the application.

Palavras-chave: Information System; Java; JavaScript; React; SpringBoot; Thesis Management; PostgreSQL; Heroku; Cloud Architecture.

SUMÁRIO

1. Introdução

1.1. Contexto e Motivação

O processo de conclusão de uma monografia envolve muitos passos durante sua construção e desenvolvimento. Porém, com o atual contexto de pandemia, isso se tornou ainda mais complicado e assim estagnou muitos discentes na iminência do fim da graduação.

O projeto desenvolvido por [SOUZA e ANDRADE, 2015] de informatização da gerência de projetos finais foi desenvolvido em 2015 e nunca foi colocado em prática, mesmo em um ambiente cliente-servidor, como foi sua concepção original. Algumas tecnologias usadas estão desatualizadas, porém a necessidade de se estabelecer uma comunicação à distância se tornou presente diante da pandemia estabelecida desde março de 2020.

A proposta deste trabalho de conclusão é utilizar metodologias atuais, bibliotecas e conceitos que não eram frequentes no ambiente de computação à época, como manter a aplicação funcional em um provedor de nuvem, além de realizar a solução para desenvolvimento web, bem como sua remodelagem visual.

1.2. Objetivos Gerais

Este trabalho tem como objetivo aprimorar (com nova arquitetura, identidade visual e desenvolvimento web) o sistema de Gerenciamento de Trabalhos de Conclusão de Curso do Bacharelado em Ciência de Computação da Universidade Federal do Rio de Janeiro, denominado Gerenciador de Projetos Finais – GeProFi [SOUZA e ANDRADE, 2015] e, com isso, acelerar o processo de comunicação docentes-discentes na construção de um projeto final, guardar todas as fases e alterações deste processo e ativar o processo da ata virtual.

1.3. Objetivos Específicos

Os objetivos específicos são:

- Implementar um novo sistema de aprovação de projetos finais do curso de Bacharelado em Ciência da Computação (BCC) para web, modernizando o sistema anterior, que era para ambiente cliente-servidor, em termos de conceitos internos de funcionamento;
- Incluir novas funcionalidades;
- Criar uma nova e moderna identidade visual para o ambiente;
- Implementar e realizar os testes da plataforma;
- Analisar os resultados.

1.4. Trabalhos Correlatos

O trabalho de [SOUZA e ANDRADE, 2015] é sumariamente a base deste trabalho de conclusão. O objetivo é automatizar e facilitar o acesso à situação em que se encontra um trabalho de conclusão momentaneamente, a comunicação entre orientador e orientados durante todo esse processo e também permitir aos alunos que haja uma área pública de interesses dos docentes, onde cada discente encontrará sua identificação temática, os trabalhos realizados na área e os docentes relacionados. Este trabalho foi feito em um ambiente local.

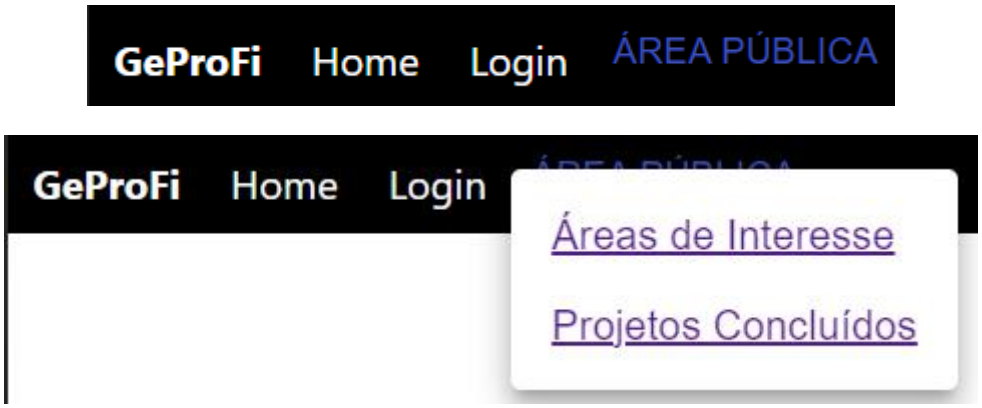
2. FUNCIONALIDADES

O capítulo de funcionalidades contém o manual de todos os passos para cada um dos processos que envolvem todo o ciclo de uso da plataforma. Existem três tipos de usuários: estudantes, professores e professores externos, além de uma área pública, com áreas de interesse e informações sobre os professores e projetos finalizados.

2.1. Área Pública

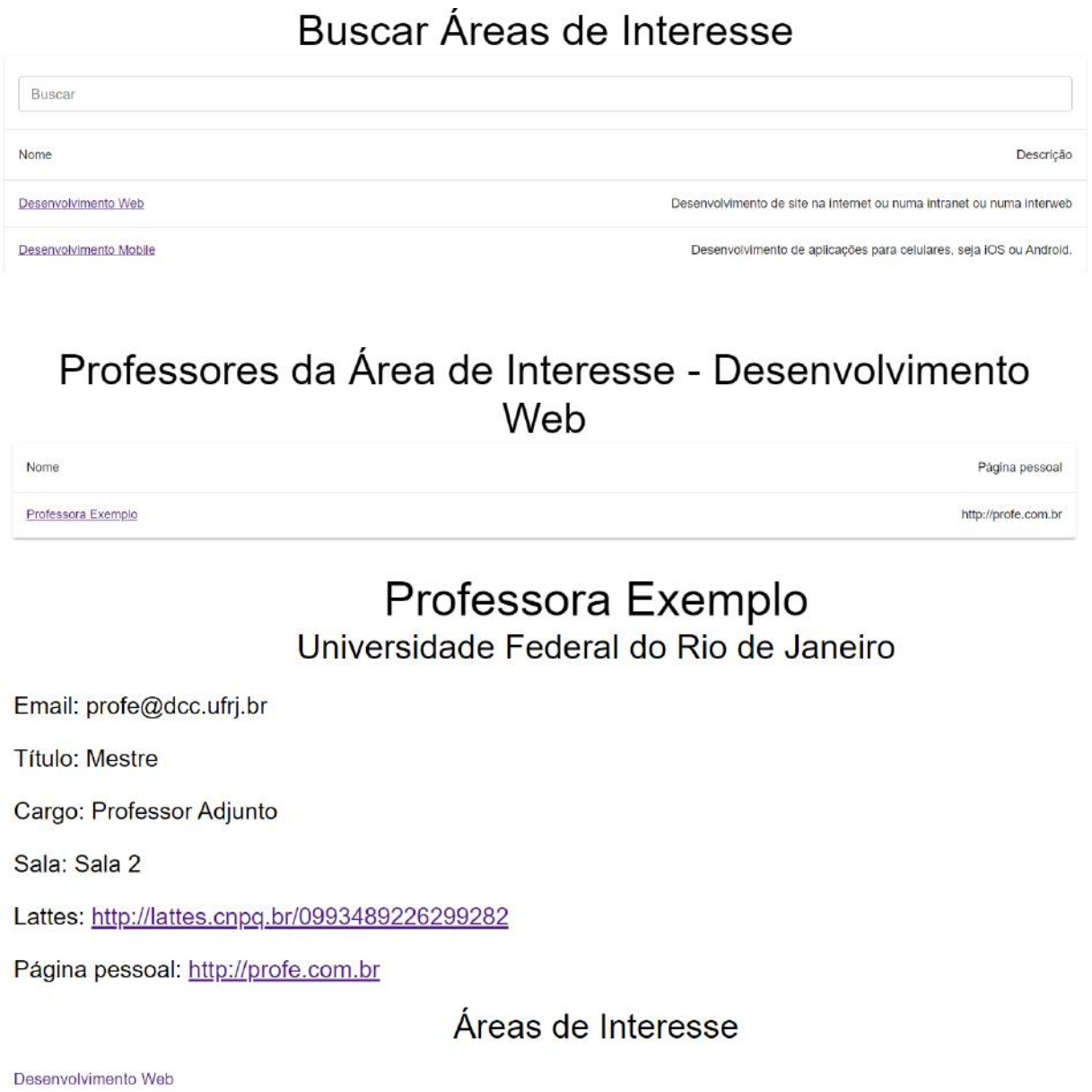
Na página inicial da plataforma, há uma barra de menus, sendo o "ÁREA PÚBLICA" onde são apresentados os itens "Áreas de Interesse" e "Projetos Finalizados", conforme figura abaixo.

Figura 1 : Barra inicial e menu após o clique em "Área Pública"

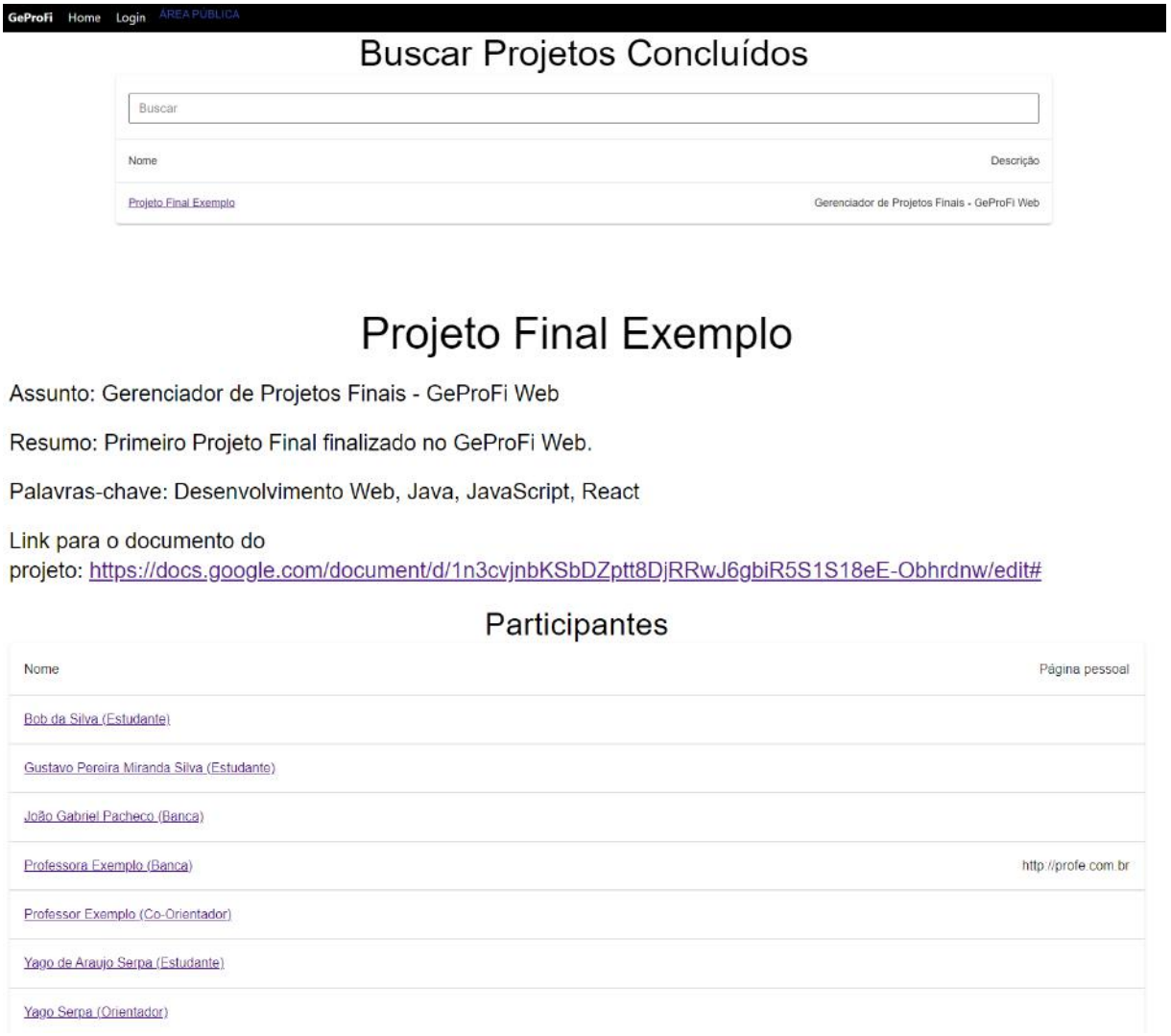


Nas Áreas de Interesse, ao clicar em alguma delas, abrirá a informação dos professores pertencentes a esta, e clicando no nome do professor, as informações deste aparecerão, como mostra a figura 2.

