UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE RUSSAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

FELIPE CESAR DE SOUSA SILVA

DESENVOLVIMENTO FRONT-END DA VERSÃO WEB DO APLICATIVO AMO (AMBIENTE DE MONITORIA ONLINE)

RUSSAS

2025

FELIPE CESAR DE SOUSA SILVA

DESENVOLVIMENTO FRONT-END DA VERSÃO WEB DO APLICATIVO AMO (AMBIENTE DE MONITORIA ONLINE)

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharelado em Engenharia de Software.

Orientador: Prof.ª Dra. Patrícia Freitas Campos de Vasconcelos.

RUSSAS

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S58d Silva, Felipe Cesar de Sousa.
Desenvolvimento front-end da versão web do aplicativo amo (ambiente de monitoria online) / Felipe Cesar de Sousa Silva. – 2025.
64 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Engenharia de Software, Russas, 2025.
Orientação: Profa. Dra. Patrícia Freitas Campos de Vasconcelos.

1. monitoria acadêmica. 2. desenvolvimento front-end. 3. aplicação web. I. Título.
CDD 005.1

FELIPE CESAR DE SOUSA SILVA

DESENVOLVIMENTO FRONT-END DA VERSÃO WEB DO APLICATIVO AMO (AMBIENTE DE MONITORIA ONLINE)

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharelado em Engenharia de Software.

Aprovada em: 04/08/2025.

BANCA EXAMINADORA

_____

Profª. Dra. Patrícia Freitas Campos de Vasconcelos (Orientador)
Universidade Federal do Ceará (UFC)

_____

Profa. Ms. Valeria Maria da Silva Pinheiro
Universidade Federal do Ceará (UFC)

_____

Prof. Ms. Hugo Nathan Barbosa Regis
Universidade do Estado do Rio Grande do Norte, UERN,

Dedico esse trabalho a Deus, e aos meus pais Francisca Iraneide da Silva e Raimundo Cezar de Souza Silva.

AGRADECIMENTOS

Agradeço a Deus por me conceder a força necessária para perseverar na busca dos meus objetivos, mesmo diante dos obstáculos que surgem pelo caminho.

Sou profundamente grato aos meus pais, Francisca Iraneide da Silva e Raimundo Cezar de Souza Silva, pelo apoio e pela presença constante em minha vida. Agradeço também à minha orientadora, Dra. Patrícia Freitas Campos de Vasconcelos, que aceitou me orientar e guiar ao longo deste trabalho, sendo fundamental para a sua realização.

Por fim, expresso minha gratidão a todo o corpo docente da Universidade Federal do Ceará - Campus Russas, que me proporcionou a oportunidade de aprimorar meus conhecimentos e ofereceu um ensino de excelência.

LISTA DE FIGURAS

Figura 1 - React DOM……..……...…….…….....22
Figura 2 - Etapas do procedimento metodológico ……...…..…….……...27
Figura 3 - Guia de estilo……..…….……...32
Figura 4 - Tela de login……..…….....33
Figura 5 - Tela de Cadastro ……..……...……...33
Figura 6 - Tela de Fórum ……....……...34
Figura 7 - Tela de selecionar curso……...……..……..…….……34
Figura 8 - Componente Card de pergunta…….……..……..…….……38
Figura 9 - Código do filtro para o fórum..……..…..……..……..…….……38
Figura 10 - Função que realiza a mudança do filtro do fórum…….…….……39
Figura 11 - Código do contexto de autenticação do sistema…….……..…….....40
Figura 12 - Tela de login implementada…….……..……..…….....41
Figura 13 - Tela de cadastro implementada…….……..……..…….....41
Figura 14 - Tela de completar cadastro implementada…….……..……..…….....41
Figura 15 - Tela de fórum implementada…….……..……..…….....42
Figura 16 - Componente Cadastrar dúvida…….……..……..…….....42
Figura 17 - Componente responder dúvida…….……..……..…….....43
Figura 18 - Módulo para realizar requisições a API…….……...…….....43
Figura 19 - Módulo para guardar, apagar e resgatar o token do localStorage…….…….....44
Figura 20 - Tela de login……...……..…….....45
Figura 21 - Context que guarda os dados do usuário……..…….....45
Figura 22 - Painel do Thunder Client……..…….....46
Figura 23 - Painel do projeto do sistema no Github.……..…….....47
Figura 24 - Painel do projeto do sistema no Github.……..…….....48
Figura 25 - Tela de seleção de curso……...……..……..…….....50
Figura 26 - Feedback de erro na tela login……...……..…….....50
Figura 27 - Feedback de campos vazios na tela login……..…..…….....51
Figura 28 - Erro no formato de email na tela cadastro.……..…….....52
Figura 29 - Erro de email já existente.……...……..…..…….....53
Figura 30 - Telas relacionadas a cadastrar dúvida e fórum.……...…….....54
Figura 31 - Erro de entrada vazia.……...……..……..…….....55
Figura 32 - Estrutura do código-fonte……...……..…….....57

Figura 33 - Feedback mais claro de registrar dúvida.……...……..…….....57
Figura 34 - Implementação da tela Fórum.……...……..…….....58

LISTA DE QUADROS

Quadro 1 - Principais tags do HTML……...……..……..…….....20
Quadro 2 - Comparação com os trabalhos relacionados…..……...…….…….....26
Quadro 3 - Requisitos funcionais……...……...28
Quadro 4 - Disponibilidade de documentação e comunicação……...……...…...30
Quadro 5 - Tamanho do arquivo JavaScript gerado…….…...……...…...30
Quadro 6 - Tempo de renderização da página……..…….……...…....31
Quadro 7 - Tempo de renderização da página……..…….……...…....31
Quadro 8 - Tempo de renderização da página……..…….……...…....31
Quadro 9 - Técnicas usadas……..……...…….……...…....36

LISTA DE ABREVIATURAS E SIGLAS

AMO	Ambiente de Monitoria Online
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
RF	Requisitos funcionais
UFC	Universidade Federal do Ceará

RESUMO

A monitoria acadêmica é um pilar fundamental na formação de alunos na Universidade, com isso o Projeto de Apoio ao Ensino (PAE) da UFC (Universidade Federal do Ceará) criou o aplicativo AMO (Ambiente de Monitoria Online) que visa auxiliar a monitoria. Porém existem problemas como a restrição de plataforma e o baixo alcance por o aplicativo dá suporte apenas a dispositivos mobile. O intuito dessa pesquisa é criar uma aplicação web que junto da sua versão mobile possa construir um aprendizado colaborativo em que os alunos, monitores e professores possam trocar conhecimento por meio desse ambiente virtual. O sistema conta com todas as funcionalidades do aplicativo mobile denominado AMO, como função para realizar agendamentos virtuais ou remotos e Fórum para interação dos estudantes, com isso melhorando a comunicação entre os usuários. Neste trabalho são relatados os tópicos do desenvolvimento do sistema como definição do aplicativo, escolha do framework, análise do protótipo e da documentação, e com base nisso o desenvolvimento do código focando em criar componentes reutilizáveis, criação das telas e dando início a criação de telas iniciais. Pretende-se ainda fazer a integração com o back-end por meio da API REST usada no AMO mobile e aplicações das técnicas de verificação e validação para testar a interface e encontrar possíveis erros no sistema. Por fim, este trabalho resultou na criação de uma aplicação web com intuito de aprimorar a interação entre alunos e monitores e expandir o alcance do aplicativo AMO na UFC. Espera-se que essa ferramenta aprimore o aprendizado melhorando o alcance e eliminando a restrição de plataforma que a versão web possui.

Palavras-chave: monitoria acadêmica; desenvolvimento front-end; aplicação web.

ABSTRACT

Academic tutoring is a fundamental pillar in the education of students at the University, so the PAE Project (Teaching Support Project) of UFC (Federal University of Ceará) created the AMO application (online tutoring environment) to assist tutoring. However, there are problems such as platform restrictions and low reach because the application only supports mobile devices. The purpose of this research is to create a web application that, together with its mobile version, can build collaborative learning in which students, tutors and teachers can exchange knowledge through this virtual environment. The system has all the functionalities of the mobile application called AMO, such as a function to make virtual or remote appointments and a Forum for student interaction, thus improving communication between users. This work reports on the topics of the system development such as defining the application, choosing the framework, analyzing the prototype and documentation, and based on this, the development of the code focusing on creating reusable components, creating screens and starting the creation of initial screens. The aim is also to integrate with the back-end through the REST API used in AMO mobile and to apply verification and validation techniques to test the interface and find possible errors in the system. Finally, this work resulted in the creation of a web application with the aim of improving the interaction between students and tutors and expanding the reach of the AMO application at UFC. It is expected that this tool will enhance learning by improving the reach and eliminating the platform restriction that the web version has.

Keywords: academic monitoring; front-end development, web application.

SUMÁRIO

1 INTRODUÇÃO..... 14

2 OBJETIVOS..... 16

2.1 Objetivo geral..... 16

2.2 Objetivos específicos..... 16

3 FUNDAMENTAÇÃO TEÓRICA..... 17

3.1 Monitoria acadêmica..... 17

3.2 Ambiente de Monitoria Online (AMO)..... 17

3.3 Desenvolvimento front-end..... 18

3.3.1 Aplicação web..... 18

3.3.2 JavaScript..... 19

3.3.3 HTML..... 20

3.3.4 CSS..... 20

3.3.5 React.js..... 21

3.4 API Rest..... 22

3.4.1 AXIOS..... 23

4 TRABALHOS RELACIONADOS..... 24

5 PROCEDIMENTOS METODOLÓGICOS..... 27

5.1 Definição do sistema..... 28

5.2 Analisar a documentação..... 28

5.3 Escolha do framework..... 30

5.3.1 Comparação de frameworks..... 30

5.4 Definição do Projeto de Interface..... 31

5.5 Desenvolvimento do sistema..... 34

5.5.1 Criação de componentes..... 34

5.5.2 Criação de telas iniciais..... 35

5.5.3 Ferramentas auxiliares..... 35

5.5.4 Integração com o back-end..... 35

5.6 Técnicas de verificação e validação..... 35

6 RESULTADOS..... 37

6.2 Definição do sistema.....37

6.3 Escolha do Framework.....37

6.4 Criação dos componentes..... 37

6.5 Criação das telas.....40

6.6 Integração com o back-end..... 43

6.7 Uso de ferramentas auxiliares.....46

6.8 Aplicação das técnicas de verificação e validação.....48

6.8.1 Teste de caixa preta..... 49

6.8.3 Revisão por Pares (Peer Review)..... 55

7 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS..... 59

REFERÊNCIAS..... 60

1 INTRODUÇÃO

A monitoria é uma ação ofertada pelas instituições de ensino superior, voltada para estudantes mais avançados em seu nível de formação, que possuem mais facilidade com determinada disciplina. Com isso, esses alunos serão responsáveis por auxiliar e ajudar outros colegas, sempre com a orientação direta de um professor (Costa et al., 2021).

Segundo Oliveira (2024), o Ambiente de Monitoria Online, conhecido como AMO, foi desenvolvido como um aplicativo móvel pelo Projeto de Extensão PAE (Projeto de Apoio ao Ensino) da Universidade Federal do Ceará Campus Russas. Essa ferramenta desempenha um papel fundamental no auxílio a alunos e monitores, facilitando o acompanhamento de atividades e interações acadêmicas. Contudo, sua restrição ao uso apenas em dispositivos móveis apresenta desafios significativos. Entre as limitações, destacam-se as dificuldades relacionadas à variabilidade de dispositivos e sistemas operacionais, a necessidade de atualizações constantes para diferentes plataformas, a experiência do usuário, que pode ser impactada em telas menores ou por problemas de desempenho em dispositivos de baixa capacidade. É válido também ressaltar o alto custo de publicação do aplicativo nas lojas de aplicativos como na AppStore fazendo com que torne inviável sua publicação. Além disso, usuários sem acesso a *smartphones* ou com restrições de conectividade acabam enfrentando barreiras adicionais, prejudicando a inclusão e a eficiência do sistema.

Com a expansão do uso da internet, aplicações web (sistemas, páginas, sites, jogos, entre outros) se destacam de outros tipos de aplicações em vários pontos, como a diversidade e globalização de usuários uma vez que traz usabilidade e interfaces mais ricas, diversificadas, adaptadas a diferentes resoluções e dispositivos distintos (Mattos et al., 2009).

O principal argumento deste trabalho é que expandir o ambiente de monitoria para a plataforma web não apenas supera as limitações técnicas e de acessibilidade presentes no uso exclusivo de dispositivos móveis, mas também proporciona uma experiência mais acessível, inclusiva e eficiente para toda a comunidade acadêmica. Ao projetar a interface e garantir a qualidade do sistema por meio de métodos sistemáticos de verificação e validação, a solução proposta poderá atender a uma maior diversidade de usuários, consolidando-se como uma ferramenta estratégica no suporte às atividades de ensino-aprendizagem. Assim, este estudo não se limita à criação do sistema, ele busca contribuir para uma transformação no acesso e na interação no contexto da monitoria acadêmica, alinhando-se às demandas contemporâneas por ambientes digitais acessíveis, inclusivos e de alta qualidade.

Este trabalho está estruturado em sete capítulos, cada um abordando aspectos cruciais

para o desenvolvimento e a compreensão do sistema AMO web. O Capítulo 1, "Introdução", apresenta o contexto e a relevância do projeto. No Capítulo 2, "Objetivos", são definidos as metas gerais e específicas a serem alcançadas. O Capítulo 3, "Fundamentação Teórica", explora os conceitos e teorias que embasam o desenvolvimento, como monitoria acadêmica, desenvolvimento front-end e APIs REST. O Capítulo 4, "Trabalhos Relacionados", analisa pesquisas anteriores que contribuíram para este estudo. O Capítulo 5, "Procedimentos Metodológicos", detalha as etapas de desenvolvimento, desde a definição do sistema até as técnicas de verificação e validação. O Capítulo 6, "Resultados", apresenta os resultados obtidos. Por fim, o Capítulo 7, "Considerações Finais e Trabalhos Futuros", conclui o trabalho e sugere direções para futuras pesquisas.

