CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
Faculdade de Tecnologia da Praia Grande
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

TCC MENTOR

SISTEMA AUXILIAR NA CRIAÇÃO DE TCC’s

Gustavo de Rezende Garcia
Heitor Pedro de Godoi
Nycolas da Guia Santos

PRAIA GRANDE
2025

Gustavo de Rezende Garcia
Heitor Pedro de Godoi
Nycolas da Guia Santos

TCC MENTOR

SISTEMA AUXILIAR NA CRIAÇÃO DE TCCs

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia da Praia Grande, como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.
Orientador: Prof. Joseffe Barroso de Oliveira

PRAIA GRANDE
2025

AGRADECIMENTOS

Manifestamos nossa sincera gratidão a Deus, por nos conceder a dádiva da vida e por fortalecer nossa caminhada diante dos desafios enfrentados ao longo deste curso.

Aos nossos familiares, dedicamos nosso sincero reconhecimento pelo apoio incondicional, pela paciência durante os momentos de dedicação exclusiva a este projeto e pelo incentivo que nos manteve firmes mesmo diante das dificuldades.

Aos nossos professores e orientadores, manifestamos nosso profundo agradecimento pelo conhecimento compartilhado, pelas valiosas orientações e pelo compromisso em nos guiar nessa importante etapa de nossa formação profissional. Cada correção, sugestão e palavra de estímulo foi fundamental para o aprimoramento deste trabalho e de nossa trajetória acadêmica.

Por fim, agradecemos aos colegas e amigos que, de diferentes formas, contribuíram com seu apoio e compreensão durante esse processo, tornando essa conquista ainda mais significativa.

RESUMO

O TCC Mentor é uma plataforma inovadora desenvolvida para facilitar a comunicação entre orientadores e alunos durante a elaboração de trabalhos de conclusão de curso. O sistema combina ferramentas de mentoria inteligente com um assistente virtual especializado, resolvendo problemas como interpretação de feedbacks e organização centralizada do processo de orientação.

Com recursos como análise automatizada de feedbacks e suporte a dúvidas acadêmicas em tempo real, a plataforma busca otimizar o tempo dos orientadores e reduzir a ansiedade dos alunos, garantindo um fluxo de trabalho mais eficiente e transparente.

Palavras-chave: Mentoria acadêmica; Comunicação orientador-aluno; Feedback automatizado; Gestão de TCC.

ABSTRACT

TCC Mentor is an innovative platform designed to streamline communication between advisors and students during the development of final-year projects. The system integrates intelligent mentoring tools with a specialized virtual assistant, addressing challenges such as feedback interpretation and centralized coordination of the advising process. Featuring automated feedback analysis and real-time academic support, the platform aims to optimize advisors' time, reduce student anxiety, and ensure an efficient, transparent workflow.

Keywords: Academic mentoring, Advisor-student communication, Automated feedback, Thesis management.

LISTA DE ABREVIAÇÕES E SIGLAS

API	Application Programming Interface
CSS	Cascading Style Sheets
DER	Diagrama Entidade Relacionamento
HTML	HyperText Markup Language
POO	Programação Orientada a Objetos
REST	Representational State Transfer
SCRUM	Sprint Cycle Review Update Meeting
SQL	Structured Query Language
UML	Unified Modeling Language
IA	Inteligência Artificial

LISTA DE FIGURAS

Figura 1 - Protótipo de baixa fidelidade. 24
Figura 2 - Protótipo de alta fidelidade da página inicial. 25
Figura 3 - Protótipo de alta fidelidade da página login. 26
Figura 4 – Caso de uso aluno. 35
Figura 5 – Caso de uso coordenador. 37
Figura 6 – Diagrama de classes. 39
Figura 7 – Diagrama de Entidade e Relacionamento. 41
Figura 8 - Diagrama de Atividade 47
Figura 9 - Figma 50
Figura 10 - Github 51
Figura 11 - Visual Studio Code 52
Figura 12 - MySQL 54
Figura 13 - JavaScript 55
Figura 14 - NodeJS 56
Figura 15 - HTML 57
Figura 16 - CSS 58
Figura 17 - draw.io 59
Figura 18 - Rotas de Usuário representadas no código. 64
Figura 19 - Rotas de Projeto representadas no código. 66
Figura 20 - Rotas de comunicação representadas no código.. 68
Figura 21 - Todas as páginas da aplicação Web 69

LISTA DE TABELAS

Tabela 1 - Dicionário de Dados da Tabela de Usuários 42
Tabela 2 - Dicionário de Dados da Tabela de Projetos 43
Tabela 3 - Dicionário de Dados da Tabela de Alunos por Projeto 44
Tabela 4 - Dicionário de Dados da Tabela de Mensagens com a IA 44
Tabela 5 - Dicionário de Dados da Tabela de Mensagens com o Orientador 45

SUMÁRIO

1 INTRODUÇÃO **11**
2 PROBLEMATIZAÇÃO DE PESQUISA **13**
3 SOLUÇÃO E HIPÓTESE **15**
3.1 OBJETIVO GERAL 16
3.2 OBJETIVOS ESPECÍFICOS 17
4 JUSTIFICATIVA **18**
4.1 RELEVÂNCIA SOCIAL 19
5 METODOLOGIA **20**
6 PLANEJAMENTO DO PROJETO **22**
6.1 PROTOTIPAÇÃO 23
6.2 PROTÓTIPO DE BAIXA FIDELIDADE 23
6.3 PROTÓTIPO DE ALTA FIDELIDADE 24
6.4 ENTIDADES DO SISTEMA 26
6.5 REQUISITOS FUNCIONAIS 27
6.6 REQUISITOS NÃO FUNCIONAIS 32
6.7 DIAGRAMAS UML (LINGUAGEM DE MODELAGEM UNIFICADA) 33
6.7.1 DIAGRAMAS DE CASO DE USO 34
6.7.2 DIAGRAMA DE CLASSES 37
6.7.3 DIAGRAMA DE ENTIDADE E RELACIONAMENTO 39
6.7.4 DIAGRAMA DE ATIVIDADE 45
7 TECNOLOGIAS E FERRAMENTAS **49**
7.1 FIGMA 49
7.2 GITHUB 50
7.3 VISUAL STUDIO CODE 51
7.4 MYSQL 53
7.5 JAVASCRIPT 54
7.6 NODEJS 55
7.7 HTML (EJS) 56
7.8 CSS 57
7.9 DRAW.IO 59
8 DESENVOLVIMENTO **60**

8.1 BANCO DE DADOS 60
8.2 BACK-END 62
8.2.1 ROTAS DE USUÁRIO 63
8.2.2 ROTAS DE PROJETO 65
8.2.3 ROTAS DE COMUNICAÇÃO 67
8.3 FRONT-END 68
8.3.1 PÁGINAS 69
9 CONSIDERAÇÕES FINAIS 67

1 INTRODUÇÃO

A elaboração de um Trabalho de Conclusão de Curso (TCC) é uma etapa fundamental na formação acadêmica, representando não apenas a consolidação dos conhecimentos adquiridos ao longo da graduação, mas também um desafio significativo para estudantes e orientadores. Esse processo, no entanto, é frequentemente marcado por dificuldades de comunicação, desencontros de expectativas e problemas na gestão do tempo, que podem comprometer a qualidade do trabalho final e até mesmo levar ao abandono do projeto. Foi a partir da vivência desses problemas em nossa própria jornada acadêmica que surgiu a motivação para desenvolver o TCC Mentor, uma plataforma inovadora que busca transformar a experiência da orientação de TCCs.

Durante o desenvolvimento de nossos projetos de TCC, enfrentamos inúmeros obstáculos na comunicação com nossos orientadores. Apesar da disponibilidade de ferramentas como Microsoft Teams e e-mail, percebemos que esses canais genéricos não atendiam às necessidades específicas do processo de orientação acadêmica. Mensagens importantes se perdiam em longas threads de conversa, os feedbacks muitas vezes chegavam de forma fragmentada e difícil de interpretar, e o agendamento de reuniões se tornava um jogo de cartas marcadas, com horários que raramente se alinhavam.

O cenário atual da orientação acadêmica revela uma contradição significativa: em uma era de comunicação digital instantânea e soluções tecnológicas avançadas, o processo de orientação de TCCs ainda segue modelos ultrapassados, que não acompanharam a evolução das necessidades dos alunos e das possibilidades oferecidas pela tecnologia. Enquanto outras áreas da educação incorporaram ferramentas digitais para melhorar a aprendizagem e a interação entre professores e alunos, a orientação de trabalhos acadêmicos permanece, em grande parte, presa a métodos tradicionais que já não atendem às demandas do século XXI.

É neste contexto que o TCC Mentor se apresenta como uma solução inovadora. Ao contrário das plataformas genéricas de comunicação, nosso sistema foi desenvolvido especificamente para atender às necessidades do processo de orientação acadêmica. A plataforma não apenas facilita a comunicação entre

orientadores e orientandos, mas também incorpora recursos inteligentes que ajudam a interpretar feedbacks, organizar cronogramas e gerenciar documentos, tudo em um ambiente centralizado e projetado para otimizar o fluxo de trabalho acadêmico.

A relevância deste projeto vai além da solução de problemas práticos. Ao facilitar o processo de orientação, o TCC Mentor tem o potencial de impactar positivamente a qualidade dos trabalhos acadêmicos produzidos, reduzir as taxas de evasão nos cursos de graduação e, em última instância, contribuir para a produção científica nacional. Em um momento em que as instituições de ensino superior buscam aumentar sua eficiência e melhorar seus indicadores de qualidade, soluções como a que propomos representam um passo importante na modernização dos processos acadêmicos.

O desenvolvimento do TCC Mentor foi guiado por três princípios fundamentais: clareza na comunicação, eficiência no processo e acessibilidade na orientação. A plataforma foi concebida não apenas para resolver os problemas que identificamos em nossa própria experiência, mas para criar um novo padrão na relação entre orientadores e orientandos, um padrão que aproveite todo o potencial das tecnologias digitais para tornar o processo de orientação mais produtivo, menos estressante e, acima de tudo, mais eficaz na condução dos alunos à conclusão bem-sucedida de seus trabalhos acadêmicos.

Neste contexto, o presente trabalho tem como objetivo apresentar o desenvolvimento do TCC Mentor, desde sua concepção até a implementação das funcionalidades principais. Ao longo dos próximos capítulos, detalharemos a fundamentação teórica que embasa nosso projeto, a metodologia utilizada em seu desenvolvimento, as tecnologias empregadas e os resultados obtidos até o momento. Acreditamos que esta solução representa um avanço significativo na forma como entendemos e praticamos a orientação acadêmica, e esperamos que ela possa servir como modelo para iniciativas similares em outras instituições de ensino.

2 PROBLEMATIZAÇÃO DE PESQUISA

O processo de orientação de Trabalhos de Conclusão de Curso nas instituições de ensino superior brasileiras enfrenta desafios significativos que impactam diretamente na qualidade dos trabalhos e na experiência acadêmica dos estudantes. A relação entre orientador e orientando, fundamental para o sucesso do TCC, muitas vezes se mostra problemática devido a diversos fatores estruturais e comunicacionais.

Estudos indicam que grande parte dos desafios enfrentados na elaboração de TCCs está associada à comunicação deficiente entre orientadores e orientandos. Segundo Leite Filho e Martins (2004), “muitos problemas surgidos durante o processo de construção do trabalho estariam ligados à relação orientador-orientando”, o que evidencia a importância de uma comunicação clara e contínua durante todo o desenvolvimento acadêmico.

Essa barreira comunicacional gera retrabalho, desalinhamento de expectativas e, em casos mais graves, pode levar ao abandono do projeto.

A ausência de um ambiente centralizado para gestão do TCC também se mostra prejudicial. Muitos estudantes enfrentam dificuldades em organizar versões do trabalho, prazos e demandas específicas, enquanto os orientadores carecem de ferramentas adequadas para acompanhar o progresso de seus orientandos de forma estruturada.

Esses desafios se tornam ainda mais evidentes quando consideramos o perfil diversificado dos estudantes universitários, muitos dos quais conciliam os estudos com trabalho e outras responsabilidades. A falta de flexibilidade nos modelos tradicionais de orientação acadêmica acaba por excluir aqueles que não conseguem se adaptar aos horários rígidos e métodos convencionais.

O desenvolvimento do TCC Mentor apresenta desafios técnicos significativos, pois exige a integração de tecnologias inovadoras em um contexto acadêmico que tradicionalmente carece de soluções tecnológicas especializadas para orientação. A construção da plataforma demandou uma análise minuciosa dos fluxos de trabalho

acadêmicos e a adaptação de ferramentas de IA para atender às necessidades específicas da relação orientador-orientando, combinando conhecimentos técnicos com uma profunda compreensão das dinâmicas pedagógicas envolvidas no processo de orientação.

3 SOLUÇÃO E HIPÓTESE

O desenvolvimento do TCC Mentor surgiu como resposta aos desafios recorrentes no processo de orientação acadêmica, propondo uma abordagem tecnológica especializada para aprimorar a comunicação e o acompanhamento de trabalhos científicos. Esta solução foi concebida para transformar a dinâmica tradicional de orientação, substituindo métodos fragmentados por um ambiente integrado e eficiente.

No cerne da plataforma está um sistema de comunicação dedicado que elimina as limitações dos canais convencionais. Ao contrário de e-mails ou aplicativos de mensagens, onde informações importantes se perdem, o TCC Mentor organiza as interações de forma estruturada, garantindo que cada feedback e orientação seja preservado e facilmente acessível. Este fluxo comunicativo contínuo permite um diálogo mais produtivo entre orientadores e orientandos.

Um dos diferenciais inovadores da plataforma é seu mecanismo de análise inteligente de feedbacks. O sistema auxilia na transformação de comentários técnicos complexos em orientações mais claras e acessíveis. Esta funcionalidade reduz significativamente os mal-entendidos frequentes no processo de revisão e permite que os estudantes aproveitem melhor as correções recebidas.