Figura 2 : Telas da Área de Interesse e Áreas de Interesse do Professor visitado



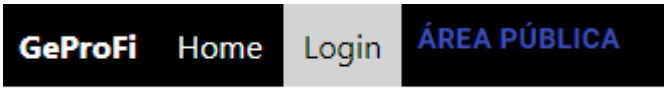
Nos projetos concluídos, é possível ver quem fez parte do projeto e também o link para o arquivo da defesa. Clicando nos participantes, também é possível ver informações em seus perfis, como mencionado na Área de Interesse.



2.2. Login e Esqueci Minha Senha

Na página inicial da plataforma, há uma barra de menus com um botão “Login”, exibido na figura 3.

Figura 3 : Barra com login selecionado



Ao clicar nesse botão o usuário é levado para a página de login, onde deve inserir seu e-mail e senha, como visto na figura 4 abaixo.

Figura 4 : Página de Login

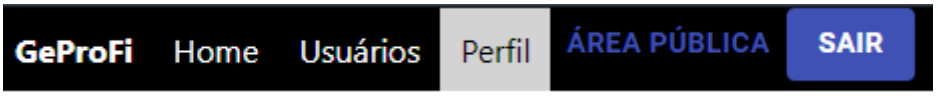
Há, nessa mesma página de login, um botão “Esqueci minha senha”, que abre um *pop-up* onde o usuário deve digitar seu e-mail para receber o link de troca de senha, como descrito na figura 5.

Figura 5 : *Pop-up* do “Esqueci minha senha” e tela de nova senha a ser inserida

2.3. Editar Perfil e Trocar Senha

Conectado à plataforma, um novo menu se abrirá na barra, “Perfil”, como na figura 6 abaixo.

Figura 6: Barra com perfil selecionado



Ao abrir este menu, o usuário edita suas informações pessoais e troca a sua senha, descrito pela figura 7.

Figura 7 : Página do perfil do usuário

Para efetuar a mudança de senha, o usuário deve fornecer a senha atual, exibido na figura 8. Caso não saiba, é recomendado clicar em “Esqueci minha senha”.

Figura 8: Trecho de mudança de senha

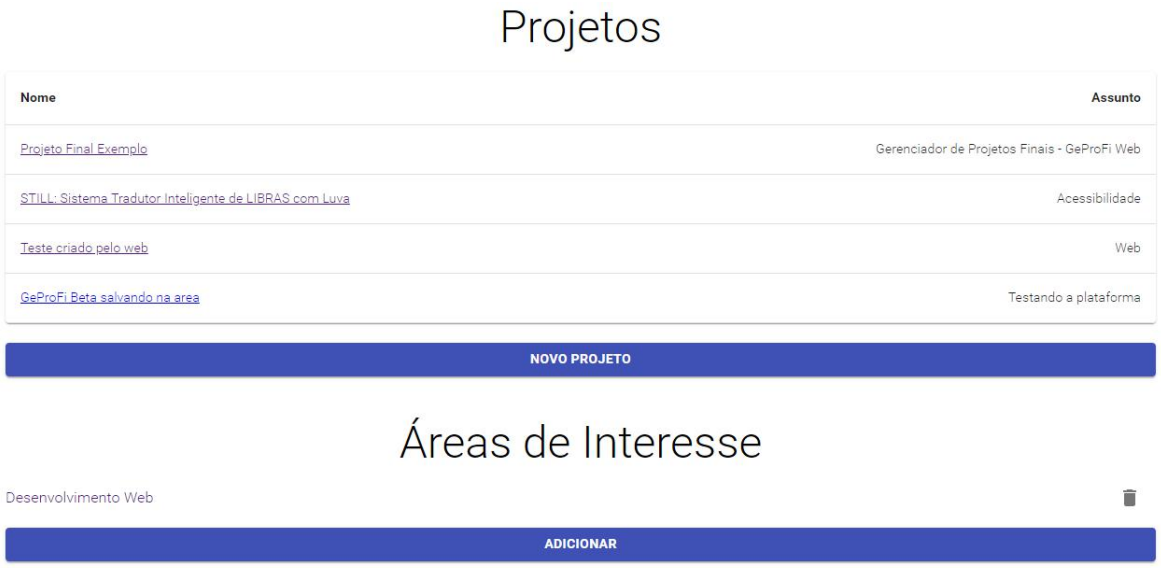
2.4. Página Inicial

A página inicial do usuário conectado ao sistema tem mais ou menos opções dependendo do tipo deste.
Estudantes e professores externos podem somente ver a lista de projetos que estão inseridos. Na figura 9 pode-se notar o nome e assunto dos projetos.

Figura 9: Lista de projetos em sua página

Já professores, além dessa mesma lista, podem criar novos projetos, ver quais áreas de interesse eles fazem parte, se conectar a áreas já existentes, criar novas, ou desconectar-se delas, mostrado na figura 10.

Figura 10 : Página de projetos e áreas de interesse do professor, que pode criá-las e editá-las



2.5. Criar Novo Projeto Final

Esta página é exclusiva para uso de professores (professores externos não podem fazê-lo). Na página inicial, clique no botão “NOVO PROJETO”, da figura 11.

Figura 11: Tela com o botão “Novo Projeto”

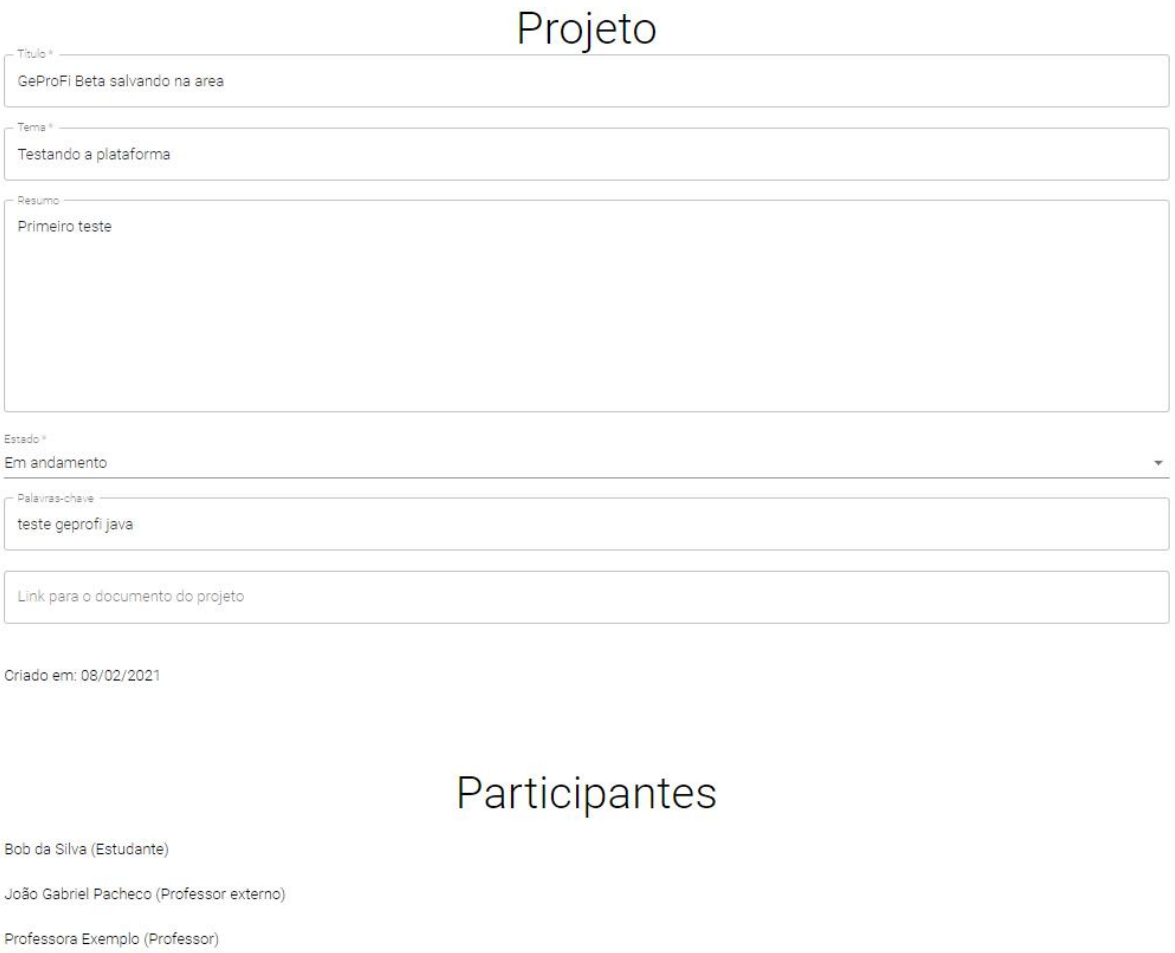
Preencha os dados da nova tela e clique em “CRIAR”, da figura 12.