2 OBJETIVOS

2.1 Objetivo geral

Desenvolver uma solução de front-end do aplicativo AMO mobile para WEB, usando React.

2.2 Objetivos específicos

- Construir a interface de usuário, considerando a plataforma web;
- Aplicar o framework React no desenvolvimento front-end;
- Analisar a construção do sistema web a partir de técnicas de verificação e validação;
- Disponibilizar um ambiente de interação independente do dispositivo.

3 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão abordados os conceitos e teorias utilizados durante a construção deste trabalho

3.1 Monitoria acadêmica

Segundo Garcia (2013) a monitoria acadêmica pode ser definida como uma modalidade de ensino e aprendizagem que atende as necessidades de formação universitária na medida em que envolve o graduando nas atividades de organização, planejamento e execução do trabalho docente. De certo modo, é um esforço pedagógico em que o professor orienta e recebe auxílio de um monitor, que tem maior conhecimento em sobre outros alunos, esse processo contribui para o ensino-aprendizagem da turma. Outra definição dada pelo Gonçalves *et al*. (2021), pode se entender como uma ferramenta para auxiliar no processo de aprendizagem que vai contribuir no crescimento profissional tanto do discente quanto do docente.

De acordo com Souza *et al*. (2016) a monitoria ajuda a melhorar o ensino e a unir a teoria com a prática. Ela lida com dúvidas, desenvolve habilidades técnicas e faz com que os alunos se sintam mais seguros para encarar a vida profissional. Além disso, ajuda a consolidar o progresso acadêmico, a operação de equipamentos e o interesse em ensinar.

A monitoria tem uma grande importância na formação acadêmica do aluno pois exerce um papel fundamental de iniciação a docência, com objetivo de prepará-los para realizar atividades docentes melhorando a qualidade do ensino. A monitoria é importante também para o desenvolvimento de habilidades e conhecimentos, seja em qualquer área específica contribuindo para o processo de aprendizagem do aluno monitor.

3.2 Ambiente de Monitoria Online (AMO)

O AMO é um aplicativo de apoio ao ensino que foi desenvolvido pelo projeto PAE (Projeto de Apoio ao Ensino) da Universidade Federal do Ceará, o aplicativo está disponível para download na plataforma PlayStore. Ele tem como principal objetivo auxiliar os alunos, monitores e professores no processo de aprendizagem através de vários recursos que a aplicação disponibiliza. Dentre as funcionalidades do AMO, ressalta-se a possibilidade de agendar monitorias com determinado monitor, tanto online ou presencialmente, deixando o

acesso a monitoria mais fácil para os alunos, o aplicativo também implementou um Fórum onde os usuários realizam suas perguntas sobre qualquer tipo de assunto relacionado a sua disciplina.

Outra funcionalidade que se destaca é a opção de favoritar dúvidas, possibilitando que os estudantes salvem conteúdos para que possam consultar em outro momento. A visualização de horários dos monitores também é possível no AMO, pois o mesmo disponibiliza uma seção que vai mostrar os horários de atendimento de cada monitor. Com todas essas funcionalidades. O AMO visa entregar um ambiente de aprendizado interativo e que seja acessível, fornecendo suporte acadêmico e contribuindo para o desempenho dos estudantes durante a conclusão do seu curso.

3.3 Desenvolvimento front-end

O *front-end* tem a função de fazer a comunicação entre o usuário e o sistema, através de uma interface gráfica, que normalmente é consumida através de um navegador, seja em qualquer dispositivo (Liu et al., 2017). O front-end desempenha um papel fundamental na exibição de dados e no gerenciamento das respostas às solicitações dos usuários. Ele representa o componente do aplicativo responsável por fornecer serviços visuais e interativos, incluindo informações essenciais sobre o projeto, relatórios de progresso e a finalização de aplicativos para as partes envolvidas no gerenciamento de projetos de pesquisa. Além disso, por meio da interface gráfica, o *front-end* recebe as solicitações dos usuários, encaminha elas para o *back-end*, onde a lógica do serviço e o processamento de dados são executados, e posteriormente retorna os resultados ao usuário (Liu et al., 2017).

3.3.1 Aplicação web

Com o aumento do acesso à internet nessa última década, aliado ao crescente uso de dispositivos móveis, o desenvolvimento de sistemas vem aumentando cada vez mais (Mattos et al). Diferentemente das aplicações comuns, que exigem instalação em computadores locais pelo usuário, os sistemas web podem ser acessados facilmente através de um navegador, independentemente do dispositivo utilizado, com isso trazendo vantagens como a facilidade na distribuição e atualizações do sistema, uma vez que as modificações são diretamente atreladas ao servidor, evitando qualquer tipo de necessidade de influência nos dispositivos (Sommerville, 2011).

O conceito de aplicação Web é utilizado para descrever softwares que operam no ambiente da internet. Pressman (2002) define uma aplicação Web como qualquer tipo de aplicação, desde uma página simples até um site completo. No entanto, essa definição revela-se apenas parcialmente adequada aos propósitos deste trabalho, uma vez que não contempla, de forma abrangente, a complexidade inerente ao desenvolvimento de software. Para que as dificuldades e os desafios característicos dessa área sejam devidamente analisados e compreendidos, é imprescindível adotar uma abordagem mais aprofundada. Tal complexidade é essencial para discutir, com o devido rigor, os aspectos técnicos, as limitações e as soluções envolvidas no processo de criação e manutenção de sistemas Web.

Segundo Conallen (1999) uma aplicação Web é um sistema que implementa lógica de negócios , em outras palavras, "uma estrutura que organiza operações de dados e processos do sistema que, quando invocadas, alteram o estado do negócio envolvido", o objetivo principal da aplicação não é apenas informar o usuário, mas também fazer a interação com o mesmo. Para Paula Filho(2003), uma aplicação web é um software com arquitetura distribuída, uma parte funciona sob o protocolo Hypertext Transfer Protocol (HTTP), com isso possibilita que o usuário acesse a interface em qualquer navegador e em qualquer dispositivo.

Com base nessas características de fácil acesso, o trabalho atual propõe o desenvolvimento de uma aplicação web para auxiliar os alunos da UFC Campus Russas e melhorar o alcance do AMO mobile. A escolha por ser uma aplicação web se resume pela necessidade de acessibilidade multiplataforma, para que alunos utilizem o sistema direto do seu navegador, e em qualquer dispositivo.

3.3.2 JavaScript

De acordo com Cunha (2024), o Javascript é um pilar fundamental do desenvolvimento front-end, é responsável por fazer o comportamento dinâmico das interfaces que irão ser executadas pelo navegador. Diferente do HTML e CSS, que são basicamente linguagens de marcação e de estilização, o Javascript é uma linguagem de programação que fornece a criação de interfaces Web dinâmicas, de maneira que o conteúdo exibido possa ser atualizado em tempo real com base na interação do usuário com o sistema. Ainda segundo Cunha (2024), o JavaScript pode ser classificado como uma linguagem de *script*, pois seu código é compilado direto de um ambiente de execução, sem precisar de um processo de compilação antecipado. Os navegadores atuais possuem mecanismo para realizar a

interpretação e executar os scripts em JavaScript, concedendo o desenvolvimento de aplicações Web complexas.

3.3.3 HTML

Segundo Cunha (2024) a linguagem HTML (HyperText Markup Language) foi projetada por Tim Berners-Lee no ano de 1990, a mesma é classificada como "linguagem de marcação de texto" como sugerido pelo nome, refere-se a uma tecnologia que possibilita fazer toda a estrutura de textos da página web. Ao longo do tempo passou por inúmeras evoluções, até chegar na sua versão recente denominada HTML5, sendo oficializada como padrão em 2016.

A estrutura de uma página Web é composta por meio da "marcação" do HTML, definindo seus elementos visuais a partir do uso de tags que se comunicam com o navegador a maneira de como esses conteúdos serão exibidos, permitindo a organização e harmonia dos elementos que compõem uma página (Cunha, 2024). O HTML traz uma vasta gama de marcadores para diferentes funções, em que é possível verificar no Quadro - 1.

Quadro 1 - Principais tags do HTML

Tags	Descrição
(p, h1, h2, h3)	Estabelece títulos e subtítulos hierárquicos.
(div, section, nav, footer)	Auxiliam na separação lógica dos conteúdos dentro da página.
(table)	Para criação de tabelas personalizadas.
(img)	Possibilita a inserção de imagens

Fonte: Elaborado pelo autor(2024) .

3.3.4 CSS

O Cascading Style Sheets (CSS), é uma linguagem de estilização para a definição da apresentação de elementos HTML. O termo "Cascading" , que traduzido seria "Cascata" , se refere à maneira de como as regras de estilos serão aplicadas, levando em consideração a ordem e a especialidade dos seletores. Sua primeira versão foi projetada em 1996, mas o css atual, denominado com CSS3, teve início em 1999 e vem recebendo atualizações para melhorar suas funções e evoluir no desenvolvimento Web (Cunha, 2024).

De acordo com Cunha (2024), o CSS possibilita montar a estilização de qualquer elemento de uma página, como textos, imagens, divs, tabelas e etc, com isso trazendo um controle preciso sobre aspectos visuais da interface, incluindo tamanhos de layouts, cores, fontes, bordas e posicionamentos de telas. A linguagem ainda possibilita uma adaptação do design com relação ao meio de exibição, sendo elas telas, impressões ou dispositivos móveis, possibilitando designs responsivos e adaptáveis. Com todos esses recursos do CSS, os desenvolvedores aprimoram a experiência do usuário, entregando interfaces que se ajustam a diferentes resoluções e telas diferentes. Dessa maneira o CSS estabelece um papel de extrema importância no desenvolvimento web, deixando interfaces mais robustas e atraentes.

3.3.5 React.js

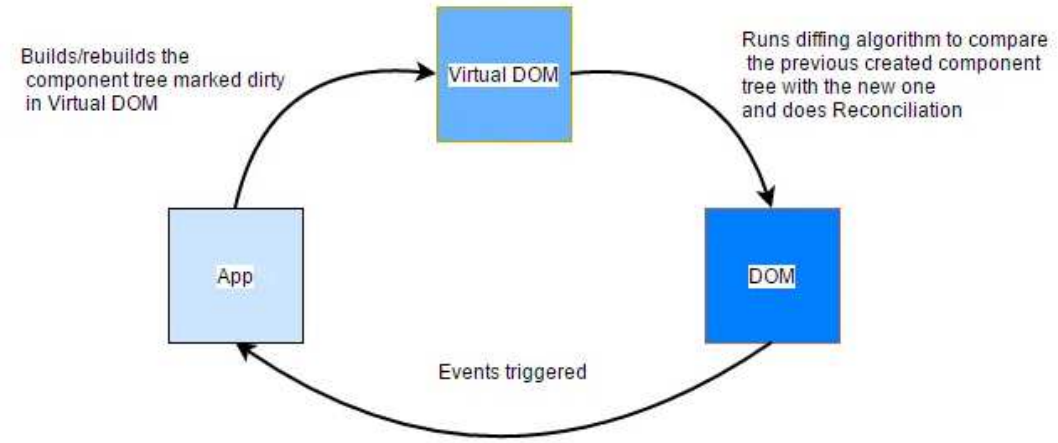
React.js é um *framework* desenvolvido para construir interfaces de usuários, foi criada por engenheiros do facebook, desde seu lançamento vem fazendo muito sucesso nas comunidades javascript, e passou a ser uma ferramenta bastante escolhida de muitas equipes de desenvolvimento por sua forma dinâmica de implementar interfaces (Mark, 2018).

O React tem várias vantagens, mas umas das principais está relacionada à forma eficiente em que ele interage com o Document Object Model (DOM). O DOM é um tipo de árvore que simboliza todos os elementos e estados de uma página web, possibilitando que os navegadores possam renderizar todo o conteúdo corretamente (Danielsson, 2016).

Com tudo, fazer modificações diretamente na estrutura pode ser um processo desagradavel, podendo afetar o desempenho do sistema, uma vez que cada alteração pode solicitar uma nova renderização de página, afetando a experiência do usuário. Uma forma de otimizar esse processo, o React introduziu o conceito de Virtual DOM, a Figura 1 apresentada descreve o ciclo de vida e a interação entre os principais componentes do React no processo de renderização e atualização da interface do usuário, com um foco central no conceito do Virtual DOM. Este diagrama é fundamental para compreender como o React otimiza as operações no Document Object Model (DOM) real, agindo como uma representação mais suave do DOM real do Javascript, em vez de alterar os elementos da página, o React faz alterações na virtual DOM, que serve mais como uma cópia, comparando as diferenças entre o estado anterior e o novo, processo esse denominado (*reconciliation),* e só depois atualizar com eficiência os elementos necessários no DOM real. Com esse comportamento reduz consideravelmente o número de manipulações diretas na estrutura principal na página, gerando mais desempenho na aplicação (Kumar; Singh, 2016).

Perante essas características, o React se apresenta como uma solução ideal para o desenvolvimento do sistema AMO web proposto neste trabalho. Sua capacidade de criar interfaces dinâmicas e reutilizáveis facilita a implementação das funcionalidades necessárias, proporcionando uma experiência de usuário fluida e eficiente. Assim, a escolha pelo React como tecnologia principal para o front-end deste sistema está alinhada com os objetivos do projeto, garantindo maior modularidade, desempenho e manutenção do código.

Figura 1 - React DOM



Fonte: Kurian(2024).

3.4 API Rest

Segundo Gowda (2024), o Representational State Transfer (REST) é uma abordagem arquitetônica amplamente utilizada para o desenvolvimento de serviços Web em aplicações modernas. Sua simplicidade, portabilidade e compatibilidade com o protocolo HTTP fazem do REST uma escolha adequada para sistemas distribuídos que precisam suportar múltiplas aplicações. APIs que seguem esse paradigma, conhecidas como RESTful, utilizam endpoints e verbos HTTP para acessar e manipular recursos entre cliente e servidor, tornando-se uma solução padronizada e versátil. No entanto, a eficácia das APIs REST depende diretamente da adoção de boas práticas durante seu design e implementação.

