AdminStepTCC: Sistema para Gerenciamento dos Processos que compõem os Trabalhos de Conclusão de Curso

Raylla Maria da Silva Araújo[1] , Michele Cristiani Barion[2]

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP
CEP 13183-250 – Hortolândia – SP – Brazil
rayllamariaaraujo@gmail.com[1]; michele_barion@hotmail.com[2]

Abstract. *Manual, repetitive and decentralized processes often lead to error and lack of process integrity. In order to avoid this, it is necessary to have a mechanism that centralizes and automates the processes, in order to guarantee their integrity. In view of this, this work proposes the development of a web software to manage Course Completion Works (TCC) of an educational institution. Through the system, instead of managing the stages of the TCCs being carried out through spreadsheets manually fed by the advisors, steps such as the search for an advisor, sending the proposal to the collegiate, development, qualification and defense would be managed through the system. This work was carried out following the steps of software development such as requirements gathering, planning, development, testing and validation. Based on this, the system aims to promote the automation of the management of Course Completion Assignments in Educational Institutions, by allowing the centralization of processes and facilitating their management.*

Resumo. *Processos manuais, repetitivos e descentralizados resultam, muitas vezes, em erros e falta de integridade dos processos. A fim de se evitar tal cenário, faz-se necessário um mecanismo que centralize e automatize os processos. Em vista disso, este trabalho propõe o desenvolvimento de um software web para gerenciar Trabalhos de Conclusão de Curso (TCC), tendo em vista um curso superior de uma instituição pública de ensino. Assim, em vez do gerenciamento das etapas dos TCCs serem realizadas através de planilhas alimentadas manualmente pelos orientadores, as etapas como a busca por orientador, envio da proposta ao colegiado, desenvolvimento, qualificação e defesa seriam gerenciados através da aplicação. Este trabalho foi realizado seguindo as etapas de desenvolvimento de software como levantamento de requisitos, planejamento, desenvolvimento, testes e validação. Como resultado, apresenta-se uma ferramenta que ficará disponível em um repositório para download e possível implantação para uso de qualquer Instituição de Ensino que possui a atividade de TCC no âmbito do curso.*

1. Introdução

Na literatura, no que diz respeito ao desenvolvimento de *software*, muito se menciona a importância de explorar o reuso de código, considerando as boas práticas do *SOLID*[1] [MARTIN 2019]. A importância do reuso e de seguir as boas práticas não se aplica apenas a

[1] *SOLID* é um acrônimo que representa cinco princípios de *design* de *software* que foram introduzidos por Robert C. Martin e são amplamente reconhecidos como boas práticas para escrever código limpo, manutenível e escalável. Os cinco princípios do *SOLID* são: o Princípio da Responsabilidade Única (*Single Responsibility Principle - SRP*), o Princípio do Aberto/Fechado (*Open/Closed Principle - OCP*), o Princípio da Substituição de Liskov (*Liskov Substitution Principle - LSP*), o Princípio da Segregação de Interfaces (*Interface Segregation Principle - ISP*) e o Princípio da Inversão de Dependência *(Dependency Inversion Principle - DIP*) [MARTIN 2019].

software, mas também a qualquer elemento que se queira gerenciar de forma organizada, com facilidade de entendimento e manutenção. Com isto em vista, o gerenciamento adequado e efetivo de qualquer objeto da realidade, mesmo que intangível, é de extrema importância para que se alcancem resultados esperados e satisfatórios.

Tendo como cenário o Instituto Federal de São Paulo (IFSP) - Campus Hortolândia, atualmente o gerenciamento de Trabalho de Conclusão de Curso (TCC) é realizado através de planilhas com várias abas que possuem dados replicados de um sistema terceiro, além de duplicação de dados entre abas da própria planilha. Logo, pode-se afirmar que em alguns momentos ao gerenciar o andamento dos TCCs ocorre retrabalho da parte dos orientadores. Além disso, há possibilidades de erros e perda da integridade dos dados, uma vez que todo controle é feito de forma manual. Segundo os autores [LAUDON e LAUDON 2016], utilizar banco de dados e sistemas da informação aumenta a disponibilidade e integridade dos dados, ao mesmo tempo que diminui a redundância e o custo de desenvolvimento e manutenção, sendo relevante até de forma estratégica em uma organização, uma vez que se consegue centralizar as informações e avaliá-las para tomada de decisões.

Diante do exposto propõe-se, então, o desenvolvimento de uma aplicação *web* que permita o monitoramento dos TCCs em todas as etapas dos seus trâmites, desde o envio da proposta ao Colegiado de Curso até a apresentação à banca e, assim, finalização das etapas do processo. O sistema proposto também visa possibilitar aos professores e coordenadores a consulta de alunos que estão sem orientador, além dos TCCs em andamento, assim como suas áreas de interesse para desenvolver futuras orientações. Além disso, busca a consulta por parte dos alunos quanto à relação de professores disponíveis para orientação diante de suas áreas de especialidade.

Para abordagem da proposta, além da presente introdução, este trabalho apresenta na seção 2 abordagem dos trabalhos correlatos, considerando suas características e a relação destes com a aplicação proposta. Na seção 3 é apresentado o referencial teórico, considerando um viés da pesquisa bibliográfica. Na seção 4 é apresentada a metodologia, além dos materiais utilizados. Na seção 5 apresenta todas as etapas do desenvolvimento da proposta, desde a sua análise até a implementação das funcionalidades e, para fechar, na seção 6 a abordagem dos resultados finais, além de sugestões futuras para complementar a proposta apresentada no decorrer das seções deste trabalho.

2. Trabalhos Correlatos

Foram analisados três trabalhos correlatos desenvolvidos especificamente para o IFSP - Campus Hortolândia, todavia cada um com sua função específica. Esses projetos foram concebidos com o propósito de automatizar processos que anteriormente eram realizados de forma manual. Os trabalhos incluem:

- PAE docs: Plataforma de Gerenciamento do Programa de Assistência Estudantil do IFSP Campus Hortolândia [ALVES 2019] - além de sua correlação de contexto, devido a sua aplicação ser proposta para ser utilizada no IFSP, há similaridade com o fato de haver várias etapas no trâmite do programa de assistência estudantil gerenciado no sistema. Apesar da tecnologia utilizada no desenvolvimento ser a linguagem *PHP* para Android, este projeto também se assemelha na utilização de *API Rest*.

- Implementação do Sistema Gerenciador de Estágio do IFSP Hortolândia baseado em microsserviços [SALVINO 2019] - a sua correlação se dá devido a utilização de tecnologias propostas para serem utilizadas no projeto como a linguagem de programação Java, *Framework back-end Spring* e *Framework Front-end Angular*, além da geração de relatórios.

- Portal de gerenciamento de inscrições para cursos de extensão oferecidos pelo campus Hortolândia [SCHIAROLLI 2021] - assemelha-se ao projeto pelo seu enfoque na automatização de trâmites e processos que eram realizados de forma manual no IFSP, centralizando-os em um sistema.

Pode-se observar na Tabela 1 a correlação das funcionalidades dos trabalhos correlatos com o sistema proposto, onde todos possuem a aplicação no mesmo contexto e enfoque em automatizar processos.

Tabela 1. Relação dos trabalhos correlatos e similaridades com o trabalho proposto

Projetos	**Contexto de aplicação acadêmico**	**Gerenciamento de Etapas**	**Sistema para automatizar processos**	**Geração de Relatórios**
Trabalho Proposto	X	X	X	X
PAE Docs	X	X	X	
Gerenciador Estágio	X		X	X
Gerenciamento de inscrições	X		X	

Já na Tabela 2 é feito um comparativo entre a aplicação proposta e os trabalhos correlatos quanto às tecnologias utilizadas, o qual pode se observar que todas as aplicações foram desenvolvidas em *APIs* no *back-end* para consultar a base de dados.

Tabela 2. Relação de tecnologias usadas nos trabalhos correlatos e o trabalho

Projetos	**JAVA**	***Framework Angular***	***Framework Spring***	***API***
Trabalho Proposto	X	X	X	X
PAE Docs				X
Gerenciador Estágio	X	X	X	X
Gerenciamento de inscrições				X

3. Referencial Teórico

Foi feito um estudo bibliográfico através de livros e *sites*, considerando, principalmente a escolha das ferramentas para os processos de análise e desenvolvimento, sendo estas apresentadas na seguinte ordem: artefatos para documentação das funcionalidades,

frameworks, *APIs*, linguagens para desenvolvimento *front-end* e *back-end*, além da definição do *dataset*.

3.1. Artefatos para documentação das funcionalidades

Requisitos de *software* são descrições das funcionalidades e características de uma aplicação que se pretende desenvolver. Entre os tipos de requisitos há os funcionais e não funcionais, em que os funcionais se referem às funcionalidades do sistema, ou seja, o que este deve fazer e, os não funcionais que envolvem aspectos como desempenho, segurança, usabilidade e escalabilidade do sistema. Os requisitos podem ser coletados em entrevistas e/ou questionários, por exemplo, com definição de perguntas abertas, fechadas ou mistas [PRESSMAN 2021].

A *Unified Modeling Language* (*UML*) é uma linguagem de modelação padrão que se baseia em diagramas para descrever e documentar especificidades do projeto de um sistemas de *software*, e é implementadas através de ferramentas como StarUML e o ArgoUML. Um dos diagramas *UML* usados na fase inicial de um projeto de *software* é o caso de uso que visa representar as especificações funcionais do sistema a partir da visão do usuário [BEZERRA 2016].

Quanto ao banco de dados pode ser citado o Diagrama Entidade Relacionamento (DER) que é um diagrama que apresenta a modelagem da estrutura lógica de uma base de dados relacional através de um alto nível de abstração. Para isso, faz-se uso representações gráficas chamadas de entidades, relacionamentos e atributos, sendo estas a representação de um objeto de existência no mundo real, as relações entre estes e as características destes, respectivamente [REZENDE 2005].

3.2. Linguagens, *frameworks front-end* e *back-end e APIs*

HTML é uma linguagem de marcação de hipertexto, que é a base para a criação de páginas *web* [SILVA 2008], já que os documentos deste tipo podem ser interpretados por navegadores.

Já o CSS é o acrônimo de *Cascading Style Sheets*, isto é, folhas de estilo em cascata. É uma especificação que define como os elementos que compõem uma página ou aplicação *web* serão exibidos, sendo responsável, assim, por configurar o *layout* das páginas *web* [SILVA 2008].

Neste conjunto de linguagens *web* pode ser citado o *JavaScript* (*JS*), sendo uma linguagem de programação estruturada e dinâmica para desenvolvimento de páginas interativas. Juntamente com o *HTML* e o *CSS*, o *JS* é uma das principais tecnologias *Word Wide Web* [FLANAGAN 2012].

Outra linguagem de código aberto e que foi construída através do *JavaScript* é o *TypeScript* (*TS*). Possui todos os recursos disponibilizados pelo *JS*, além de funcionalidades adicionais, tais como a tipagem estática e interfaces que tornam mais fácil a detecção e a prevenção de erros durante o processo de desenvolvimento. [MICROSOFT 2022].

Quanto a simplificação do ambiente de desenvolvimento, além de evitar funções e repetições de bibliotecas comuns em um projeto, busca-se o uso de *frameworks* que, conforme apresenta [MINETTO 20007], são implementações e funcionalidades genéricas, ou seja, que

já estão prontas e que servem de base para vários projetos de desenvolvimento distintos. Um exemplo que pode ser citado e que será usado na proposta é o *Angular* que é um framework *web* orientado a componentes, sendo muito utilizado para a criação de *Single-Page Applications* (*SPA*), isto é, para a criação de páginas *web* que ao receberem uma solicitação de atualização, por exemplo, não exige que toda a página seja atualizada, e, sim apenas o componente referente a requisição. O *framework* trabalha em conjunto com as linguagens *HTML*, *CSS* e *TypeScript* para criar aplicações *web front-end* [GOOGLE 2022].

Nas opções de linguagens de programação para *web* cita-se o Java que usa o paradigma da programação orientada a objetos e que ao longo do tempo possibilitou o desenvolvimento de aplicações por meio da plataforma *Java EE* (*Java Enterprise Edition*), consistindo em uma série de especificações que criam a estrutura para o desenvolvimento de aplicações voltadas à plataforma *web* [MELO 2007].

Outra solução a ser usada na proposta sendo importante para interligar plataformas e ferramentas, permitindo a comunicação e o trânsito de dados é a *API* (*Application Programming Interface* - Interface de Programação de Aplicação) que é um conjunto de rotinas e padrões de programação que possibilita o acesso a uma aplicação *web*. Utiliza-se uma *API*, geralmente, quando se quer promover um contrato de serviços entre duas aplicações que há uma arquitetura cliente-servidor que estabelecem uma relação de requisição e resposta. Enfatiza-se que o *endpoint* de uma *API* trata de um endereço utilizado para a comunicação de uma *API* e um serviço externo. Todo *endpoint* é associado a um método que define a sua função ao realizar a comunicação. Entre os métodos de um *endpoint* são encontrados: *GET*, *POST*, *PUT*, *PATCH* e *DELETE* [AMAZON 2022].