A hipótese central que fundamenta este projeto sustenta que a implementação do TCC Mentor pode transformar qualitativamente a experiência de orientação. Espera-se observar uma melhoria substancial na eficácia comunicativa, com redução expressiva de ruídos e retrabalhos. O tempo antes dedicado a questões burocráticas e organizacionais deverá ser redirecionado para o essencial desenvolvimento acadêmico. Paralelamente, a plataforma tem potencial para diminuir a ansiedade dos estudantes através de um acompanhamento mais transparente e sistemático.

A relevância desta iniciativa torna-se evidente ao constatar que a maioria das instituições de ensino ainda depende de ferramentas genéricas e métodos convencionais para gerir o complexo processo de orientação. O TCC Mentor emerge como uma alternativa especializada, projetada especificamente para otimizar esta relação pedagógica. Ao concentrar-se nos pontos críticos identificados -

comunicação, documentação e suporte técnico - a plataforma oferece uma resposta concreta aos principais desafios enfrentados por orientadores e orientandos no desenvolvimento de trabalhos acadêmicos.

3.1 OBJETIVO GERAL

A partir dos conhecimentos adquiridos ao longo do curso, desenvolvemos a plataforma TCC Mentor, que visa aprimorar o processo de orientação acadêmica, proporcionando aos orientadores e orientandos um ambiente organizado e eficiente para o desenvolvimento de trabalhos científicos. A plataforma busca facilitar a comunicação, simplificar o acompanhamento das versões do trabalho e oferecer suporte automatizado para dúvidas frequentes, promovendo uma experiência mais clara e produtiva para ambos os lados.

O TCC Mentor diferencia-se por colocar o estudante no centro do processo, garantindo que ele tenha acesso a feedbacks compreensíveis, orientações estruturadas e recursos que aumentem sua autonomia na elaboração do trabalho. Simultaneamente, oferece aos orientadores ferramentas que otimizam seu tempo, permitindo um acompanhamento mais eficaz sem sobrecarga de demandas burocráticas.

Mais do que criar apenas mais uma ferramenta digital, nosso objetivo é transformar a relação de orientação em um processo mais transparente, colaborativo e menos estressante. Queremos que os alunos se sintam mais confiantes em seu desenvolvimento acadêmico e que os orientadores possam dedicar seu tempo ao que realmente importa: a qualidade do conteúdo e da pesquisa. Ao mesmo tempo, a plataforma servirá como um meio eficiente para padronizar e organizar o fluxo de trabalho, beneficiando tanto as instituições de ensino quanto seus alunos e professores.

3.2 OBJETIVOS ESPECÍFICOS

Este projeto visa desenvolver uma plataforma digital focada exclusivamente em melhorar a comunicação acadêmica durante o processo de orientação de trabalhos científicos. O objetivo principal consiste em criar um sistema de mensagens especializado que organize as interações por projetos e etapas de desenvolvimento, substituindo eficazmente os meios convencionais como e-mails e mensageiros instantâneos.

Como segundo objetivo específico, propõe-se desenvolver ferramentas de análise textual que auxiliem os orientadores na formulação de feedbacks mais claros e objetivos. Estas funcionalidades serão projetadas para identificar possíveis melhorias na estruturação do texto acadêmico, sugerindo ajustes que tornem o conteúdo mais coeso e alinhado às normas científicas.

O terceiro objetivo central é a criação de um ambiente digital intuitivo e acessível, com interfaces adaptadas às necessidades específicas de orientadores e orientandos. A plataforma priorizará a simplicidade de uso, garantindo que todos os usuários possam navegar e utilizar as funcionalidades sem dificuldades técnicas, independentemente de seu nível de familiaridade com ferramentas digitais.

4 JUSTIFICATIVA

A criação do TCC Mentor responde a uma demanda crítica no cenário acadêmico atual: a carência de ferramentas adequadas para otimizar o processo de orientação de trabalhos científicos. A plataforma foi concebida para superar as limitações dos métodos tradicionais, que frequentemente resultam em comunicação ineficiente, desorganização e sobrecarga tanto para orientadores quanto para orientandos. Ao integrar tecnologias inovadoras em um ambiente unificado, a solução oferece mecanismos aprimorados para troca de feedbacks, gestão documental e suporte acadêmico imediato, preenchendo uma lacuna evidente no mercado de ferramentas educacionais.

A importância desta iniciativa se revela em múltiplas dimensões do ecossistema acadêmico. Estudantes se beneficiarão de um acompanhamento mais transparente e de maior independência em sua produção científica. Orientadores contarão com recursos eficazes para gerenciar suas atividades de mentoria, enquanto as instituições de ensino terão à disposição instrumentos valiosos para monitorar e aprimorar seus processos educacionais. O caráter inovador da plataforma reside em sua capacidade de reinventar a dinâmica de orientação, incorporando funcionalidades inteligentes como análise textual avançada e assistência automatizada para questões metodológicas.

Ao abordar diretamente as deficiências do modelo atual, o TCC Mentor promove uma transformação significativa na elaboração de trabalhos acadêmicos, com potencial para diminuir índices de evasão e elevar o padrão da produção científica. Esta iniciativa exemplifica a aplicação estratégica de tecnologias educacionais, atendendo às exigências contemporâneas por modernização no ensino superior e estabelecendo um paradigma adaptável a diversas instituições e áreas do conhecimento.

4.1 RELEVÂNCIA SOCIAL

A dificuldade no desenvolvimento de trabalhos de conclusão de curso é um desafio recorrente no ensino superior, afetando tanto estudantes quanto orientadores. Muitas vezes, a falta de ferramentas adequadas para organizar o processo de orientação gera atrasos, retrabalhos e até mesmo o abandono de projetos promissores. Uma plataforma especializada nesse apoio poderia transformar essa realidade, tornando a produção acadêmica mais acessível e eficiente para toda a comunidade universitária.

Do ponto de vista social, facilitar a elaboração de pesquisas científicas impacta diretamente na qualidade do conhecimento produzido. Trabalhos bem estruturados e com acompanhamento adequado têm maior potencial para gerar soluções aplicáveis a problemas reais da sociedade, desde questões de saúde pública até inovações tecnológicas. Além disso, ao reduzir as barreiras na fase final da graduação, contribui-se para a formação de profissionais mais qualificados e preparados para os desafios do mercado de trabalho.

A implementação de soluções digitais para orientação acadêmica representa ainda um avanço na modernização do ensino superior. Ao otimizar esse processo, cria-se um ambiente mais propício para a pesquisa e o desenvolvimento intelectual, fortalecendo a relação entre universidades e sociedade. Essa integração é essencial para que o conhecimento acadêmico possa ser transformado em benefícios concretos para a população em geral.

5 METODOLOGIA

A metodologia adotada neste projeto combinou pesquisa aplicada e desenvolvimento iterativo para criar uma solução tecnológica que atenda efetivamente às necessidades dos estudantes durante o processo de orientação de TCC. Inicialmente, conduziu-se uma pesquisa exploratória através de formulário online com estudantes que estavam desenvolvendo ou haviam concluído seus trabalhos de conclusão de curso. O questionário buscou identificar os principais desafios enfrentados na comunicação com orientadores, organização do material de pesquisa e compreensão das orientações recebidas. As respostas coletadas foram analisadas qualitativamente, permitindo mapear os problemas mais recorrentes e relevantes que seriam abordados pela plataforma.

Para o desenvolvimento da solução, adotou-se uma abordagem ágil adaptada, inspirada nos princípios do Scrum mas ajustada às particularidades do projeto acadêmico. O trabalho foi organizado em ciclos curtos de desenvolvimento que incluíram planejamento das funcionalidades prioritárias, implementação incremental e testes contínuos. Essa abordagem permitiu incorporar feedbacks e ajustar a plataforma de forma iterativa, garantindo que cada componente desenvolvido respondesse adequadamente às necessidades identificadas na pesquisa inicial. A equipe realizou reuniões frequentes para acompanhamento do progresso e tomada de decisões colaborativas, mantendo o foco nos objetivos principais do projeto.

A fase de validação envolveu a utilização da plataforma por um grupo de estudantes durante um semestre letivo completo, em condições reais de trabalho acadêmico. O acompanhamento desse período de uso gerou ideias valiosas que permitiram refinamentos na interface e nas funcionalidades. Observou-se especialmente a interação dos usuários com o sistema de comunicação integrado e as ferramentas de análise textual, que foram sendo aprimoradas conforme os padrões de uso se tornavam mais claros. Essa integração contínua entre pesquisa, desenvolvimento e validação prática foi fundamental para criar uma ferramenta que não apenas resolvesse problemas teóricos, mas que se mostrasse efetiva no cotidiano acadêmico dos estudantes. A metodologia adotada, portanto, garantiu que

a solução desenvolvida estivesse ancorada tanto em evidências concretas quanto na experiência real de uso, cumprindo seu propósito de transformar positivamente o processo de orientação de TCCs.

6 PLANEJAMENTO DO PROJETO

A aplicação desenvolvida tem como objetivo principal facilitar a vida dos estudantes durante a criação do Trabalho de Conclusão de Curso (TCC). Por meio da plataforma web, os alunos conseguem organizar suas ideias, receber feedbacks dos orientadores e acompanhar o progresso do seu projeto de forma mais eficiente. O sistema visa centralizar as informações do TCC em um único lugar, otimizando o tempo e reduzindo o estresse envolvido nesse processo acadêmico.

Para o desenvolvimento do back-end da aplicação, foi utilizada a tecnologia Node.js em conjunto com o framework[1] Express[2]. Essa combinação foi escolhida por sua leveza e eficiência na criação de APIs REST[3], possibilitando uma comunicação rápida e direta com o banco de dados, além de oferecer uma estrutura clara e modular para o desenvolvimento das rotas e lógica do servidor.

No front-end, foi adotado o uso de HTML e CSS puro, sem a utilização de frameworks visuais. Essa decisão teve como foco a simplicidade e o controle total sobre o layout e estilo da aplicação, garantindo uma interface objetiva e funcional voltada exclusivamente para navegadores web.

O banco de dados utilizado no projeto foi o MySQL, uma das soluções mais populares e confiáveis do mercado. A escolha se deu pela robustez, facilidade de integração com Node.js e boa performance mesmo em aplicações de pequeno a médio porte.

Todas as tecnologias utilizadas possuem como base a linguagem JavaScript, que se destaca por sua versatilidade e amplo suporte da comunidade. O JavaScript pode ser usado tanto para o lado do cliente quanto para o lado do servidor, permitindo que desenvolvedores criem aplicações inteiras usando uma única

---

[1] Framework é um conjunto de bibliotecas, que abordam funcionalidades, e estruturas, para o desenvolvimento de aplicações. https://balta.io/blog/o-que-e-um-framework [1]

[2] Express é um popular framework web estruturado, escrito em JavaScript que roda sobre o ambiente node.js em tempo de execução. https://developer.mozilla.org/pt-BR/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs [2]

[3] API REST é uma interface de programação de aplicações (API) que segue os princípios de design da arquitetura REST. REST é a sigla para "Representational State Transfer" (transferência de estado representacional), um conjunto de regras e diretrizes sobre como você deve criar uma API web. https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api [3]

linguagem, sendo essencial para projetos que exigem dinamismo e integração entre as partes da aplicação.

6.1 PROTOTIPAÇÃO

Para proporcionar uma visão geral de como funcionaria o fluxo dos sistemas, foram desenvolvidos dois tipos de protótipos. O primeiro foi um protótipo de baixa fidelidade, que serviu como ponto de partida para conceber como a aplicação seria feita e como seria concebida. Posteriormente, foi criado um protótipo de alta fidelidade, projetando a aparência final do sistema, sendo muito próximo do resultado final.

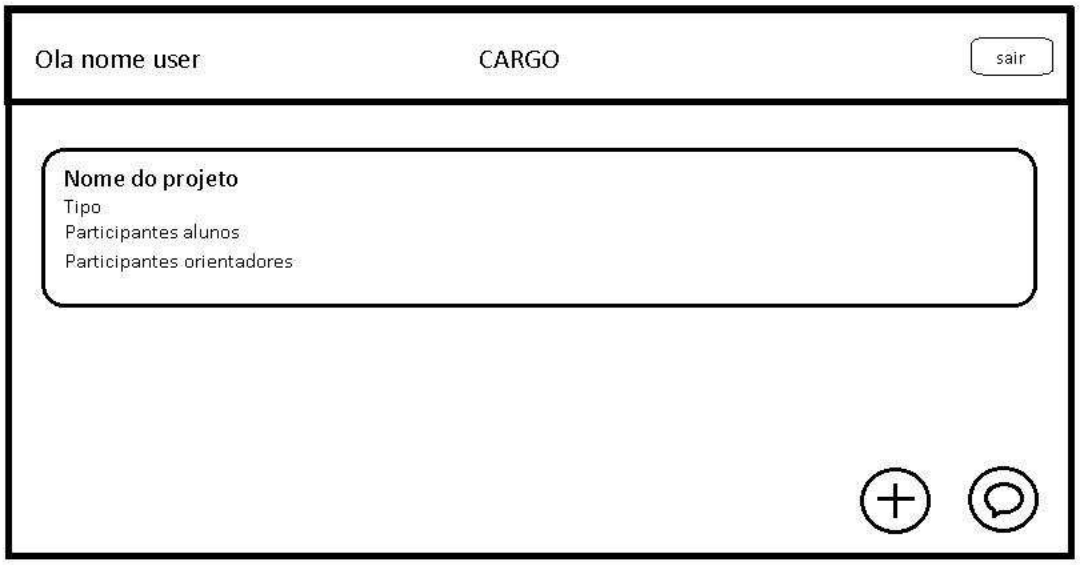
6.2 PROTÓTIPO DE BAIXA FIDELIDADE

Protótipos de baixa fidelidade são esboços iniciais que validam conceitos de design rapidamente. Usados em UX, ajudam a identificar falhas e descartar ideias ineficientes antes do desenvolvimento, assim como define o site Medium:

> “Um protótipo de baixa fidelidade é bastante utilizado em fases iniciais e exploratórias de um projeto para validar um conceito e decidir se uma ideia tem ou não valor funcional. Devem ser rápidos, rudimentares e baratos.” (MEDIUM, 2021)

Assim como o diagrama de caso de uso, foi utilizado a ferramenta Draw IO para desenvolver o diagrama de classes, abaixo um exemplo de como era o sistema como um protótipo de baixa fidelidade:

Figura 1 - Protótipo de baixa fidelidade



Fonte: Elaborado pelos autores, 2025.