Além disso, Gowda (2024) destaca que a adoção dessas diretrizes proporciona benefícios em termos de estabilidade, confiabilidade e proteção dos serviços expostos pela

API. Por meio da aplicação dessas boas práticas, os desenvolvedores conseguem aprimorar a experiência de uso da API, tornando-a mais intuitiva e acessível para outros programadores. Por exemplo, a utilização de nomenclaturas claras e descritivas para os *endpoints*, bem como o mapeamento adequado das operações CRUD (Creat Read Update Delete) para os métodos HTTP correspondentes, promove consistência e previsibilidade na interação com a API. Dessa forma, não apenas facilita o processo de desenvolvimento, como também melhora a experiência dos consumidores da API nos aplicativos cliente.

3.4.1 AXIOS

Axios é uma biblioteca JavaScript popular, baseada em Promises, utilizada para realizar requisições HTTP (XHR no navegador e requisições HTTP no Node.js). Ela oferece uma interface simples e poderosa para interagir com APIs RESTful, facilitando o envio de dados para o servidor e o recebimento de respostas. Uma das principais vantagens do Axios é a sua capacidade de lidar com requisições assíncronas de forma elegante, utilizando Promises para gerenciar o sucesso ou falha das operações. Além disso, o Axios possui funcionalidades como interceptores de requisição e resposta, que permitem adicionar lógica personalizada antes ou depois de uma requisição ser enviada ou recebida, e a capacidade de cancelar requisições, o que é útil em cenários onde a resposta não é mais necessária. Sua compatibilidade com navegadores e ambientes Node.js, juntamente com sua vasta comunidade e documentação clara, tornam-no uma escolha robusta para o desenvolvimento de aplicações web modernas que dependem fortemente da comunicação com o *back-end*.

Para este projeto, a escolha do Axios foi ideal devido à arquitetura RESTful do *back-end* do AMO mobile, garantindo uma comunicação eficiente e padronizada entre o *front-end* e o *back-end* da aplicação web.

4 TRABALHOS RELACIONADOS

Nesta seção serão abordados trabalhos que contribuíram para o desenvolvimento dessa pesquisa, sendo descritos trabalhos voltados para o desenvolvimento de sistemas para a educação e acompanhamento acadêmico.

A plataforma "Graduando", desenvolvida por Reis e Brito (2021), é um sistema feito para ajudar a monitoria acadêmica em universidades. Foi construída usando as tecnologias Laravel, Bootstrap, HTML, CSS e JavaScript, a plataforma concede aos alunos acesso a informações como os horários das monitorias, enviar mensagens para os monitores, realizar publicações e participar de discussões no fórum. Além do suporte aos alunos, o sistema também disponibiliza acesso para funcionários da instituição de ensino, possibilitando a gestão de cursos, alunos, professores e demais colaboradores, como a equipe administrativa da secretaria.

Esse sistema possui algumas similaridades com o trabalho dessa pesquisa, pois tem como principal objetivo melhorar o acompanhamento acadêmico dos estudantes por meio de um sistema. No entanto, a plataforma "Graduando" se difere do sistema AMO em alguns aspectos, ela não permite que alunos façam o agendamento em tempo real, e se limita apenas a exibição dos horários dos monitores, não tendo a funcionalidade de agendar atendimentos.

O trabalho realizado por Oliveira (2024), contempla o desenvolvimento da versão mobile do sistema AMO, considerando um conjunto de recursos que visam auxiliar no aprendizado de alunos, como a possibilidade de agendar horários como monitores de forma remota ou presencial, acesso a fórum de discussões, (onde alunos, monitores e professores possam publicar suas perguntas, responder e interagir entre si) dentre outras funcionalidades. O aplicativo é voltado aos alunos da Universidade Federal do Ceará Campus Russas, que têm dificuldades em disciplinas específicas contempladas por projetos. Contudo, vale ressaltar que a ferramenta desenvolvida por Oliveira (2024) foi projetada para mobile usando o framework React-Native, que não tem suporte a desktops, se limitando apenas ao acesso em dispositivos móveis. Esta pesquisa se assemelha com a pesquisa de Oliveira (2024) por se tratar do desenvolvimento do AMO, contudo aplicado ao ambiente WEB. Para este desenvolvimento ser possível, será utilizado a biblioteca React.

O trabalho de Abreu et al. (2016) consiste no desenvolvimento de uma plataforma denominada “Monitoria Virtual” , que tem como objetivo facilitar a interação entre monitores e estudantes no quesito monitoria acadêmica. O sistema foi introduzido com a linguagem PHP e usa o banco de dados MySQL, sua estrutura é composta por dois módulos principais, sendo

um voltado para os estudantes, e o outro para monitores. A ferramenta vem com a ideia de disponibilizar diversos ambientes, um para cada disciplina, operando como um meio digital para mesclar a relação entre os alunos e os monitores no processo de aprendizagem. A plataforma tem como principal funcionalidade permitir aos alunos caminhar suas dúvidas aos monitores, que logo em seguida podem responder diretamente no sistema. Com isso ampliando a acessibilidade dos estudantes ao suporte didático, sem precisar marcar encontros presenciais.

A respeito das semelhanças entre a solução dada por Abreu et al. (2016) e a atual pesquisa, está na ênfase do processo de digitalização da prática de monitoria, concedendo ao aluno a possibilidade de um contato remoto com monitores e suportes acadêmicos. Quanto às diferenças entre as duas plataformas, identificou-se a ausência de um fórum interativo que permitisse colaboração entre os alunos. Além disso, o estudo de Abreu et al (2016) não prevê a possibilidade de estudantes realizarem agendamentos para encontros síncronos com os monitores. Contudo esta pesquisa sugere a implementação desses recursos, permitindo um ambiente onde os estudantes não ficam limitados a interagir apenas com monitores, podendo discutir entre si, aumentando a aprendizagem coletivamente. Além da possibilidade de agendamento de reuniões remotas, a partir de uma abordagem customizada. Embora ambos os trabalhos compartilhem o objetivo de aprimorar a experiência da monitoria com o auxílio de sistemas, a pesquisa atual pretende expandir essa proposta oferecendo funcionalidades que melhoram o aprendizado colaborativo entre os alunos e monitores.

O Quadro 2 a seguir apresenta os principais critérios usados para a comparação entre os trabalhos relacionados a esta pesquisa. É importante observar que dois trabalhos possuem bate-papo com o monitor, dois trazem suporte a fórum para discussão, apenas um possibilita o agendamento de monitoria e três são sistemas web.

Quadro 2 - Comparação com os trabalhos relacionados

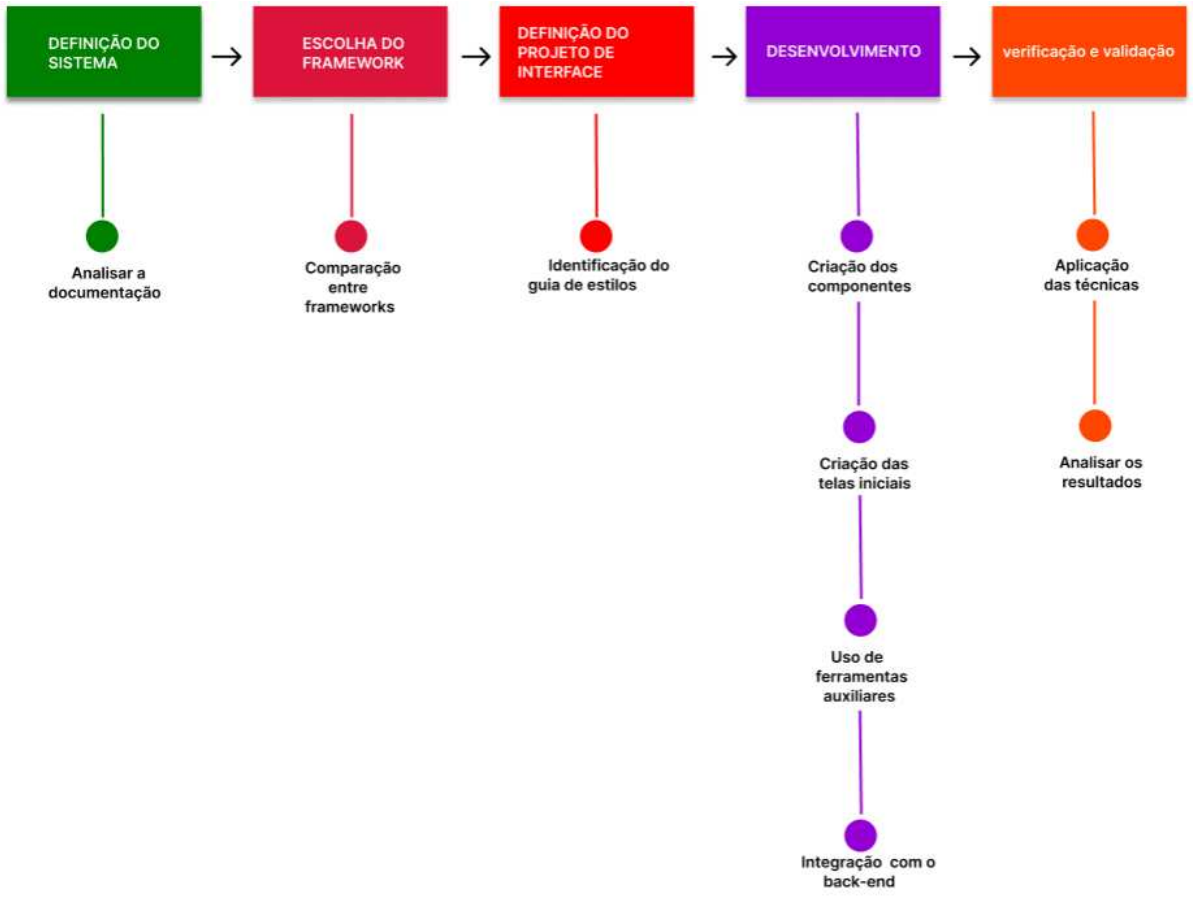
Trabalho	Possui batepapo direto com o monitor?	Possui fórum para discussão?	Framework utilizado	Permite o agendamento de monitoria?	Plataforma web ou mobile?
Abreu et al (2016)	Não	Não	Não	Não	Web
Reis e Brito (2021)	Sim	Sim	Laravel	Não	Web
Oliveira (2024)	Sim	Sim	React-Native	Sim	Mobile
Este trabalho	Sim	Sim	React.js	Sim	Web

Fonte: Elaborado pelo autor (2024).

5 PROCEDIMENTOS METODOLÓGICOS

Esta seção mostra as atividades que serão realizadas para atingir os objetivos propostos neste trabalho. O foco está na obtenção de informações essenciais para o desenvolvimento de um sistema voltado ao auxílio da monitoria acadêmica. Para garantir um processo estruturado e eficiente, as etapas foram organizadas da seguinte forma: definição do sistema, escolha do framework, projeto de interface, desenvolvimento do sistema, além da aplicação de técnicas de verificação e validação para assegurar a qualidade e o correto funcionamento do sistema.

Figura 2 - Etapas do procedimento metodológico



Fonte: Elaborado pelo autor(2024).

5.1 Definição do sistema

Nesta etapa ocorreu a definição do sistema educacional a ser desenvolvido. O sistema definido foi o AMO proposto no projeto PAE para ajudar a comunidade acadêmica da Universidade Federal do Ceará (UFC). O PAE é um projeto de extensão da UFC campus de Russas, e o sistema a ser desenvolvido é baseado no aplicativo (AMO) do próprio PAE, que visa auxiliar a realização da monitoria acadêmica. Aumentando o seu alcance em multiplataformas, diferente do (AMO) mobile, esse sistema propõe ser acessível para grande maioria dos dispositivos, como desktops e tablets. Como o sistema vai ter as mesmas funcionalidades da sua versão mobile, a maioria dos requisitos foram reutilizados.

5.2 Analisar a documentação

Nesta fase aconteceu a análise da documentação, contendo a especificação dos requisitos, o guia de estilo e o protótipo inicial sugerido pelo projeto PAE para o sistema AMO. Cada documento foi fundamental para o desenvolvimento do sistema. A documentação de requisitos abrange as funcionalidades importantes da solução, enquanto o guia de estilo e o protótipo inicial constroem as diretrizes visuais e de usabilidade, sendo útil como base para a ideia da interface e da experiência do usuário. Com isso foram criadas reuniões entre os times *front-end, back-end e design* para analisar os requisitos funcionais em comparação com o protótipo desenvolvido, além de também analisar o guia de estilo e *design*. Foram reaproveitados 20 requisitos funcionais, conforme apresentado no Quadro 6.

Quadro 3 - Requisitos funcionais

Código	Título	Descrição	Status
[RF01]	Autenticação	O sistema deve permitir que o usuário insira um endereço de E-mail válido e uma senha para acessar a conta.	Implementado
[RF02]	Cadastrar usuário	O sistema deve permitir o cadastro do usuário	Implementado
[RF03]	Recuperar senha	O sistema deve permitir ao usuário recuperar sua senha.	Não Implementado
[RF04]	Selecionar curso	O sistema deve permitir que o usuário possa selecionar o curso na qual ele quer ver as disciplinas.	Implementado
[RF05]	Selecionar	O sistema deve permitir que o usuário	Implementado

Código	Título	Descrição	Status
	disciplina	possa selecionar a disciplina na qual ele quer ver as dúvidas.	
[RF06]	Listar as dúvidas	O sistema deve exibir todas as dúvidas da disciplina	Implementado
[RF07]	Cadastro de dúvida	O sistema deve permitir que o usuário cadastre uma dúvida.	Implementado
[RF08]	Apagar dúvida	O sistema deve permitir que o aluno criador da dúvida ou o monitor e professor exclua a dúvida selecionada.	Não Implementado
[RF09]	Responder dúvida	O sistema deve permitir que os usuários possam responder à dúvida.	Implementado
[RF10]	Marcar resposta como correta	O sistema deve permitir que o professor ou o monitor marque ou desmarque respostas como corretas.	Não Implementado
[RF11]	Filtrar dúvidas	O sistema deve permitir que os usuários possam filtrar dúvidas no fórum por mais ou menos curtidas e entre as mais recentes.	Implementado
[RF12]	Pesquisar dúvidas	O sistema deve permitir que o usuário pesquise uma dúvida no fórum pelo nome digitado em uma barra de pesquisa no topo da tela.	Implementado
[RF13]	Curtir dúvidas	O sistema deve permitir que os usuários curtam as dúvidas	Implementado
[RF14]	Trocar de disciplina	O sistema deve permitir ao usuário trocar de disciplina	Não Implementado
[RF15]	Solicitar agendamento	O sistema deve permitir que o aluno possa solicitar um agendamento.	Não Implementado
[RF16]	Visualizar agendamento	O sistema deve permitir que o usuário possa visualizar os dados do agendamento.	Não Implementado
[RF17]	Cancelar agendamento	O sistema deve permitir que o aluno, monitor e professor possa cancelar as solicitações de agendamentos.	Não Implementado
[RF18]	Editar dados do usuário	O sistema deve permitir que o usuário altere seus dados como email, data de nascimento, curso e foto de perfil	Não Implementado

Código	Título	Descrição	Status
[RF19]	Possuir Fórum	O sistema deve oferecer um fórum onde os usuários possam interagir entre si.	Implementado
[RF20]	Redefinir senha	O sistema deve permitir que o usuário redefina sua senha antiga para uma nova.	Não Implementado

Fonte: Elaborado pelo autor(2024).