Um outro exemplo de *framework open source* muito usado para facilitar o desenvolvimento de aplicações *web*, como *APIs* em Java, é o *Spring.* Uma das suas vantagens é que não exige um servidor para que a aplicação execute, utilizando padrões de projeto e os conceitos de inversão de dependência e inversão de controle [SPRING 2022].

Cita-se, também, neste trabalho, as ferramentas *Swagger* e *Postman*. A primeira trata-se de uma ferramenta de código aberto que permite a documentação de *APIs* [SWAGGER 2022]. Já a segunda é uma plataforma de *API* para criar e consumir de forma simples, facilitando as etapas do desenvolvimento [POSTMAN 2022].

A Figura 1 apresenta as camadas da *API*. Considerando a ilustração e a sua aplicação através dessa proposta de trabalho, é importante explicar que no *Controller* são mapeados os *endpoints* da *API* que serão acessados pelo *front-end* para exibir as informações persistidas no banco de dados. Já na camada de *Model* são adicionadas as classes que serão transformadas em tabelas posteriormente na camada de *Repository*. Além da camada *Repository* que é feita a persistência dos modelos no banco de dados através do *JPA* e, por fim, no *Service* é feita a intermediação da camada de *Controller* e *Repository*, permitindo a segregação de responsabilidades na *API*.

Considerando a etapa de teste de *software*, apresenta-se o modelo de Ponta a Ponta (*End-to-End*) que é uma forma de teste de *software* que visa verificar o funcionamento de um sistema ou aplicativo como um todo, do início ao fim, simulando cenários de uso real. Isso inclui testar a integração entre diferentes componentes do sistema, a interação com interfaces de usuário e a validação dos fluxos de trabalho completos [HUMBLE 2013].

Quanto ao banco de dados, a proposta aplica os conceitos do modelo relacional usando como Sistema Gerenciador de Banco de Dados (SGBD) o MySQL, sendo um banco de dados de código aberto e estando entre os mais utilizados para fins comerciais [ALVES 2009 e ORACLE 2019]. No entanto, as operações no *dataset* não se darão por instruções *SQL* e sim através do *Spring* com a *JPA* (*Java Persistence API*), conforme será apresentado na seção de desenvolvimento.

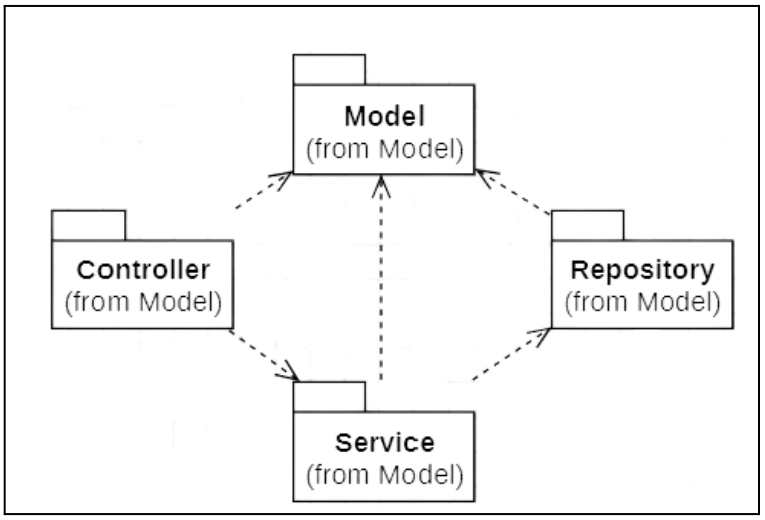

Figura 1. Desenho das camadas da API

4. Metodologia e materiais

O trabalho seguiu as etapas de desenvolvimento de um *software*, conforme define a literatura [PRESSMAN 2021] e seguiu a metodologia incremental, sendo essas apresentadas na seção 5 que irá tratar os processos de implementação.

Como apresenta a Figura 2, a aplicação do modelo incremental seguiu as cinco etapas de cada incremento, sendo a comunicação, o planejamento, a modelagem, a construção e a implantação. Enfatiza-se que esse modelo tem como base a reunião inicial com os envolvidos para definir o escopo dos objetivos do projeto, as implementações considerando diferentes versões das etapas, até que as funcionalidades estejam adequadas com a proposta. As etapas são seguidas sequencialmente e os incrementos são entregues com a finalização de cada etapa.

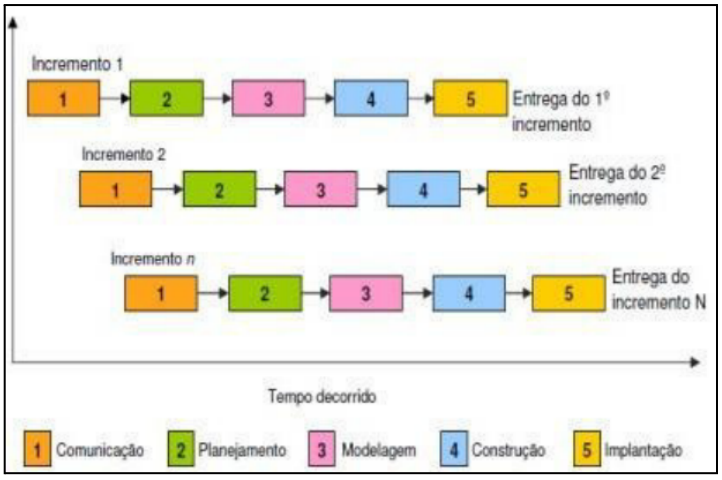

Figura 2. Etapas do Modelo Incremental

Inicialmente foi realizado um estudo de trabalhos correlatos, analisando a viabilidade da execução do projeto, principalmente no que tange ao uso das ferramentas escolhidas (conforme exposto na seção 2). Em seguida foi feito um levantamento de requisitos com os *stakeholders* através de entrevistas. Também é importante enfatizar que a pesquisa bibliográfica por meio de livros e *sites* técnicos foram fundamentais para o embasamento teórico, além do conhecimento para codificação das funcionalidades (abordados na seção 3).

Uma vez definido o escopo do projeto e suas funcionalidades, foi feita a implementação da aplicação *web* através das etapas e das tecnologias adotadas:

- Diagrama de Casos de Uso para modelagem das funcionalidades da aplicação;
- Diagrama de Classe para a modelagem das principais classes da regra de negócio;
- Diagrama Entidade-Relacionamento para visão estrutural do banco de dados;
- *Framework Angular front-end* para a criação das telas;
- A linguagem de programação *JAVA* e o *framework Spring back-end* para a construção da *API*;
- O Sistema de Gerenciamento de Banco de Dados (SGBD) MySQL, para a persistência dos dados da aplicação.

5. Desenvolvimento

Esta seção retrata as fases do desenvolvimento do projeto como um todo, sendo elas: levantamento e documentação dos requisitos, modelo de dados do sistema e implementação da proposta. Considera-se para essa etapa seis incrementos, como são descritos a seguir:

- Incremento 1/6: estudo de viabilidade do desenvolvimento da aplicação, através do estudo dos trabalhos correlatos e da realização de entrevistas para entender melhor as necessidades do problema a ser solucionado com o sistema. Após essa coleta e abstração foi desenvolvido o Diagrama de Casos de Uso com as funcionalidades do sistema.
- Incremento 2/6: configuração dos projetos *front-end* e *back-end,* com seus respectivos *frameworks* e iniciado o desenvolvimento das principais entidades e *endpoints*: Aluno, Professor e Proposta. Além disso, foram criados os componentes login e cadastro de aluno e professor, além da integração com a *API* e com as telas desenvolvidas.
- Incremento 3/6: criação dos elementos de tela das etapas da proposta e implementado no *front-end* o cadastro de proposta integrado com a *API*. Também foram implementadas as telas de listagem de alunos e professores.
- Incremento 4/6: criação das telas para apresentação das listagens de propostas e adicionado paginação. Além disso, foi implementada a geração de relatórios para alunos, professores e propostas na *API* e no *front-end*.
- Incremento 5/6: implementação da consulta de aluno, professor e proposta no front-end. Além disso, criação das entidades Curso e Áreas, adicionando os *endpoints* na *API* e as listagens, visualização e cadastro de cursos e áreas nas telas.
- Incremento 6/6: separação das visualizações dos perfis de acesso, adicionado logo de instituição de ensino nos cadastros dos cursos e adicionado gráficos da relação quantidade de alunos x status propostas de TCC nos relatórios de alunos e propostas, criando relações separadas por cursos. Além disso foi gerado o Diagrama de Classes separado por camadas e o Diagrama Entidade-Relacionamento.

É importante ressaltar que cada incremento teve uma duração média de quatro semanas, considerando o cronograma do projeto.

5.1. Levantamento de requisitos

Inicialmente foi realizada uma entrevista[2] com 14 perguntas abertas com a coordenadora do Curso de Análise e Desenvolvimento de Sistemas do IFSP - Campus Hortolândia para levantamento das informações associadas às regras de negócios referentes aos trâmites do Trabalho de Conclusão de Curso no Campus. A base para a entrevista foi a planilha utilizada para o gerenciamento de TCCs, o qual todas as etapas são preenchidas e gerenciadas manualmente pela coordenação, além dos professores-orientadores. Foram dois encontros, sendo esses primordiais para esclarecer várias questões quanto aos campos relevantes para o gerenciamento dos trabalhos e as suas etapas de desenvolvimento. Também é importante destacar que as etapas do processo de TCC desse curso se dá por meio das normas definidas em documento próprio[3].

Pontos relevantes levantados na entrevista após a abstração das informações coletadas:

- Criação de uma sessão para que os alunos possam registrar suas áreas de interesse para realização do TCC;
- Criação de uma sessão para que os professores possam registrar suas especialidades;
- Criação de algoritmo que permita o "*match*"[4] de aluno e professor para sugerir a possível combinação de orientando e orientador com base nas áreas de interesse de ambos e o semestre atual do aluno;
- Tendo uma relação atual de alunos, ter opções de relatórios para apresentar alunos com o TCC em andamento e os que estão com pendências, tais como não encaminharam a proposta ao Colegiado, não qualificaram e/ou defenderam;
- Disponibilização de agenda pública com as bancas de defesa de TCCs marcadas.

5.1.1. Requisitos Não-Funcionais

Para a presente versão do sistema proposto, foram levantados os seguintes requisitos não-funcionais, descritos na Tabela 3.

---

[2] As perguntas criadas e usadas para a entrevista podem ser consultadas através do link: https://drive.google.com/file/d/1rei-QPqc61VyKMG1CK4DuUPZTuN1X5Cb/view?usp=sharing.

[3] O documento pode ser consultado através do link: https://hto.ifsp.edu.br/portal/images/IFSP/Cursos/Coord_ADS/TCC/Normas_Procedimentos_TCC.pdf.

[4] *match* é um termo em inglês que, nesse contexto utilizado, refere-se à ação de combinar ou corresponder duas ou mais coisas que são semelhantes ou compatíveis. Disponível em: https://dictionary.cambridge.org/dictionary/english-portuguese/match.

Tabela 3. Requisitos Não-Funcionais

RNF	Descrição
RNF-1	O sistema foi desenvolvido utilizando padrão de *design* de *APIs RESTful* para facilitar a integração. Este é um requisito de interoperabilidade.
RNF-2	O sistema segue o padrão de arquitetura *Spring MVC*, separando as responsabilidades em camadas *Model, Controller. Service* e *Repository*. Este é um requisito de arquitetura de *software*.
RNF-3	O sistema possui documentação da *API* em *Swagger*. Este é um requisito de documentação.
RNF-4	O sistema é de fácil manutenção, uma vez que a arquitetura segue as boas práticas do *clean code*[5] . Este é um requisito de manutenibilidade.
RNF-5	A aplicação rodará em ambiente *web*, tendo todas as páginas na linguagem em português. Este é um requisito de usabilidade.
RNF-6	O sistema responde uma resposta ao cliente em menos de 3s. Este é um requisito de desempenho.
RNF-7	Os serviços de servidor serão executados através do sistema operacional Windows. Este é um requisito de compatibilidade.
RNF-8	O servidor de banco de dados e o sistema ficarão instalados em uma máquina de acesso para apenas um usuário. Será implantado um servidor local. Este é um requisito de disponibilidade.

5.1.2. Requisitos Funcionais

A Tabela 4 demonstra os requisitos funcionais, considerando o cenário do usuário e a primeira versão desta aplicação.

Tabela 4. Requisitos Funcionais

RF	Descrição
RF-1	O sistema deve ser capaz de realizar cadastro de dois usuários: aluno e professor. Para isso serão solicitados dados como nome, email e telefone, além das áreas de interesse dos usuários para facilitar uma futura busca por orientador/orientando.
RF-2	O sistema deve possuir 3 níveis de acesso: Aluno, Professor e Coordenador.
RF-3	No perfil do aluno deve ser possível visualizar informações dos professores, assim como suas áreas de interesses, e visualizar dados da sua proposta de TCC.
RF-4	No perfil do Professor o sistema deve permitir visualizar os alunos cadastrados no sistema, assim como suas áreas de interesse, visualizar as propostas dos alunos com TCC em andamento e editar suas informações. Além disso, deve ser possível visualizar todos os professores.