Figura 12: Tela de Novo Projeto

2.6. Ver Projeto Final

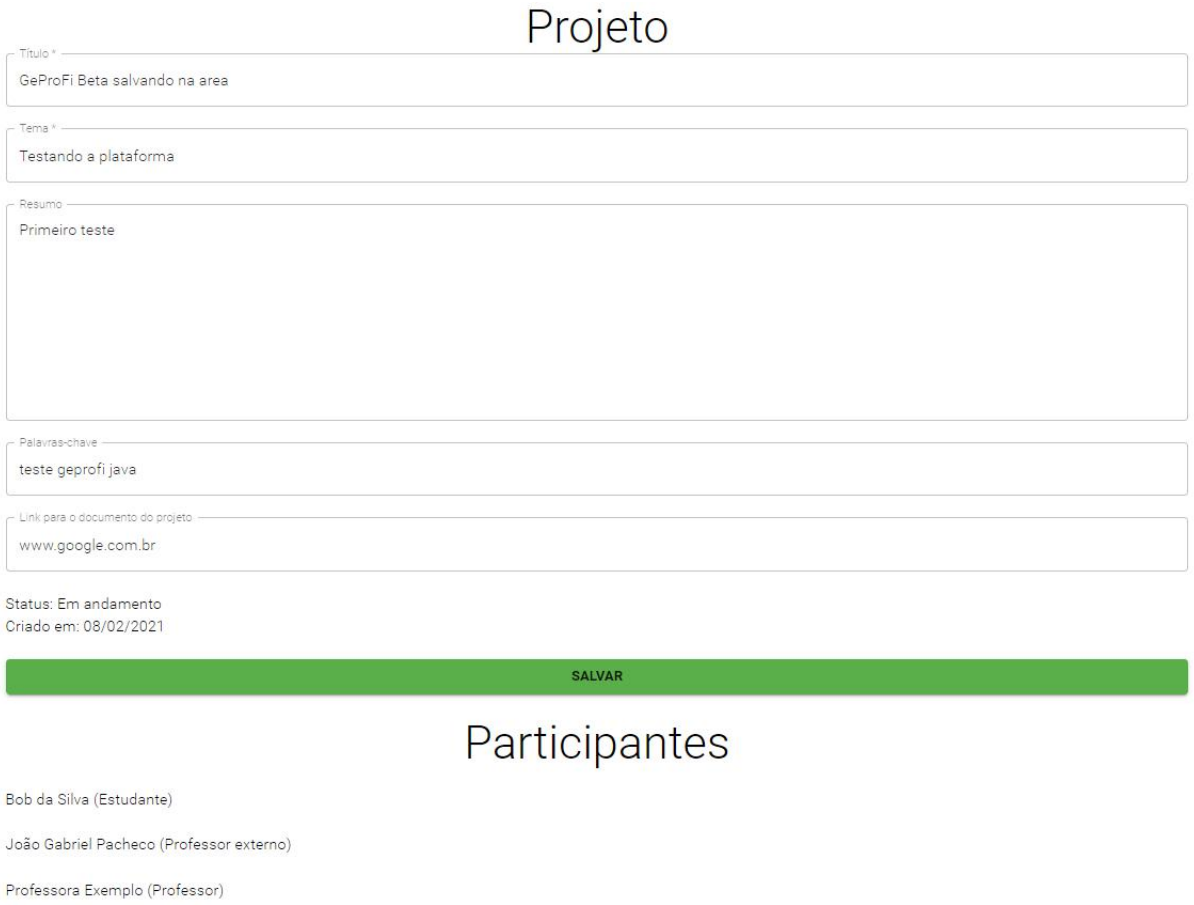
Para ver os dados de um projeto, basta clicar em seu nome na página inicial. Esta página tem mais ou menos opções dependendo do tipo de função.
Professores externos podem somente ver os dados do projeto e seus participantes, demonstrado na figura 13.

Figura 13: Projeto e seus participantes



Estudantes podem editar dados do projeto, mas não seu status e nem os participantes. A figura 14 mostra a visão do aluno no projeto.

Figura 14: Visão do aluno à tela de um projeto em andamento

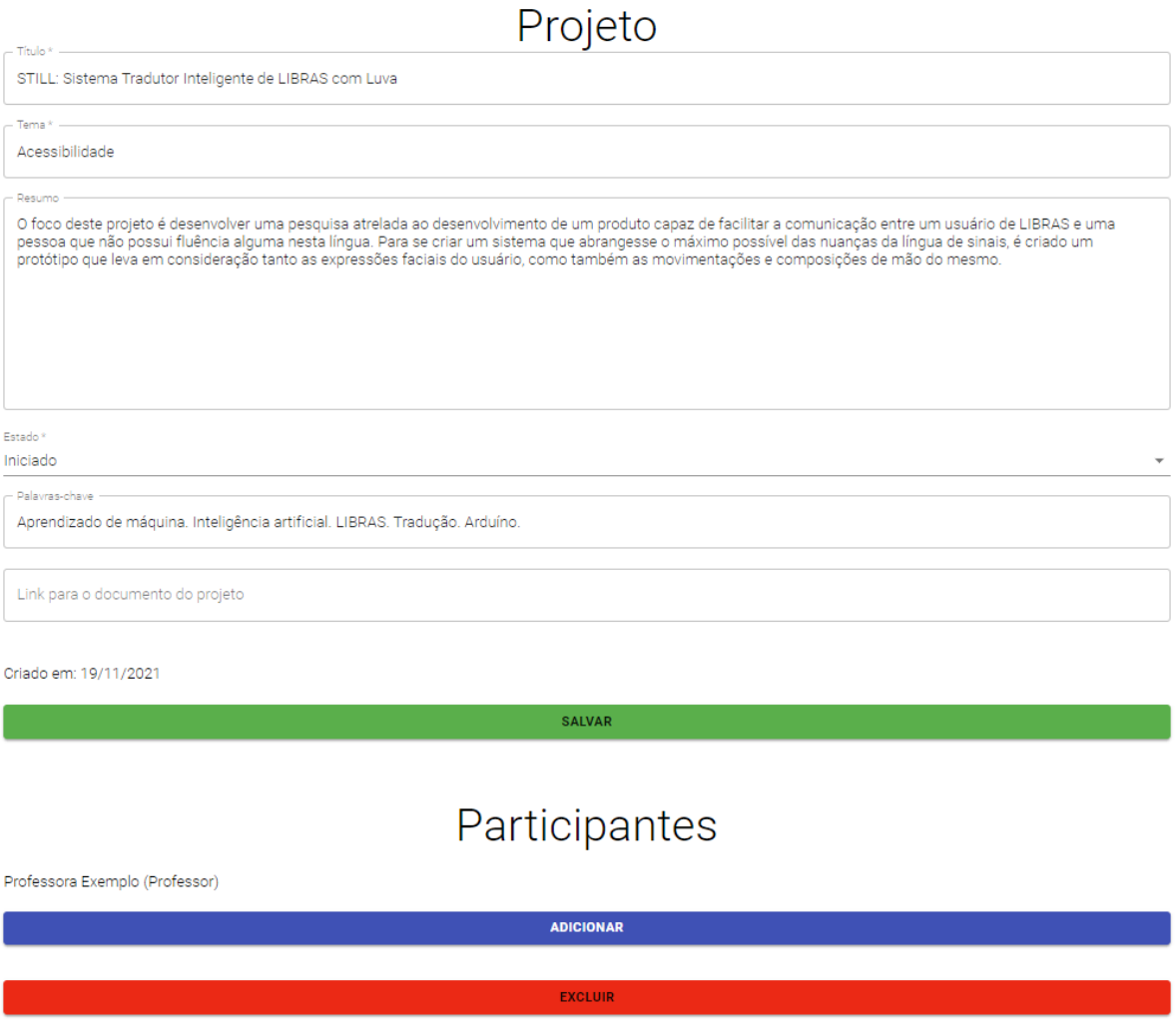


Ao adicionar ou alterar o *link* para o documento do projeto, o professor recebe um e-mail contendo o endereço do texto para posteriores revisões. Após a revisão por parte do professor, o estudante recebe um e-mail para ver as sugestões e alterações, e pode requisitar uma nova revisão através do botão "Mandar para revisão", conforme figura 15.

Figura 15: Campo do link para o documento do projeto com botão de "Mandar para revisão"

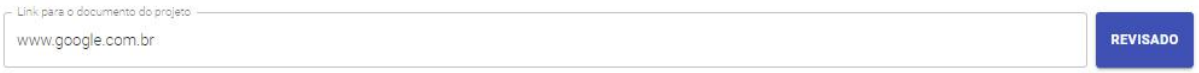
Já os professores podem adicionar novos participantes, editar o projeto e seu status, como visto na figura 16.

Figura 16: Visão do professor à tela de um projeto



Além disso, os professores podem revisar um texto já entregue pelo aluno quando o projeto estiver em andamento, basta clicar em “Revisado”, demonstrado na figura 17. O aluno será informado por e-mail sobre a revisão.

Figura 17: Campo do link para o documento do projeto com botão de “Revisado”



2.7. Marcar Defesa

Para marcar a defesa, é necessário trocar o Estado do projeto, de “Em Andamento” para “Pronto para defesa”. Para isso, uma data, hora e local iniciais deverão ser preenchidos para a disparar um e-mail aos participantes do projeto, conforme figura 18.

Figura 18: Campos de dados necessários para marcação de uma defesa

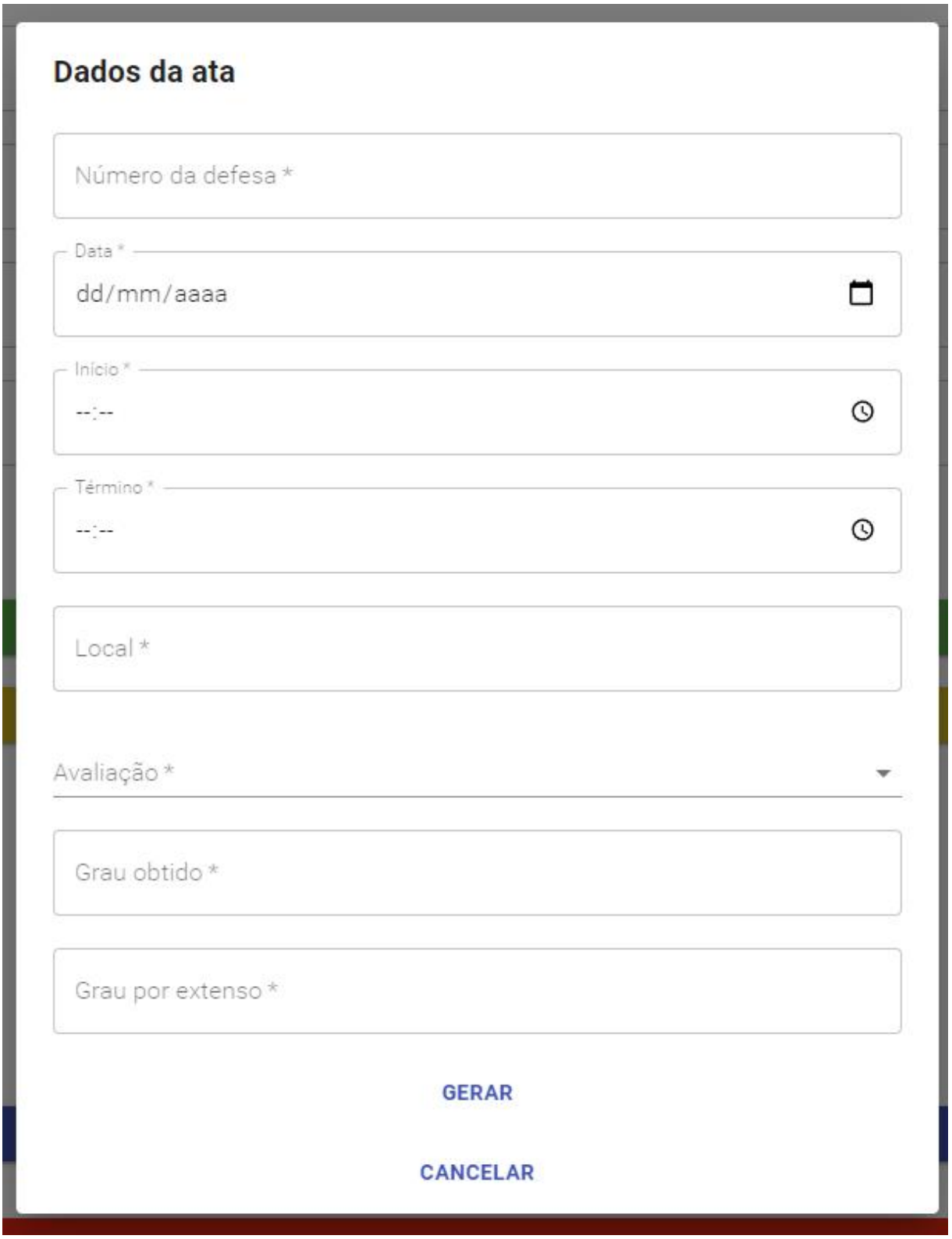
2.8. Gerar Ata

Após a defesa ser realizada, o professor pode gerar a ata clicando no botão “Gerar ata”, ilustrado na figura 19.