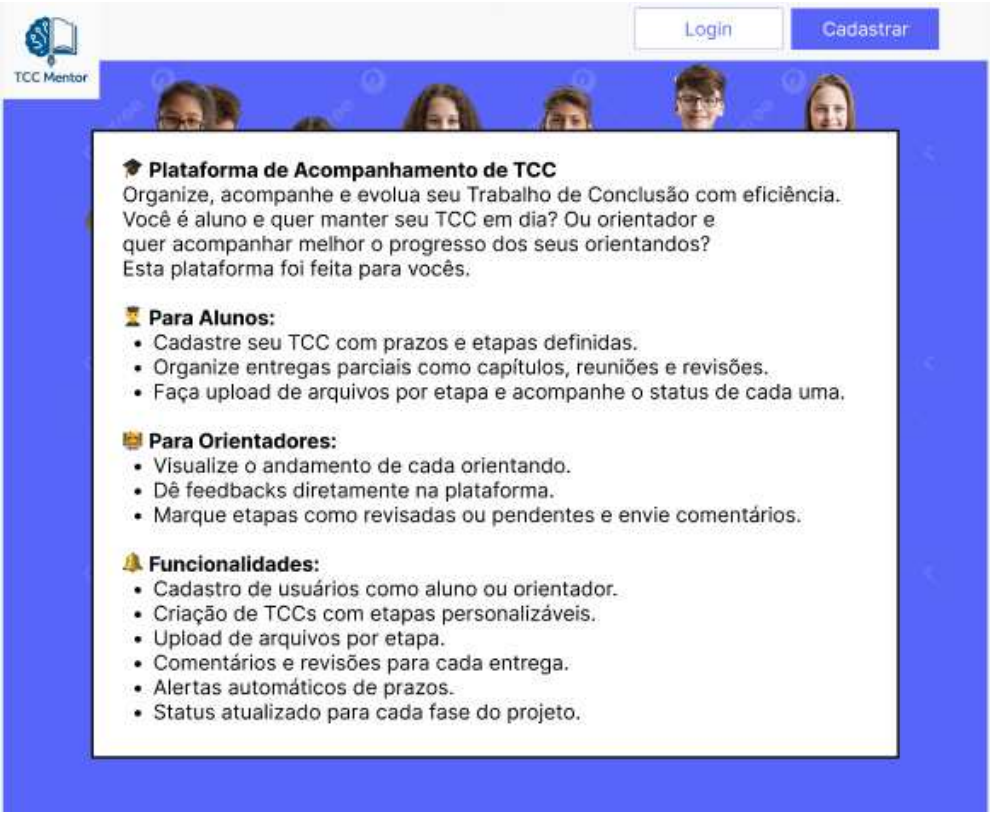
6.3 PROTÓTIPO DE ALTA FIDELIDADE

Posteriormente, após os avanços na ideia e sua implementação,baseado no que o protótipo de baixa fidelidade nos permitiu ver, as telas do protótipo de alta fidelidade do sistema são criadas. Ficando mais parecido com a versão final do aplicativo

> "Um protótipo de alta fidelidade deve se aproximar ao máximo dos aspectos visuais e funcionais do produto final, incluindo o conteúdo, fluxo de navegação e interações. São muito utilizados para testes e validação com usuários, ou para vender uma ideia, pois ver o produto "funcionando" costuma gerar fascinação." (MEDIUM, 2021)

Após a finalização de testes do protótipo, foi possível iniciar a programação das telas do próprio sistema. Abaixo, as principais telas que foram desenvolvidas para serem integradas ao sistema:

Figura 2 - Protótipo de alta fidelidade da página inicial.



Fonte: Elaborado pelos autores, 2025.

Figura 3 - Protótipo de alta fidelidade da página Login.

Fonte: Elaborado pelos autores, 2025.

6.4 ENTIDADES DO SISTEMA

Pensar cuidadosamente nas entidades que compõem um sistema é uma etapa essencial para iniciar o desenvolvimento de qualquer projeto. No contexto da mentoria de TCC, as entidades representam os principais elementos que terão participação ativa dentro da aplicação, como orientadores, orientandos, projetos e mensagens. Cada uma dessas entidades possui características específicas, chamadas de atributos, que ajudam a identificá-las e diferenciá-las no funcionamento geral do sistema.

De forma prática, uma entidade é uma representação clara de algo do mundo real que possui relevância para o funcionamento do sistema. Ao defini-las, estamos mapeando tudo aquilo que precisa ser controlado, registrado ou manipulado no ambiente digital da plataforma. Essas definições são fundamentais para a organização da estrutura interna do projeto.

A identificação das entidades permite não apenas dar início à modelagem do banco de dados, onde cada entidade será transformada em uma tabela com colunas correspondentes aos seus atributos, mas também direciona o desenvolvimento das demais camadas do sistema. No front-end, por exemplo, as entidades são o ponto de partida para criar interfaces e componentes que respeitem as regras de negócio, permitindo uma navegação intuitiva para orientadores e orientandos.

Já no back-end, essas entidades serão representadas por objetos ou classes responsáveis por interagir com o banco de dados, manipulando as informações conforme as operações solicitadas. Dessa forma, elas se tornam o elo entre o armazenamento dos dados e a lógica da aplicação.

Após uma análise detalhada dos processos envolvidos na plataforma, foram definidas as seguintes entidades como base para o desenvolvimento do sistema:

- Usuário;
- Mensagem;
- Projeto;
- Inteligência Artificial;

Essas entidades serão utilizadas em toda a estrutura da aplicação, desde a modelagem de dados até os diagramas UML, assegurando que cada parte do sistema esteja alinhada com os objetivos do projeto e suas funcionalidades específicas.

6.5 REQUISITOS FUNCIONAIS

Os requisitos funcionais descrevem o comportamento esperado do sistema, especificando de que forma ele deve operar para satisfazer as demandas dos usuários.

Esses requisitos são perceptíveis ao usuário, pois correspondem às funcionalidades acessadas por meio da interface da aplicação. Abaixo, são listados os requisitos funcionais levantados:

[RF001] CADASTRAR USUÁRIO

Descrição do caso de uso: O sistema deverá permitir o cadastro de alunos e de coordenadores, com cada um desses cadastros com as devidas informações necessárias a serem fornecidas.

Prioridade: Alta **Usabilidade:** Normal **Desempenho:** Alto

Situação: Incluído **Segurança:** Alta

[RF002] AUTENTICAR USUÁRIO

Descrição do caso de uso: O sistema deverá permitir o cadastro de alunos e de coordenadores, com cada um desses cadastros com as devidas informações necessárias a serem fornecidas.

Prioridade: Alta **Usabilidade:** Normal **Desempenho:** Alto

Situação: Incluído **Segurança:** Alta

[RF003] REALIZAR LOGIN E LOGOFF

Descrição do caso de uso: O sistema deverá fornecer em sua interface a possibilidade do aluno/coordenador conseguir se conectar e desconectar da aplicação.

Prioridade: Normal **Usabilidade:** Normal **Desempenho:** Normal

Situação: Incluído **Segurança:** Normal

[RF004] RECUPERAR SENHA

Descrição do caso de uso: O sistema deverá permitir iniciar o processo de recuperação de senha por e-mail e permitir a redefinição segura após validações.

Prioridade: Alta **Usabilidade:** Alta **Desempenho:** Normal

Situação: Incluído **Segurança:** Alta

[RF005] CONFIRMAR E-MAIL

Descrição do caso de uso: Após o cadastro, o sistema deverá enviar um link de confirmação por e-mail e ativar a conta do usuário mediante validação do token.

Prioridade: Alta **Usabilidade:** Normal **Desempenho:** Normal

Situação: Incluído **Segurança:** Alta

[RF006] ATUALIZAR CONTA

Descrição do caso de uso: O sistema deverá permitir que usuários atualizem seus dados pessoais (nome, e-mail, cargo), garantindo flexibilidade e personalização.

Prioridade: Normal **Usabilidade:** Alta **Desempenho:** Normal

Situação: Incluído **Segurança:** Normal

[RF007] ATUALIZAR SENHA

Descrição do caso de uso: O sistema deverá permitir que o usuário altere sua senha de forma segura e criptografada, sem necessidade de recuperação por esquecimento.

Prioridade: Normal **Usabilidade:** Normal **Desempenho:** Normal

Situação: Incluído **Segurança:** Alta

[RF008] DELETAR CONTA

Descrição do caso de uso: O sistema deverá permitir que o usuário solicite a exclusão de sua conta, com confirmação via e-mail antes da remoção definitiva.

Prioridade: Normal **Usabilidade:** Normal **Desempenho:** Normal

Situação: Incluído **Segurança:** Alta

[RF009] ADICIONAR PROJETO

Descrição do caso de uso: O sistema deverá permitir que orientadores ou alunos criem novos projetos, desde que os dados sejam válidos e respeitem as regras (limite de membros, orientador obrigatório).

Prioridade: Alta **Usabilidade:** Alta **Desempenho:** Normal

Situação: Incluído **Segurança:** Alta

[RF010] EDITAR PROJETO

Descrição do caso de uso: O sistema deverá permitir a atualização das informações do projeto, como nome, descrição ou dados adicionais.

Prioridade: Normal **Usabilidade:** Alta **Desempenho:** Normal

Situação: Incluído **Segurança:** Normal

[RF011] CONVIDAR ALUNO PARA PROJETO

Descrição do caso de uso: O sistema deverá permitir o envio de convites para alunos participarem de projetos, validando as regras de limite e vinculação.

Prioridade: Alta **Usabilidade:** Alta **Desempenho:** Normal

Situação: Incluído **Segurança:** Alta

[RF012] ADICIONAR ALUNO AO PROJETO

Descrição do caso de uso: O sistema deverá registrar a associação entre o aluno e o projeto após a aceitação do convite.

Prioridade: Alta **Usabilidade:** Normal **Desempenho:** Normal

Situação: Incluído **Segurança:** Normal

[RF013] REMOVER ALUNO DO PROJETO

Descrição do caso de uso: O sistema deverá permitir a remoção controlada de um aluno do projeto, respeitando a integridade dos dados.

Prioridade: Normal **Usabilidade:** Normal **Desempenho:** Normal

Situação: Incluído **Segurança:** Normal

[RF014] ENVIAR E-MAIL PARA DELETAR PROJETO

Descrição do caso de uso: Antes da exclusão definitiva, o sistema deverá enviar um e-mail de confirmação ao responsável.

Prioridade: Normal **Usabilidade:** Normal **Desempenho:** Normal

Situação: Incluído **Segurança:** Alta

[RF015] DELETAR PROJETO

Descrição do caso de uso: Após a confirmação por e-mail, o sistema deverá excluir completamente o projeto e seus vínculos.

Prioridade: Alta **Usabilidade:** Normal **Desempenho:** Normal

Situação: Incluído **Segurança:** Alta

[RF016] ACESSAR PROJETO POR URL

Descrição do caso de uso: O sistema deverá permitir o acesso direto a um projeto via URL personalizada.

Prioridade: Normal **Usabilidade:** Alta **Desempenho:** Normal

Situação: Incluído **Segurança:** Normal

[RF017] EXIBIR TELA INICIAL

Descrição do caso de uso: O sistema deverá exibir um dashboard com resumo dos projetos, atalhos e informações para o usuário logado.

Prioridade: Normal **Usabilidade:** Alta **Desempenho:** Normal

Situação: Incluído **Segurança:** Normal

[RF018] ENVIAR MENSAGEM NO PROJETO

Descrição do caso de uso: O sistema deverá permitir o envio de mensagens entre os participantes de um projeto, com opção de anexar arquivos PDF.

Prioridade: Alta **Usabilidade:** Alta **Desempenho:** Normal

Situação: Incluído **Segurança:** Normal

[RF019] BUSCAR MENSAGENS DO PROJETO

Descrição do caso de uso: O sistema deverá retornar o histórico de mensagens trocadas entre os participantes de um projeto.

Prioridade: Normal **Usabilidade:** Normal **Desempenho:** Alto

Situação: Incluído **Segurança:** Normal

[RF020] ANALISAR MENSAGEM COM IA

Descrição do caso de uso: O sistema deverá processar a mensagem via IA e retornar uma análise interpretativa para auxiliar a compreensão.

Prioridade: Alta **Usabilidade:** Alta **Desempenho:** Alto
Situação: Incluído **Segurança:** Alta

[RF024] CHATBOX COM IA
Descrição do caso de uso: O sistema deverá permitir interação em tempo real com uma IA, auxiliando dúvidas e ações rápidas.
Prioridade: Alta **Usabilidade:** Alta **Desempenho:** Alto
Situação: Incluído **Segurança:** Alta

6.6 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais estabelecem critérios de qualidade e restrições sob as quais o sistema deve operar, como desempenho, confiabilidade, segurança, usabilidade e compatibilidade, e são distintos dos requisitos funcionais, que definem as tarefas que o sistema deve executar. Esses requisitos, também chamados de “atributos de qualidade”, influenciam diretamente decisões de arquitetura e implantação do sistema (DEVMEDIA, 2024).