5.3 Escolha do framework

A escolha de um framework é essencial para conseguir resultados satisfatórios durante o processo de desenvolvimento. A escolha do framework para utilizar neste trabalho foi fundamentada em uma pesquisa detalhada sobre fatores importantes, conforme relatado no estudo de Ferreira et al (2018).

5.3.1 Comparação de frameworks

De acordo com o trabalho de Ferreira et al (2018), que tem como finalidade, auxiliar desenvolvedores a escolher o framework mais adequado para seus projetos. Este trabalho avaliou de forma sensata os frameworks React, Angular e Vue, comparando suas documentações e comunidade, tamanho de arquivos gerados por cada tecnologia e tempo de renderização de páginas, como apresentados nos Quadros 3,4 e 5.

Quadro 4 - Disponibilidade de documentação e comunicação

	Angular	Vue	React
Fórum integrado	Não	Sim	Sim
Chat	Gitter	Discord	Discord
Github	Sim	Sim	Sim
StackOverflow	Sim	Sim	Sim

Fonte: Ferreira et al (2018).

Quadro 5 - Tamanho do arquivo JavaScript gerado

Framework	Tamanho do arquivo gerado
Angular	16 KB
Vue	11 KB
React	6 KB

Fonte: Ferreira et al (2018).

Quadro 6 -Tempo de renderização da página

Framework	Load
Angular	1.17 s
Vue	767 ms
React	788 ms

Fonte: Ferreira et al (2018).

Perante os resultados ilustrados foi possível perceber que todos eles apresentam uma ampla comunidade e documentação detalhada. No quesito de tamanho de arquivos e consumo de rede, o React apresentou superioridade em relação aos outros dois, já quando se trata de velocidade no carregamento da página o Vue tem um ganho mínimo imperceptível sobre o React, o Angular mesmo sendo um framework muito completo foi o que menos se destacou nessa pesquisa.

Como os frameworks comparandos são excelentes opções para desenvolvimento web, o autor deste trabalho escolheu o React por já ter experiência com a biblioteca através de outros projetos realizados, e também pela versão mobile do AMO ser construída com React-Native.

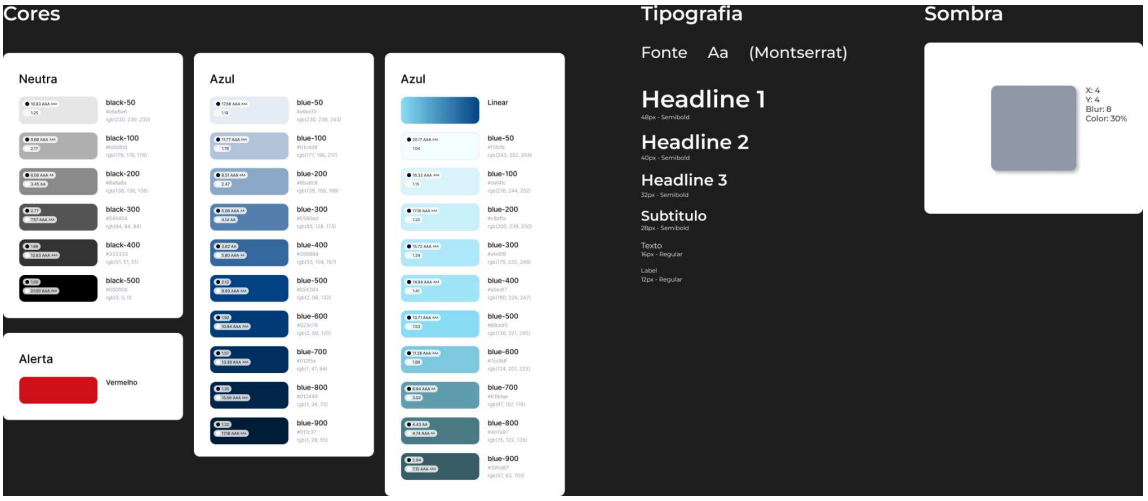
5.4 Definição do Projeto de Interface

Nesta etapa, foram definidos o estilo e o design a ser seguido durante o desenvolvimento do sistema.

O guia de estilos foi desenvolvido aproveitando a paleta de cores do AMO mobile, utilizando-se da ferramenta Figma. O guia é um conjunto de diretrizes que define a identidade visual contendo as paletas de cores a serem utilizadas, e a padronização do design de uma

aplicação, sendo essencial para garantir uma experiência do usuário. O guia de estilo está apresentado na Figura 3.

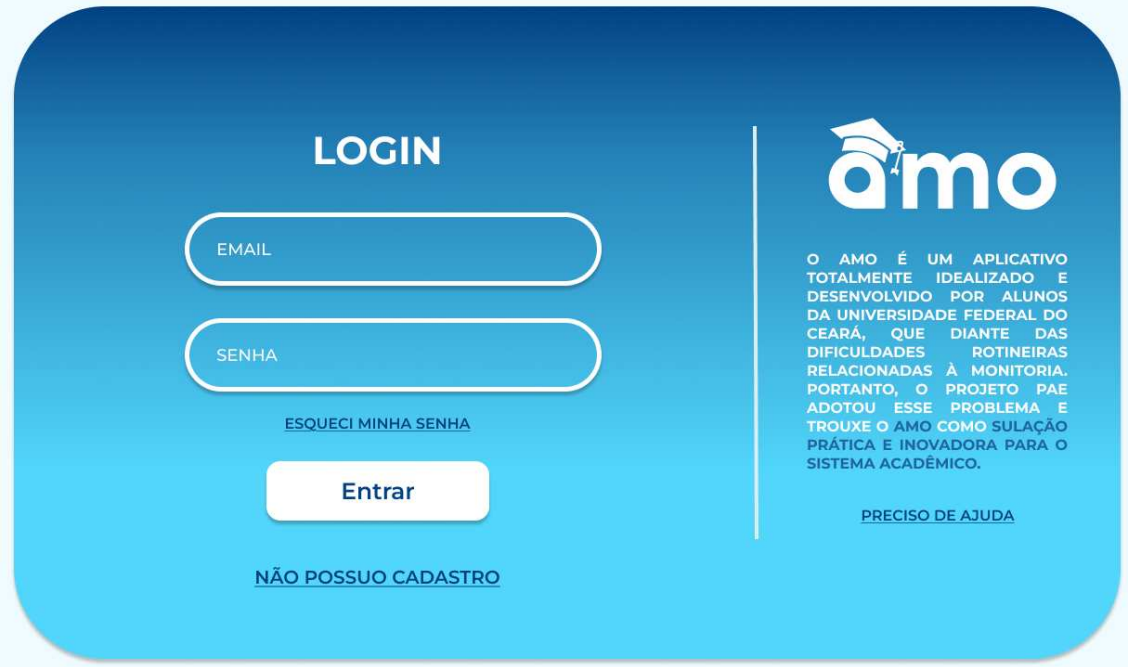
Figura 3 - Guia de estilo



Fonte: Elaborado pelo autor (2024).

Como exemplo do projeto de interface desenvolvido, foram especificados os protótipos de alta fidelidade do sistema. Os requisitos RF01, RF02, RF04 e RF19 foram prototipados usando-se da ferramenta Figma e são exibidos respectivamente nas Figuras 4, 5, 6 e 7. Observe que é possível notar que as cores do guia mostrados na Figura 3 foram aplicados, sendo prova da utilização do guia de estilo proposto.

Figura 4 - Tela de login

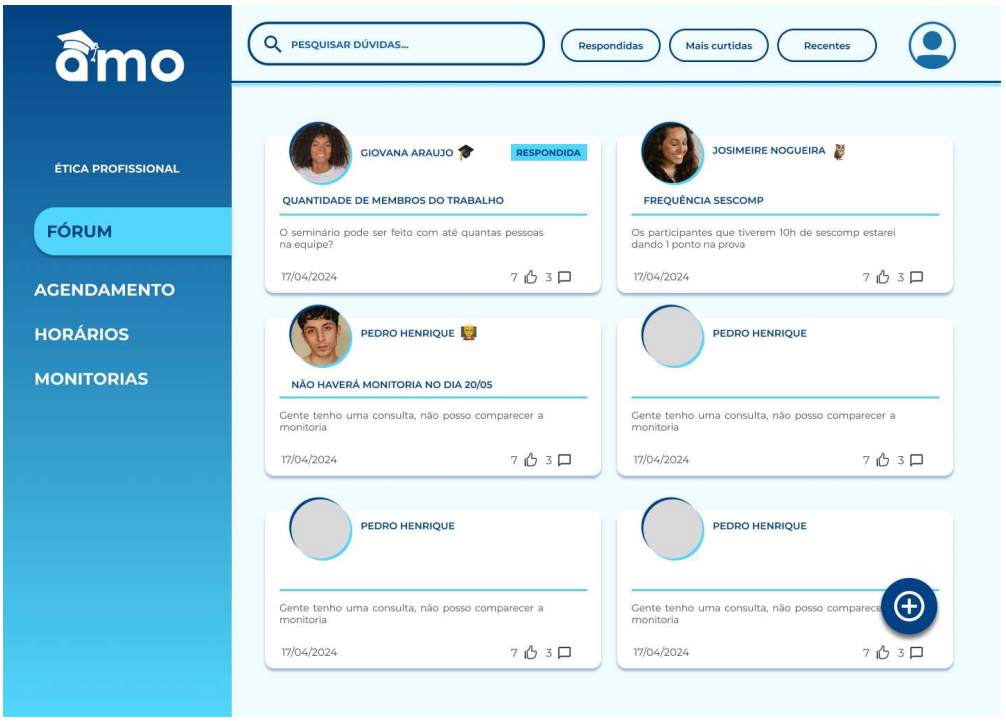


Fonte: Elaborado pelo autor (2024).

Figura 5 - Tela de Cadastro

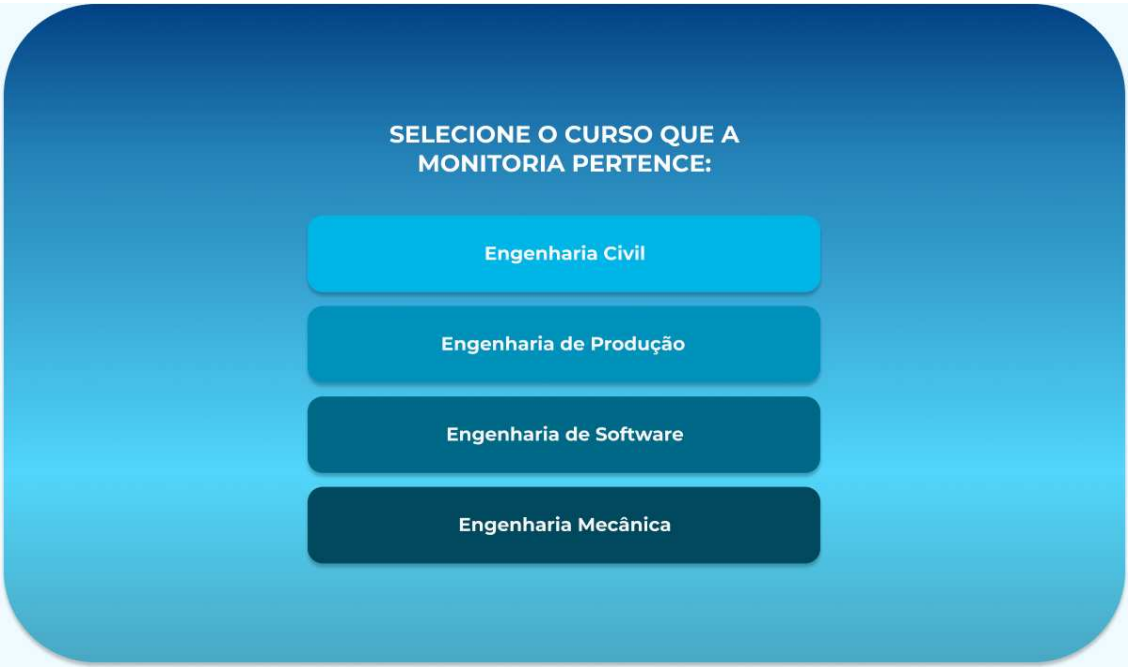
Fonte: Elaborado pelo autor (2024).

Figura 6 - Tela de Fórum



Fonte: Elaborado pelo autor (2024).

Figura 7 - Tela de selecionar curso



Fonte: Elaborado pelo autor (2024).

5.5 Desenvolvimento do sistema

Nesta etapa foi iniciado o desenvolvimento em si, iniciando pela criação dos componentes individuais da interface, com isso garantindo a reutilização do código. Prosseguindo teve a criação das telas do sistema, conforme as diretrizes fornecidas no protótipo e no guia, e por fim a integração com o back-end e as ferramentas auxiliares. Como o front-end já foi iniciado, detalhes serão apresentados na seção de resultados preliminares.