[5] *Clean Code* ou código limpo refere-se à prática de escrever código de fácil entendimento e manutenção. O *Clean Code* é pautado em princípios para escrever código limpo e de alta qualidade como legibilidade, manutenibilidade e divisão de responsabilidades [MARTIN 2009].

RF-5	Já no perfil do coordenador o sistema deve possibilitar todo o acesso do perfil de professor, além do cadastro, edição, consulta e listagem de cursos e áreas de interesse.
RF-6	O sistema deve permitir a geração de relatórios de propostas e alunos por cursos.
RF-7	O sistema deve permitir a busca de propostas, alunos e professores por filtros como status da proposta, curso e área de interesse, respectivamente.

5.2. Diagrama de Casos de Uso

Considerando a Tabela 4 que traz resumidamente a descrição das funcionalidades e as características do *software* proposto, a Figura 3 apresenta o diagrama de caso de uso diante dos diferentes usuários e as suas respectivas funcionalidades.

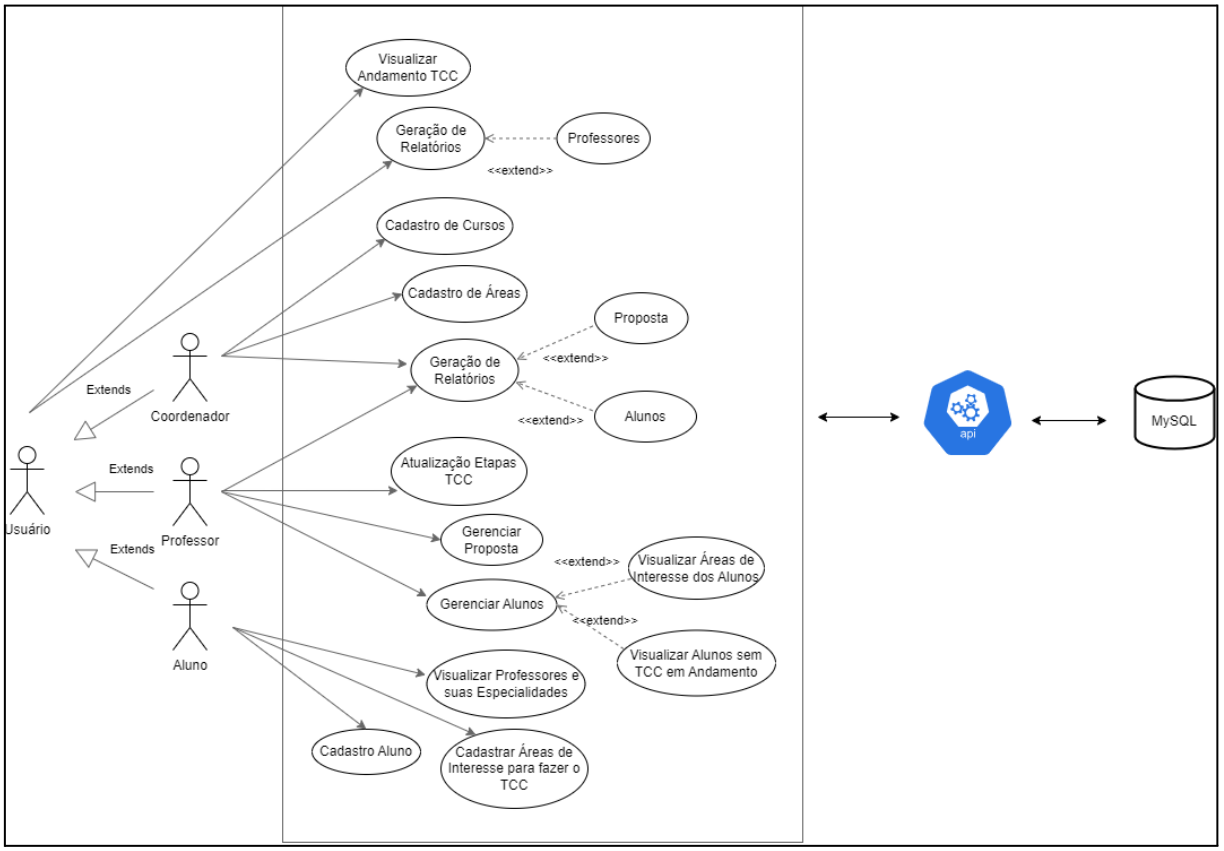

Figura 3. Diagrama de Caso de Uso

5.3. Desenvolvimento *APIs*

A partir das funcionalidades levantadas na entrevista foi feito o desenvolvimento da estrutura do projeto de *API* utilizando a linguagem JAVA e o *framework Spring*. O projeto foi estruturado com as camadas *Controller*, *Service*, *Repository* e *Model*, como foi explicado na Figura 1.

Assim, foi realizado o desenvolvimento referente às classes associadas ao Aluno, Professor, Proposta, Área e Curso com os *endpoints* referentes a consulta (*GET*), listagem (*GET*), inserção (*POST*) e alteração (*PUT/PATCH*). Além disso, também foi feita a

configuração do banco de dados MySQL através do *framework Spring*, além do teste manual dos *endpoints* mapeados na sua requisição no *Postman* à sua persistência no banco de dados.

Na Figura 4 é apresentada a visão macro dos diagramas de classe separadas pelas camadas *Model*, *Controller*, *Service* e *Repository*. A relação das camadas se dá na seguinte ordem: *Controller*, *Service* e *Repository*, sendo a camada *Model* acionada pelas três camadas. As figuras numeradas de 5 até 8 detalham separadamente as camadas da Figura 4.

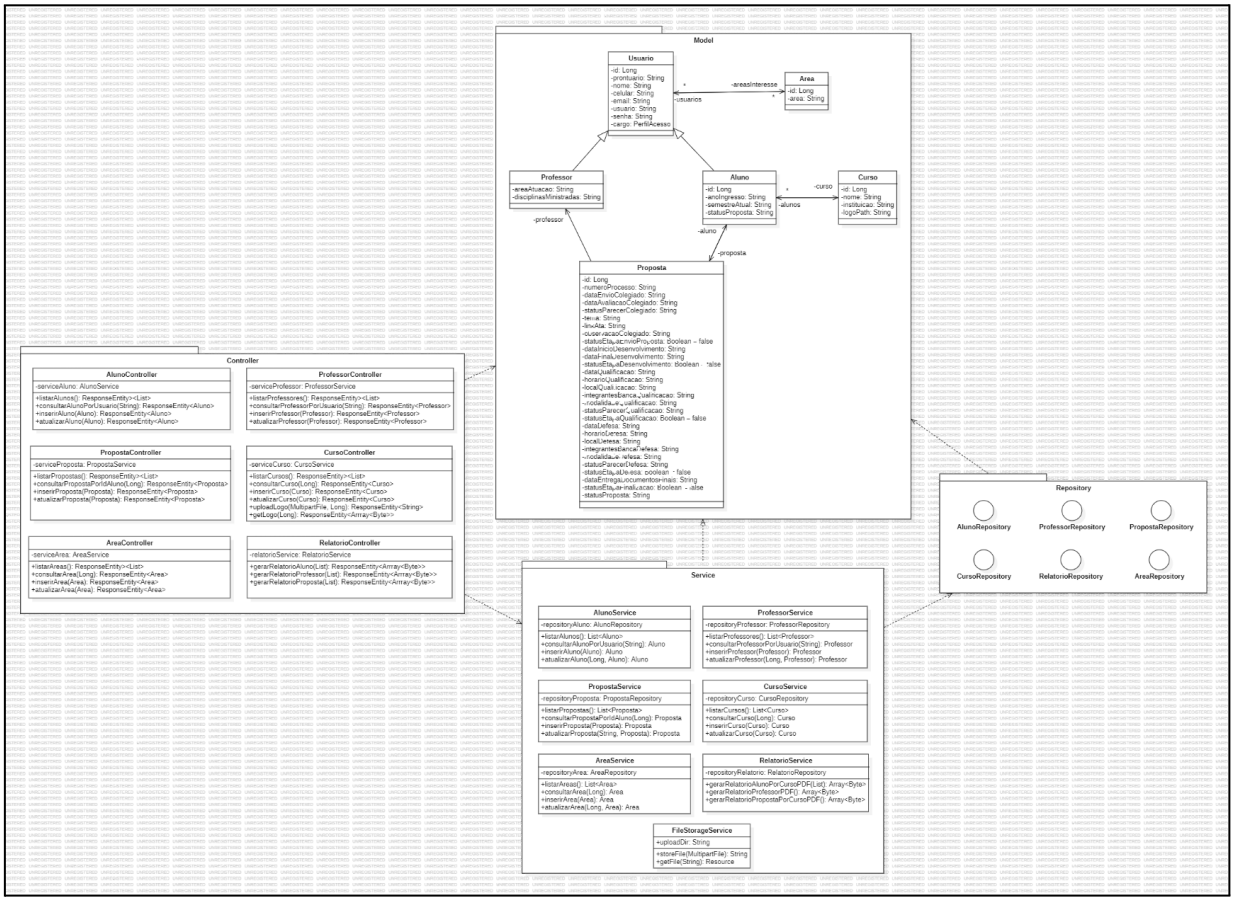

Figura 4. Diagrama de classe separado por camadas

A Figura 5 apresenta a camada *Model* com o mapeamento das classes que se transformam em entidades pelo *JPA*, criando, assim, a base de dados. Além disso, essas classes são usadas para o mapeamento dos objetos de cada entidade durante as requisições.

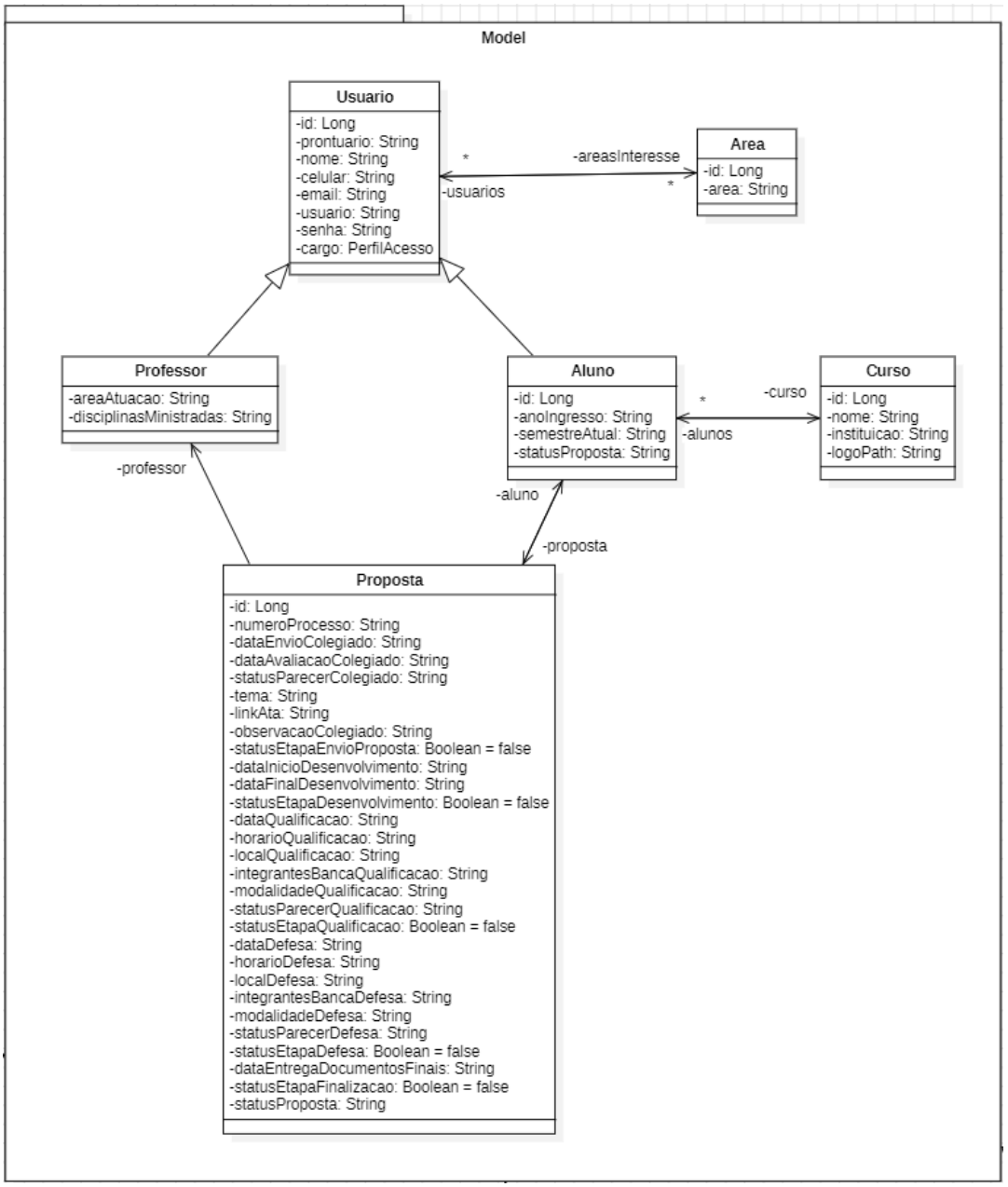

Figura 5. Diagrama da camada model

Já a Figura 6 aborda a *Controller*, que trata da camada de *view*, ou seja, a camada que interage com o *front-end.* Na *Controller* são mapeados os *endpoints* com os métodos *HTTP* como *GET*, *POST* e *PUT* permitindo, assim, com que o *front-end* consulte e altere os dados do *database.*

Quanto às regras de negócio e o processamento dos dados, a Figura 7 traz a camada de *Service*, sendo essa utilizada para isolar a regra de negócio da *Controller* e manter o código reutilizável.