No contexto deste projeto, que envolve o desenvolvimento de uma aplicação Web, tornou-se indispensável considerar aspectos não funcionais desde as etapas iniciais. Essa atenção visa garantir que o sistema não apenas funcione corretamente, mas também atenda aos padrões esperados de eficiência, acessibilidade e robustez exigidos pelas diferentes plataformas de uso. A seguir, apresentam-se os requisitos não funcionais definidos para esta aplicação:

[RNF001] SEGURANÇA DE AUTENTICAÇÃO
Descrição do caso de uso: O sistema deverá criptografar senhas e realizar validações para proteger os dados dos usuários.
Prioridade: Alta **Usabilidade:** Normal **Desempenho:** Alto
Situação: Incluído **Segurança:** Alta

[RNF002] ARMAZENAMENTO DE ARQUIVOS

Descrição do caso de uso: Os arquivos PDF enviados nas mensagens devem ser armazenados de forma organizada e segura, com vínculo direto à mensagem e projeto correspondente.

Prioridade: Normal **Usabilidade:** Normal **Desempenho:** Normal

Situação: Incluído **Segurança:** Alta

[RNF003] CONEXÃO COM A INTERNET

Descrição do caso de uso: O sistema deve manter uma conexão de maneira ativa em todos os momentos com a internet para o envio e uso de dados.

Prioridade: Alta **Usabilidade:** Normal **Desempenho:** Normal

Situação: Incluído **Segurança:** Média

6.7 DIAGRAMAS UML (LINGUAGEM DE MODELAGEM UNIFICADA)

Com o mapeamento das entidades do sistema concluído, iniciamos a elaboração dos diagramas UML. Conforme explicado pelo Lucidchart (2024), a UML é uma linguagem de modelagem visual padronizada que combina riqueza semântica e sintática, sendo amplamente utilizada para projetar, documentar e implementar sistemas de software complexos.

Esses diagramas se baseiam no paradigma de Programação Orientada a Objetos (POO), que permite representar entidades do mundo real como objetos de software. Essa abordagem traz vantagens significativas ao desenvolvimento, incluindo maior facilidade de manutenção, melhor escalabilidade e a promoção de práticas de codificação mais robustas e eficientes.

Como existem vários tipos de diagramas UML que podem ser criados para servir como uma ferramenta para o desenvolvimento do software, utilizou-se neste projeto três destes diagramas, sendo eles:

- **Diagrama de Caso de Uso**: Representação visual que demonstra a interação entre usuários (atores) e as funcionalidades do sistema. Mapeia os principais cenários de uso e requisitos funcionais de forma clara e objetiva.
- **Diagrama de Classe**: Utilizado para ilustrar a estrutura de um software, exibindo todas as classes do sistema, atributos, métodos e as relações entre as classes estabelecidas.
- **Diagrama de Entidade e Relacionamento**: É uma representação gráfica que é utilizada no design de banco de dados, sendo usada para modelar os dados e as relações entre eles.

No próximo tópico, será detalhado a definição de cada um desses diagramas e sua importância para o desenvolvimento da aplicação, dando ênfase maior para cada um deles e suas devidas explicações e conceitos.

6.7.1 DIAGRAMAS DE CASO DE USO

Na linguagem de modelagem unificada (UML), o diagrama de caso de uso resume os detalhes dos usuários do seu sistema (também conhecidos como atores) e as interações deles com o sistema. (LUCIDCHART, 2021).

Essa é uma ferramenta muito importante para a modelagem dos requisitos do software, sendo uma representação gráfica simples e intuitiva, facilitando o entendimento das funcionalidades oferecidas pelo sistema e deixando claro quais caminhos os desenvolvedores devem percorrer.

Os diagramas de caso de uso são representados por diversas formas,mas seus elementos mais consciente estes são:

- **Atores**: São representados por bonecos no estilo palito, representando os usuários ou sistemas externos que irão interagir com o sistema.
- **Caso de Uso**: Representado por um círculo (ou elipse), são utilizadas para representar uma sequência específica de ações e interações dos atores com o sistema.

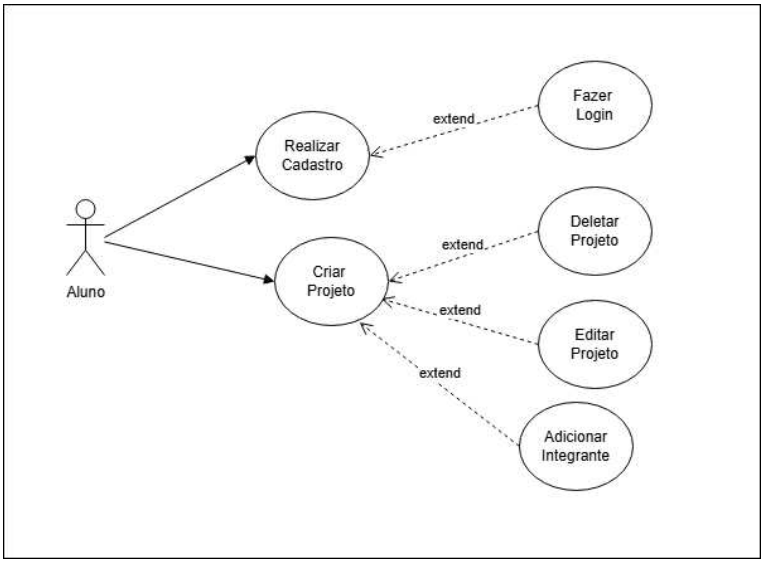
- **Relações**: As relações podem ser associações (linhas entre atores e casos de uso), generalização (definir hierarquias entre os atores),relações como inclusão (include),quando uma ação ocorre quando o ator faz algo e extensão (extend), quando o ator pode fazer uma ação.

A criação deste diagrama foi o primeiro passo para o desenvolvimento da aplicação, servindo como referência de como funcionaria diversas das funcionalidades do software, os atores identificados foram:

- **Aluno**: Se o usuário escolher se cadastrar como aluno, ele poderá criar e editar as propriedades do projeto (TCC) assim como seus integrantes, podendo apenas estar em 1 (um) projeto.
- **Coordenador**: Coordena o projeto, assume uma posição mais “passiva” em relação ao projeto, não pode deletar ou alterar as propriedades do projeto, mas pode estar em múltiplos projetos ao mesmo tempo.

Para fazer o diagrama de caso de uso, utilizamos a ferramenta Draw IO, sendo uma ferramenta simples e fácil de usar para a criação de diagramas UML de maneira rápida e efetiva, abaixo está disponível as representações gráficas dos diagramas, detalhando todos os processos envolvidos.

Figura 4 – Caso de uso aluno.



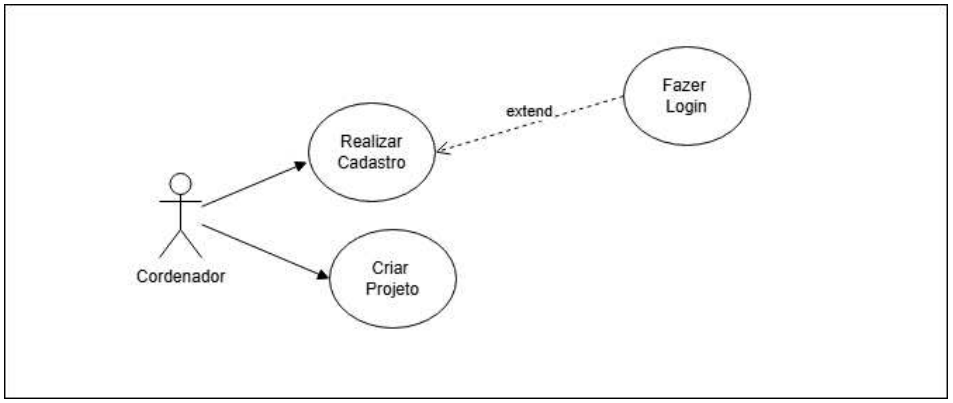
Fonte: Elaborado pelos autores, 2025.

A figura acima descreve um dos diagramas de caso de uso, especificamente focado no aluno. O diagrama apresenta dois casos de uso principais e suas extensões. Conforme demonstrado na imagem a seguinte especificação e detalhamento dele é descrito abaixo:

- Criar Projeto: Este caso de uso permite que o usuário crie um projeto.
- Deletar Projeto: Este caso de uso estendido permite que o aluno (junto da maioria dos integrantes) exclua o projeto, seja por ele ter terminado ou por que não avançou.
- Editar Projeto: Este caso de uso estendido permite que os alunos editem o projeto, caso ele mude de tema.
- Adicionar membro: Este caso de uso estendido permite que os alunos adicionem outro membro ao projeto respeitando o limite máximo de alunos.
- Realizar Cadastro: Este caso de uso estendido permite que os usuários se cadastrem no Sistema.
- Fazer login: Este caso de uso inclui o login.

Cada uma das extensões é representada por uma linha pontilhada no diagrama, indicando que são comportamentos adicionais que se estendem dos casos de uso principais, e que podem ser executados dependendo da necessidade do aluno.

Figura 5 – Caso de uso coordenador.



Fonte: Elaborado pelos autores, 2025.

A figura acima descreve o diagrama de caso de uso para o Coordenador no sistema. O diagrama apresenta dois casos de uso principais, um com uma extensão Conforme demonstrado na imagem a seguinte especificação e detalhamento dele é descrito abaixo:

- Criar Projeto: Este caso de uso permite que o coordenador crie vários projetos
- Realizar Cadastro: Este caso de uso estendido permite que os usuários se cadastrem no Sistema.
- Fazer login: Este caso de uso inclui o login.

As extensões são representadas por linhas pontilhadas no diagrama, indicando que são comportamentos adicionais aos casos de uso principais e podem ser acionados conforme necessário.

6.7.2 DIAGRAMA DE CLASSES

Responsáveis por estabelecer todas as classes do sistema, seus atributos e as relações entre elas, os diagramas de classes são ferramentas essenciais no processo de desenvolvimento de software. Eles facilitam a compreensão da estrutura do sistema e funcionam como guia para implementação. Segundo Booch, Rumbaugh e Jacobson (2005), “um diagrama de classes descreve a estrutura de um sistema, mostrando suas classes, atributos, operações e os relacionamentos entre objetos.” A partir dessa perspectiva, é possível perceber como esses diagramas promovem uma visão clara do sistema, favorecendo tanto o planejamento quanto a codificação.

Esses diagramas oferecem diversos benefícios, como a visualização da modelagem de dados, uma melhor compreensão da arquitetura geral da aplicação, e uma representação gráfica que orienta a programação e a integração entre os componentes do sistema.

Para a construção das classes presentes neste sistema, utilizamos como base o diagrama de caso de uso previamente elaborado, adaptando e expandindo as entidades conforme a necessidade da aplicação. As principais classes identificadas foram:

- Usuário: Classe central do sistema, que representa tanto alunos quanto orientadores. Possui atributos como id, nome, cargo, endereçoEmail e senha. Entre os métodos associados estão: autenticar(), atualizarConta(), deletarConta(), criarProjeto(), entrarChat() e entrarChatBox() com a IA.

- Projeto: Representa os trabalhos em desenvolvimento. Seus atributos incluem id, nome, descrição, tipo e projetoURL. Está associada a vários usuários e permite métodos como editarProjeto(), deletarProjeto() e convidarUsuario().

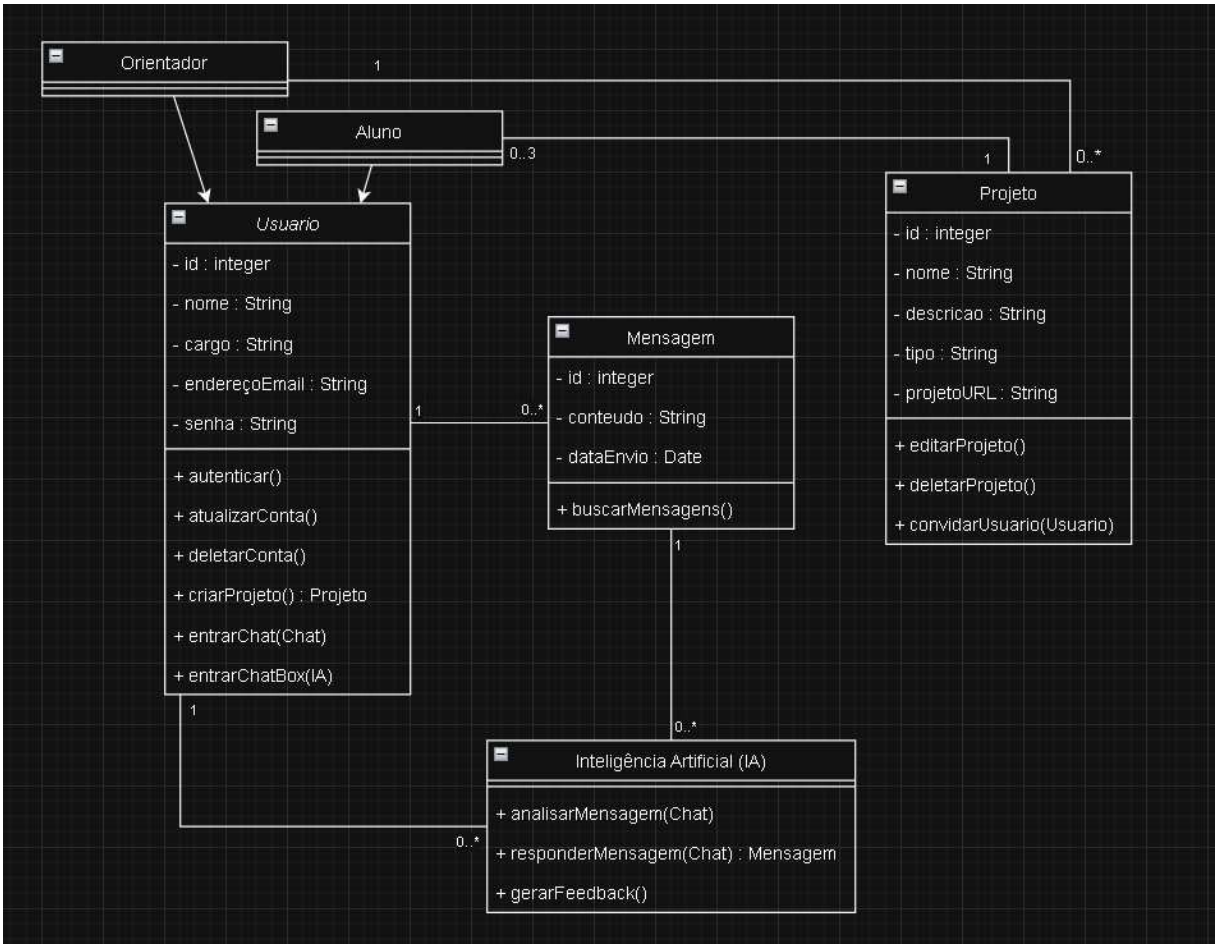
- Mensagem: Responsável por armazenar as conversas entre usuários ou com a inteligência artificial, contendo os atributos id, conteúdo e dataEnvio. Possui a operação buscarMensagens() e está associada diretamente tanto ao Usuário quanto à IA.