5.5.1 Criação de componentes

Nesta fase, foi realizado o planejamento e a criação de componentes reutilizáveis, que são partes cruciais da interface e podem ser inseridas em diversas áreas do sistema ou aplicação. A descoberta dos possíveis componentes mais visíveis no protótipo é essencial, uma vez que trará otimização no desenvolvimento, trazendo um código mais limpo e uniforme. Quando explorado de maneira eficiente os componentes reduzem o tempo de desenvolvimento e minimizam o trabalho entregando um código padrão do projeto. Sabendo disso, foram desenvolvidos elementos fundamentais como o botão principal para as ações primárias, os campos de entrada (inputs) para a coleta padronizada de dados, os cards de perguntas para a exibição da mesma no Fórum, os toasts de notificação para fornecer feedback ao usuário, e a barra de navegação (navbar). Juntos, esses componentes garantem a coesão visual e funcional em toda a aplicação.

5.5.2 Criação de telas iniciais

A criação das telas iniciais deve ser baseada no protótipo, seguindo fielmente o fluxo de navegação entre as telas. Além disso, é crucial seguir o guia de estilo definido. Durante essa fase, será feita a integração do maior número possível de componentes reutilizáveis, possibilitando não só acelerar o desenvolvimento, mas também vai disponibilizar uma experiência constante e coerente para o usuário.

5.5.3 Ferramentas auxiliares

As ferramentas auxiliares que foram utilizadas nesse trabalho foram Thunder Client, Visual Studio Code, Git e GitHub, ambos fornecendo um papel muito importante na otimização do desenvolvimento do projeto. O Visual Studio Code é um editor de código-fonte gratuito muito popular entre desenvolvedores que usam JavaScript, que oferece ferramentas

básicas para o desenvolvimento do código. O Thunder Client é uma extensão do Visual Studio Code com o objetivo de realizar os testes de APIs REST através de uma interface, possibilitando testar possíveis rotas do back-end e verificar dados retornados.

O Git, se trata de um sistema de controle de versão que concede ao usuário acompanhar modificações no código do projeto, identificando problemas e podendo alterar de forma prática as alterações do código. Já o GitHub é uma plataforma de hospedagem que facilita a colaboração, possibilitando o compartilhamento de trabalhos, e melhora o gerenciamento sobre as tarefas entre os desenvolvedores.

5.5.4 Integração com o back-end

Nesta fase foi realizada a integração com o back-end, logo após a finalização da interface web. Esse processo dá início a implementação de toda a lógica do sistema e as suas funcionalidades, como fazer perguntas e responder dúvidas no fórum. É importante ressaltar que será usado o mesmo back-end do AMO mobile, que é baseado na estrutura REST, e com isso será feita a comunicação do back-end com front-end.

5.6 Técnicas de verificação e validação

Nesta etapa do desenvolvimento, foram feitas as aplicações e análises dos resultados das técnicas de V&V, com foco em assegurar que o sistema a ser desenvolvido atenda aos padrões de qualidade e eficiência. A implementação dessas técnicas vai agregar no desenvolvimento, pois vai entregar dados relevantes sobre o comportamento da interface do projeto e possíveis inconsistências. Neste trabalho serão utilizadas as seguintes técnicas

Quadro 9 - Técnicas usadas

Técnicas	Descrição
Teste de Caixa Preta (Black-box Testing)	Relacionado a parte de Validação, esse teste visa avaliar as funcionalidades do sistema sem levar em consideração o código fonte
Revisão por Pares (Peer Review)	Relacionado a parte de Verificação, essa técnica tem o intuito de inspecionar o código e identificar possíveis erros antes da execução do sistema

Fonte: Elaborado pelo autor (2024).

6 RESULTADOS

Nesta seção são descritos os resultados obtidos a partir do desenvolvimento, iniciativa do sistema, aplicação do framework e caracterização do protótipo.

6.2 Definição do sistema

O sistema definido foi a versão web do aplicativo AMO, proposto no Projeto de Apoio ao Ensino para a comunidade acadêmica russana (PAE) da Universidade Federal do Ceará(UFC). A ideia da criação da plataforma web é aumentar o alcance e deixar o sistema disponível para multiplataformas, visto que o aplicativo mobile estava sofrendo na questão de atingir o público docente, uma vez que só dispositivos móveis poderiam usar.

6.3 Escolha do Framework

O sistema desenvolvido neste trabalho trata-se de uma versão web do aplicativo AMO já existente, mantendo a mesma proposta funcional, baseado em uma arquitetura REST, como a versão mobile foi feita em React-Native, o framework mais adequado para esse trabalho seria o React, que é voltado para desenvolvimento de aplicações web

Assim, o framework escolhido foi o React, que foi comparado com outros frameworks no trabalho de teste(2021). Além disso, considerando as habilidades prévias dos integrantes da equipe, que já possuem habilidades em JavaScript e desenvolvimento web com React.

6.4 Criação dos componentes

Nesta etapa, foram desenvolvidos os componentes reutilizáveis para diversas telas do sistema. Essa abordagem garante que o código desses componentes seja facilmente customizável e reutilizável, sem a necessidade de reescrita completa, otimizando o tempo de desenvolvimento e evitando retrabalho. Alguns componentes podem ser vistos nas Figuras

Figura 8 - Componente Card de pergunta

Fonte: Elaborado pelo autor(2025).

Além de realizar a criação de componentes reutilizáveis, também foram feitos hooks próprios, hooks fazem parte do sistema nativo da biblioteca React criados para chamar funções que podem ser reaproveitadas em várias partes do código. Os hooks são essenciais para o funcionamento eficiente do React e sua criação personalizada pode ser vantajosa. Eles permitem encapsular lógicas de componentes de forma independente, promovendo a reutilização e a organização do código de maneira mais eficaz.

O código do hook apresentado na Figura 9 apresenta o sistema de filtrar as perguntas do Fórum, em recentes, respondidas e mais curtidas, mudando em tempo real, já na figura mostra a função para implementar a funcionalidade de filtrar.

Figura 9 - Código do filtro para o fórum

```
export default function Forum() {
  const [filters, setFilters] = useState({
    recent: true,
    responded: false,
    mostLiked: false,
    text: "",
  });
```

Fonte: Elaborado pelo autor(2025).

Figura 10 - Função que realiza a mudança do filtro do fórum

```
306   // Função para alternar filtros
307   const handleFilterChange = (filterType) => {
308     setFilters((prev) => {
309       // Primeiro, desativa todos os filtros
310       const newFilters = {
311         recent: false,
312         responded: false,
313         mostLiked: false,
314         text: prev.text, // Mantém o texto de busca
315       };
316
317       // Ativa apenas o filtro selecionado (ou desativa se já estiver ativo)
318       if (!prev[filterType]) {
319         newFilters[filterType] = true;
320       }
321
322       return newFilters;
323     });
324   };
325
326   // Função para alterar texto de busca
327   const handleSearchChange = (searchText) => {
328     setFilters((prev) => ({
329       ...prev,
330       text: searchText,
331     }));
332   };
333
```

Fonte: Elaborado pelo autor(2025).

E para evitar repetição de código, foram criados módulos de códigos separados que podem ser reutilizados em diferentes partes da aplicação. Isso torna o código mais organizado e facilita a manutenção. Um exemplo é o código da Figura 11, que se trata de um hook personalizado do React que fornece acesso ao contexto de autenticação de todo o sistema, permitindo que qualquer componente da aplicação consuma as funções e dados de autenticação.

Figura 11 - Código do contexto de autenticação do sistema

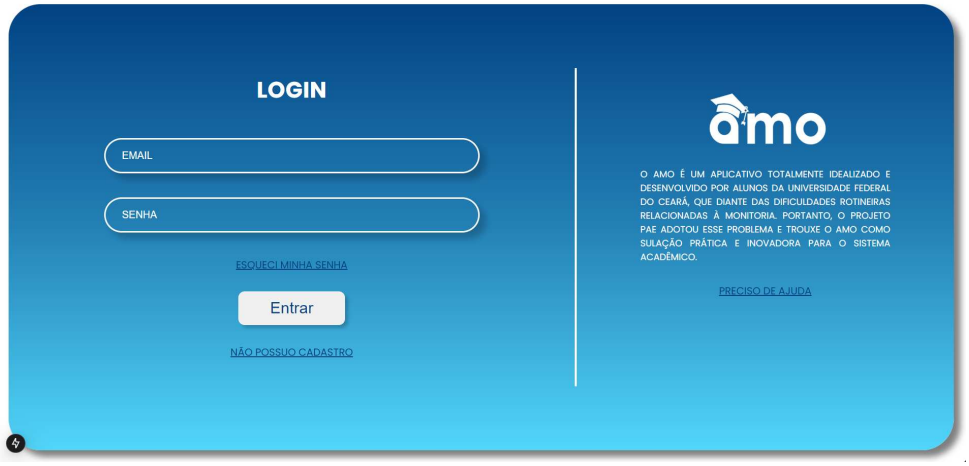
Fonte: Elaborado pelo autor(2025).

6.5 Criação das telas

Nesta fase do desenvolvimento, foram construídas as telas do sistema com base no protótipo. Para isso, foram utilizados componentes e módulos reutilizáveis sempre que possível, o que contribui para uma estrutura mais organizada, reduz o retrabalho e facilita a manutenção do sistema no futuro. As principais interfaces implementadas nesta etapa estão relacionadas à autenticação de usuários, ao fórum.

Na tela de login, o usuário deve inserir suas credenciais para acessar o sistema. Caso ainda não possua uma conta, ele poderá criá-la clicando em não possuo cadastro, que vai redirecionar para tela de registro, onde será informado um e-mail e uma senha. Em seguida, o sistema solicita que o usuário insira um código de verificação enviado por e-mail, a fim de validar o endereço informado. Por fim, o usuário é direcionado a uma tela onde deve completar seu cadastro inserindo suas informações pessoais. Essas telas estão ilustradas nas Figuras 12,13 e 14.

Figura 12 - Tela de login implementada



Fonte: Elaborado pelo autor(2025).

Figura 13 - Tela de cadastro implementada

Fonte: Elaborado pelo autor (2025).

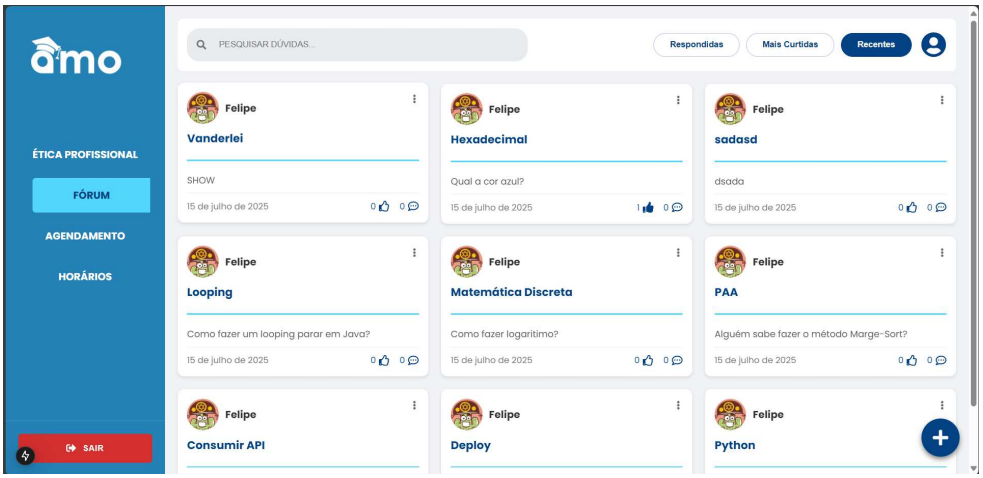
Figura 14 - Tela de completar cadastro implementada

Fonte: Elaborado pelo autor (2025).

A Figura 15 ilustra a interface principal do fórum, um dos módulos centrais do aplicativo. Nessa tela, o usuário pode visualizar as perguntas publicadas por outros usuários, bem como o número de curtidas e comentários relacionados a cada uma. Já na Figura 16 exibe a tela de cadastro de dúvidas no fórum.

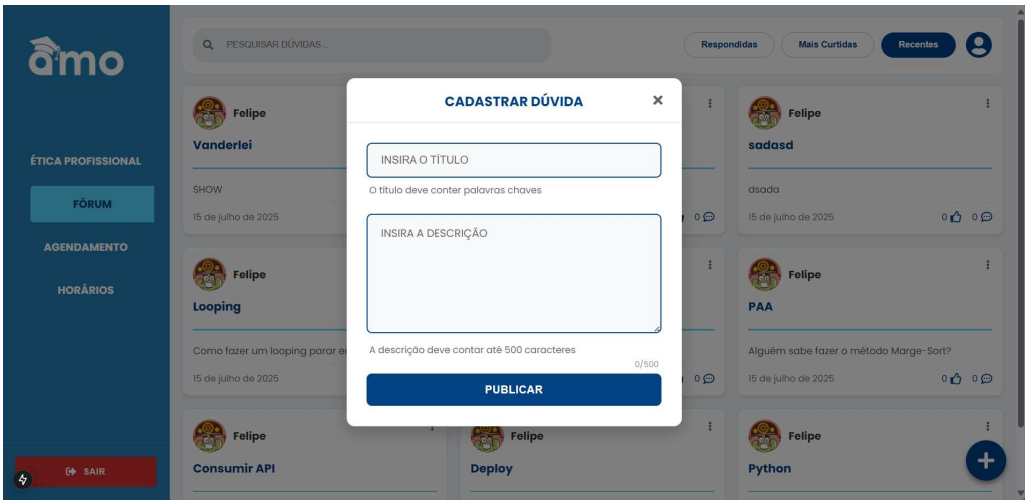
A Figura 17 demonstra a seção de comentários de uma pergunta específica, onde os usuários podem interagir entre si, respondendo à pergunta inicial ou comentando as respostas de outros participantes.