E por fim, é abordada a camada de *Repository*, possuindo as interfaces que implementam o *JPA* e que permite a inclusão e alteração dos dados na base, como mostra a Figura 8.

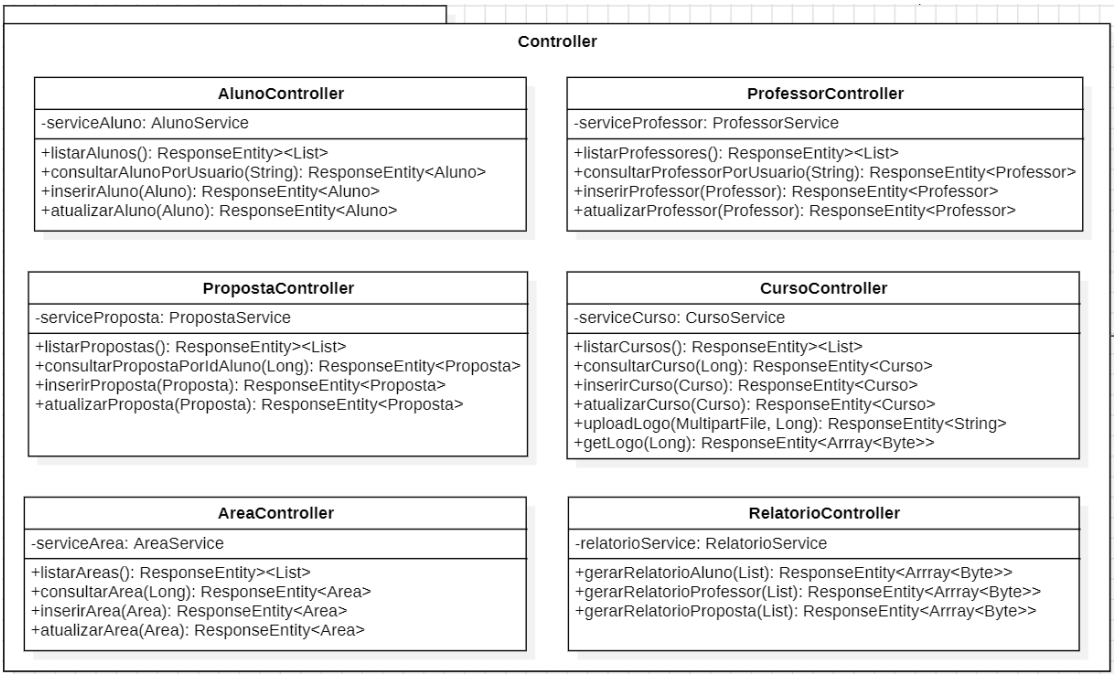

Figura 6. Diagrama da camada controller

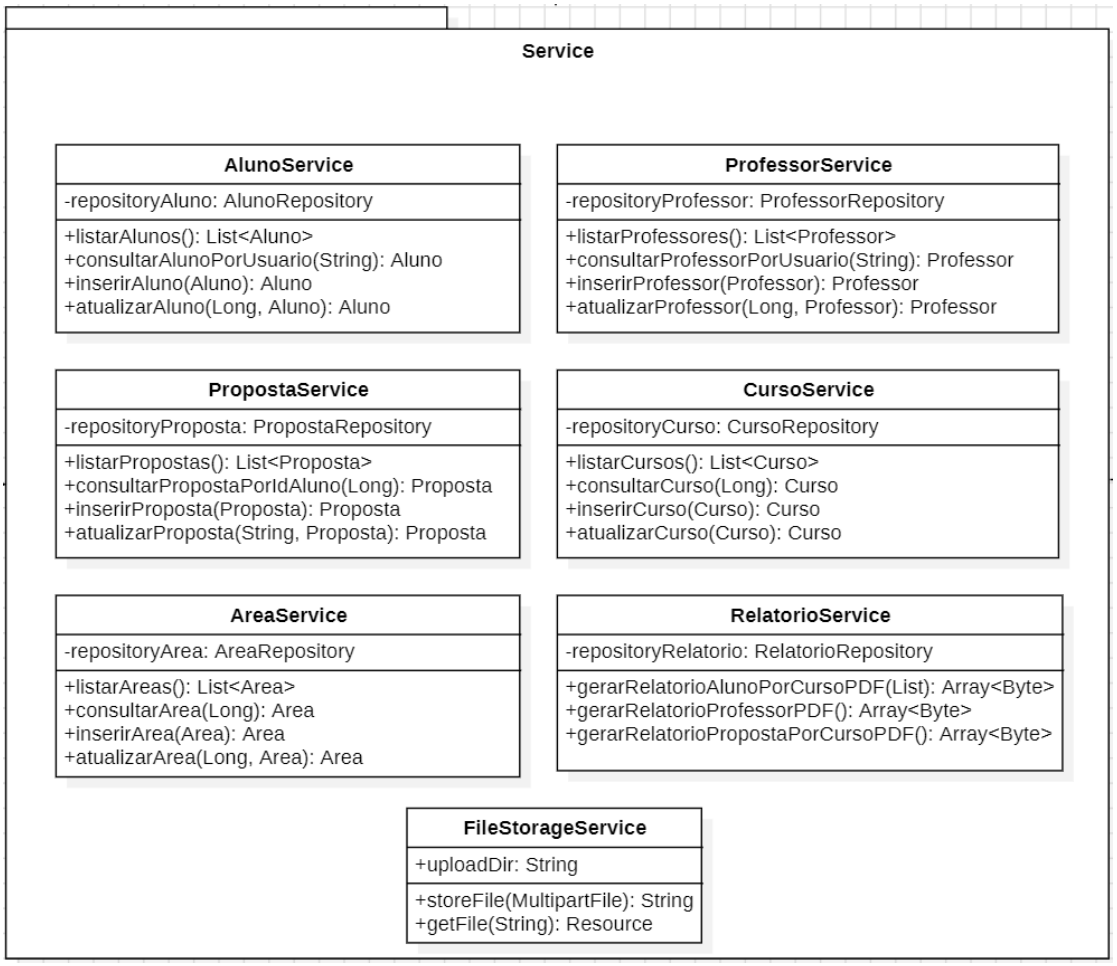

Figura 7. Diagrama da camada service

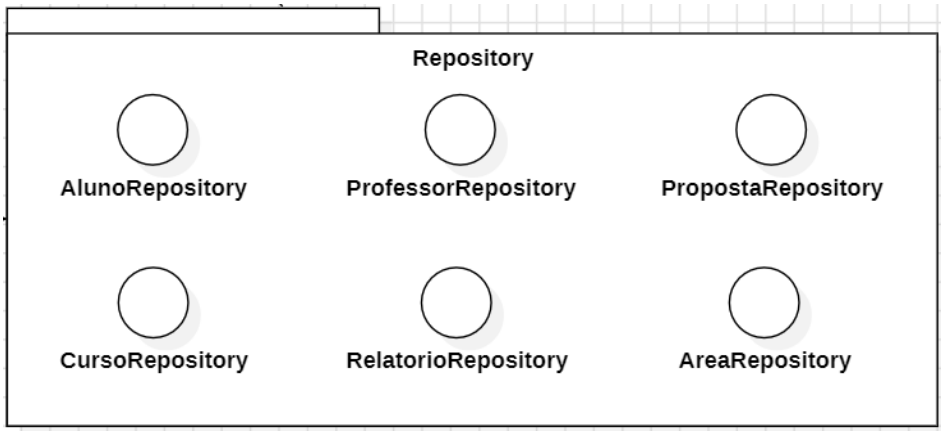

Figura 8. Representação da camada repository

A fim de documentar a *API* e facilitar seu entendimento para qualquer um que possa acessá-la, foi utilizado o *Swagger* que possibilita a visualização dos *endpoints*, *headers*, *body*, *parameters* e os *responses* de cada rota mapeada na *API*, como mostra nas figuras abaixo.

Na Figura 9 é exibido os *endpoints* mapeados na camada de *Controller* da entidade aluno, que serão requisitados pelo *front-end* para cadastrar, listar, consultar e atualizar os alunos.

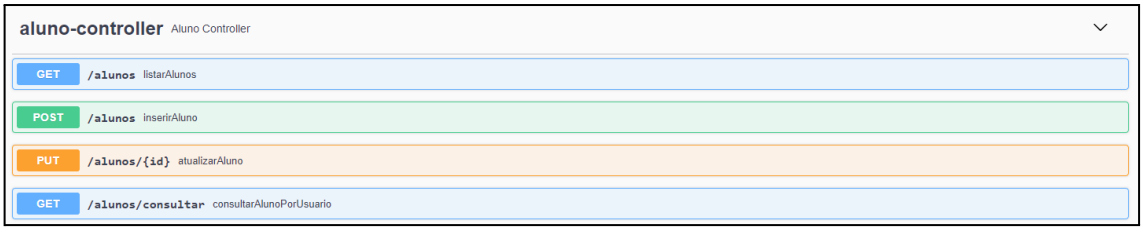

Figura 9. Documentação dos *endpoints* da entidade Aluno

A partir da Figura 10 observa-se que os *endpoints* mapeados possuem as mesmas ações da entidade aluno, contudo agora no perfil de professor.

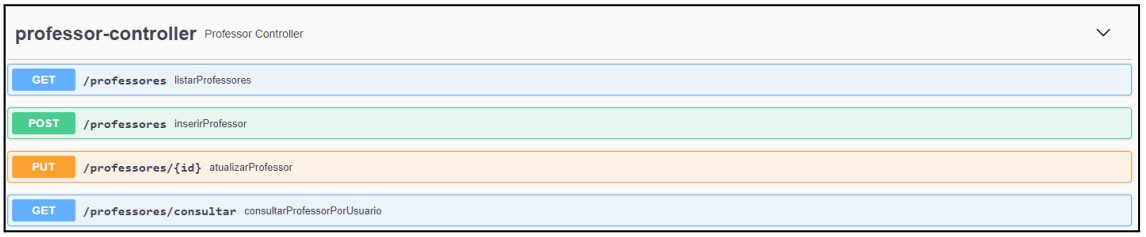

Figura 10. Documentação dos *endpoints* da entidade Professor

Já a Figura 11 exibe os *endpoints* mapeados na camada de *Controller* da entidade proposta. A requisição desses *endpoints* será feita pelo *front-end* para cadastrar, listar, consultar e alterar as propostas de TCC dos alunos.

Figura 11. Documentação dos *endpoints* da entidade Proposta

Quanto à geração de relatórios, a Figura 12 mostra os *endpoints* referentes às listagens de alunos, professores e propostas.

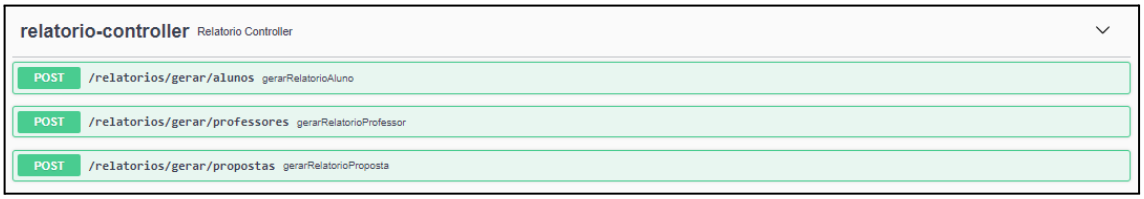

Figura 12. Documentação dos *endpoints* de Relatório

A Figura 13 aborda os *endpoints* referentes a listagem, consulta, atualização e ao cadastro dos cursos, assim como, o cadastro e consulta do logotipo da instituição de ensino do curso.

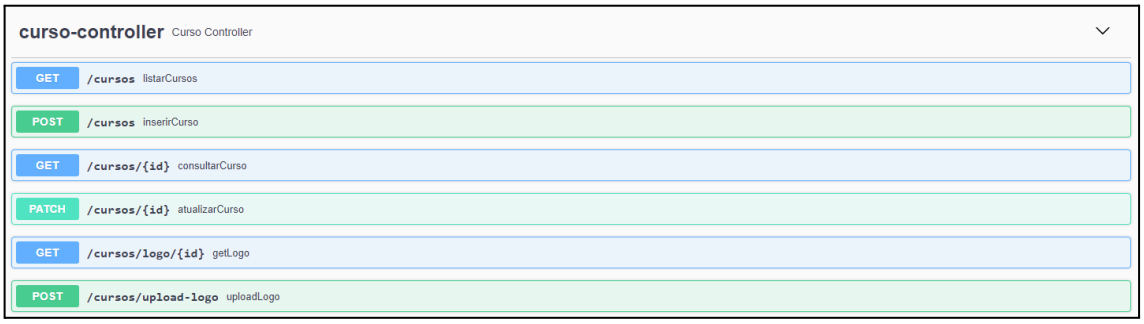

Figura 13. Documentação dos *endpoints* da Entidade Curso

E, por fim, a Figura 14 traz os *endpoints* referentes a listagem, consulta e atualização quanto ao cadastro das áreas.