Figura 19: Botão de “Gerar ata”

Ao clicar no botão um *pop-up* se abre para preenchimento das informações que estarão presentes na ata, conforme figura 20.

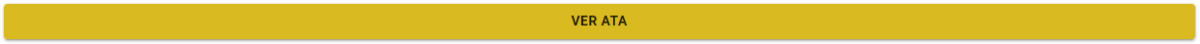
Figura 20: *Pop-up* de dados da ata



2.9. Ver Ata e Assinar Ata

Após a ata ser gerada, a página do projeto terá o botão “Ver ata”, descrito na figura 21.

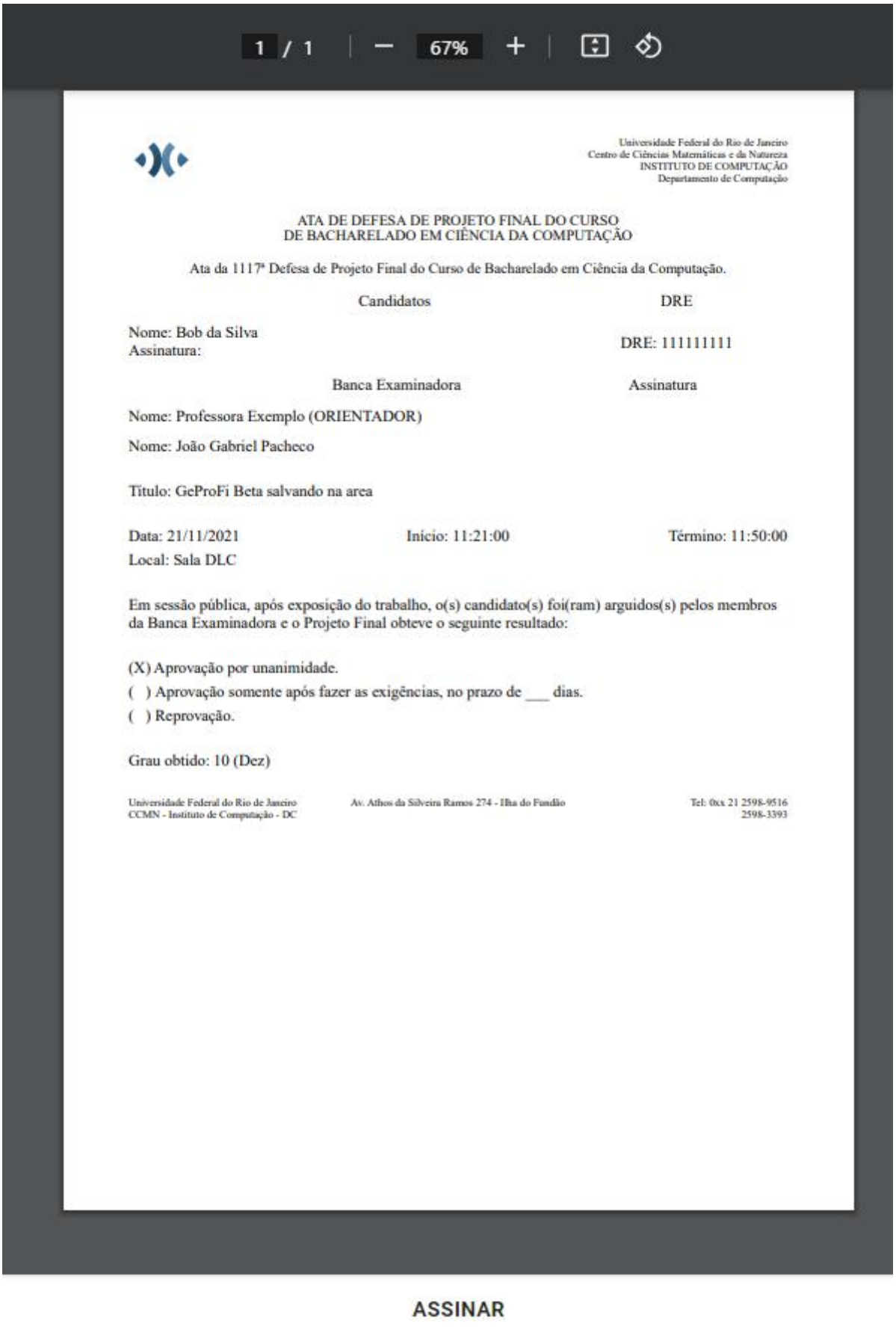
Figura 21: Botão de “Ver ata”



Enquanto todos os participantes do projeto não tiverem assinado, usuários de todos os tipos poderão ver esse botão. Assim que todas as assinaturas forem feitas, o projeto trocará o status automaticamente para “Finalizado”, e somente os professores Orientador e Co-Orientador poderão ver esse botão.

Para assinar a ata basta clicar no “Ver ata” e na página seguinte no botão “Assinar”. Um *pop-up* aparecerá para o usuário assinar por meio do *mouse*, mesa digitalizadora ou do *touchpad* como pode-se observar na figura 22.

Figura 22: Página de ata com botão de assinatura e *pop-up* de assinatura



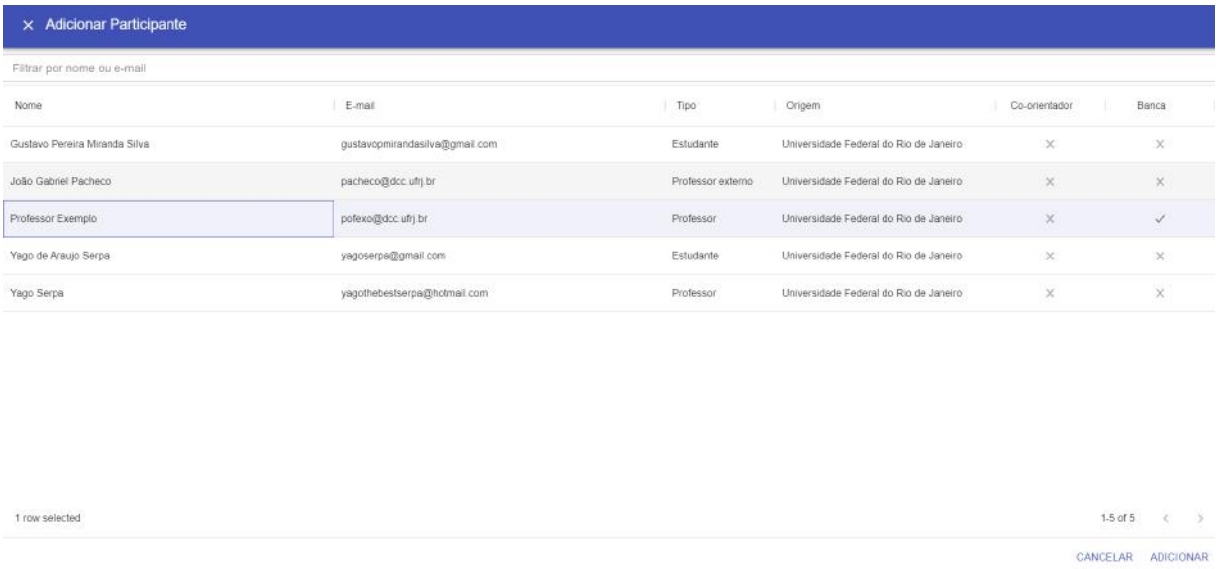
2.10. Adicionar Participantes

Na página do projeto, professores (que são orientadores ou co-orientadores) podem adicionar participantes já cadastrados na plataforma ao projeto, bastando ir ao segundo segmento da página, onde está "Participantes", e clicar em “ADICIONAR”, conforme figura 23.

Figura 23: Área de adição e remoção de participantes dentro da tela de um projeto em andamento

Uma nova janela se abrirá e o professor pode escolher a quem vai adicionar, clicando no nome do estudante, ou atribuindo a devida permissão, em casos de outros professores (banca ou co-orientador se o orientador estiver no controle), como na figura 24. Só é possível adicionar um usuário por vez, o que estiver selecionado no momento.

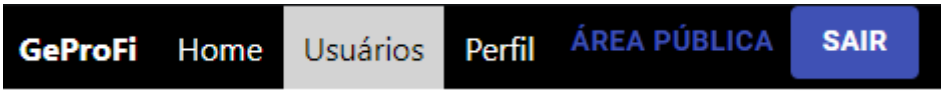
Figura 24: *Pop-up* de adicionar participante ao projeto



2.11. Ativar e Desativar Usuários

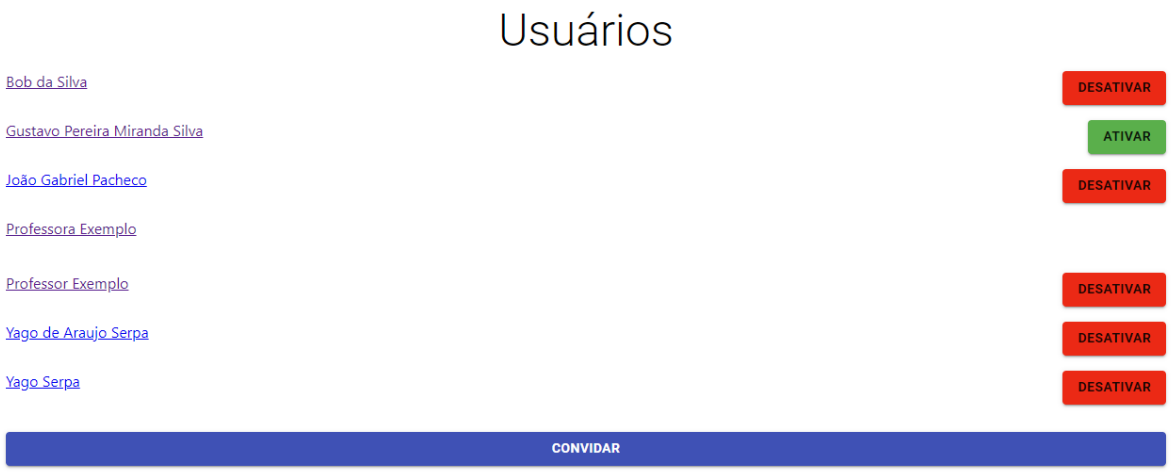
Conectado à plataforma, um novo menu se abre na barra, “Usuários”, como visto na figura 25.

Figura 25: Barra com usuários selecionado



Nessa página o professor pode ativar ou desativar usuários do sistema, clicando no botão “Desativar” ou “Ativar”, descrito na figura 26.