Figura 15 - Tela de Fórum implementada



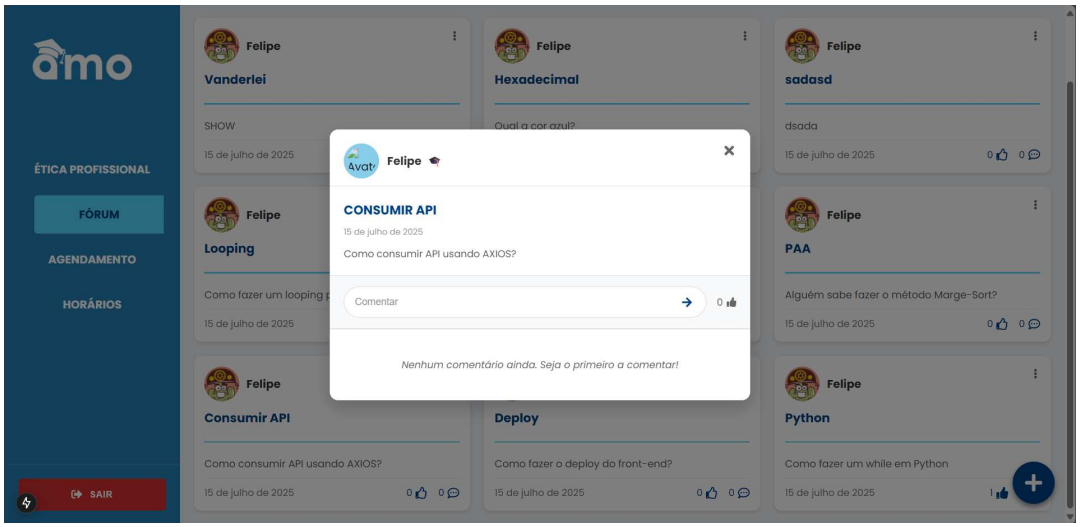
Fonte: Elaborado pelo autor(2025).

Figura 16 - Componente Cadastrar dúvida



Fonte: Elaborado pelo autor(2025).

Figura 17 - Componente responder dúvida



Fonte: Elaborado pelo autor(2025).

6.6 Integração com o back-end

A comunicação com o back-end é realizada por meio de uma URL previamente definida. Para tornar o código mais limpo e evitar repetição desnecessária dessa URL em diferentes trechos do sistema, foi criado um módulo denominado "api", responsável por centralizar esse endereço. Com isso, todas as requisições REST, como GET, POST, PUT, entre outras, podem ser feitas de maneira padronizada e organizada. Esse módulo centralizado pode ser visualizado na Figura 18.

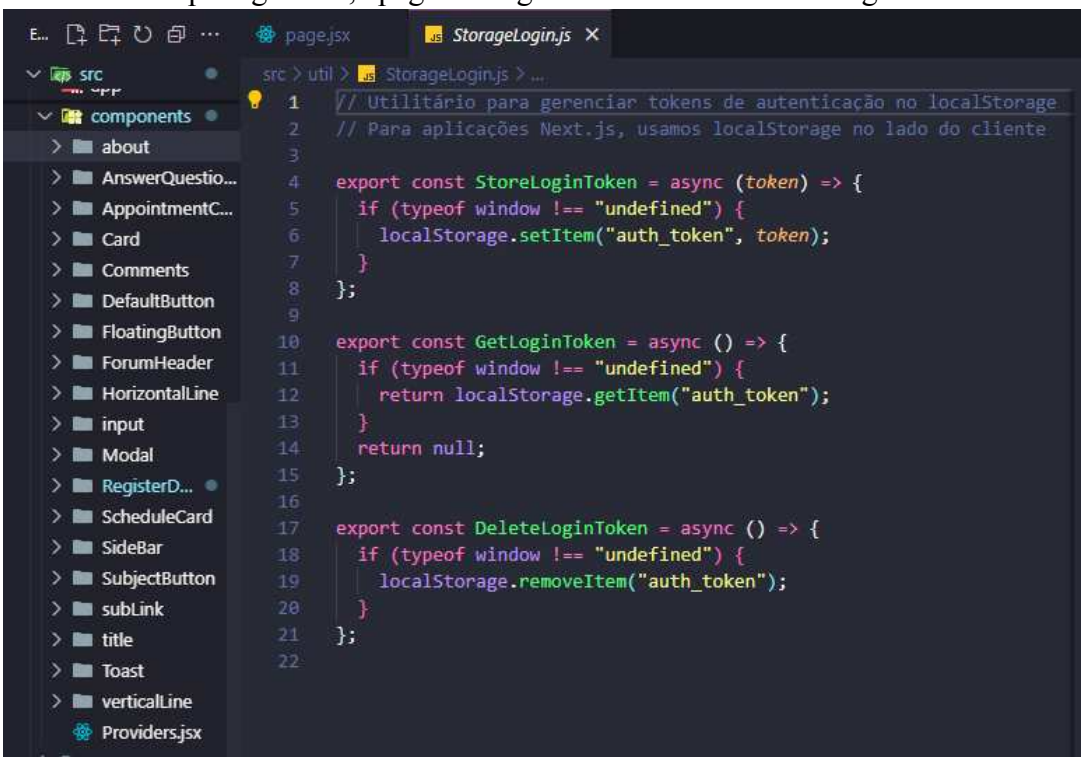
Figura 18- Módulo para realizar requisições a API.

```
1  import axios from "axios";
2
3  const api = axios.create({
4    baseURL: "http://127.0.0.1:8000/",
5    headers: {
6      "Content-Type": "application/json",
7      Accept: "application/json",
8    },
9    withCredentials: false, // Importante para CORS
10   timeout: 10000, // Timeout de 10 segundos
11 });
12
```

Fonte: Elaborado pelo autor(2025).

Também é utilizado o *hook context* do React que desempenha um papel crucial na integração com o *back-end*, evitando enviar requisições em excesso para o servidor assim minimizando o uso do mesmo. Durante o processo de autenticação, o *token* de autenticação do usuário é armazenado localmente no localStorage do navegador, a Figura 29 mostra as principais funções no código relacionadas ao *token*. Isso elimina a necessidade de realizar login repetidamente a cada acesso ao sistema, já que o *token* é enviado automaticamente em cada requisição.

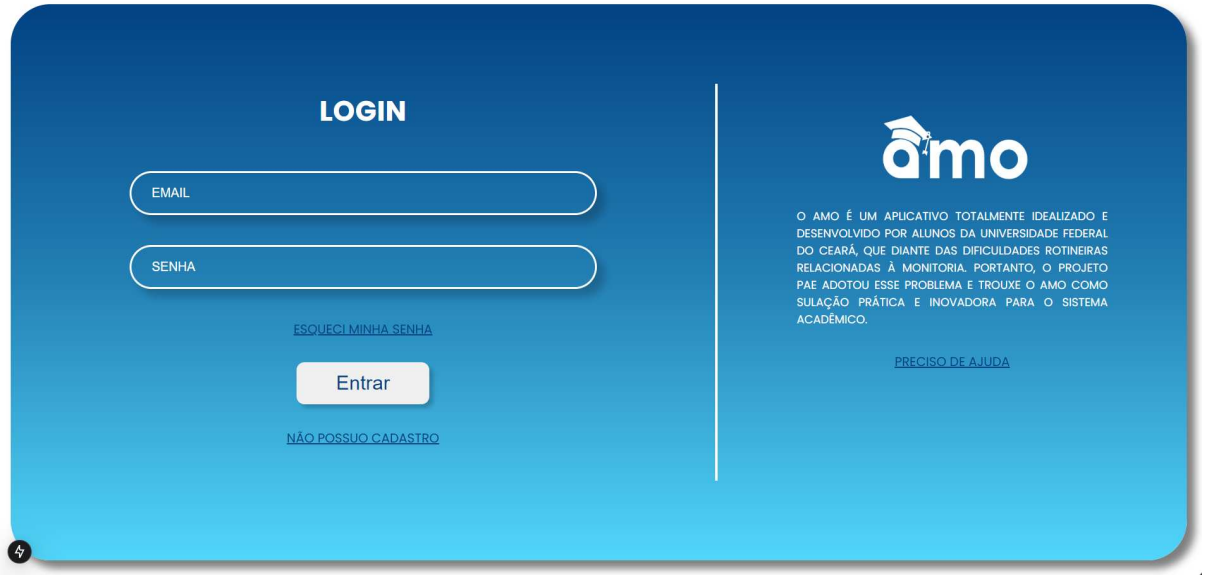
Figura 19- Módulo para guardar, apagar e resgatar o token do localStorage



Fonte: Elaborado pelo autor(2025).

Com base nisso as informações relacionadas ao usuário que está utilizado o aplicativo também são armazenadas após realizar a autenticação no sistema por meio da tela de login mostrada na Figura 20. Dessa forma, não é necessário requisitar constantemente ao servidor as informações sobre o usuário como nome ou seu cargo. Essas informações são armazenadas na memória e acessadas conforme necessário através do Context como pode ser visto na Figura 21, proporcionando uma experiência mais ágil e eficiente para o usuário e otimizado para o front-end e back-end.

Figura 20- Tela de login



Fonte: Elaborado pelo autor(2025).

Figura 21- Context que guarda os dados do usuário.

```
async function Login(user) {
  try {
    console.log("🔄 Tentando fazer login...");

    const response = await api.post(API_ENDPOINTS.LOGIN, {
      username: user.email,
      password: user.password,
    });

    console.log("✅ Resposta recebida:", response.status);

    const token = response.data.token;
    const userData = await GetUser(token);

    setUser({ ...userData });

    await StoreLoginToken(token);

    if (
      userData.perfil.curso === null ||
      userData.perfil.entrada === null ||
      userData.perfil.nome_completo.length < 1
    ) {
      return { erro: "usuario incompleto!" };
    }

    return undefined;
  } catch (error) {
    console.error("❌ Erro no login:", error);
```

Fonte: Elaborado pelo autor(2025).

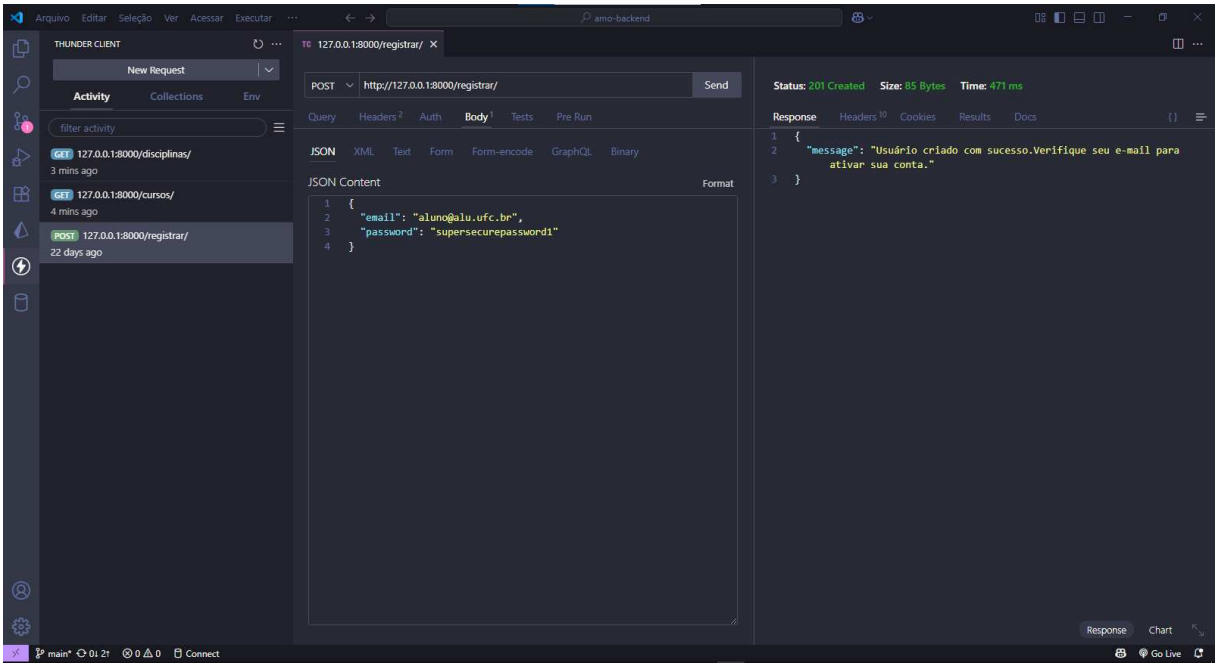
6.7 Uso de ferramentas auxiliares

Nesta seção, são apresentadas as ferramentas que foram essenciais para o desenvolvimento do sistema. Dentre elas, destaca-se o Thunder Client, uma extensão do Visual Studio Code utilizada amplamente durante o processo de integração com o back-end. Essa ferramenta permite testar de forma prática e rápida as rotas de APIs REST diretamente no ambiente de desenvolvimento, facilitando a verificação de requisições HTTP como GET, POST, PUT e DELETE (Thunder Client, 2025).

O Thunder foi utilizado para testar rotas antes da implementação real no front-end, sendo assim, por meio da extensão era possível visualizar uma prévia de como as rotas funcionam e se estavam conforme o necessário para serem utilizadas no sistema.