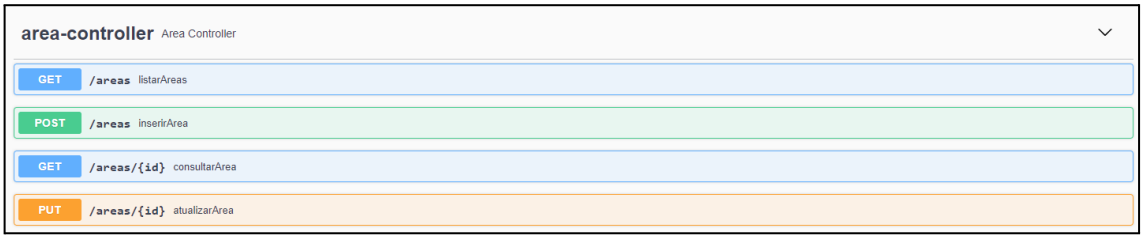

Figura 14. Documentação dos *endpoints* da Entidade Area

5.4. Abordagem do desenvolvimento do *front-end*

Conforme ferramentas especificadas nas seções anteriores deste trabalho, o *front-end* foi desenvolvido através do *framework Angular* e da linguagem *Typescript*, possuindo 3 níveis de acesso: Aluno, Professor e Coordenador (RF-2). Seguindo os requisitos abordados no diagrama de caso de uso através da Figura 3, são definidas as seguintes funcionalidades para cada perfil:

- No perfil do aluno é possível visualizar a proposta de TCC, caso já tenha sido cadastrada, e a listagem de professores, assim como suas áreas de interesses, podendo filtrar por área de interesse e gerar relatório destes.
- Já no perfil do professor é possível visualizar os alunos e as as propostas de TCC em andamento e editar suas informações. Além disso, deve ser possível visualizar todos os professores.
- O perfil do coordenador possui todos os acessos do perfil do professor, além do cadastro de cursos e áreas de interesse.

Enfatiza-se que possíveis dados apresentados a partir das telas são meramente valores registrados para os testes diante dos incrementos.

A Figura 15 apresenta a tela de login com os campos usuário e senha. Caso ainda não tenha cadastro, é necessário clicar no *link* “cadastre-se aqui” para realizar a inclusão de um novo usuário. Na lateral direita antes de iniciar o login é exibida uma listagem pública das defesas agendadas a partir da data atual.

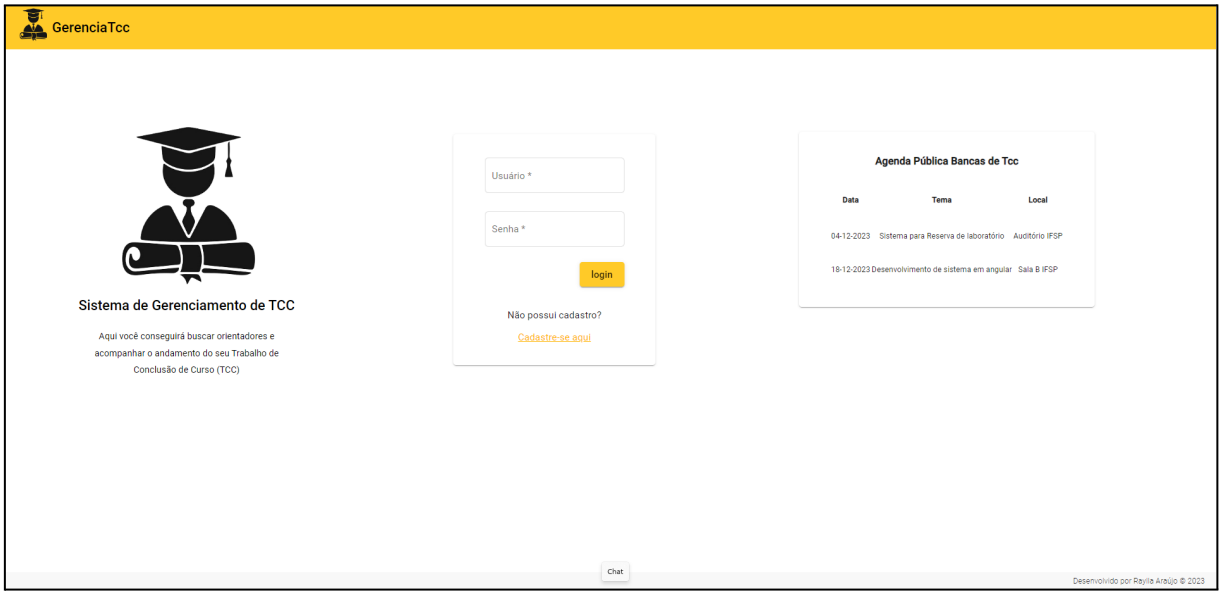

Figura 15. Tela de Login

Considerando o processo de inclusão de novos usuários, a Figura 16 ilustra o cadastro de um novo aluno (RF-1). São solicitadas informações pessoais, como nome, prontuário e dados de usuário, além do seu curso e áreas de interesse.

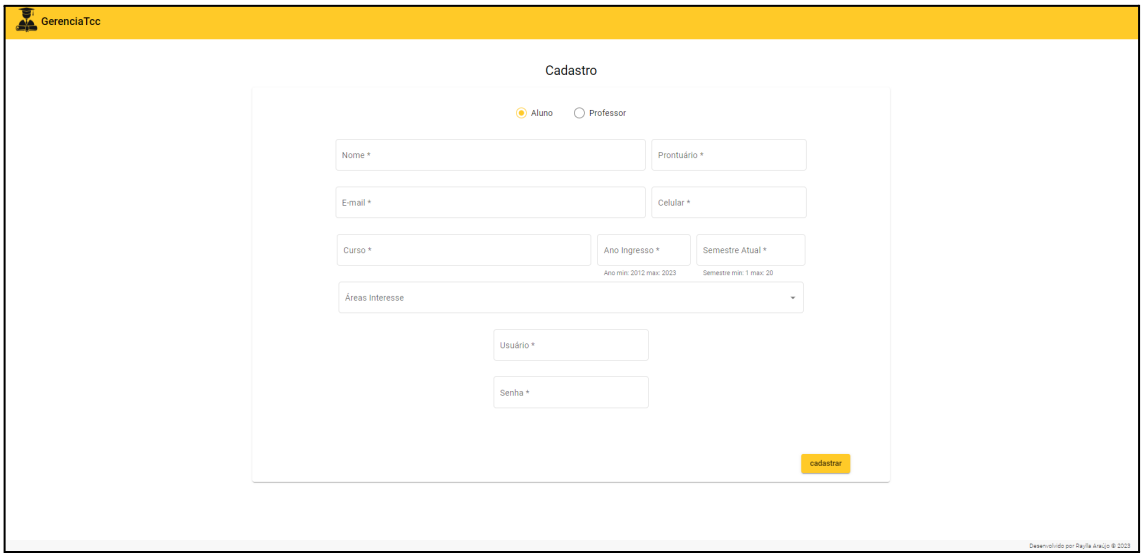

Figura 16. Tela de Cadastro de Aluno

Já a Figura 17 traz a tela para cadastro de usuário no perfil "professor" (RF-1). Nessa tela, além de informações pessoais, traz como diferencial o cadastro das áreas de interesse.

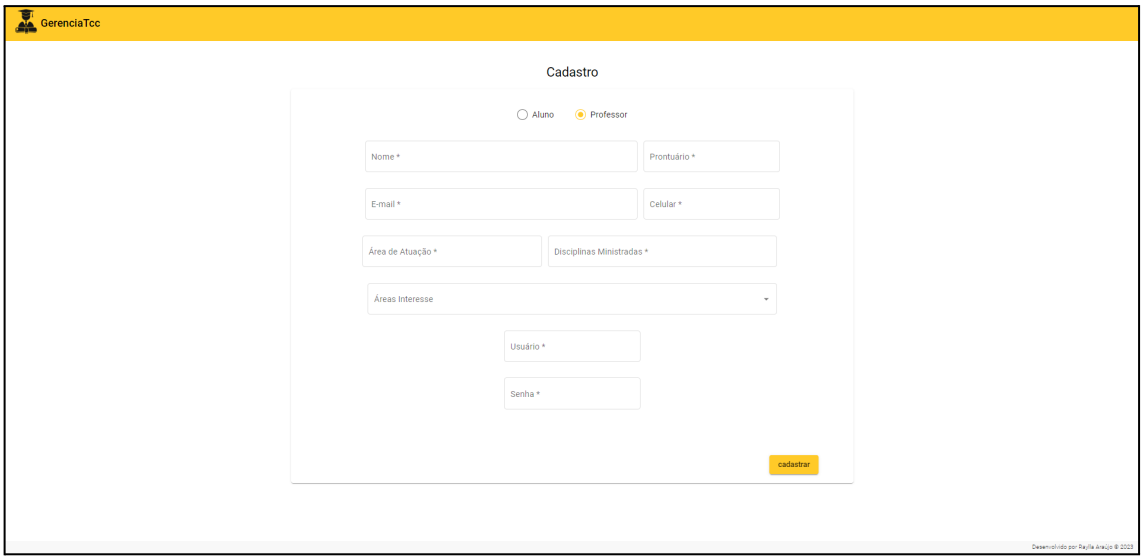

Figura 17. Tela de Cadastro de Professor

Considerando a funcionalidade associada ao envio da proposta, a Figura 18 traz esta visualização possuindo campos preenchidos pelo orientador referentes à proposta do aluno, como o tema, data de envio, parecer do Colegiado de Curso e *link* da ata (RF-3). As visões dos perfis aluno, professor e coordenador se diferenciam quanto ao acesso de alteração dos dados da proposta que só podem ser feitas pelo orientador da proposta.

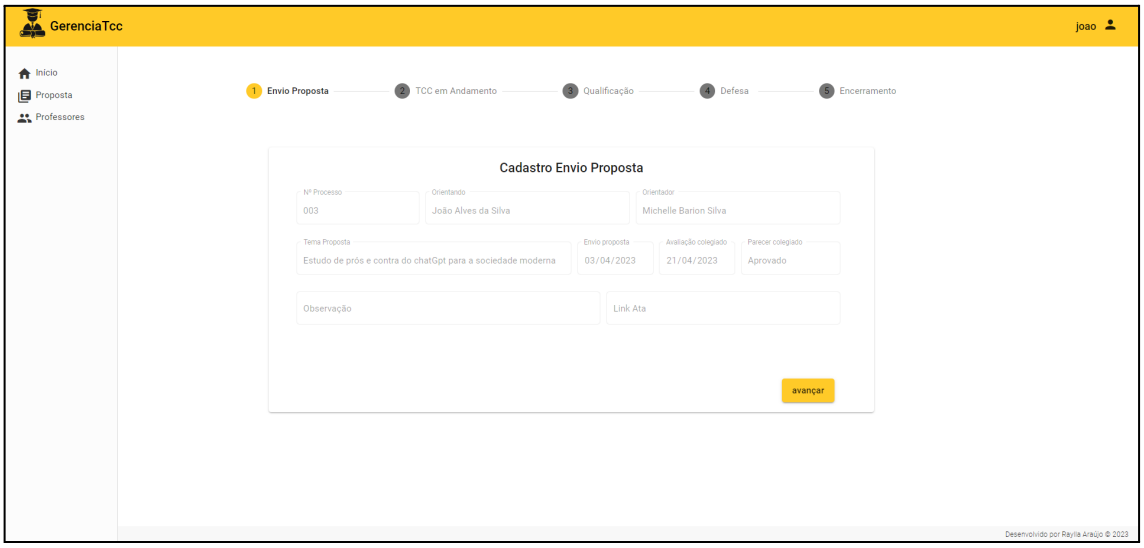

Figura 18. Tela etapa Envio de Proposta ao Colegiado no perfil aluno

O aluno também pode visualizar por meio do seu acesso às etapas do andamento do TCC, como é demonstrado na Figura 19 (RF-3).