- Inteligência Artificial (IA): Classe voltada para análise automatizada das mensagens. Tem os métodos analisarMensagem(), responderMensagem() e gerarFeedback(), atuando com várias mensagens ao mesmo tempo.

- Aluno e Orientador: Representam os papéis distintos de usuários no sistema. Um orientador pode estar vinculado a até três alunos, refletindo a multiplicidade 0..3.

O diagrama de classes foi desenvolvido utilizando a ferramenta Draw.io, que permitiu a representação visual clara das relações e operações entre os elementos do sistema.

Figura 6 - Diagrama de Classes



Fonte: Elaborado pelos autores, 2025

6.7.3 DIAGRAMA DE ENTIDADE E RELACIONAMENTO

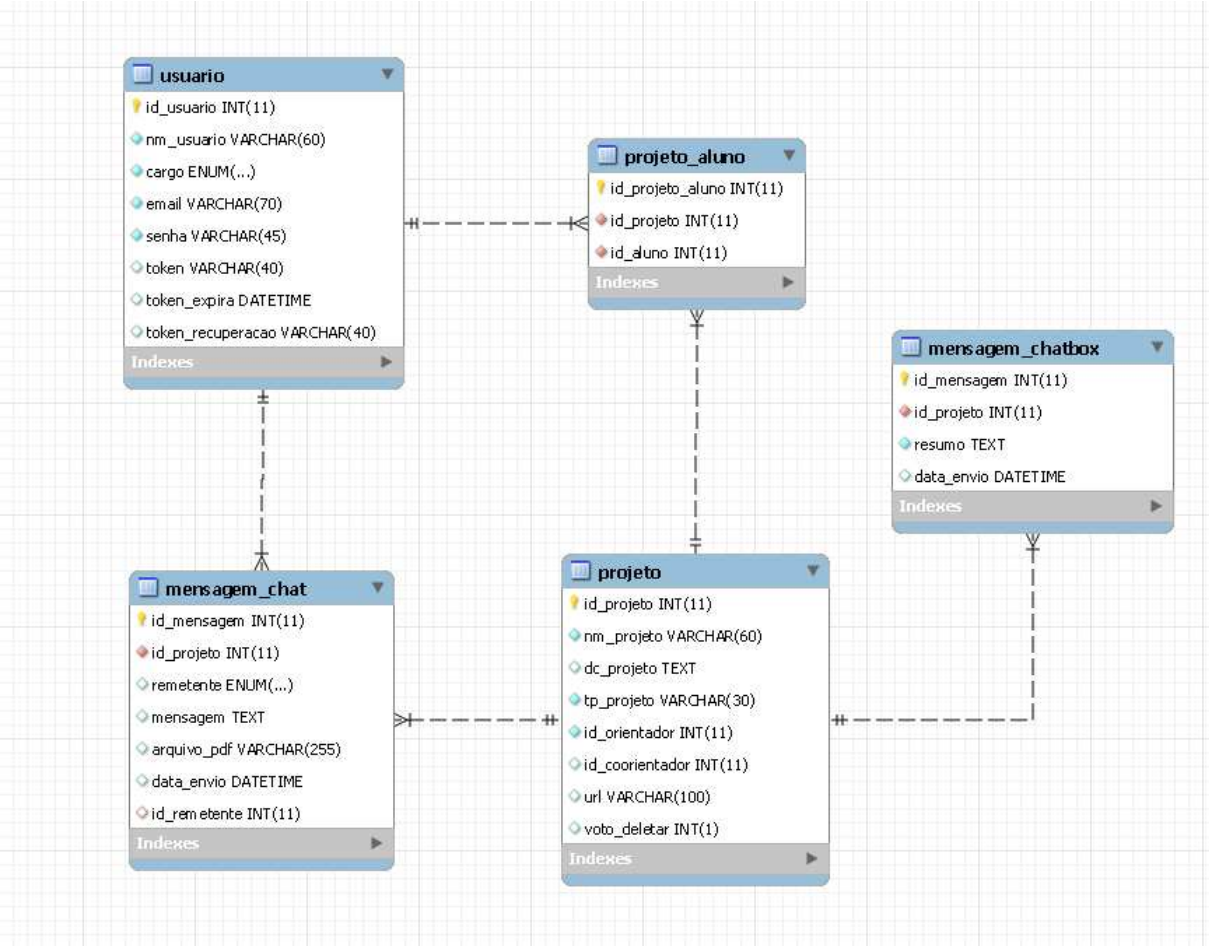
Segundo Pressman e Maxim (2021), os diagramas de entidade e relacionamento (DER) são ferramentas essenciais no processo de modelagem de dados, permitindo representar visualmente como os elementos de um sistema se conectam em um banco de dados. Eles são amplamente utilizados durante o planejamento e análise de sistemas, especialmente na construção de bancos de dados relacionais, e contam com símbolos padronizados para identificar entidades, seus atributos e os vínculos entre elas.

Os DERs são formados por três componentes principais: entidades, atributos e relacionamentos. Cada um tem um papel crucial no entendimento da estrutura dos dados. De forma resumida:

- Entidades: São os objetos ou elementos principais do sistema, que possuem existência própria. Exemplos incluem "Cliente", "Produto" ou "Pedido".
- Atributos: São as propriedades que descrevem cada entidade, como "CPF" para um Cliente ou "Preço" para um Produto.
- Relacionamentos: Mostram como as entidades estão conectadas entre si. Eles podem ser:
    - Um para Um (1:1): Uma entidade se relaciona com apenas uma instância da outra.
    - Um para Muitos (1:N): Uma entidade se conecta a várias instâncias da outra.
    - Muitos para Muitos (N:N): Várias instâncias de uma entidade estão associadas a várias instâncias da outra.

Para a criação desse diagrama, é utilizada a ferramenta MySQL Workbench, que oferece não apenas recursos para modelagem de dados, mas também funcionalidades para o desenvolvimento completo do banco de dados. Trata-se de um software bastante versátil e amplamente utilizado na área. Na figura abaixo, é apresentada a representação gráfica do diagrama de entidade e relacionamento.

Figura 7 - Diagrama de Entidade e Relacionamento



Fonte: Elaborado pelos autores, 2025.

Abaixo é possível visualizar o dicionário de dados correspondente ao nosso Diagrama de Entidade e Relacionamento. De acordo com Silva e Santos (2022), um dicionário de dados é “um repositório central que contém definições e descrições detalhadas sobre os elementos de dados, incluindo seus significados, formatos, origem, relacionamentos e formas de utilização.” Ele serve como base para garantir a consistência e o entendimento claro das informações no desenvolvimento de sistemas.

Tabela 1 - Dicionário de Dados da Tabela de Usuários

Tabela	Usuário			
Descrição	Armazena todos os usuários cadastrados			
Observações				
CAMPOS				
NOME	**DESCRIÇÃO**	**TIPO DE DADO**	**TAMANHO**	**RESTRIÇÕES**
id_usuario	Código de identificação do usuário	Int	11	PK / Auto Increment
nm_usuario	Nome do usuário	Varchar	60	Not Null
cargo	Função do Usuario (Aluno ou Orientador)	Enum	ALUN, ORIE	Not Null
email	Email do usuario	Varchar	70	Not Null
senha	Senha do Usuário	Varchar	45	Not Null
token	Token para ativação da conta pelo email	Varchar	40	Defautl
token_expira	Validade para o token de ativação de conta	Datetime		Default
token_recuperacao	Token para recuperação de senha	Varchar	40	Default

Tabela 2 - Dicionário de Dados da Tabela de Projetos

Tabela	Projeto			
Descrição	Armazena todos os projetos cadastrados porusuários			
Observações				
CAMPOS				
NOME	**DESCRIÇÃO**	**TIPO DE DADO**	**TAMANHO**	**RESTRIÇÕES**
id_projeto	Código de identificação de cada projeto	Int	11	PK / Auto Increment
nm_projeto	Nome do projeto	Varchar	60	Not Null / Unique
dc_projeto	Descrição do projeto	Text		Default
tp_projeto	Tipo do projeto (Relatório Técnico, Monográfia ou Artigo)	Varchar	30	Not Null
id_orientador	Identificador do orientador associado ao projeto	Int	11	FK
url	Url referente ao projeto	Varchar	100	Default
voto_deletar	Votos para deletar o projeto	Int	1	Default

Tabela 3 - Dicionário de Dados da Tabela de Alunos por Projeto

Tabela	projeto_aluno			
Descrição	Armazena a conexão dos alunos com os projetos			
Observações	Possui duas chaves estrangeiras, uma de usuario e outra de projeto.			
CAMPOS				
NOME	**DESCRIÇÃO**	**TIPO DE DADO**	**TAMANHO**	**RESTRIÇÕES**
id_projeto_aluno	Identificador da ponte aluno/projeto	Int	11	PK / Auto Increment
id_projeto	Identificador do projeto	Int	11	FK
id_aluno	Identificador do usuario aluno	Int	11	FK

Tabela 4 - Dicionário de Dados da Tabela de Mensagens com a IA

Tabela	mensagem_chatbox			
Descrição	Armazena todos as mensagens com a Inteligência Artificial			
Observações	Possui uma chave estrangeira, da tabela projeto			
CAMPOS				
NOME	**DESCRIÇÃO**	**TIPO DE DADO**	**TAMANHO**	**RESTRIÇÕES**
id_mensagem	Identificador da mensagem	Int	11	PK / Auto Increment
id_projeto	Identificador do projeto	Int	11	FK
resumo	Resumo da ultima mensagem enviada	Text		Not Null
data_envio	Data de envio da mensagem	Datetime		Default

Tabela 5 - Dicionário de Dados da Tabela de Mensagens com o Orientador

Tabela	mensagem_chat			
Descrição	Armazena todos as mensagens com o orientador			
Observações	Possui duas chaves estrangeiras, uma de usuario e outra de projeto.			
CAMPOS				
NOME	**DESCRIÇÃO**	**TIPO DE DADO**	**TAMANHO**	**RESTRIÇÕES**
id_mensagem	Identificador da mensagem	Int	11	PK / Auto Increment
id_projeto	Identificador do projeto	iNT	11	FK
remetente	Cargo do remetente	Enum	ALUN, ORIE, IA	Default
mensagem	Conteúdo da mensagem	Text		Default
arquivo_pdf	Arquivo PDF	Varchar	255	Default
data_envio	Data de envio da mensagem	Datetime		Default
id_remetente	Identificador do remetente	Int	11	FK

6.7.4 DIAGRAMA DE ATIVIDADE

Continuando a modelagem do sistema, o Diagrama de Atividade foi elaborado para detalhar o fluxo de trabalho de funcionalidades específicas. Este diagrama é um dos mais importantes da UML para representar a dinâmica do sistema. Segundo Guedes (2011), o diagrama de atividade "descreve os passos a serem percorridos para a

conclusão de uma atividade, mostrando o fluxo de controle entre eles". Em outras palavras, ele funciona como um fluxograma que ilustra a sequência de ações e decisões, desde o início até a conclusão de um processo.

A principal importância do diagrama de atividade reside na sua capacidade de modelar a lógica de negócio de forma clara e visual. Ele ajuda a equipe de desenvolvimento a compreender os passos necessários para executar uma funcionalidade, identificar condições, laços de repetição e atividades que podem ocorrer em paralelo. Essa visualização detalhada é fundamental para traduzir os requisitos de um caso de uso em uma lógica de implementação concreta e livre de ambiguidades.