Na Figura 22 é possível observar o painel principal da extensão, onde tem a exibição das principais rotas de *back-end* listadas. Sempre que uma nova rota era implementada, ela passava por um teste antes e assim tinha-se o resultado prévio do funcionamento da API para ser incorporada ao sistema. Essa abordagem sequencial garantiu um desenvolvimento eficaz e confiável na incorporação do *back-end* ao aplicativo, assim garantindo otimização e confiabilidade

Figura 22 - Painel do Thunder Client



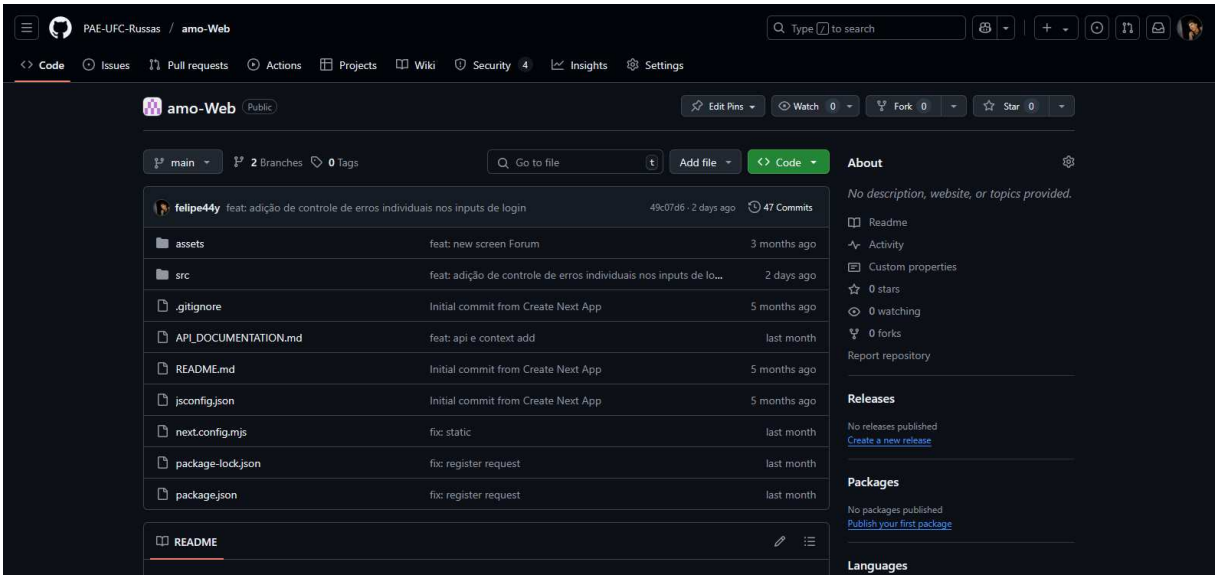
Fonte: Elaborado pelo autor(2025).

O Git foi uma tecnologia fundamental para o controle de versão do sistema. Com ele, foi possível manter um registro organizado das diferentes versões do código, permitindo acompanhar o histórico de adições, remoções e mudanças realizadas ao longo do desenvolvimento. Essa abordagem possibilitou acompanhar o progresso do projeto e garantir que o trabalho estivesse alinhado com as funcionalidades definidas na documentação.

O código precisava ser disponibilizado ao público e a outros membros do projeto por ser open-source e a ferramenta escolhida foi o GitHub, uma plataforma online essencial para o compartilhamento de código fonte, permitindo que os desenvolvedores hospedem seus repositórios publicamente em um banco de dados online.

Utilizando o Git para controle de versão, a plataforma possibilita também que outros desenvolvedores contribuam para os projetos. O GitHub é a ferramenta mais utilizada para o gerenciamento de controle de versão online (Cosentino, 2017). O código está disponível atualmente no site como é possível ver na Figura 23, onde os membros do projeto realizavam o desenvolvimento colaborativo da aplicação.

Figura 23 - Painel do projeto do sistema no Github.



Fonte: Elaborado pelo autor(2025).

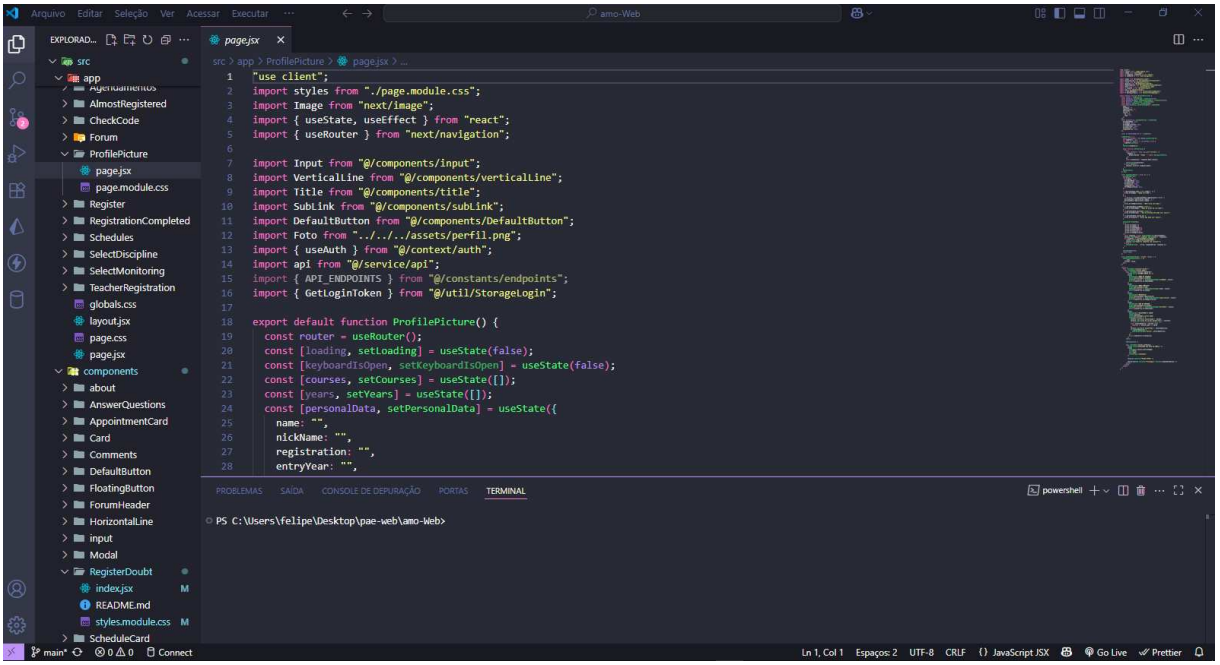
O Visual Studio Code foi outra ferramenta utilizada no desenvolvimento do sistema. Ele trata-se de um editor de código-fonte leve, multiplataforma e altamente extensível, que oferece suporte a diversas linguagens de programação e conta com uma ampla gama de extensões que otimizam o processo de desenvolvimento. Com sua interface intuitiva e recursos integrados como realce de sintaxe, terminal embutido e integração com sistemas de

controle de versão como o Git, o Visual Studio Code contribuiu significativamente para a produtividade e organização do projeto.

Durante o desenvolvimento, o editor foi configurado com extensões específicas, como o Thunder Client, para facilitar o teste das rotas da API, além de ferramentas de formatação automática e análise de código que ajudaram a manter a padronização e qualidade do código-fonte. Sua leveza e rapidez também permitiram um fluxo de trabalho mais eficiente, tanto em máquinas pessoais quanto nos ambientes de desenvolvimento da equipe.

Considerado um dos ambientes de desenvolvimento mais populares da atualidade, o Visual Studio Code destaca-se por sua flexibilidade, grande comunidade de usuários e constante atualização por parte da Microsoft (Microsoft, 2025).

Figura 24 - Interface do Visual Studio Code.



Fonte: Elaborado pelo autor(2025).

6.8 Aplicação das técnicas de verificação e validação

Nesta etapa são apresentados os resultados das aplicações das técnicas de verificação e validação. As técnicas utilizadas foram: teste de caixa preta e revisão por pares.

6.8.1 Teste de caixa preta

Os testes de caixa preta foram aplicados com o objetivo de validar o comportamento do sistema web de monitoria a partir da perspectiva do usuário final, sem considerar a lógica interna do código. Essa técnica consiste em fornecer entradas ao sistema e observar se as saídas correspondem aos resultados esperados, com base nos requisitos funcionais definidos.

Para a aplicação dos testes, foram definidos três cenários principais baseados nas funcionalidades principais que já estão implementadas no sistema, tais como autenticação, cadastro de usuários, seleção de monitorias e fazer perguntas no fórum. Em cada cenário, foram utilizados dados válidos e inválidos, de modo a verificar se o sistema se comporta corretamente em diferentes situações.

Na funcionalidade de login, foram aplicados testes de caixa preta para verificar o comportamento do sistema diante de diferentes tipos de entrada de dados. O objetivo foi garantir que o sistema respondesse adequadamente tanto a dados válidos quanto a entradas inválidas ou incompletas.

1. Entrada válida

- Descrição: Usuário preenche o campo de e-mail e senha com credenciais corretas, previamente cadastradas no sistema.
- Resultado esperado: O sistema realiza a autenticação e redireciona o usuário para a página de seleção de Cursos.
- Resultado obtido: O sistema se comportou conforme o esperado, direcionando corretamente o usuário para a tela de seleção de cursos que está sendo mostrada na Figura 25.

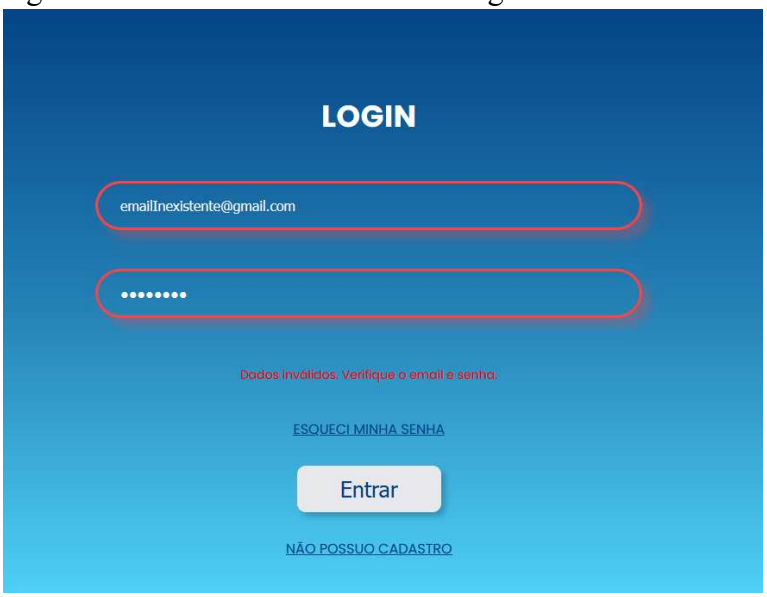
Figura 25 - Tela de seleção de curso

Fonte: Elaborado pelo autor(2024).

2. Entrada inválida

- Descrição: Usuário informa um e-mail inexistente ou uma senha incorreta.
- Resultado esperado: O sistema não realiza o login e exibe uma mensagem de erro informando que os dados estão incorretos.
- Resultado obtido: O sistema apresentou a mensagem de erro conforme o esperado, impedindo o acesso.

Figura 26- Feedback de erro na tela login

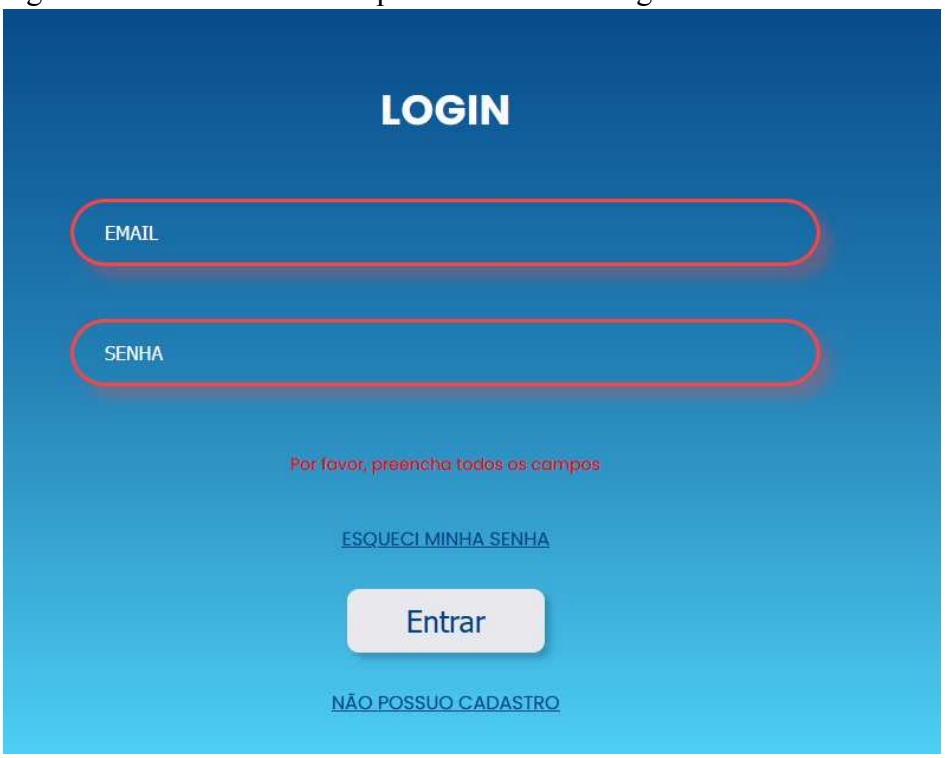


Fonte: Elaborado pelo autor(2024).

3. Entrada vazia.

- Descrição: O usuário tenta realizar o login deixando o campo de e-mail e/ou senha em branco.
- Resultado esperado: O sistema bloqueia a tentativa de login e exibe uma mensagem solicitando o preenchimento dos campos obrigatórios.
- Resultado obtido: O sistema respondeu corretamente, impedindo o envio do formulário e exibindo a mensagem de aviso.

Figuras 27- Feedback de campos vazios na tela login



Fonte: Elaborado pelo autor(2024).

A tela de cadastro foi submetida a testes de caixa preta com o intuito de validar a criação de novos usuários no sistema, garantindo que os dados fornecidos fossem tratados corretamente e que os erros comuns fossem adequadamente identificados e comunicados ao usuário.

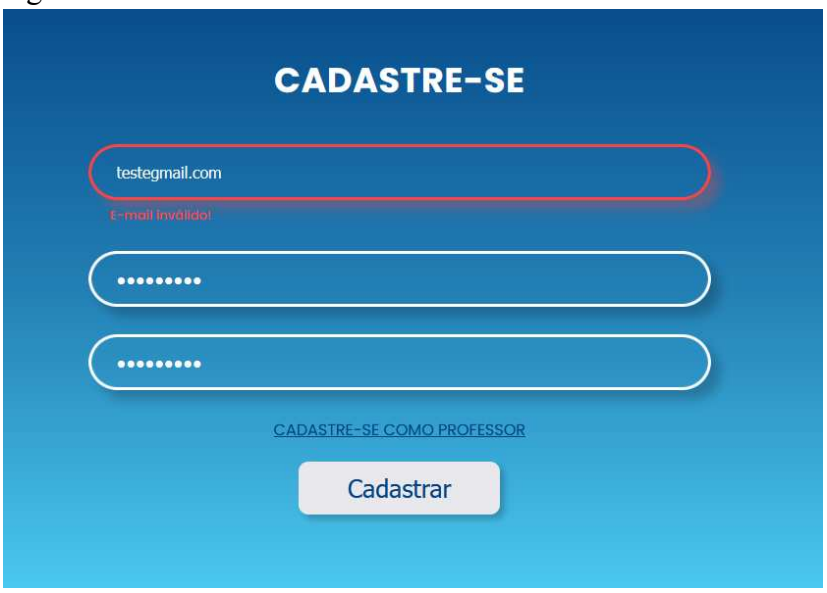
1. Entrada válida

- Descrição: O usuário preenche corretamente todos os campos obrigatórios, incluindo nome, e-mail em formato válido, senha com no mínimo seis caracteres e seleção do tipo de usuário.
- Resultado esperado: O sistema cadastra o novo usuário e redireciona para a tela de login com uma mensagem de confirmação.
- Resultado obtido: O sistema se comportou conforme o esperado, realizando o cadastro e direcionando adequadamente.