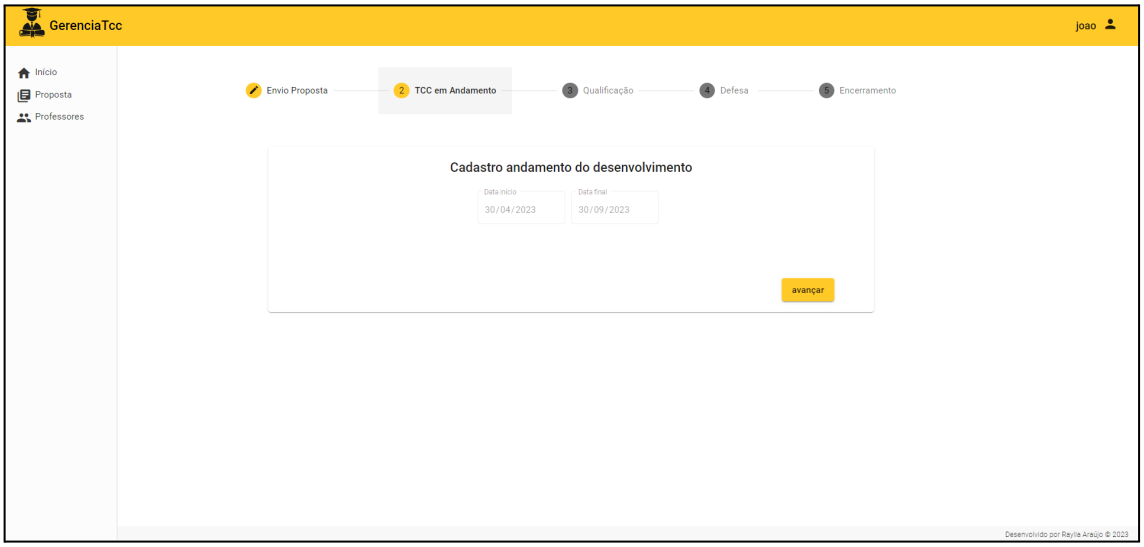

Figura 19. Tela etapa TCC em andamento no perfil aluno

Outras informações da etapa de qualificação, o aluno visualiza no seu perfil, considerando os campos de modalidade, local da qualificação, data, horário, banca e status do parecer, como é apresentada a tela a partir da Figura 20 (RF-3).

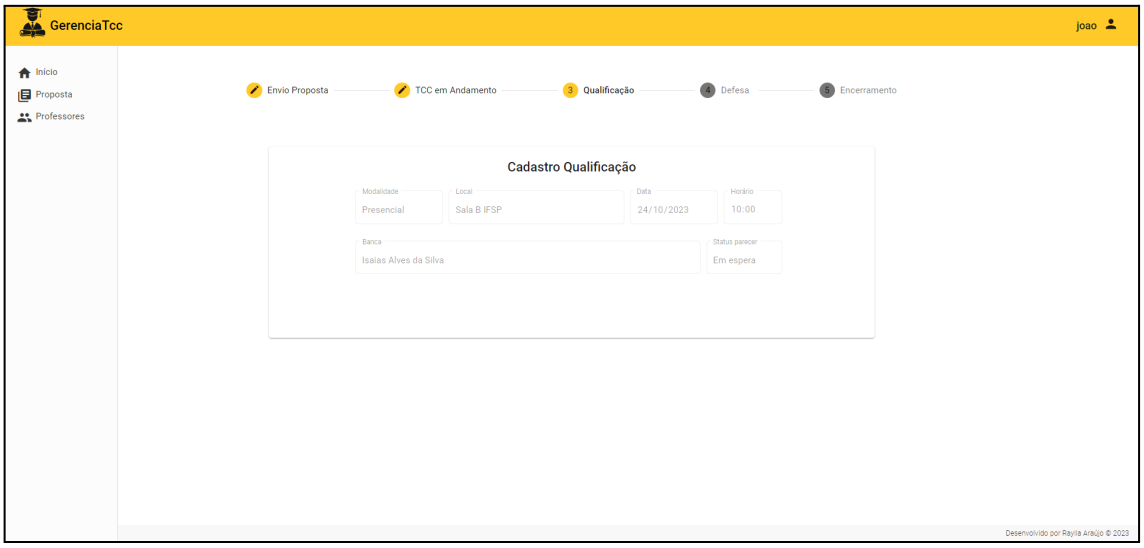

Figura 20. Tela etapa qualificação no perfil aluno

Após a qualificação, o aluno visualiza as informações associadas à próxima etapa, ou seja, a defesa. Na Figura 21 é apresentada a tela para visualização da etapa de defesa, considerando os campos modalidade, local da defesa, data, horário, banca e status do parecer (RF-3).

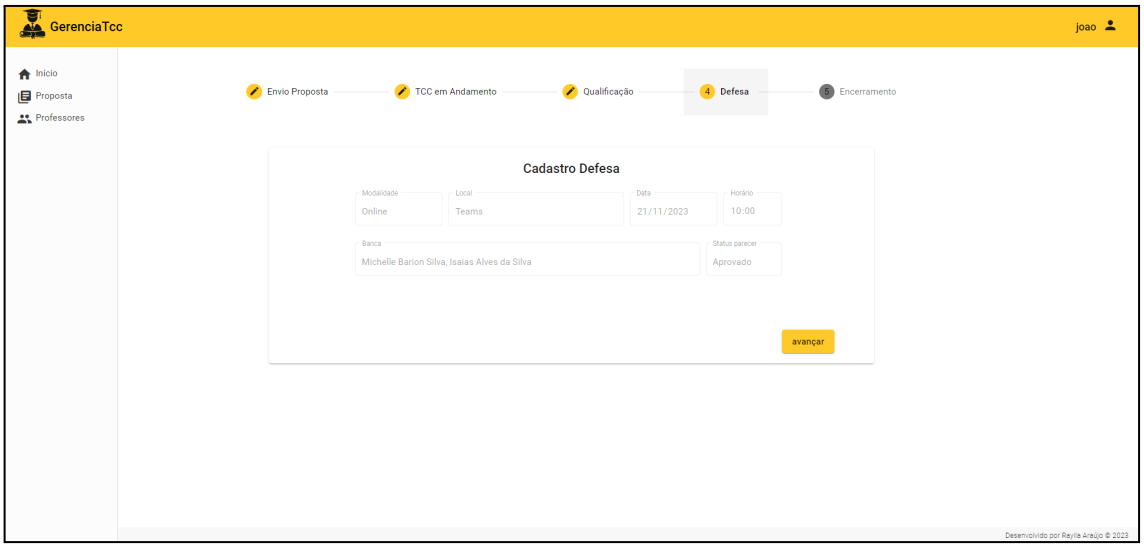

Figura 21. Tela etapa defesa no perfil aluno

Já a Figura 22 ilustra a etapa de encerramento com apresentação dos campos para entrega dos documentos finais e, assim, com a conclusão dos trâmites do TCC (RF-3).

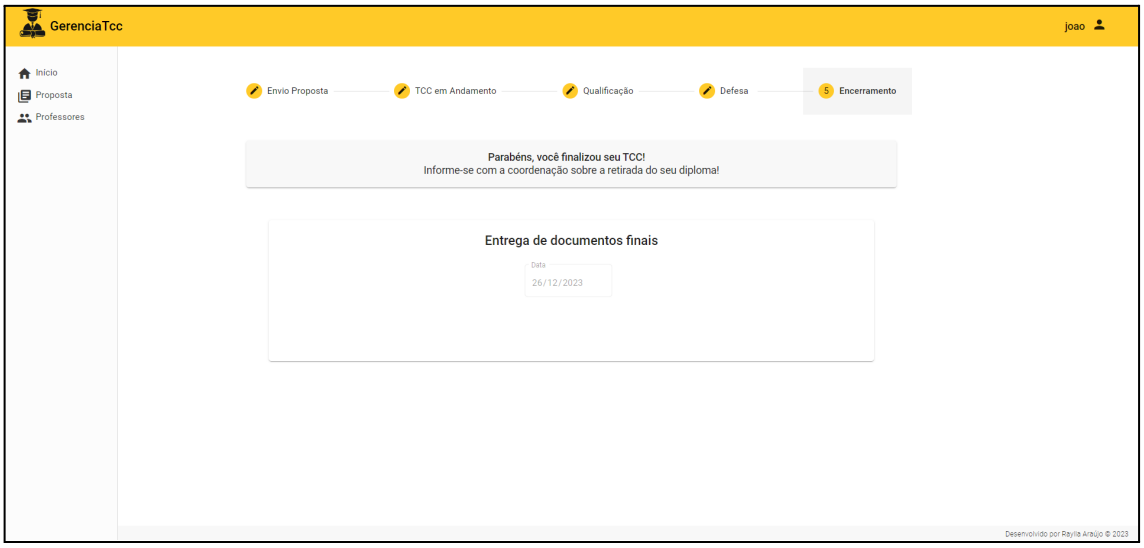

Figura 22. Tela etapa encerramento no perfil aluno

Ainda na abordagem do perfil aluno, a Figura 23 traz a funcionalidade "listagem de professores" com uma pré-visualização de informações como nome, e-mail e áreas de interesses. Através da lupa de cada registro é possível visualizar mais informações do professor (RF-3). Nesta tela é possível também filtrar os professores pelos valores dos campos da tabela (RF-7) e gerar uma lista do tipo pdf (RF-6).

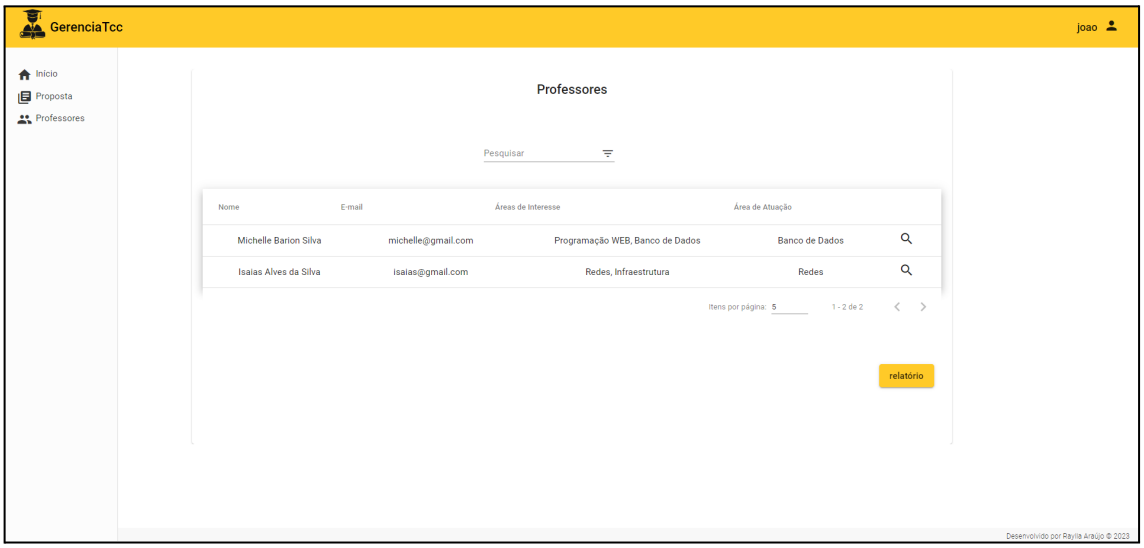

Figura 23. Tela de listagem de professores no perfil aluno

Quanto ao login no perfil professor, a Figura 24 apresenta a funcionalidade associada a listagem de propostas, sendo exibidos os dados previamente da proposta, como número do processo, tema, professor orientador, curso e status da mesma (RF-4). A partir da tabela é possível visualizar a proposta clicando na lupa no registro da proposta. Como mencionado anteriormente, a visualização é semelhante a da visão do aluno. Também é possível nessa tela filtrar pelos valores dos campos da tabela (RF-7) e gerar relatório em pdf com as propostas filtradas a partir do botão "relatório" (RF-6). No botão cadastrar é possível inserir um novo registro de proposta preenchendo os campos apresentados na visualização da visão do aluno (Figura 18).

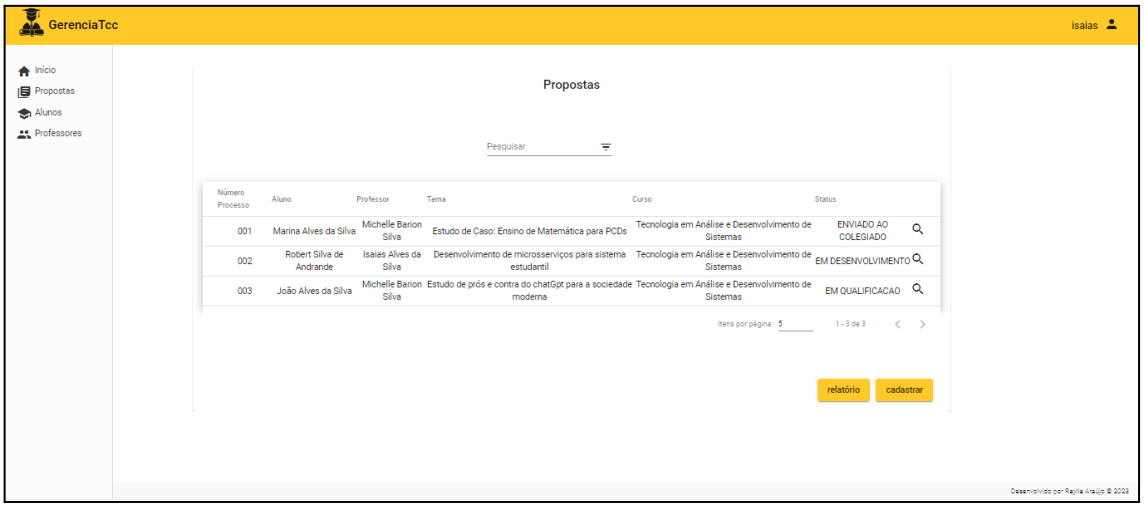

Figura 24. Tela de Listagem de Propostas no perfil professor

A Figura 25 apresenta a listagem de alunos na visão do professor, que se assemelha com a visão do coordenador (RF-4). Na listagem é possível ver previamente informações dos alunos como prontuário, nome, curso e áreas de interesse, além de visualizar detalhadamente o aluno clicando no ícone de lupa. Além disso, é possível filtrar os alunos pelos valores dos campos, como áreas de interesses (RF-7) e exibir relatórios ao clicar no botão "relatório" (RF-6).