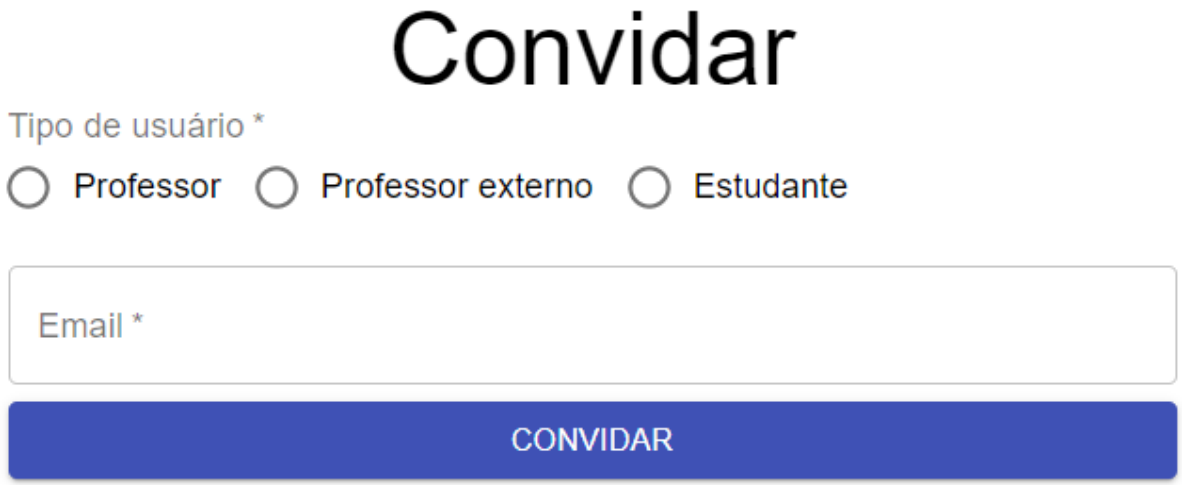
Figura 26: Lista de usuários ativados e desativados



2.12. Convidar Usuários

Na página de usuários, ao clicar em “CONVIDAR”, o professor irá para uma nova tela, onde pode inserir a função que o convidado terá (professor, professor externo ou estudante) e seu e-mail, ilustrada na figura 27. O convidado receberá um *link* para completar seu registro no sistema.

Figura 27: Tela de convidar usuário



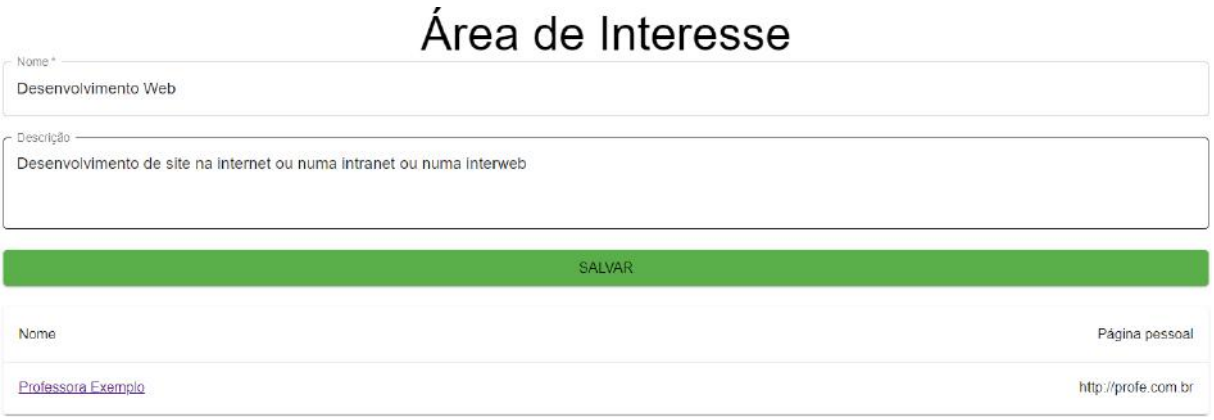
2.13. Editar Área de Interesse

Na página inicial, os professores podem clicar em uma Área de Interesse existente e fazer modificações a esta, listado abaixo na figura 28.

Figura 28: Página de Áreas de Interesse

Basta, na tela seguinte, clicar no botão “SALVAR”, mostrada na tela 29.

Figura 29: Tela de edição de área de interesse



2.14. Registro

Ao clicar no link recebido por e-mail, o usuário preenche todas as informações pertinentes e clica em "REGISTRAR" para finalizar e fazer uso da plataforma, de acordo com sua função, abaixo exibido pela figura 30.

Figura 30: Página de Registro

3. Tecnologias Utilizadas

3.1. Orientação a Objetos

Para o desenvolvimento do sistema utilizamos Orientação a Objetos (OO), que consiste numa abstração e representação simplificada do mundo real, facilitando a leitura e manutenção do código. É uma abordagem que permite uso da modelagem e programação baseada nos dados que temos, com um viés analítico mais robusto e mais realista [MENDES, 2009].

3.2. UML

A *Unified Modeling Language* (UML) é uma linguagem de notação usada para descrever a arquitetura, o design e a implementação de projetos e pode ser representada em diferentes diagramas que caracterizam cada parte do sistema [RUMBAUGH, 1996].

A UML é aceita como modelo padrão na construção e gerenciamento de requisitos para projetos desenvolvidos utilizando OO [DOBING e PARSONS, 2006]. Uma vez construídos, os diagramas são usados para agilizar a modelagem e o desenvolvimento, já que definem de forma clara o escopo do sistema e direcionam os desenvolvedores à solução.

3.3. Heroku

Heroku é um provedor em nuvem que permite a desenvolvedores e empresas a fazer builds, monitorar e escalar aplicações, sem toda a dor de cabeça sobre infraestrutura que tínhamos anteriormente. Possui bancos de dados escaláveis e instâncias que mantêm as aplicações disponíveis 24/7. Estamos usando o nível gratuito, que permite um banco com 10000 registros, 20 conexões simultâneas e máximo de 4 horas de *downtime* por mês. Após 30 minutos de inatividade, fica dormindo (*idle*) até receber novas requisições [HEROKU, 2021].

3.4. PostgreSQL

PostgreSQL é um sistema gerenciador de banco de dados relacional com mais de 30 anos de desenvolvimento ativo, de código aberto e que usa o padrão da linguagem SQL para consultas. Possui uma documentação bem elaborada que facilita seu uso [PSQL, 2021].

3.5. Spring

Spring é um framework para Java focado em velocidade, simplicidade e produtividade que, dentre suas muitas aplicações, pode ser utilizado para desenvolvimento de backend, no caso deste trabalho. Ele também garante segurança contra os ataques mais comuns e tem uma vasta comunidade com guias e tutoriais que ajudam no aprendizado e desenvolvimento de projetos [SPRING, 2021].

3.6. Java

A linguagem de programação utilizada no desenvolvimento do backend do projeto foi Java. Criada no ano de 1995 por James Gosling da Sun Microsystem, hoje propriedade da Oracle [TEC, 2009], é uma linguagem orientada a objetos, derivada das linguagens C e C++, e apresenta algumas características interessantes que motivam o seu uso, como as operações de baixo nível automatizadas. Possui um vasto conjunto de bibliotecas para as mais diversas necessidades.

Diferente de outras linguagens, que quando compiladas se tornam código nativo, a linguagem Java é compilada para um bytecode que é executado em uma máquina virtual, permitindo que, após compilado, o sistema possa ser executado em qualquer computador, independente do sistema operacional e arquitetura.

3.7. React

React é uma biblioteca Javascript de código aberto para criar interfaces de usuário em páginas web, desenvolvido pelo Facebook em 2013. É utilizado em sites de grandes empresas como

Netflix, Dropbox, o próprio Facebook, entre outros. É uma biblioteca simples, de fácil aprendizado, rápida, fácil de testar, com grande capacidade de reutilização de componentes e uma comunidade imensa que estende suas funcionalidades [REACT, 2021].

3.8. Javascript

A linguagem de programação utilizada no desenvolvimento do frontend do projeto foi Javascript [JS, 2021]. É uma linguagem de programação de scripts de alto nível, padronizada pela ECMA International (European Computer Manufacturers Association) criada por Brendan Eich (Netscape) em 1995. Foi pensada como um complemento para Java e tem sintaxe semelhante. É, atualmente, a principal linguagem para programação *client-side* em navegadores web, mas também é bastante utilizada do lado do servidor, através de frameworks.

3.9. DBeaver

DBeaver [DBEAVER, 2021] é uma ferramenta que pode ser utilizada com qualquer sistema de gerenciamento de banco de dados. É um ambiente completo que, uma vez conectado ao banco, permite criar, alterar e apagar tabelas e registros contidos nele, tanto na forma de consultas quanto por interface gráfica.

3.10. SQL Power Architect

O SQL Power Architect [PWR, 2021] é um programa que permite criar modelos Entidade-Relacionamento, modelos lógicos e, além disso, disponibiliza códigos de criação das tabelas e atualização de colunas, caso se altere o tratamento para todos os bancos relacionais (no nosso caso, utilizamos para o PostgreSQL).

3.11. IntelliJ

Para a programação do sistema foi utilizada a IDE (Integrated Development Environment) IntelliJ [INTELLIJ, 2021]. Desenvolvida pela empresa tcheca JetBrains em janeiro de 2001,

vem sendo atualizada com melhorias e novas funcionalidades constantemente. Foi inicialmente projetada para o desenvolvimento em Java, mas seu framework é utilizado como base para IDEs de outras linguagens. Como é escrita em Java, pode ser utilizada em diversos sistemas operacionais.

3.12. Git/GitHub

O Git é um sistema de controle e versionamento gratuito e de código aberto, utilizado para versionar o código do projeto, seja do backend ou do frontend [GIT, 2021]. É a plataforma onde hospedamos os códigos e as versões do projeto. É possível dividir o código principal em ramificações e trabalhar em diferentes *features* em paralelo para convergir mais rapidamente em um produto pronto.

3.13. Google Meet

Plataforma da Google onde as conferências para alinhamento do projeto entre os integrantes e orientação aconteceram durante todo o desenvolvimento do mesmo.

3.14. Google Docs

Plataforma da Google onde o Trabalho de Conclusão do Curso foi escrito e editado, respeitando as normas impostas pelo Departamento de Ciência da Computação.

3.15. Material UI

Framework criado pelo Google para React com o propósito de facilitar o uso de componentes em implementações de interface de usuário [MUI, 2021]. Auxiliou bastante na reutilização de código de alguns componentes do projeto.