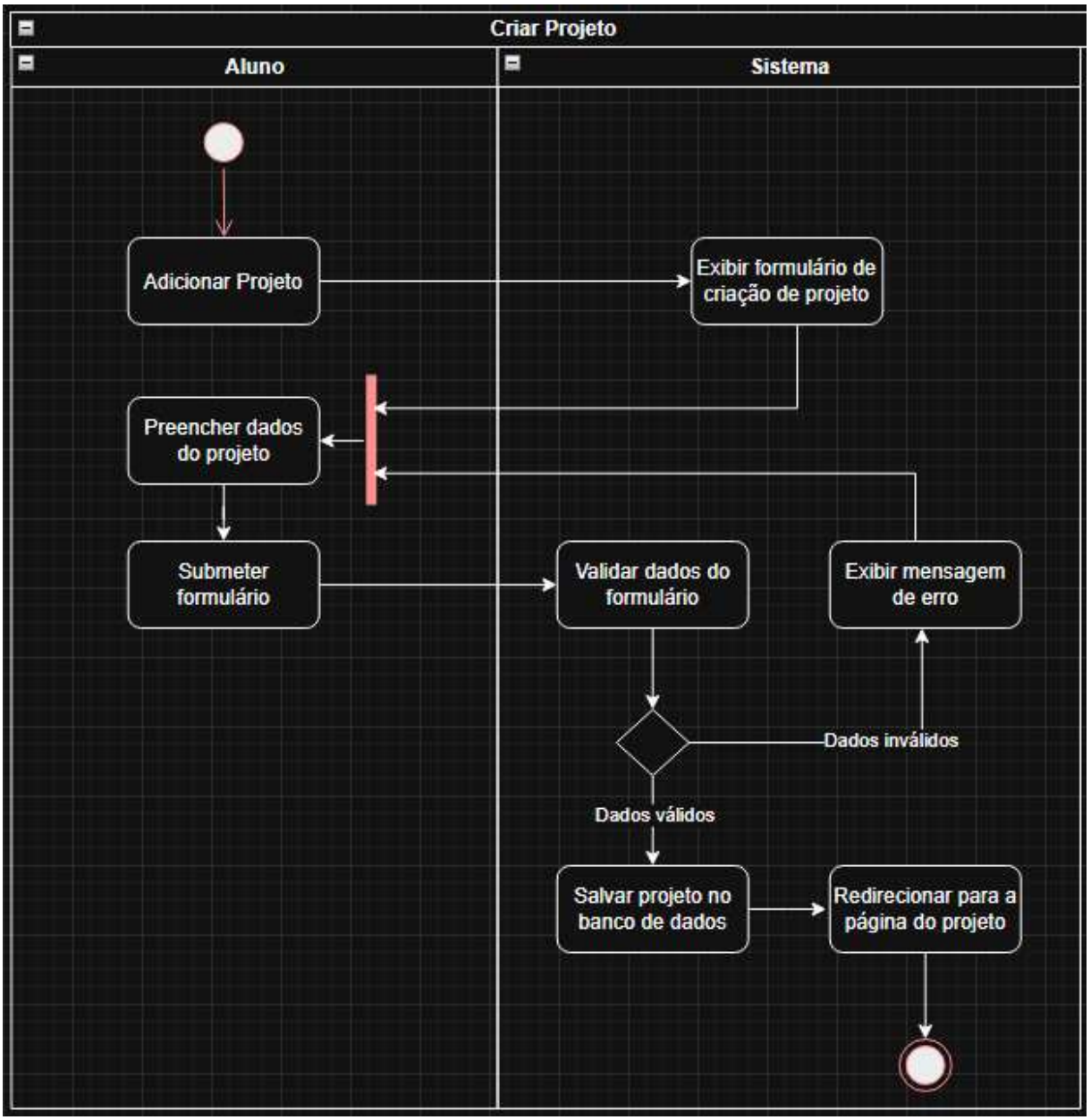
Os diagramas de atividade são compostos por um conjunto de símbolos padronizados para representar o fluxo, dos quais os principais são:

- Nó Inicial e Nó Final**:** Representados por um círculo preenchido e um círculo preenchido dentro de outro, respectivamente. Indicam o começo e o fim do fluxo.
- Ação**:** Representada por um retângulo com cantos arredondados, descreve uma tarefa ou passo a ser executado pelo sistema ou por um ator.
- Fluxo de Controle: Setas que conectam as ações, indicando a ordem em que ocorrem.
- Nó de Decisão e Nó de Mesclagem: Ambos representados por um losango. O nó de decisão ramifica o fluxo com base em uma condição, enquanto o nó de mesclagem unifica diferentes fluxos em um só.
- Nós de Bifurcação (Fork) e Junção (Join): Barras que dividem o fluxo em atividades paralelas (fork) ou sincronizam múltiplos fluxos em um único (join).
- Raias (Swimlanes): Divisões no diagrama (geralmente colunas) que agrupam atividades de acordo com o responsável por executá-las, como "Aluno", "Sistema" ou "Coordenador".

Para este projeto, o diagrama de atividade foi utilizado para detalhar o fluxo do caso de uso "Criar Projeto", uma das funcionalidades centrais do sistema. A sua elaboração permitiu mapear cada etapa, desde a interação inicial do usuário até a

confirmação final pelo sistema, garantindo que todas as validações e processos fossem considerados. A ferramenta Draw.io foi novamente utilizada para a criação deste diagrama, dada a sua flexibilidade.

Figura 8 - Diagrama de Atividade



Fonte: Elaborado pelos autores, 2025.

O diagrama acima ilustra o fluxo de trabalho para a criação de um novo projeto, utilizando raias para distinguir as ações realizadas pelo Aluno e as processadas pelo Sistema. A seguir, a descrição detalhada do fluxo, conforme representado na imagem:

1. O fluxo se inicia na raia "Aluno", que realiza a ação "Adicionar Projeto".
2. Em resposta, o Sistema executa a ação "Exibir formulário de criação de projeto".
3. O controle retorna ao Aluno, que realiza as ações de "Preencher dados do projeto" e, em seguida, "Submeter formulário".
4. Após a submissão, o Sistema inicia a "Validar dados do formulário". Um nó de decisão verifica se os dados são válidos.
5. Se a condição for "Dados inválidos", o sistema executa a ação "Exibir mensagem de erro" e o fluxo retorna para a etapa "Preencher dados do projeto", criando um ciclo para que o aluno corrija as informações.
6. Se a condição for "Dados válidos", o fluxo segue para a ação "Salvar projeto no banco de dados".
7. Após salvar, o sistema executa a etapa final "Redirecionar para a página do projeto".
8. O processo é concluído, e o fluxo atinge o nó de atividade final.

6.7.5 DIAGRAMA DE SEQUÊNCIA

Dando continuidade à modelagem do sistema, foi elaborado o **Diagrama de Sequência** com o objetivo de representar, de maneira temporal e detalhada, as interações entre os atores e o sistema durante a execução da funcionalidade "Adicionar Projeto".

Segundo Guedes (2011), o diagrama de sequência é utilizado para "mostrar como os objetos interagem por meio do envio de mensagens ao longo do tempo". Em

outras palavras, ele demonstra **quem faz o quê, em qual ordem**, e como essas ações se comunicam dentro de um determinado caso de uso.

No diagrama representado, temos dois participantes principais:

- **Aluno**: ator responsável por iniciar a ação de adicionar um projeto.

- **Sistema**: entidade que responde às ações do aluno com funcionalidades e validações internas.

A sequência inicia quando o **Aluno** aciona a opção **"Adicionar Projeto"**. Em resposta, o **Sistema** exibe o formulário de criação. Após o preenchimento dos dados pelo aluno, os dados são enviados para o sistema, que realiza uma **validação**.

Esse momento é representado por um **nó de decisão**, indicando que o fluxo pode seguir por dois caminhos distintos:

- Se os **dados forem válidos**, o sistema executa a ação de **salvar o projeto**.

- Caso contrário, o sistema responde com a **exibição de uma mensagem de erro**, retornando o fluxo ao usuário.

Esse tipo de diagrama é fundamental para compreender as **trocas de mensagens** entre o usuário e o sistema, evidenciando a ordem cronológica dos eventos e o comportamento esperado da aplicação. Ele ajuda a equipe de desenvolvimento a identificar responsabilidades, dependências entre objetos e validar a lógica de comunicação entre camadas do sistema.

Elementos Utilizados no Diagrama:

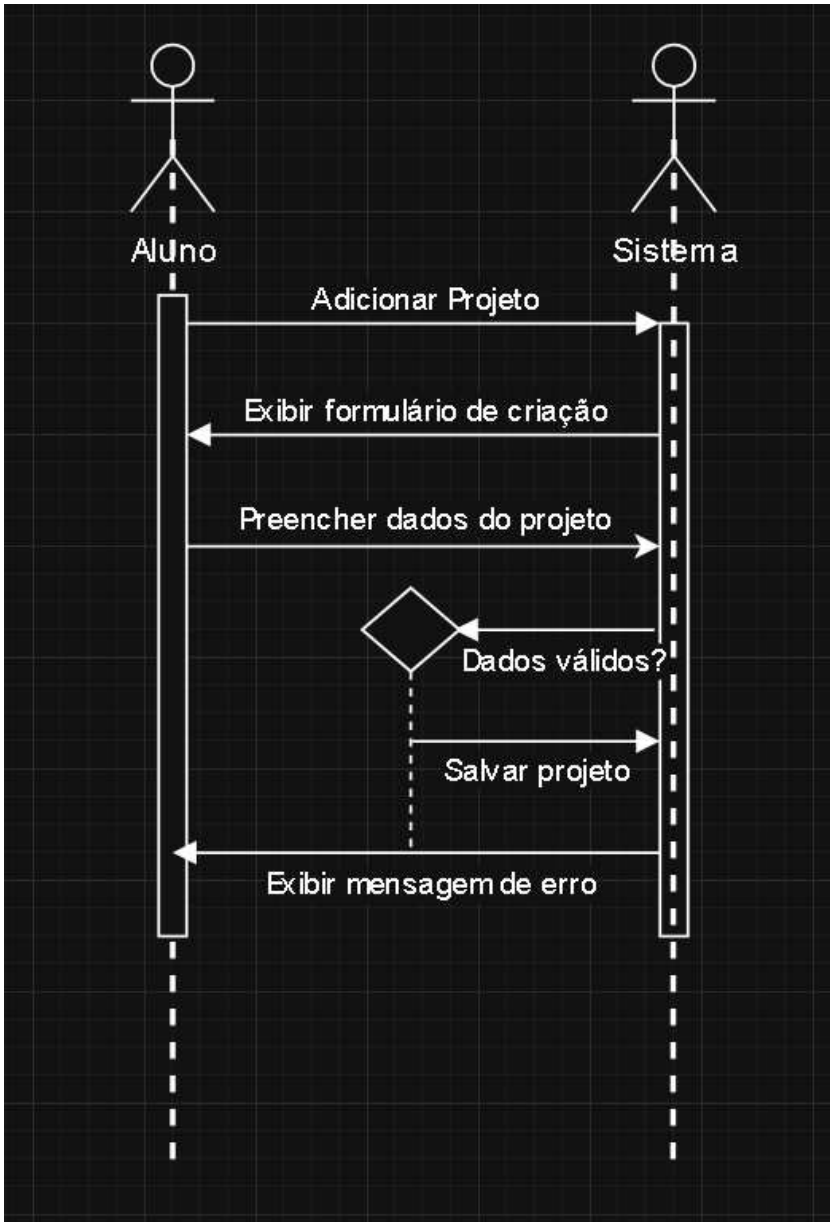
- **Atores**: Representados como bonecos com lifelines (linhas verticais).

- **Mensagens síncronas**: Linhas com setas indicando chamadas diretas de um participante ao outro.

- **Condições (Decisão)**: Representadas por losangos, mostram bifurcações no fluxo com base em condições lógicas.

- **Atividades do sistema**: Mostradas como ações reativas ao input do usuário, como exibir formulário, validar dados ou salvar.

A construção foi realizada utilizando a ferramenta Draw.io, permitindo uma visualização clara das interações e facilitando a comunicação entre os membros da equipe de desenvolvimento.


Aluno
Sistema
Adicionar Projeto
Exibir formulário de criação
Preencher dados do projeto
Dados válidos?
Salvar projeto
Exibir mensagem de erro

7 TECNOLOGIAS E FERRAMENTAS

Aqui são detalhados os principais recursos tecnológicos empregados no desenvolvimento da plataforma. Desde soluções de infraestrutura até linguagens de programação, cada componente foi selecionado para otimizar desempenho, segurança e usabilidade ao longo do ciclo de vida do projeto.

7.1 FIGMA

O Figma se consolidou como uma das principais ferramentas de design de interface e prototipagem, destacando-se por sua abordagem colaborativa e funcionamento baseado em nuvem. Sua arquitetura em navegador elimina a necessidade de instalação local, proporcionando acessibilidade imediata e compatibilidade multiplataforma. No contexto do projeto, o Figma foi adotado como solução central para a criação de protótipos de alta fidelidade Sua capacidade de edição simultânea permitiu que toda a equipe participasse ativamente do processo criativo, facilitando a visualização e interação de ideias em tempo real. Os diferenciais da ferramenta - particularmente sua natureza colaborativa nativa e a independência de sistemas operacionais - otimizaram significativamente o fluxo de trabalho. Essa escolha tecnológica não apenas agilizou o desenvolvimento das interfaces, mas também promoveu uma integração mais eficiente entre os membros da equipe, resultando em um processo de design mais dinâmico e alinhado com os objetivos do projeto.

Figura 9 - Figma

Fonte: Figma, 2022.

7.2 GITHUB

O GitHub se estabeleceu como a principal plataforma de hospedagem de código-fonte baseada no sistema Git. Desde seu lançamento em 2008, tornou-se essencial para desenvolvedores, oferecendo um ambiente completo para gestão de projetos e trabalho em equipe. Sua arquitetura permite o controle preciso de versões enquanto facilita a colaboração entre programadores, seja em projetos open source ou corporativos.

A plataforma se destaca por funcionalidades como pull requests, que permitem a revisão sistemática de código, e branches, que possibilitam o desenvolvimento paralelo de funcionalidades. O sistema de Issues ajuda no gerenciamento de tarefas, enquanto o GitHub Actions automatiza processos de integração contínua. Essas ferramentas transformaram a maneira como equipes desenvolvem software, tornando o fluxo de trabalho mais eficiente e organizado.

No contexto deste projeto, o GitHub foi fundamental para manter a integridade do código-fonte. Todos os membros da equipe trabalharam em um repositório central, onde podiam enviar suas contribuições, revisar alterações e resolver

conflitos de forma sistemática. A integração nativa com ambientes de desenvolvimento como o Visual Studio Code agilizou operações diárias, desde commits até merges.

Como observou Linus Torvalds, criador do Git, o verdadeiro valor do controle de versão está na eficiência colaborativa. O GitHub incorpora esse princípio, tendo sido decisivo para a qualidade e organização do nosso processo de desenvolvimento. A plataforma não apenas armazenou nosso código, mas estruturou toda a metodologia de trabalho da equipe.

Figura 10 - Github

Fonte: LogosWorld, 2020.

7.3 VISUAL STUDIO CODE

Desenvolvido pela Microsoft e lançado em 2015, o Visual Studio Code (VS Code) rapidamente se estabeleceu como um dos editores de código mais populares do mundo. Sua combinação única de leveza, versatilidade e poder o torna ideal para desenvolvedores em qualquer linguagem ou plataforma.