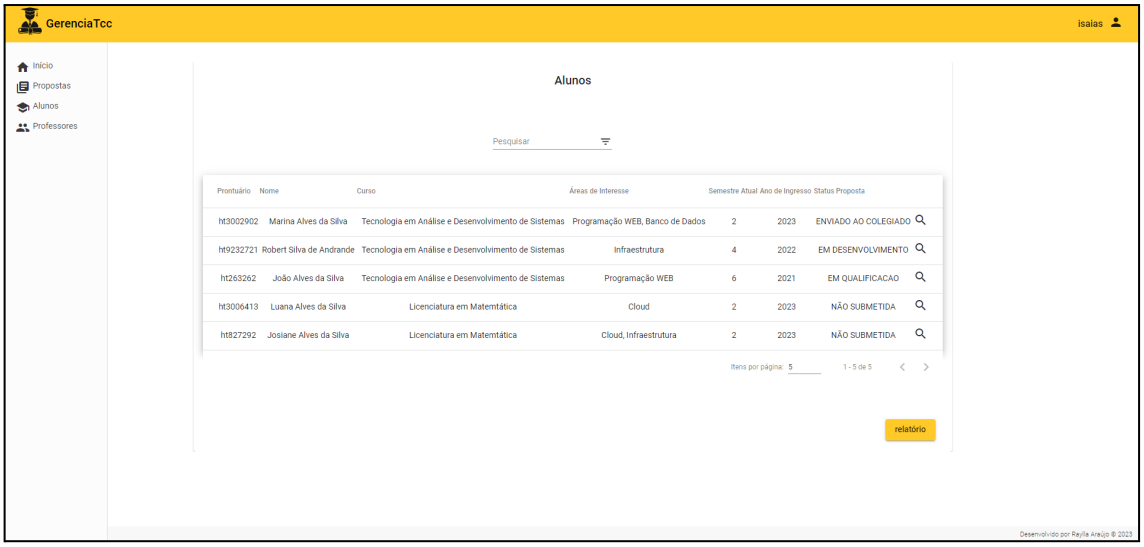

Figura 25. Tela de Listagem de Alunos e Áreas de Interesses no perfil professor

Ao clicar no botão "relatório" é possível gerar um pdf (RF-6) com os alunos filtrados (RF-7), sendo exibida a listagem de alunos por curso e um gráfico da relação alunos do curso X status proposta, como mostra a Figura 26.

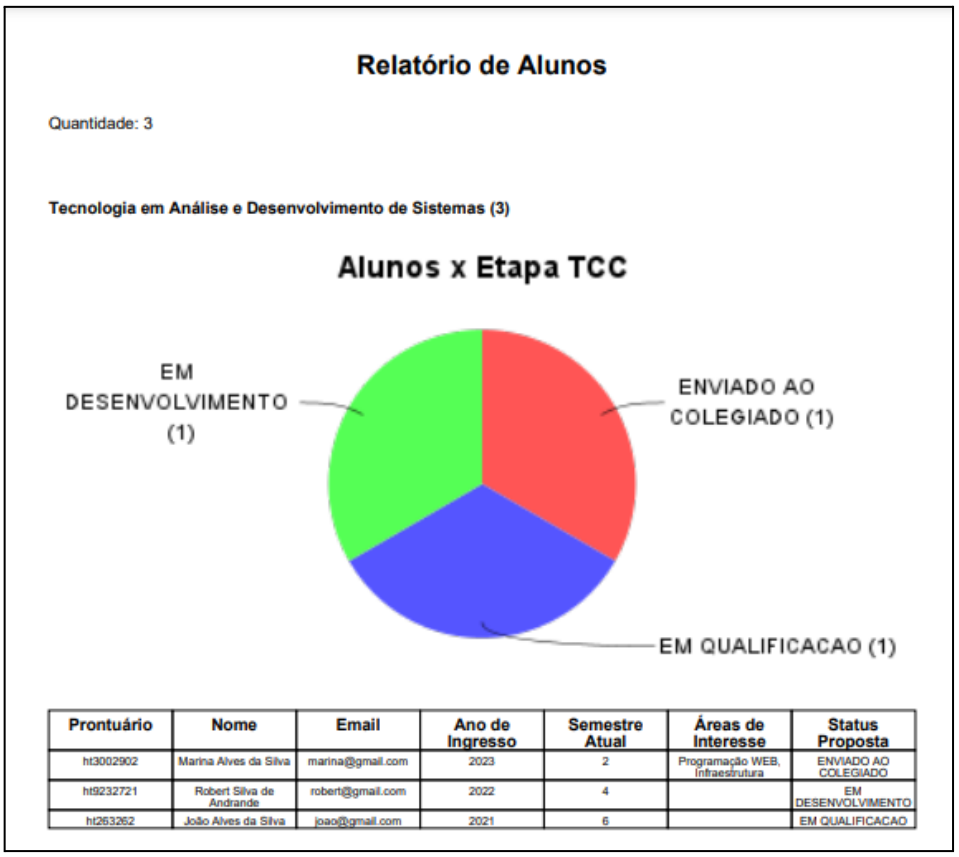

Figura 26. Relatório de alunos por curso

Ao clicar no ícone de lupa da listagem de aluno, como mostrado na Figura 25, é possível detalhar as informações dos alunos, como celular e e-mail, como é apresentado na Figura 27 (RF-4).

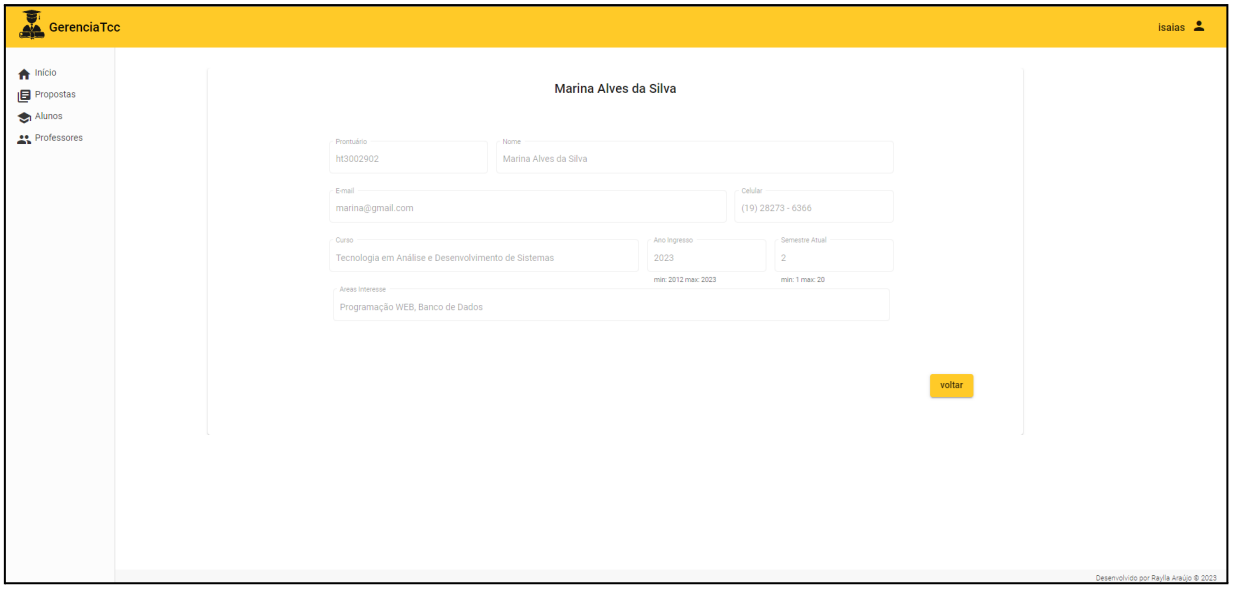

Figura 27. Tela de consulta de aluno

Quanto a visão no perfil coordenador, a Figura 28 mostra a listagem de cursos, podendo filtrar através do campo de busca, detalhar o curso e cadastrar novos cursos (RF-5).

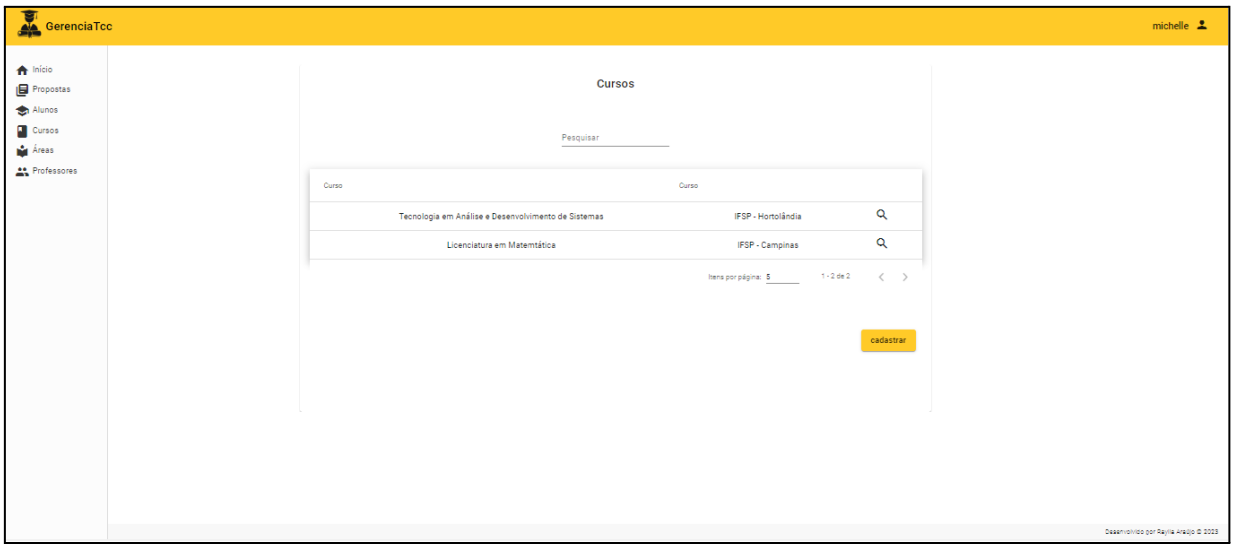

Figura 28. Tela de Listagem de cursos no perfil coordenador

Já a Figura 29 traz a funcionalidade associada à visualização da consulta de um curso ao clicar no ícone de lupa da listagem (RF-5).

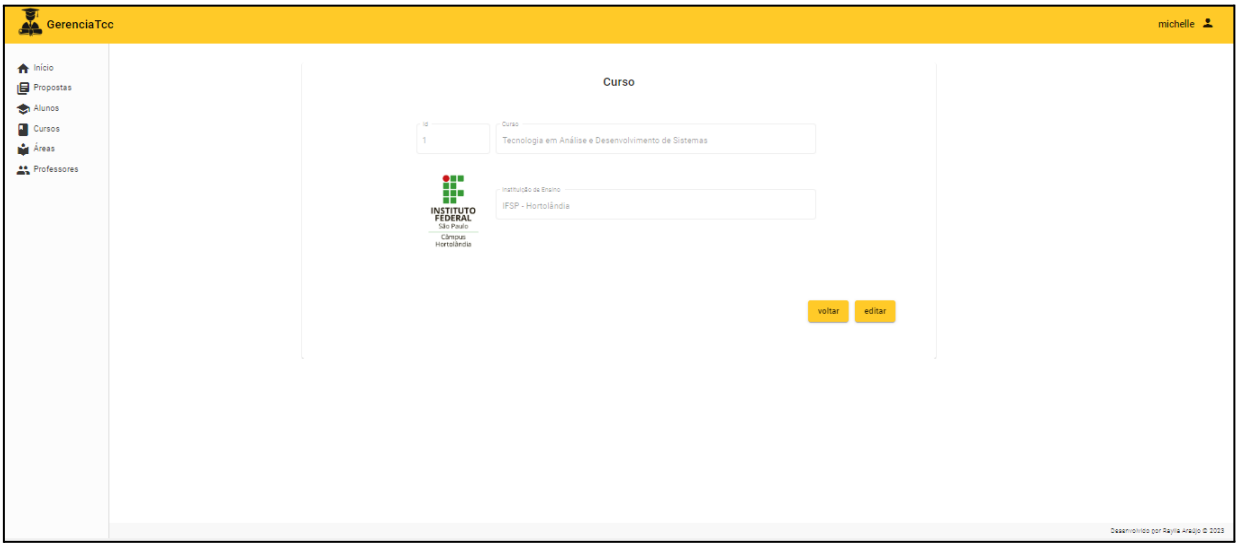

Figura 29. Tela de visualização de curso no perfil coordenador

E para concluir as funcionalidades no perfil coordenador, a Figura 30 apresenta a listagem de áreas, podendo filtrar através do campo de busca, detalhar a área e cadastrar nova área (RF-5).