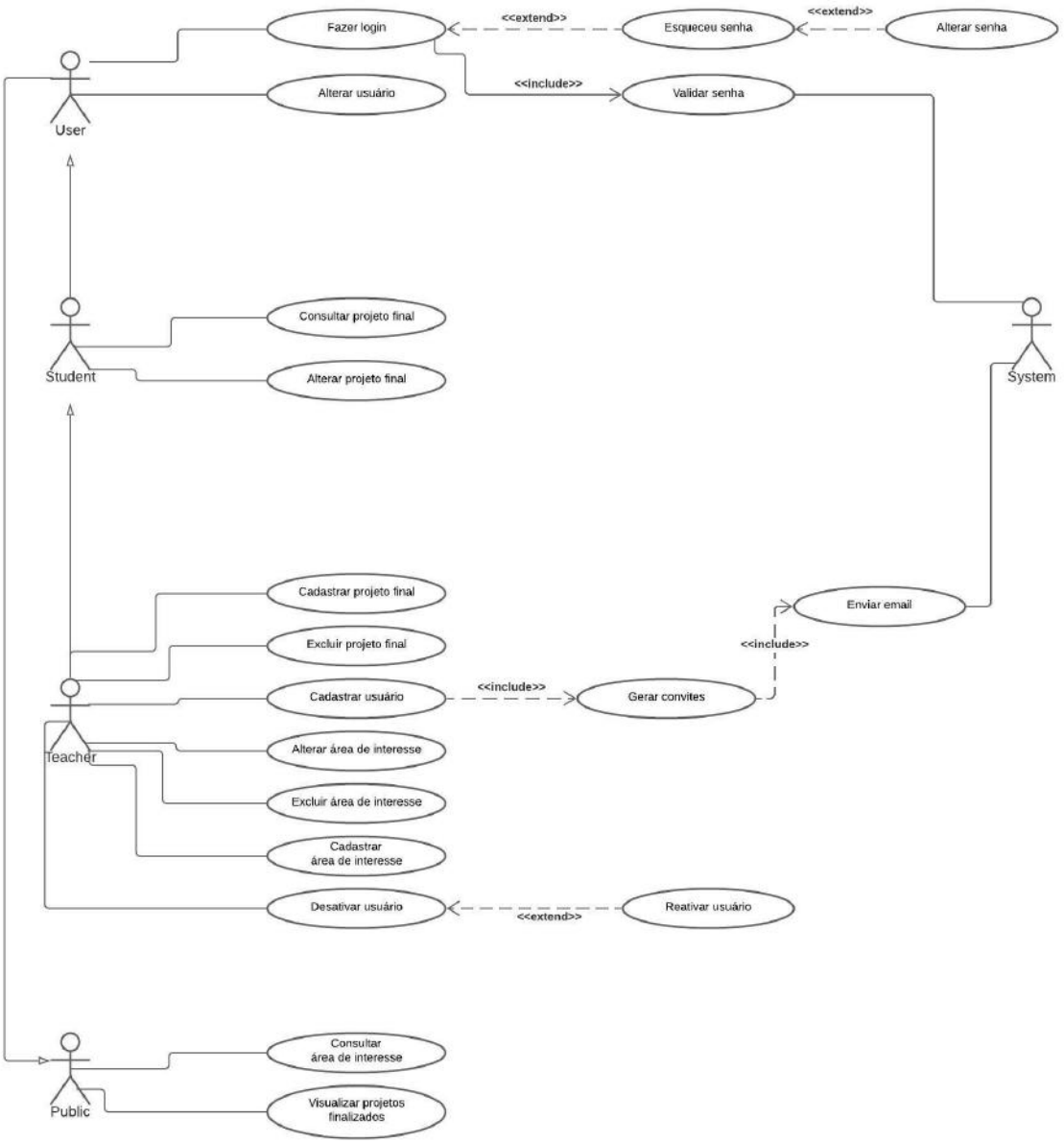
3.16. Características Técnicas da Aplicação

A aplicação possui aproximadamente 3 mil linhas de código em seu frontend e 2 mil linhas em seu backend. Está funcionando em um servidor do plano gratuito do Heroku, com 512 MB de RAM e 1 CPU Share (não discrimina o clock), além de também não informar o armazenamento disponível. O plano gratuito não garante disponibilidade 24/7, com isso, mantemos a aplicação de pé por meio de um comando lançado a cada 29 minutos (em 30 minutos os componentes dormem automaticamente por conta de gastos do próprio Heroku), *curl -I http://geprofi-front.herokuapp.com/public/projects*, e funciona 18 horas por dia, das 6 às 23:59. Acredita-se que uma máquina simples (2 GB RAM, 2GHz CPU, 10 GB HD) consiga rodar tranquilamente esta aplicação. Para instalar as dependências do frontend, basta seguir a documentação de [FRONTEND, 2022], o mesmo vale para o backend, em [BACKEND, 2022].

4. Diagrama de casos de uso

O diagrama de casos de uso resume as interações dos diferentes tipos de usuários do sistema, chamados de “atores”, com as funcionalidades do mesmo, chamados de “caso de uso”, além de definir as relações de dependência e/ou associação entre os casos de uso, determinando a modelagem dos requisitos do sistema de acordo com sua composição. Isso dá sustentação ao uso por parte dos atores e suporte ao comportamento do sistema [GUTIERREZ, 2011]. O diagrama detalhado de casos de uso se encontra na imagem abaixo.

Figura 31: Diagrama de Casos de Uso com atores



4.1. Área Pública

A área pública da plataforma é o local onde usuários não cadastrados podem acessar projetos já finalizados e pesquisar áreas de interesse e seus respectivos professores.

4.2. Estudante

O estudante pode editar seus dados, ver e alterar dados dos projetos ao qual está conectado e requisitar revisão de alterações no texto do projeto, além de assinar a ata após a defesa.

4.3. Professor

O Professor pode editar seus dados, convidar usuários (tanto para participar do projeto, quanto para fazer parte da banca), desativar e reativar usuários, criar, editar e excluir áreas de interesse, criar, ver e alterar projetos ao qual está conectado, marcar defesa, gerar e assinar ata.

4.4. Professor Externo

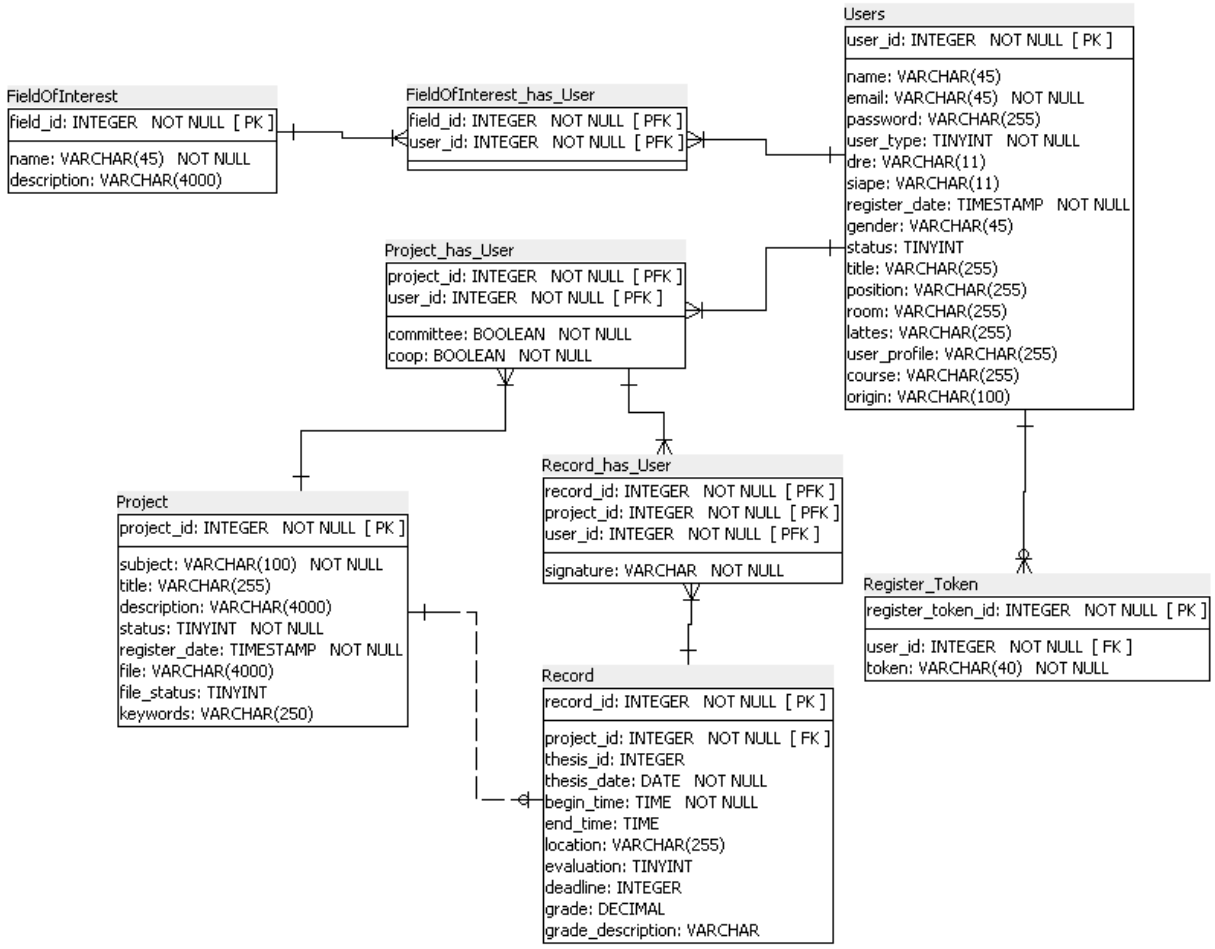
O Professor Externo pode editar seus dados, ver os projetos ao qual está conectado e assinar a ata após a defesa.

5. METODOLOGIAS APLICADAS AO DESENVOLVIMENTO DO GEPROFI

5.1. Modelo Lógico

Neste tópico, há menção às tabelas, conexões entre tabelas e seus conteúdos. O dicionário de dados completo está nos apêndice A e o modelo lógico na figura 32 abaixo.

Figura 32: Modelo Lógico do Banco de Dados



5.1.1. FieldOfInterest

A tabela FieldOfInterest armazena as informações sobre as áreas de interesse registradas na plataforma.

5.1.2. FieldOfInterest_Has_User

A tabela FieldOfInterest_Has_User é a tabela-conexão entre as tabelas de Usuários (Users) e de Áreas de Interesse (FieldOfInterest).

5.1.3. Users

A tabela Users guarda as informações dos usuários cadastrados no sistema.

5.1.4. Register_Token

A tabela Register_Token salva os *tokens* para verificar se um convite de novo usuário é válido.

5.1.5. Project

A tabela Project tem todas as informações dos projetos presentes na plataforma.

5.1.6. Project_Has_User

A tabela Project_Has_User é a tabela-conexão entre as tabelas de Usuários (Users) e de Projetos (Project), além da *flag* que determina se um professor é membro da banca, orientador ou co-orientador do projeto.

5.1.7. Record

A tabela Record salva todos os dados das atas dos projetos e é usada para gerar o arquivo final da Ata em PDF.

5.1.8. Record_Has_User

A tabela Record_Has_User é a tabela-conexão entre as tabelas de Usuários (Users) e de Atas (Record) e armazena as assinaturas da ata dos membros do projeto.

5.2 Backend

Como previamente mencionado, o backend do projeto foi desenvolvido em Java. Serão pontuados aqui cada módulo e suas respectivas funções.

5.2.1. Database Config

Esse módulo contém um único arquivo onde estão armazenadas as informações do banco de dados ao qual o backend se conecta. Por questões de segurança, os dados sensíveis (como usuário, senha, endereço IP do banco, tamanho de pool de conexões) foram definidos em um arquivo diferente chamado application.properties, que não fica salvo no repositório do Git, ao qual o DatabaseConfig.java faz referência. Ainda são guardados nesse arquivo o endereço de conexão ao banco, o usuário e a senha e o máximo de conexões permitidas do *backend* ao banco.