O verdadeiro diferencial do VS Code está em seu ecossistema de extensões. Com milhares de opções disponíveis, desde formatadores como Prettier até ferramentas de linting como ESLint, o editor pode ser completamente personalizado

para atender às necessidades específicas de cada projeto. Essa flexibilidade, combinada com recursos nativos como terminal integrado, depurador poderoso e suporte completo ao Git, cria um ambiente de desenvolvimento excepcionalmente produtivo.

Durante este projeto, o VS Code provou ser indispensável. Seus recursos de IntelliSense aceleraram significativamente a escrita de código, enquanto a integração direta com o GitHub simplificou todo o fluxo de versionamento - desde commits até a resolução de conflitos. O editor também facilitou a navegação em bases de código complexas através de seu sistema de busca e referências cruzadas.

Como bem observou Satya Nadella, CEO da Microsoft, ferramentas que colocam o desenvolvedor no centro são o futuro da programação. O VS Code exemplifica essa visão, tendo sido fundamental para manter nossa produtividade alta e nosso fluxo de trabalho organizado em todos os estágios do projeto. Seu desempenho consistente, seja para edição rápida ou desenvolvimento intensivo, confirmou por que se tornou o editor preferido de tantos profissionais.

Figura 11 - Visual Studio Code

Fonte: Visual Studio Code, 2021.

7.4 MYSQL

Como um dos sistemas de gerenciamento de banco de dados mais consolidados no mercado, o MySQL oferece uma combinação poderosa de desempenho, confiabilidade e facilidade de uso. Desenvolvido como software de código aberto e mantido pela Oracle, este SGBD relacional se tornou fundamental para aplicações que demandam estruturação rigorosa de dados.

Sua arquitetura baseada em tabelas relacionais provou ser ideal para nosso projeto, permitindo modelar com precisão as entidades do sistema e seus relacionamentos complexos. Implementamos estruturas otimizadas para usuários, projetos e interações, garantindo tanto a integridade dos dados quanto a eficiência nas operações. O suporte nativo a transações ACID foi particularmente valioso para manter a consistência das informações em operações críticas.

A integração com nossa stack tecnológica, especialmente com o back-end em Node.js, ocorreu de maneira fluida através de bibliotecas especializadas. Essa conexão eficiente entre as camadas da aplicação permitiu operações rápidas e seguras no banco de dados. Complementamos o ambiente com o MySQL Workbench, que proporcionou uma interface visual para administração, modelagem e otimização de consultas.

A escolha do MySQL se justificou plenamente ao longo do desenvolvimento, atendendo com excelência aos requisitos de desempenho para consultas complexas, segurança no acesso aos dados e escalabilidade para crescimento futuro. Sua maturidade como tecnologia e a vasta documentação disponível aceleraram a implementação e solução de desafios, consolidando-o como componente essencial da nossa arquitetura de dados.

Figura 12 - MySQL

Fonte: MySQL, 2011.

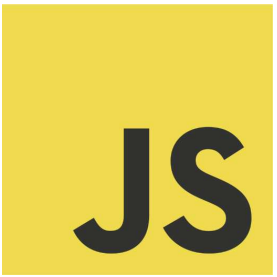
7.5 JAVASCRIPT

O JavaScript foi essencial para desenvolver a interatividade e dinamismo do sistema. Utilizamos a linguagem para criar respostas instantâneas às ações do usuário, validar dados antes do envio ao servidor e conectar a interface com os serviços backend de forma fluida. Sua capacidade de manipular elementos da página em tempo real permitiu construir uma experiência ágil e intuitiva.

A linguagem mostrou toda sua versatilidade ao integrar perfeitamente o frontend com as APIs do sistema. Implementamos lógica para tratamento de eventos, atualizações dinâmicas de conteúdo e comunicação assíncrona eficiente. Como destacou Brendan Eich, o JavaScript realmente atua como a "cola" da web moderna - no nosso projeto, foi fundamental para unir interface, regras de negócio e dados em uma aplicação coesa e responsiva.

A escolha do JavaScript se justifica plenamente, oferecendo o equilíbrio ideal entre performance e produtividade no desenvolvimento. Sua natureza ubíqua na web e constante evolução permitiram implementar soluções robustas mantendo a compatibilidade com todos os dispositivos e navegadores modernos.

Figura 13 - JavaScript



Fonte: World Vector Logo, 2024.

7.6 NODEJS

O Node.js revolucionou o desenvolvimento backend ao trazer o JavaScript para o lado do servidor. Construído sobre o motor V8 do Chrome, ele combina alto desempenho com um modelo não-bloqueante orientado a eventos - perfeito para aplicações que demandam concorrência e escalabilidade.

No projeto, o Node.js foi a espinha dorsal da nossa arquitetura backend. Ele permitiu criar uma API robusta que processa requisições de forma eficiente, integra-se perfeitamente com o banco de dados e mantém a consistência dos dados. Sua natureza assíncrona foi crucial para lidar com múltiplas conexões simultâneas sem comprometer a performance.

A modularidade do Node.js, aliada ao ecossistema npm, nos permitiu estruturar o código em camadas lógicas (controladores, serviços, rotas), facilitando a manutenção e expansão do sistema. A possibilidade de usar JavaScript em todo o stack acelerou o desenvolvimento e reduziu a curva de aprendizado para a equipe.

A escolha do Node.js se mostrou acertada: entregamos uma aplicação rápida, escalável e com tempo de resposta otimizado - tudo aproveitando uma linguagem já

dominada pelos desenvolvedores. Seu modelo de I/O não-bloqueante provou ser ideal para nossas necessidades de concorrência e processamento assíncrono.

Figura 14 - NodeJS

Fonte:Node.js, 2020.

7.7 HTML (EJS)

O HTML provou ser muito mais que simples marcação no nosso projeto - foi a base sobre a qual construímos interfaces ricas e funcionais. Ao adotar o EJS como engine de templates, demos um salto qualitativo, transformando páginas estáticas em componentes dinâmicos que respondem inteligentemente aos dados do servidor.

Essa combinação nos permitiu desenvolver uma arquitetura frontend onde cada elemento tinha propósito claro. Desde a estruturação semântica do conteúdo até a renderização condicional de componentes, o EJS ampliou as possibilidades do HTML tradicional sem sacrificar sua simplicidade. A reutilização de templates para elementos comuns não apenas economizou linhas de código, como garantiu consistência em toda a aplicação.

Como bem destacou Tim Berners-Lee, o HTML é a linguagem que dá forma à informação. No nosso caso, o EJS foi o complemento perfeito para tornar essa informação viva e contextual. Páginas deixaram de ser documentos estáticos para

se tornarem interfaces inteligentes, capazes de se adaptar aos dados e à interação do usuário em tempo real.

A integração perfeita entre EJS e nosso backend Node.js criou um fluxo de dados fluido, onde as atualizações no servidor se refletiam instantaneamente na interface. Essa sinergia resultou em tempos de desenvolvimento mais curtos e uma base de código mais limpa e organizada - prova de que mesmo tecnologias consolidadas como o HTML podem surpreender quando combinadas com as abordagens certas.

Figura 15 - HTML

Fonte: W3C, 2011.

7.8 CSS

O CSS transformou nossa estrutura HTML em uma experiência visual coesa e atraente. Mais do que simples estilização, ele nos permitiu estabelecer uma identidade visual consistente em toda a aplicação, garantindo não apenas beleza, mas usabilidade e acessibilidade.

No projeto, o CSS mostrou sua verdadeira força quando combinado com técnicas modernas. Utilizamos variáveis CSS para criar um sistema de design

unificado, onde cores, espaçamentos e tipografia podiam ser controlados globalmente. Os seletores avançados nos permitiram estilizar componentes de forma precisa, sem prejudicar a semântica do HTML.

A adoção do Flexbox e Grid melhorou nosso layout, tornando-o responsivo por natureza. Elementos que antes complicados agora se organizavam de maneira sólida em qualquer tamanho de tela.

Com a mentalidade certa, o CSS bem aplicado passa despercebido - o usuário só vê o resultado final: interfaces limpas, intuitivas e visualmente harmoniosas. No nosso caso, ele foi o elo entre a estrutura crua do HTML e a experiência polida que entregamos aos usuários, provando que, na web, forma e função andam sempre juntas.

Figura 16 - CSS

Fonte: CSS-Next Community Group, 2025.

7.9 DRAW.IO

O draw.io é uma ferramenta gratuita e online para criação de diagramas, fluxogramas, organogramas, UML e outros modelos visuais.

Foi usado no início do projeto para primeiro fazer algumas telas do protótipo de baixa fidelidade, e assim que a ideia foi se solidificando, fizemos o Diagrama de Caso de Uso e o Diagrama de Classes nele.

Figura 17 - draw.io

Fonte: Wikipedia, 2021.

8 DESENVOLVIMENTO

Neste tópico, são descritas todas as etapas que compõem o desenvolvimento da aplicação TCC Mentor, abordando cada uma de forma detalhada, com ênfase em sua relevância no processo de construção e finalização do sistema. Cada componente do desenvolvimento teve um papel essencial na consolidação do projeto, contribuindo para a sua completude e funcionalidade.

No front-end, as entidades desempenham um papel estruturante, servindo como referência para a elaboração das interfaces e dos componentes visuais da aplicação. Essas interfaces foram projetadas com base nas regras de negócio previamente definidas, assegurando que os usuários finais — alunos e orientadores — possam interagir com o sistema de maneira clara, intuitiva e eficiente.

No back-end, as entidades representam as estruturas de dados que interagem diretamente com o banco de dados, seja no envio ou na recuperação de informações. Elas são modeladas de acordo com a lógica do sistema, funcionando como abstrações fundamentais para o processamento e persistência dos dados.

De forma integrada, essas entidades estão presentes em todos os âmbitos do desenvolvimento da aplicação: no front-end, no back-end, na modelagem do banco de dados, na documentação de requisitos e na construção dos diagramas UML. Sua utilização transversal confere coesão ao projeto e garante a consistência entre os diferentes módulos da aplicação.

8.1 BANCO DE DADOS

Dado que o sistema lida com informações sensíveis e relevantes, que precisam ser armazenadas de forma segura para posterior reutilização, tornou-se imprescindível a adoção de um banco de dados como solução para a gestão desses

dados. Segundo a definição apresentada pela Oracle (2024), um banco de dados é caracterizado como:

> “Uma coleção organizada de informações – ou dados – estruturadas, normalmente armazenadas eletronicamente em um sistema de computador. Um banco de dados é geralmente controlado por um sistema de gerenciamento de banco de dados (DBMS). Juntos, os dados e o DBMS, juntamente com os aplicativos associados a eles, são chamados de sistema de banco de dados, geralmente abreviados para apenas banco de dados.”

Com base nessa definição, evidencia-se a relevância do uso de um sistema de banco de dados para garantir a organização, integridade e acessibilidade dos dados gerados e manipulados pela aplicação. Ao se observar que determinadas entidades do sistema possuíam dependência ou conexão lógica entre si, optou-se pela utilização de um banco de dados relacional, o qual oferece suporte nativo para esse tipo de estrutura.

Conforme descrito pela IBM (2024), os bancos de dados relacionais são projetados para organizar os dados em tabelas compostas por linhas e colunas, onde cada tabela possui um identificador único (chave primária), que permite a criação de relacionamentos com outras tabelas por meio de chaves estrangeiras. Essa abordagem é fundamental para manter a consistência e integridade dos dados em sistemas que exigem múltiplas entidades interconectadas.

Além disso, a mesma fonte destaca a importância da linguagem SQL (Structured Query Language), desenvolvida com o propósito de possibilitar a interação entre os usuários e os sistemas de gerenciamento de banco de dados relacionais. Com o passar do tempo, o uso da SQL se tornou tão difundido e consolidado neste contexto, que muitos bancos relacionais passaram a ser informalmente denominados como “bancos de dados SQL”. Esse fenômeno ilustra a padronização e consolidação do uso da linguagem SQL em soluções que demandam manipulação robusta e eficiente de dados relacionais.

A seguir, é apresentada a representação dos modelos das tabelas do banco de dados:

8.2 BACK-END

O back-end da aplicação é o responsável por fazer a ponte entre o sistema de banco de dados e o ambiente de front-end, sendo essencial para que os usuários consigam visualizar, inserir ou modificar informações por meio da interface. De acordo com a Alura (2023), o back-end é a parte do sistema que gerencia os bastidores da aplicação, lidando com dados, lógica de negócio, autenticação, integrações e outros processos que não são visíveis ao usuário final.

No nosso projeto, o back-end foi desenvolvido usando Node.js, utilizando um conjunto de bibliotecas que ajudaram a facilitar tanto a comunicação com o banco quanto o gerenciamento de requisições e respostas. Entre as tecnologias empregadas, tivemos: Express para montar o servidor e as rotas, MySQL como banco de dados, EJS para renderizar as páginas, Multer para lidar com upload de arquivos, Form-Data para envio de formulários, Express-Session para gerenciar sessões de usuário, Axios para requisições HTTP e Nodemailer para envio de e-mails automatizados.

Segundo o blog da Rocketseat (2022), APIs são um dos principais meios de comunicação entre aplicações web e servidores, permitindo a troca de dados de forma organizada e segura. No nosso caso, mesmo sem utilizar o padrão RESTful, estruturamos rotas específicas para cada funcionalidade do sistema, organizadas de acordo com as entidades presentes. Essas rotas são responsáveis tanto por receber e processar dados enviados pelo front-end, quanto por buscar informações no banco e retorná-las para serem exibidas ao usuário.

Com isso, o back-end assume um papel fundamental para garantir que a aplicação funcione de forma estável, dinâmica e segura, sempre atuando nos bastidores para manter tudo fluindo.