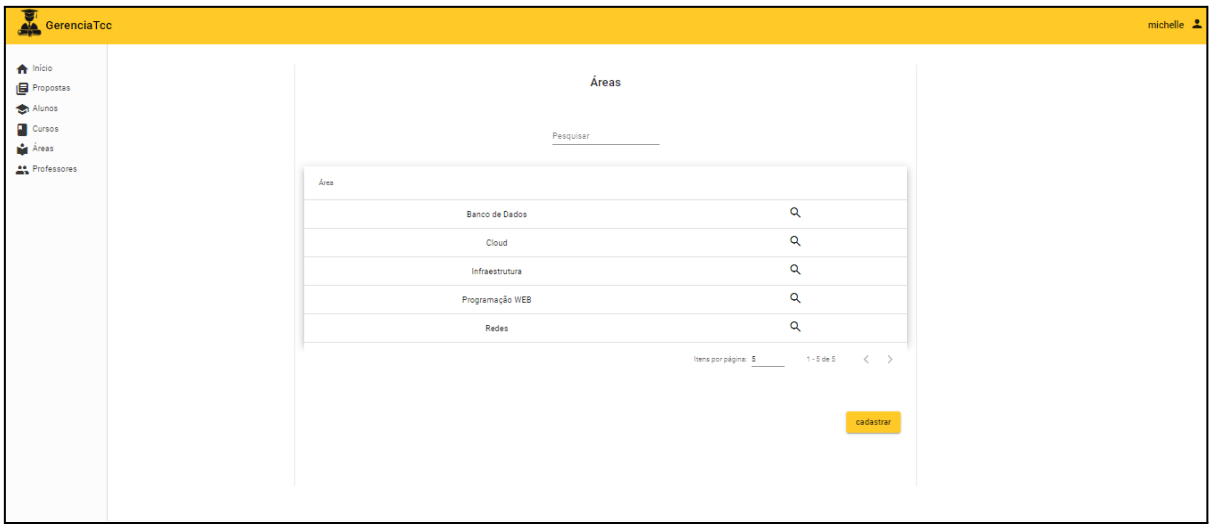

Figura 30. Tela de Listagem de áreas de interesse no perfil coordenador

5.5. *Dataset* da implementação

O desenvolvimento do banco de dados é feito via *Java Persistence API* (JPA) que é encapsulado pelo *framework Spring*. Este utiliza as classes mapeadas com a *notation Entity* para gerar as tabelas, suas colunas e seus relacionamentos. Sendo assim, o banco de dados possui sua geração automática, sem necessidade de criação manual das tabelas e relacionamentos.

A fim de ilustrar o *dataset*, a Figura 31 traz o DER gerado através do MySQL.

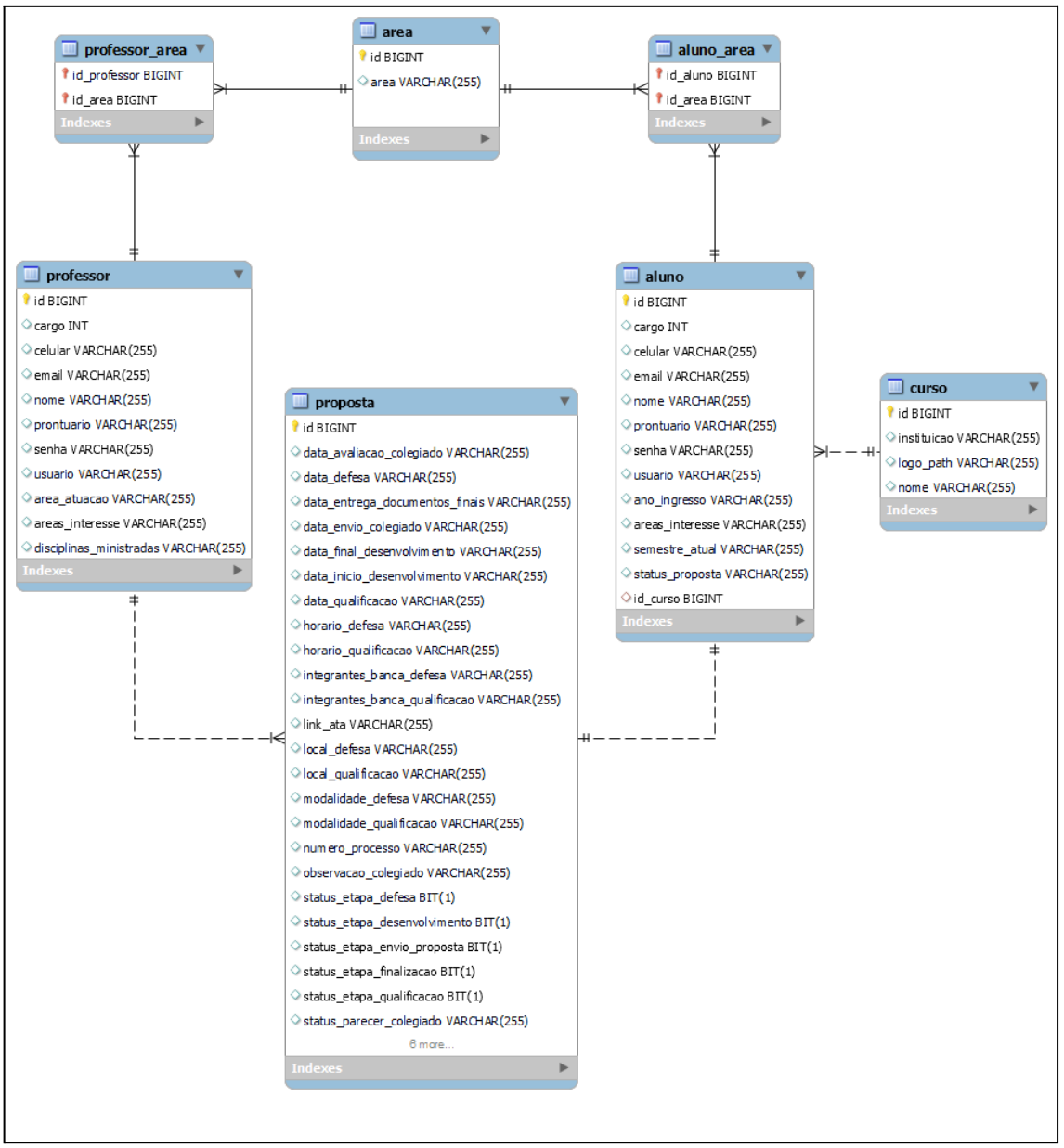

Figura 31. Diagrama Entidade-Relacionamento (DER)

6. Conclusão e trabalhos futuros

Como conclusão, vale salientar a importância do sistema proposto dado os problemas enfrentados tanto pelos orientadores quanto os orientandos, em que os primeiros precisam inserir todos os dados dos seus orientandos em uma planilha com várias abas e atualizar constantemente a cada alteração de status do TCC. Já os segundos, os orientandos, têm grandes dificuldades em encontrar orientadores disponíveis com as habilidades técnicas necessárias para as suas ideias de Trabalho de Conclusão de Curso.

Seguindo o cronograma, foi realizado inicialmente um estudo de trabalhos correlatos e entrevistas para conhecimento das regras de negócio a serem implementadas no sistema. Partindo disso, foi criado o diagrama de casos de uso com as funcionalidades do sistema e realizado o desenvolvimento da *API* e das telas do sistema. Por fim, foram realizados testes manuais, testando a aplicação de ponta a ponta.

A partir do funcionamento da aplicação, espera-se que esta possa exercer sua função social contribuindo para a automação do gerenciamento de TCCs, facilitando a vida dos docentes e discentes, ao promover organização e centralização dos processos referentes ao TCC. Foi considerado como cenário o Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do IFSP Campus Hortolândia, contudo a ferramenta ficará disponível para que outras instituições possam utilizar a codificação, fazendo possíveis ajustes quanto aos requisitos[6].

Para desenvolvimento deste trabalho e até mesmo destacando as competências e habilidades adquiridas no decorrer do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, houve aplicação de conceitos associados às disciplinas de Algoritmo e Lógica de Programação, Desenvolvimento de Sistemas *Web*, Banco de Dados, Engenharia de *Software*, Programação Orientada a Objetos, além da Metodologia de Pesquisa e Projeto de Sistemas para planejamento e desenvolvimento teórico-técnico.

6.1. Trabalhos futuros

Visa-se a implementação de um algoritmo que faça a sugestão de orientadores aos alunos, a partir do *"match"* das áreas de interesses.

Foi criada a opção da agenda pública para consulta das bancas de qualificação e defesa na tela inicial da ferramenta. No entanto, como trabalho futuro, sugere-se a implementação de um rotina que possa encaminhar um e-mail a todos os alunos com as informações dessa agenda.

Outro ponto a ser elencado é a implementação de testes unitários e integrados para garantir que o sistema não venha a possuir *bugs* em produção quando implantado.

E considerando que a Instituição de Ensino possui o sistema SUAP, torna-se como trabalho futuro o estudo sobre uma possível integração com os dados entre as aplicações para eliminação de funcionalidades associadas ao cadastro, como por exemplo, de usuários.

Referências

ALVES, D. L. (2019). "PAE docs: Plataforma de Gerenciamento do Programa de Assistência Estudantil do IFSP Campus Hortolândia".
https://hto.ifsp.edu.br/portal/images/thumbnails/images/IFSP/Cursos/Coord_ADS/Arquivos/TCCs/2019/TCC_Daniel_Luz_Alves.pdf. [Online: acessado em 23 de junho de 2022].

ALVES, W. P. (2009). "Banco de dados: teoria e desenvolvimento". Editora Érica.

AMAZON. (2022). "O que é uma API?". https://aws.amazon.com/pt/what-is/api. [Online: acessado em 23 de junho de 2022].

[6] A ferramenta proposta se encontra disponível para consulta/download através dos links: https://github.com/rayllaa/AdminStepTCC-java-spring e https://github.com/rayllaa/AdminStepTCC-angular.

BEZERRA, E. (2016). “Princípios de análise e projeto de sistemas com UML”. Editora Elsevier Academic.

FLANAGAN, D. (2012). “JavaScript: O Guia Definitivo”. Editora Bookman.

GOOGLE. (2022) “Introduction to the Angular Docs”. https://angular.io/docs. [Online: acessado em 21 de junho de 2022].

HUMBLE, J; FARLEY, D. (2013) “Entrega Contínua: Como Entregar Software de Forma Rápida e Confiável”. Editora Novatec.

LAUDON, K. C.; LAUDON, J. P. (2016) "Management Information Systems: Managing the Digital Firm". Pearson, 2016.

MARTIN, R. C. (2019) “Arquitetura Limpa: O guia do artesão para estrutura e design de software”. Editora Alta Books.

MARTIN, R. C. (2009) “Código limpo: Habilidades práticas do Agile Software”. Editora Alta Books.

MELO, J. C. S. (2007). “Guia do Java Enterprise Edition 5: desenvolvendo aplicações corporativas”. Editora Brasport.

MICROSOFT. (2022) “Criar aplicativos JavaScript usando o TypeScript”. https://docs.microsoft.com/pt-br/learn/paths/build-javascript-applications-typescript/.[Online: acessado em 23 de junho de 2022].

MINETTO, E. L. (2007). “Frameworks para Desenvolvimento PHP”. Editora Novatec.

ORACLE. (2019). “MySQL”. https://www.mysql.com/products/. [Online: acessado em 23 de maio de 2019].

POSTMAN. (2022). “What is Postman?”. https://www.postman.com/. [Online: acessado em 03 de dezembro de 2022].

PRESSMAN, R. S. (2021). “Engenharia de Software: Uma Abordagem Profissional”. Editora AMGH.

REZENDE, D. A. (2005). “Engenharia de Software e Sistemas de Informação”. Editora Brasport.

SALVINO, L. T. (2019). “Implementação do Sistema Gerenciador de Estágio do IFSP Hortolândia baseado em microsserviços”. https://hto.ifsp.edu.br/portal/images/thumbnails/images/IFSP/Cursos/Coord_ADS/Arquivos/TCCs/2019/TCC_Lenilson_Teixeira_Salvino.pdf. [Online: acessado em 23 de junho de 2022].

SCHIAROLLI, F. (2021). “Portal de gerenciamento de inscrições para cursos de extensão oferecidos pelo campus Hortolândia”. https://hto.ifsp.edu.br/cloud/s/HMd2jLrtcJW58Xw. [Online: acessado em 23 de junho de 2022].

SILVA, M. S. (2008). “Criando Sites com HTML”. Editora Novatec.

SPRING. (2022) “Spring Framework”. https://spring.io/projects/spring-framework. [Online: acessado em 23 de junho de 2022].

SWAGGER. (2022). “Documentação da API”. https://swagger.io/solutions/api-documentation/. [Online: acessado em 03 de dezembro de 2022].