5.2.2. Controllers

O *controller* define as rotas que o frontend utiliza para recuperar e/ou atualizar dados no banco de dados, bem como a implementação de cada rota e o que ela retorna. Nós temos vários *controllers* separados por área do frontend, como usuários, áreas de interesse, atas, projetos e arquivos. Na Figura 33 é mostrado o código de um deles.

Figura 33: Código do *Controller* para arquivos

```java
package br.yagoserpa.geprof.controller;

import br.yagoserpa.geprof.model.File;
import br.yagoserpa.geprof.repository.FileRepository;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class FileController {

    private final FileRepository fileRepository;

    public FileController(FileRepository fileRepository) {
        this.fileRepository = fileRepository;
    }

    @GetMapping("/api/v1/file/{id}")
    public File find(
            @PathVariable(value = "id") Integer id
    ) {
        return fileRepository.findById(id).orElse(null);
    }

    @GetMapping("/api/v1/file")
    public List<File> all() {
        return fileRepository.findAll();
    }

    @PostMapping("/api/v1/file/")
    public void insert(
            @RequestBody File file
    ) {
        fileRepository.insert(file);
    }

    @PutMapping("/api/v1/file/{id}")
    public void update(
            @PathVariable(value = "id") Integer id,
            @RequestBody File file
    ) {
        fileRepository.update(id, file);
    }

}
```

5.2.3. Filters

Os *filters* têm a função de manipular ou bloquear requisições de acordo com os objetivos da aplicação e do mesmo. Temos dois *filters* em nosso trabalho, um de CORS (*Cross-Origin Resource Sharing*) e um de JWT (*JavaScript Object Notation Web Token*). O filtro de CORS permite que as requisições HTTP (*Hypertext Transfer Protocol)* vindas do *frontend* sejam aceitas pelo *backend*, pois os dois estão hospedados em diferentes endereços de origem. O de JWT é o filtro que valida o acesso público do acesso privado. Se um usuário não logado tentar

acessar a situação de um projeto, ele será bloqueado. Na Figura 34 observa-se a configuração do *filter* do CORS.

Figura 34: Código do *Filter* de CORS

```
package br.yagoserpa.geprof.filter;

import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import javax.servlet.*;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Configuration
public class CorsFilter implements Filter {

    @Bean
    public FilterRegistrationBean<CorsFilter> corsFilterRegistration() {
        FilterRegistrationBean<CorsFilter> registration = new FilterRegistrationBean<>();
        registration.setFilter(this);
        registration.addUrlPatterns("/**/*", "/*");
        registration.setName("corsFilter");
        registration.setOrder(2);
        return registration;
    }

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) throws IOException, ServletException {
        var httpResponse = (HttpServletResponse) servletResponse;
        httpResponse.setHeader("Access-Control-Allow-Origin", "*");
        httpResponse.setHeader("Access-Control-Allow-Headers", "*");
        httpResponse.setHeader("Access-Control-Allow-Methods", "*");
        filterChain.doFilter(servletRequest, servletResponse);
    }
}
```

5.2.4. Models

Os *models* são a fonte dos atributos, onde se localizam os construtores, gets e sets de todas as tabelas que temos no banco de dados, além de tratamentos específicos para campos que são texto no frontend e vão ao banco como números. Na Figura 35 é mostrado o código da Área de Interesse.

Figura 35: Código do *Model* de Área de Interesse

```java
package br.yagoserpa.geprof.model;

import java.sql.ResultSet;
import java.sql.SQLException;

public class FieldOfInterest {

    private Integer id;
    private String name;
    private String description;

    public FieldOfInterest() {
    }

    public FieldOfInterest(ResultSet resultSet, int row) throws SQLException {
        id = resultSet.getInt("field_id");
        name = resultSet.getString("name");
        description = resultSet.getString("description");
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }
}
```

5.2.5. Repositories

Os *repositories* contêm a comunicação entre o backend e o banco de dados. Todas as consultas (select, insert, update, delete) acontecem partindo do repository. Construímos separadamente duas versões de *repository* para cada uma de nossas tabelas e para armazenar as informações geradas pelo token JWT, para que não percamos a informação de login do

usuário que pode ver informações privadas do site. Na Figura 36, descreve-se o código do Registro do Token.

Figura 36: Código do *Repository* de Registro do Token

```java
package br.yagoserpa.geprof.repository;

import br.yagoserpa.geprof.model.RegisterToken;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Component;

import java.util.List;
import java.util.Optional;

@Component
public class RegisterTokenRepositoryImpl implements RegisterTokenRepository {

    private final JdbcTemplate template;

    public RegisterTokenRepositoryImpl(
            JdbcTemplate template
    ) {
        this.template = template;
    }

    @Override
    public Optional<RegisterToken> findByUserId(Long userId) {
        List<RegisterToken> users = template.query("SELECT * FROM register_token WHERE user_id = ? LIMIT 1",
                RegisterToken::new,
                userId);
        if (users.isEmpty()) {
            return Optional.empty();
        }
        return Optional.of(users.get(0));
    }

    @Override
    public Optional<RegisterToken> findByToken(String token) {
        List<RegisterToken> users = template.query("SELECT * FROM register_token WHERE token = ? LIMIT 1",
                RegisterToken::new,
                token);
        if (users.isEmpty()) {
            return Optional.empty();
        }
        return Optional.of(users.get(0));
    }

    @Override
    public void insert(RegisterToken registerToken) {
        template.update("INSERT INTO register_token (user_id, token) VALUES (?, ?)",
                registerToken.getUserId(),
                registerToken.getToken()
        );
    }

    @Override
    public void delete(Integer id) {
        template.update("DELETE FROM register_token WHERE register_token_id = ?",
                id);
    }
}
```

5.2.6. GeprofApplication

Resumidamente, é o gatilho para iniciar o backend do projeto. Na Figura 37, é mostrado o código do GeprofApplication.

Figura 37: Código do GeprofApplication

```
package br.yagoserpa.geprof;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class GeprofApplication {

	public static void main(String[] args) {
		SpringApplication.run(GeprofApplication.class, args);
	}

}
```

5.3. Frontend

Foi desenvolvido em JavaScript utilizando React, uma biblioteca para criação de interface do usuário e Material UI, um framework para React das mesmas interfaces. Criaram-se *components*, *contexts*, *hooks*, *models* e *pages*. Além da página principal (App.js), que funciona similarmente à uma main.

5.3.1. Components

Os *components* contêm o HTML (*HyperText Markup Language*) das páginas, ou seja, toda a estrutura das páginas e subpáginas são escritas neste atributo do JavaScript. Na Figura 38, observa-se o código do componente do usuário.

Figura 38: Código do *Component* UserList

```
import React from "react";
import { Link } from "react-router-dom";
import {
  TableContainer,
  Paper,
  TableBody,
  TableRow,
  TableCell,
  Table,
  TableHead,
} from "@material-ui/core";

function UserList({ userList }) {
  return (
    <TableContainer component={Paper}>
      <Table>
        <TableHead>
          <TableRow>
            <TableCell>Nome</TableCell>
            <TableCell align="right">Página pessoal</TableCell>
          </TableRow>
        </TableHead>
        <TableBody>
          {userList.map((user) => (
            <TableRow key={user.id}>
              <TableCell component="th" scope="row">
                <Link to={`/user/${user.id}/fields`}>{user.name}</Link>
              </TableCell>
              <TableCell align="right">{user.userProfile}</TableCell>
            </TableRow>
          ))}
        </TableBody>
      </Table>
    </TableContainer>
  );
}

export default UserList;
```

5.3.2. Context

O context armazena os dados do usuário e os outros componentes do *frontend* (component, pages) consomem tais dados consultando a Context API. O código da Context API é muito grande para ser posto em uma imagem, na Figura 39 vê-se apenas um trecho.

Figura 39: Trecho de código da Context API

```
import React, { useContext, useState } from "react";
import Axios from "axios";
import LoadingOverlay from "react-loading-overlay";
import { useSnackbar } from "react-simple-snackbar";

const api = Axios.create({
  baseURL: process.env.REACT_APP_BASE_URL,
  headers: {
    Accept: "application/json",
    Authorization: "Bearer " + localStorage.getItem("@GeProFi:token"),
  },
});

const successOptions = {
  style: {
    backgroundColor: "#5AAF4B",
    color: "white",
  },
};

const errorOptions = {
  style: {
    backgroundColor: "#EA402F",
    color: "white",
  },
};

export function useApi() {
  const context = useContext(ApiContext);

  return context;
}

const ApiContext = React.createContext({});

export const ApiProvider = ({ children }) => {
  const [loading, setLoading] = useState(false);
  const [successSnackbar, _] = useSnackbar(successOptions);
  const [errorSnackbar, __] = useSnackbar(errorOptions);

  async function doCall(call, callback, errorCallback, showSuccess) {
    setLoading(true);
    call
      .then((response) => {
        setLoading(false);
```

5.3.3. Hook

O Hook no contexto do projeto tem a função de coletar o token no momento em que um usuário é convidado para fazer registro, e validar o token recebido. Na Figura 40 descreve-se o código do hook.

Figura 40 : Código do Hook

```
import { useLocation } from "react-router-dom";

function useQuery() {
  return new URLSearchParams(useLocation().search);
}

export { useQuery };
```

5.3.4. Models

Os *models* descrevem os campos que no backend e no banco têm descrição diferente (no caso, a representação é numérica e vai de 0 a 2 ou 0 a 3). Na Figura 41 vê-se o exemplo do *model* do usuário.

Figura 41: Código do *Model* do usuário

```
export const UserType = {
  EXTERNAL_TEACHER: "Professor externo",
  TEACHER: "Professor",
  STUDENT: "Estudante",
};
```

5.3.5. Pages

As *pages* fazem uso dos *components* e da Context API, carregam os dados destes e exibem no frontend. Na Figura 42 vê-se o exemplo da *page* de página pública.

Figura 42: Código da *page* de página pública

```
import React, { useState } from "react";
import { Container, Typography } from "@material-ui/core";
import FieldOfInterestList from "../components/FieldOfInterestList";
import { useEffect } from "react";
import { useApi } from "../contexts/ApiContext";

function PublicPage() {
  const { apiGet } = useApi();
  const [fieldOfInterestList, setFieldOfInterestList] = useState([]);

  useEffect(() => {
    function onListLoaded(data) {
      setFieldOfInterestList(data);
    }

    apiGet("public/field", onListLoaded);
  }, []);

  return (
    <Container component="article">
      <Typography variant="h3" component="h1" align="center">
        Buscar Áreas de Interesse
      </Typography>
      <FieldOfInterestList fieldOfInterests={fieldOfInterestList} />
    </Container>
  );
}

export default PublicPage;
```

5.3.6. App.js

O App.js concentra as rotas para cada *page* em determinado item (da página inicial para o login, de logado para a criação de projeto, etc). Na Figura 43 vemos o código do App.js.