8.2.1 ROTAS DE USUÁRIO

Essas rotas foram criadas com o intuito de alterar dados referentes aos alunos e orientadores cadastrados no sistema. Foram criadas visando melhorar a experiência no front-end. Cada uma delas foi feita para enviar e retornar apenas as informações essenciais para o funcionamento das regras de negócio, organizadas conforme suas funções específicas. Sendo elas:

- **Adicionar Conta**: Essa rota foi criada para permitir que o usuário (seja ele aluno ou orientador) consiga se registrar na plataforma. Todas as informações enviadas passam por validações rigorosas, garantindo que os dados estejam corretos e evitando registros duplicados. É essencial para permitir o acesso ao sistema;
- **Autenticar Conta**: Responsável por autenticar um usuário válido no sistema. Caso as informações de login estejam corretas, a rota devolve um token de autenticação, que será utilizado para acessar rotas protegidas, garantindo a segurança e restrição do sistema;
- **Recuperar Senha**: Essa funcionalidade é dividida em duas rotas: uma para iniciar o processo de recuperação (via e-mail) e outra para realizar a redefinição de senha. Ambas foram implementadas com foco na segurança do usuário, exigindo verificações e validações antes de permitir a alteração da senha;
- **Confirmar E-mail**: Após o cadastro, o usuário recebe um link de confirmação de e-mail. Esta rota é responsável por validar o token recebido e confirmar que o e-mail informado pertence ao usuário, ativando assim a conta na plataforma;
- **Atualizar Conta**: Rota dedicada para que o usuário consiga atualizar seus dados cadastrados, como nome, e-mail ou cargo. Isso garante maior flexibilidade e autonomia ao usuário, sendo essencial para a personalização e manutenção das informações no sistema;

- **Atualizar Senha**: Rota destinada à troca de senha por vontade do próprio usuário (e não por esquecimento). A nova senha é validada e criptografada, mantendo a segurança dos dados pessoais;
- **Deletar Conta**: Essas rotas permitem que o usuário solicite a exclusão da sua conta. A requisição de exclusão dispara um e-mail de confirmação (para evitar exclusões acidentais), e somente após a validação, a conta e seus dados são removidos da plataforma de forma segura;
- **Logout**: Utilizada para invalidar a sessão atual do usuário, encerrando seu token e garantindo que o acesso à conta seja finalizado corretamente, aumentando a segurança em ambientes compartilhados.

Figura 18 - Rotas de Usuário representadas no código.

```
app.get('/login', function (req, res) { });
app.post('/login', function (req, res) { });

app.get('/esqueciSenha', function (req, res) { });
app.post('/esqueciSenha', function (req, res) { });

app.get('/redefinirSenha', function (req, res) { });
app.post('/redefinirSenha', function (req, res) { });

app.get('/cadastro', function (req, res) { });
app.post('/cadastro', function (req, res) { });

app.get('/confirmarEmail', function (req, res) { });

app.get('/editarUsuario', function (req, res) { });
app.post('/editarUsuario', function (req, res) { });

app.post('/editarSenha', function (req, res) { });

app.post('/emailDeletarUsuario', function (req, res) { });
app.get('/deletarConta', function (req, res) { });
app.post('/deletarConta', function (req, res) { });

app.post('/quit', function (req, res) { });
```

Fonte: Elaborados pelos autores, 2025

8.2.2 ROTAS DE PROJETO

Essas rotas foram desenvolvidas com o objetivo de gerenciar os projetos dentro da plataforma, permitindo que orientadores e alunos possam criar, editar, deletar e interagir com os projetos aos quais estão vinculados. Cada rota foi cuidadosamente estruturada para tratar apenas as informações necessárias, respeitando as regras de negócio definidas (como o limite de três alunos por projeto e a obrigatoriedade de um orientador). As funcionalidades foram pensadas para garantir praticidade e segurança durante a navegação no front-end, além de manter a integridade dos dados armazenados.

- **Adicionar Projeto**: Essa rota permite que o usuário crie um novo projeto dentro da plataforma. Todas as informações necessárias para o cadastro são validadas antes de o projeto ser inserido no sistema, permitindo que ele seja gerenciado e visualizado futuramente;
- **Enviar E-mail para Deletar Projeto**: Antes da exclusão definitiva de um projeto, essa rota é responsável por enviar um e-mail de confirmação para o usuário. Esse processo serve como uma camada de segurança adicional, evitando exclusões acidentais;
- **Deletar Projeto**: Após a confirmação via e-mail, essa rota realiza de fato a exclusão do projeto selecionado. Todos os dados relacionados ao projeto são removidos da base, incluindo os vínculos com alunos e mensagens associadas, garantindo uma exclusão completa e segura;
- **Editar Projeto**: Essa rota fornece ao orientador a possibilidade de atualizar as informações de um projeto já existente, como o nome, a descrição ou outros dados relevantes. É importante para manter os dados atualizados ao longo da execução do projeto;
- **Convidar Aluno para Projeto**: Aqui os integrantes conseguem convidar alunos para participar de um projeto já existente. A rota valida se o projeto permite mais participantes (máximo de 3) e se o aluno ainda não está vinculado a outro projeto, garantindo as regras de negócio;

- **Adicionar Aluno ao Projeto**: Responsável por adicionar de fato o aluno ao projeto, após o convite ter sido aceito. Essa rota faz a associação no banco de dados entre o projeto e o aluno;
- **Remover Aluno do Projeto**: Permite que um aluno seja desvinculado de um projeto. Essa função pode ser utilizada em casos de desistência, substituição ou reorganização de equipe. A remoção é controlada para não afetar o funcionamento geral do projeto;
- **Home**: Responsável por exibir a tela inicial do usuário logado, normalmente contendo um resumo dos projetos em que ele participa, atalhos para ações rápidas e informações de destaque;
- **Acessar Projeto por URL**: Essa rota permite que um projeto seja acessado diretamente através de uma URL personalizada (`/projeto-:projectURL`). Isso facilita o compartilhamento e navegação direta para um projeto específico na plataforma.

Figura 19 - Rotas de Projeto representadas no código.

```
app.get('/home', function (req, res) { });

app.get('/projeto-:projectURL', function (req, res) { });

app.get('/adicionarProjeto', function (req, res) { });
app.post('/adicionarProjeto', function (req, res) { });

app.get('/editarProjeto', function (req, res) { });

app.get('/deletarProjeto', function (req, res) { });
app.post('/emailDelete', function (req, res) { });
app.post('/deletarProjeto', function (req, res) { });

app.post('/adicionarAluno', function (req, res) { });
app.post('/removerAluno', function (req, res) { });

app.get('/conviteProjeto', function (req, res) { });
app.post('/conviteProjeto', function (req, res) { });
```

Fonte: Elaborado pelos autores, 2025

8.2.3 ROTAS DE COMUNICAÇÃO

As rotas de mensagens foram criadas para possibilitar a comunicação direta entre os participantes de um projeto, promovendo a troca de informações, feedbacks e arquivos importantes. Também integram recursos de análise automática com IA, ajudando na interpretação das mensagens enviadas. Cada rota é responsável por uma função específica dentro do fluxo de mensagens — desde o envio e recebimento até o processamento dos dados. Foram pensadas para entregar uma experiência fluida no front-end, garantindo velocidade e organização na comunicação entre usuários.

- **Buscar Mensagens**: Essa rota retorna todas as mensagens relacionadas a um projeto específico, permitindo que os participantes visualizem o histórico completo de interações. É fundamental para manter a continuidade das conversas e facilitar a comunicação;
- **Enviar Mensagens**: Essa rota permite que um usuário envie uma nova mensagem no contexto de um projeto. Além do texto, também é possível anexar um arquivo PDF, que será armazenado e vinculado à mensagem, expandindo as possibilidades de interação entre os participantes;
- **Analisar Mensagem com IA**: Aqui, a mensagem enviada pelo usuário é processada por uma inteligência artificial, que retorna uma análise automática do conteúdo. Essa funcionalidade é especialmente útil para auxiliar o aluno a entender melhor um feedback ou comentário recebido, oferecendo interpretações e sugestões;
- **ChatBox com IA**: Essa rota conecta o usuário com uma IA em tempo real, permitindo interações diretas e respostas automáticas dentro do ambiente da plataforma. É ideal para dúvidas rápidas, explicações sobre comentários ou ajuda com ações específicas.

Figura 20 - Rotas de comunicação representadas no código.

Fonte: Elaborado pelos autores, 2025

8.3 FRONT-END

No desenvolvimento da aplicação web, foi utilizada a combinação de **HTML** e **CSS** puro (sem o uso de frameworks), garantindo maior controle sobre a estrutura e o estilo da interface. Essa escolha permitiu o desenvolvimento de páginas mais leves e diretas, com foco total na organização do conteúdo e na personalização da experiência visual para o usuário.

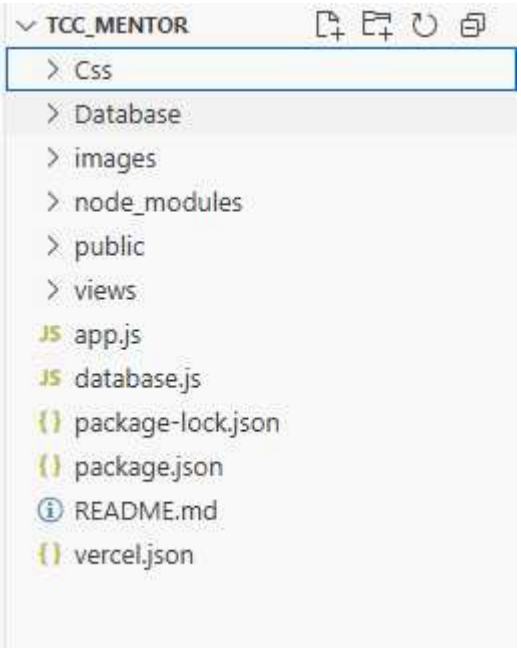
A utilização de HTML possibilitou a criação de toda a estrutura semântica da aplicação, organizando os elementos de forma clara e acessível. Já o CSS foi responsável por aplicar o estilo visual às páginas, definindo cores, espaçamentos, fontes, layouts e demais aspectos visuais que compõem a interface do sistema.

Mesmo sem o uso de bibliotecas externas ou frameworks, a separação entre estrutura (HTML) e estilo (CSS) garantiu uma boa manutenção do código e facilitou ajustes visuais conforme o projeto evoluiu. Essa abordagem também foi importante para entender e aplicar, de forma prática, os conceitos fundamentais do desenvolvimento web.

8.3.1 PÁGINAS

Nossa aplicação seguiu uma forma bem tradicional de programação, cada página tem seu próprio arquivo CSS para estilização própria, elementos e trechos que foram se repetindo foram postos no global.css, para ficar disponíveis para múltiplas páginas, permitindo assim um código menor no geral e aumentando a velocidade de carregamento.

Figura 21 - Todas as páginas da aplicação Web.



Fonte: Elaborado pelos autores, 2025.

9 CONSIDERAÇÕES FINAIS

Inicialmente, o projeto estava concebido para ser um simples administrador de TCC's, para ajudar na organização de alunos e professores, mas conforme fomos conversando com nosso orientador e debatemos entre nós mesmos, percebemos que poderemos fazer uma integração com IA o que facilitaria ainda mais o desenvolvimento de TCC's futuros.

Integrando as novas ideias com as já existentes conseguimos fazer um projeto além do que esperávamos inicialmente. Com ele facilitando a discussão de ideias, fornecendo perspectivas novas a velhas ideias e tornando a confecção do TCC algo muito mais suave.

Com a ideia definida, iniciamos o desenvolvimento mapeando as entidades principais e modelando os fluxos do sistema através de diagramas. Realizamos a escolha estratégica da plataforma de hospedagem considerando requisitos técnicos e de usuário. Cada etapa foi planejada para alinhar a solução final às necessidades do projeto.

A implementação do projeto exigiu uma distribuição estratégica de tarefas, alocando cada função conforme as especialidades dos membros da equipe. Essa organização otimizou o processo de criação, garantindo maior eficiência na construção integrada das soluções. A abordagem colaborativa permitiu potencializar os talentos individuais enquanto fortalecemos o resultado final.

Ao final do desenvolvimento, superados os desafios técnicos comuns em projetos de software, obtivemos um resultado positivo. O sistema atendeu plenamente aos objetivos propostos, incorporando todas as funcionalidades previstas inicialmente. A solução entregue correspondeu às expectativas e requisitos estabelecidos.

REFERÊNCIAS

ATE O MOMENTO. Caso de Uso - Include, Extend e Generalização. Disponível em: https://www.ateomomento.com.br/caso-de-uso-include-extend-e-generalizacao/.

FIGMA. Low-fidelity prototyping. Disponível em: https://www.figma.com/resource-library/low-fidelity-prototyping/.

LIMA, Clara. Protótipos: baixa, média ou alta fidelidade? Medium, 5 fev. 2021. Disponível em: https://medium.com/ladies-that-ux-br/prot%C3%B3tipos-baixa-m%C3%A9dia-ou-alta-fidelidade-71d897559135

LUCIDCHART. O que é UML? Disponível em: https://www.lucidchart.com/pages/pt/o-que-e-uml.

SPASOJEVIC, Anastazija. What is JavaScript?. Disponível em: https://phoenixnap.com.br/glossary/what-is-javascript.

ZADHID, Powell. Node.js vs PHP: Uma Comparação Detalhada.. Disponível em: https://kinsta.com/pt/blog/node-js-vs-php/.

NORONHA, Cristiano B. O que é um framework. Disponível em: https://balta.io/blog/o-que-e-um-framework.

RED HAT. O que é uma API REST? Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api.

MOZILLA FOUNDATION. Express e Node.js. MDN Web Docs. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs.

LEITE FILHO, Geraldo Alemandro. A relação orientador-orientando e suas influências no processo de elaboração de teses e dissertações dos programas de pós-graduação em contabilidade da cidade de São Paulo. 2004. Dissertação (Mestrado) – Universidade de São Paulo, São Paulo, 2004. Disponível em: http://www.teses.usp.br/teses/disponiveis/12/12136/tde-29012005-165626

DEVMEDIA. *Requisitos Não Funcionais e Arquitetura de Software*. Disponível em: https://www.devmedia.com.br/artigo-engenharia-de-software-3-requisitos-nao-funcionais/9525.

GUEDES, Gilleanes T. A. UML 2: Guia de Consulta Rápida. São Paulo: Novatec Editora, 2011.