Figura 43: Código do App.js

```
import { BrowserRouter as Router, Switch, Route } from "react-router-dom";
import NavHeader from "./components/NavHeader";
import LoginPage from "./pages/LoginPage";
import PublicPage from "./pages/PublicPage";
import PublicProjectsPage from "./pages/PublicProjetcsPage";
import NotFoundPage from "./pages/NotFoundPage";
import HomePage from "./pages/HomePage";
import FieldOfInterestPage from "./pages/FieldOfInterestPage";
import UserPage from "./pages/UserPage";
import { ProjectPage, PublicProjectPage } from "./pages/project";
import InvitePage from "./pages/InvitePage";
import RegisterPage from "./pages/RegisterPage";
import { NewProjectPage } from "./pages/newproject";
import { ApiProvider } from "./contexts/ApiContext";
import SnackbarProvider from "react-simple-snackbar";
import ProfilePage from "./pages/profile/ProfilePage";

function App() {
  return (
    <SnackbarProvider>
      <ApiProvider>
        <Router>
          <div id="pagina">
            <NavHeader />
            <Switch>
              <Route exact path="/">
                <HomePage />
              </Route>
              <Route path="/login">
                <LoginPage />
              </Route>
              <Route path="/public/fields">
                <PublicPage />
              </Route>
              <Route path="/public/projects">
                <PublicProjectsPage />
              </Route>
              <Route path="/public/project/:id">
                <PublicProjectPage />
              </Route>
              <Route path="/field/:id/users">
                <FieldOfInterestPage />
              </Route>
              <Route path="/user/:id/fields">
                <UserPage />
              </Route>
              <Route path="/newproject">
                <NewProjectPage />
              </Route>
              <Route path="/project/:id">
                <ProjectPage />
              </Route>
              <Route path="/invite">
                <InvitePage />
              </Route>
              <Route path="/register">
                <RegisterPage />
              </Route>
              <Route path="/profile">
                <ProfilePage />
              </Route>
              <Route>
                <NotFoundPage />
              </Route>
            </Switch>
          </div>
        </Router>
      </ApiProvider>
    </SnackbarProvider>
  );
}
```

6 Conclusão

O GeProFi auxilia em todo o processo do trabalho de conclusão de curso. Desde a dúvida do aluno por qual área de interesse seguir, pesquisar sobre os professores, encontrar os dados necessários e ver trabalhos prévios, até passos mais avançados, como a criação da ata completa e com assinatura de todos os responsáveis, seja o trabalho terminado de maneira presencial ou remota.

A maior dificuldade apresentada pelo trabalho foi a concepção de como garantir uma assinatura segura e factível para que estivesse presente na ata. A procura encontrou soluções mais fáceis e práticas, porém caras, e no fim, optou-se pela utilização de um canvas em branco que disponibiliza assinatura por meio do *mouse*, mesa digitalizadora ou do *touchpad*.

7 Trabalhos Futuros

Há muitas maneiras de aprimorar o GeProFi na conjuntura atual em que ele está. Como tratamento (edição e submissão) de arquivos dentro do ambiente ao invés de redirecionamento ao *link* do Google Drive, OneDrive, Overleaf ou qualquer outra plataforma de edição de arquivos.

Uma forma de assinatura melhor do que utilizar o periférico disponível, assinatura digital, por exemplo.

Utilizar uma plataforma em que os recursos sejam ilimitados, como Amazon Web Services (AWS), Google Cloud Platform (GCP) ou mesmo um plano dentro do próprio Heroku, todos estes pagos por utilização.

Implementar um chat entre os membros do projeto, que seja extensivo à banca conforme o projeto avança.

Generalizar o conteúdo da ata, assim todos os cursos da UFRJ poderiam aproveitar o advento desta plataforma e vincular a área de interesse às palavras-chave do projeto.

Adicionar um calendário que mostre as defesas futuras e até mesmo correntes para que traga mais engajamento à comunidade discente em relação aos projetos desenvolvidos na universidade.

Criar uma distribuição de projetos finais por áreas de interesse permite aos alunos verem exemplos dos tipos de projetos que são realizados em determinada área, podendo assim facilitar sua escolha de TCC.

REFERÊNCIAS

[BACKEND, 2022] Getting Started. **Disponível em: https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started.html#getting-started.installing .** Acesso em Janeiro de 2022.

[DBEAVER, 2021] DBeaver - Universal Database Tool. **Disponível em: https://dbeaver.io .** Acesso em Julho de 2021.

[DOBING e PARSONS, 2006] DOBING, Brian; PARSONS, Jeffrey. **How UML is used.** *Communications of the ACM*, 2006, 49. Jg., Nr. 5, S. 109-113.

[FRONTEND, 2022] Downloading and installing Node.js and npm. **Disponível em: https://docs.npmjs.com/downloading-and-installing-node-js-and-npm .** Acesso em Janeiro de 2022.

[GIT, 2021] Github. **Disponível em: https://github.com .** Acesso em Julho de 2021.

[GUTIERREZ, 2011] GUTIERREZ, Demián. **Casos de uso Diagramas de Casos de Uso.** *Gutierrez, Demián*, 2011, 1. Jg., S. 45.

[HEROKU, 2021] Heroku. **What is Heroku. Disponível em: https://www.heroku.com/what .** Acesso em Fevereiro de 2021.

[INTELLIJ, 2021] IntelliJ Idea. **Disponível em: https://www.jetbrains.com/idea/ .** Acesso em Julho de 2021.

[JS, 2021] JavaScript. **Disponível em: https://www.javascript.com .** Acesso em Julho de 2021.

[MENDES, 2009] MENDES, Douglas Rocha. ***Programação Java com ênfase em Orientação a Objetos*.** Novatec Editora, 2009.

[MUI, 2021] Material UI. **The React component library you always wanted. Disponível em: https://mui.com/.** Acesso em Setembro de 2021.

[PSQL, 2021] PostgreSQL - The world's most advanced open source relational database. **Disponível em: https:/www.postgresql.org .** Acesso em Julho de 2021.

[PWR, 2021] Power Architect. **Disponível em: http://www.bestofbi.com/page/architect .** Acesso em Julho de 2021.

[REACT, 2021] React. **Disponível em: https://reactjs.org .** Acesso em Julho de 2021.

[RUMBAUGH, 1996] JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. **The Unified Modeling Language.** *University Video Communications*, 1996.

[SOUZA e ANDRADE, 2015] SOUZA, Carina Dias Sucupira de; ANDRADE, Domingas Silva. **Gerenciador de Projetos Finais - GeProFi.** *Departamento de Ciência da Computação*, 2015.

[SPRING, 2021] Spring - Why Spring?. **Disponível em: https://spring.io/why-spring .** Acesso em Julho de 2021.

[TEC, 2009] Tecmundo. O que é Java? **Disponível em: https://www.tecmundo.com.br/programacao/2710-o-que-e-java-.htm .** Acesso em Fevereiro de 2021.

APÊNDICE A – DICIONÁRIO DE DADOS

A modelagem de dados contém todas as informações sobre o banco (suas tabelas, campos, tipos de campo e se permitem começo de campo como valor nulo ou não).

User

Atributos	Descrição	Tipo	Tam	Null	PK
user_id	Identificador do usuário no sistema	Integer	11	Não	Sim
name	Nome do usuário no sistema	Varchar	45	Não	Não
email	E-mail do usuário no sistema	Varchar	45	Não	Não
password	Código secreto do usuário para entrar no sistema	Varchar	45	Não	Não
user_type	Atributo que define o tipo de acesso que o usuário terá no sistema diferenciando as permissões e informações de acordo com o seu tipo	Tinyint	4	Não	Não
dre	Número da matrícula do aluno no curso	Integer	11	Sim	Não
siape	Número da matrícula do professor	Integer	11	Sim	Não
register_date	Data que o usuário foi cadastrado no sistema	Timestamp		Não	Não
gender	Sexo do usuário	Varchar	45	Sim	Não
status	Situação do usuário no sistema	Tinyint	4	Sim	Não
title	Título do usuário	Varchar	255	Sim	Não
position	Cargo do usuário na instituição	Varchar	255	Sim	Não
room	Sala onde é possível encontrar o professor na instituição	Varchar	255	Sim	Não
lattes	Currículo Lattes do usuário	Varchar	255	Sim	Não
user_profile	Site pessoal do usuário	Varchar	255	Sim	Não
course	Curso do usuário	Varchar	255	Sim	Não
origin	Universidade de origem do professor externo	Varchar	100	Não	Não

FieldOfInterest

Atributos	Descrição	Tipo	Tam	Null	PK
field_id	Identificador da área de interesse no sistema	Integer	11	Não	Sim
name	Nome da área de interesse no sistema	Varchar	45	Não	Não
description	Descrição detalhada da área de interesse	Varchar	4000	Sim	Não

Project

Atributos	Descrição	Tipo	Tam	Null	PK
project_id	Identificador do projeto no sistema	Integer	11	Não	Sim
subject	Tema que será abordado no trabalho	Varchar	100	Não	Não
title	Título do projeto no sistema	Varchar	255	Sim	Não
description	Descrição sobre o projeto abordando os tópicos importantes de forma resumida	Varchar	4000	Sim	Não
status	Situação do projeto no sistema	Tinyint	4	Sim	Não
register_date	Data em que o projeto foi cadastrado no sistema	Timestamp		Não	Não
keywords	Palavras-chave que definem o projeto no sistema	Varchar	250	Sim	Não
file	Link para arquivo do projeto	Varchar	4000	Sim	Não
file_status	Situação do arquivo no projeto	Tinyint	4	Sim	Não

Project_has_User

Atributos	Descrição	Tipo	Tam	Null	PK
project_id	Identificador do projeto no sistema	Integer	11	Não	Sim
user_id	Identificador do usuário no sistema	Integer	11	Não	Sim
committee	Flag que indica se é membro da banca	Boolean		Não	Não

	ou não				
coop	Flag que indica se é co-orientador ou orientador	Boolean		Não	Não

FieldOfInterest_has_User

Atributos	Descrição	Tipo	Tam	Null	PK
project_id	Identificador do projeto no sistema	Integer	11	Não	Sim
user_id	Identificador do usuário no sistema	Integer	11	Não	Sim

Record_has_User

Atributos	Descrição	Tipo	Tam	Null	PK
record_id	Identificador do arquivo no sistema	Integer	11	Não	Sim
project_id	Identificador do projeto no sistema	Integer	11	Não	Sim
user_id	Identificador do usuário no sistema	Integer	11	Não	Sim
signature	Assinatura dos envolvidos	Varchar		Não	Não

Record

Atributos	Descrição	Tipo	Tam	Null	PK
record_id	Identificador da ata no sistema	Integer	11	Não	Sim
project_id	Identificador do projeto no sistema	Integer	45	Não	Não
thesis_id	Identificador da defesa no sistema	Integer	45	Sim	Não
thesis_date	Dia da defesa no sistema	Date	45	Não	Não
begin_time	Horário de início da defesa no sistema	Time	45	Não	Não
end_time	Horário de fim da defesa no sistema	Time	11	Sim	Não

location	Local da defesa no sistema	Varchar	255	Sim	Não
evaluation	Avaliação no sistema	Tinyint	4	Sim	Não
deadline	Prazo para satisfazer as exigências no sistema	Integer	11	Sim	Não
grade	Grau obtido no sistema	Decimal		Sim	Não
grade_description	Grau obtido no sistema por extenso	Varchar	30	Sim	Não

Register_Token

Atributos	Descrição	Tipo	Tam	Null	PK
register_token_id	Identificador do registro de token no sistema	Integer	11	Não	Sim
user_id	Identificador do usuário no sistema	Integer	11	Não	Sim
token	Token que é enviado no e-mail de registro na plataforma	Varchar	40	Sim	Não