2. Entrada inválida

- Descrição: O usuário tenta se cadastrar informando um e-mail em formato incorreto (ex: "usuario@") ou uma senha com menos de seis caracteres.
- Resultado esperado: O sistema exibe mensagens de erro específicas para cada campo inválido, impedindo a conclusão do cadastro.
- Resultado obtido: O sistema apresentou as mensagens de erro corretamente e bloqueou a submissão do formulário até a correção dos dados.

Figuras 28- Erro no formato de email na tela cadastro.



Fonte: Elaborado pelo autor(2024).

3. E-mail duplicado

- Descrição: O usuário tenta se cadastrar utilizando um endereço de e-mail já existente no banco de dados.
- Resultado esperado: O sistema impede o cadastro e exibe uma mensagem de erro informando que o e-mail já está em uso.
- Resultado obtido: O sistema respondeu conforme o esperado, impedindo a duplicidade e exibindo o alerta de e-mail já registrado.

Figuras 29- Erro de email já existente.

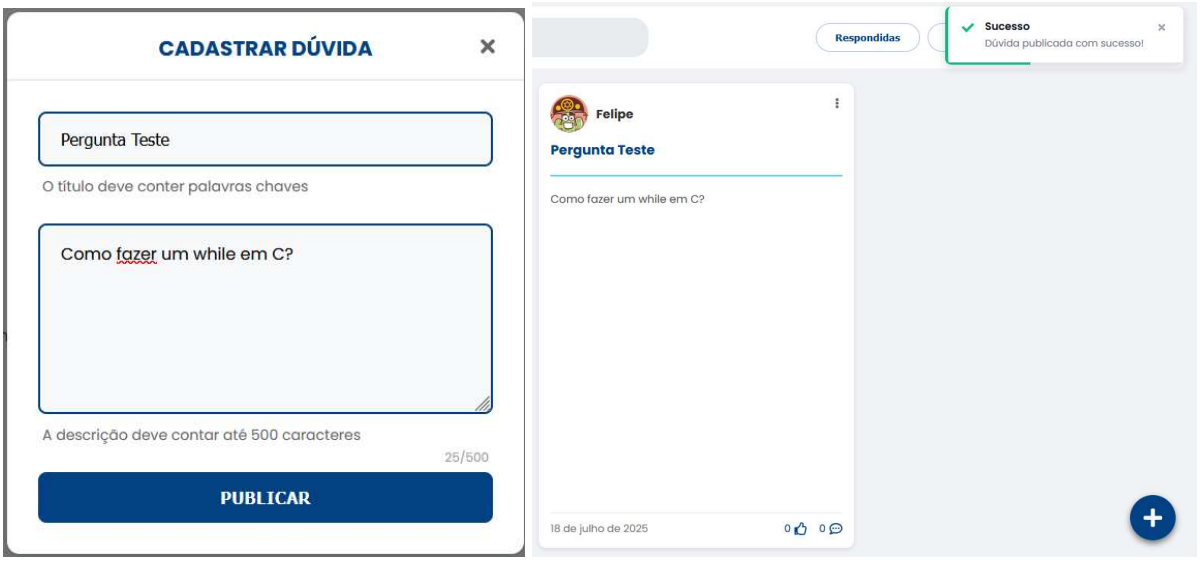
Fonte: Elaborado pelo autor(2024).

A funcionalidade de fórum de discussões do sistema foi testada com o objetivo de validar o envio e exibição de perguntas entre usuários, garantindo que o sistema permita a comunicação eficiente entre alunos, monitores e professores. Os testes buscaram verificar tanto o comportamento do sistema diante de mensagens válidas quanto a forma como ele lida com tentativas de envio inválidas.

1. Entrada válida

- Descrição: O usuário preenche o campo de texto com uma dúvida clara e relevante e envia a mensagem.
- Resultado esperado: O sistema publica a mensagem no fórum e ela se torna visível para os demais usuários, sendo incluída na lista de discussões.
- Resultado obtido: O sistema apresentou o comportamento esperado, exibindo a mensagem imediatamente após o envio.

Figura 30- Telas relacionadas a cadastrar dúvida e fórum.



Fonte: Elaborado pelo autor(2024).

2. Entrada vazia

- Descrição: O usuário tenta enviar uma mensagem sem digitar nenhum conteúdo no campo de texto.
- Resultado esperado: O sistema bloqueia o envio e dá um feedback deixando os campos obrigatórios em destaque.
- Resultado obtido: O sistema respondeu corretamente, impedindo o envio da mensagem em branco e exibindo o alerta de preenchimento necessário.

Figuras 31- Erro de entrada vazia.

Fonte: Elaborado pelo autor(2024).

6.8.3 Revisão por Pares (Peer Review)

Com o objetivo de complementar os testes funcionais e garantir maior confiabilidade ao sistema desenvolvido, foi realizada uma revisão por pares (peer review). Essa técnica consistiu na análise crítica do sistema por duas pessoas com conhecimento técnico relevante, mas que não participaram diretamente de todas as etapas do desenvolvimento. O processo de revisão por pares seguiu os seguintes passos:

Preparação: Inicialmente, foram fornecidos aos revisores o acesso ao sistema web de monitoria acadêmica, juntamente com a documentação relevante (como os requisitos funcionais). Foi solicitado que cada revisor analisasse as funcionalidades e usabilidade do sistema.

Revisão Individual: Cada revisor realizou sua análise de forma individual e independente. Eles foram orientados a documentar quaisquer bugs, inconsistências, sugestões de melhoria ou pontos de atenção em um relatório padronizado, que incluía a descrição do problema, a severidade e a sugestão de solução.

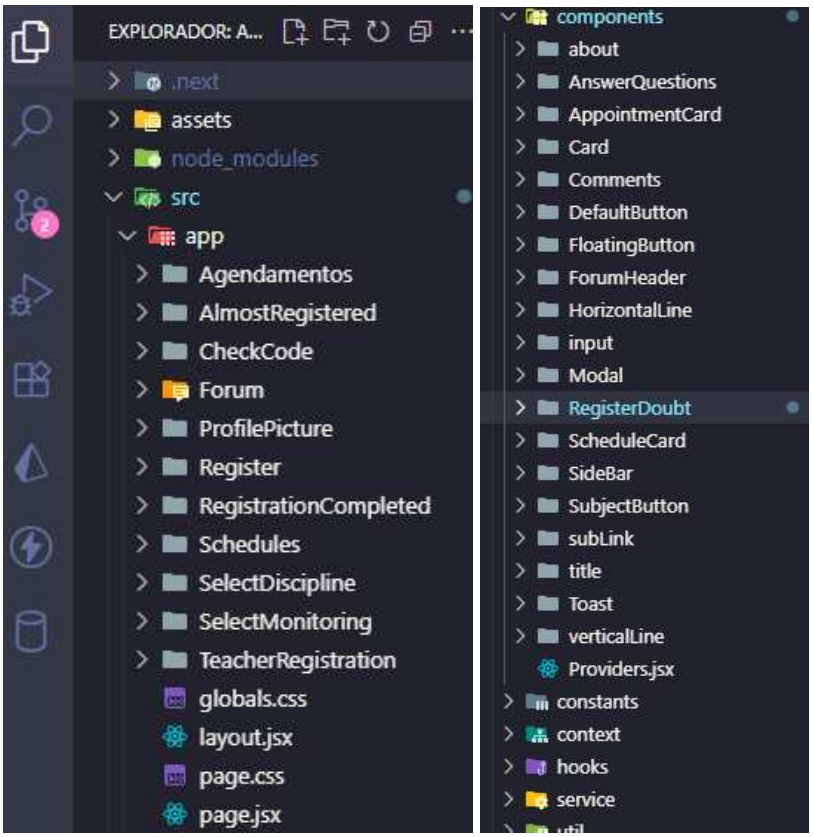
Consolidação e Discussão Conjunta: Após a conclusão das revisões individuais, foi agendada uma reunião em conjunto com os dois alunos e a equipe de desenvolvimento. Nesta reunião, os pontos levantados foram discutidos detalhadamente, permitindo esclarecer dúvidas, priorizar as correções e validar as sugestões apresentadas.

Para garantir a qualidade das análises, os revisores foram selecionados com base em dois perfis distintos. O primeiro, com um perfil interno, era um integrante da própria equipe de desenvolvimento com conhecimento específico na área de *back-end e* banco de dados. Sua contribuição foi focada em uma análise técnica aprofundada da integração e da qualidade do código das funcionalidades nas quais não atuou diretamente. Já o segundo revisor, com um perfil externo, era um aluno do 8º semestre do curso de Engenharia de Software da Universidade Federal do Ceará – Campus Russas, com conhecimentos em disciplinas como Qualidade de Software e Engenharia de Usabilidade. Essa combinação de perfis permitiu um feedback balanceado, cobrindo tanto os aspectos técnicos internos quanto a perspectiva do usuário final.

Durante a revisão, os participantes avaliaram:

- A organização do código-fonte que deve incluir clareza na nomenclatura, estrutura dos arquivos e reutilização de componentes.

Figuras 32- Estrutura do código-fonte.



Fonte: Elaborado pelo autor (2024).

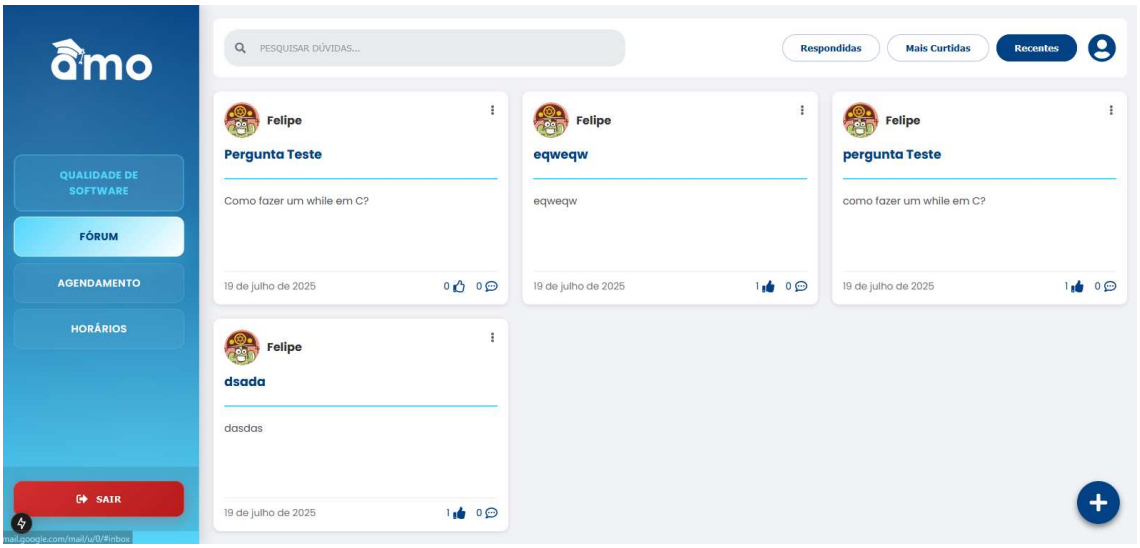
- A usabilidade do sistema, considerando a clareza das mensagens exibidas, a disposição dos elementos nas telas e o feedback fornecido ao usuário durante as interações.

Figuras 33- Feedback mais claro de registrar dúvida.

Fonte: Elaborado pelo autor(2024).

- A aderência do sistema aos requisitos definidos, verificando se as funcionalidades implementadas estavam em conformidade com os objetivos propostos. O requisito avaliado foi a integração de um Fórum.

Figuras 34 - Implementação da tela Fórum.



Fonte: Elaborado pelo autor(2024).

Ambos os revisores demonstraram satisfação com o sistema apresentado, destacando a boa estruturação do projeto e a implementação das funcionalidades propostas. Durante a análise, sugeriram pequenas melhorias voltadas à usabilidade, como a reformulação de mensagens exibidas ao usuário para torná-las mais claras e objetivas, além de ajustes pontuais no layout de algumas telas. As sugestões foram consideradas pertinentes e implementadas, contribuindo para tornar a experiência do usuário mais fluida e intuitiva.

Essa atividade contribuiu significativamente para a melhoria da qualidade do sistema, permitindo identificar detalhes que poderiam passar despercebidos por quem desenvolveu diretamente a aplicação. A revisão por pares serviu, portanto, como uma ferramenta de verificação eficaz e complementar aos testes de caixa preta aplicados.

7 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Este trabalho concentrou-se na construção da versão web do aplicativo AMO projetado para apoiar a monitoria acadêmica. O objetivo principal foi trazer um sistema que possa ser utilizado em diferentes plataformas, principalmente para desktops, a fim de superar as limitações de compatibilidade entre diferentes sistemas operacionais e dispositivos.

Com base nisto, foi desenvolvido o sistema, que oferece as mesmas funcionalidades da versão mobile, as principais como um fórum online para aprendizado colaborativo entre alunos, monitores e professores.

A elaboração deste trabalho iniciou-se com o reaproveitamento e a análise dos requisitos definidos para o aplicativo mobile. Essa abordagem teve como objetivo não apenas aprimorar a interface de usuário, mas também expandir o alcance da multiplataforma da solução. Com base nessa análise, um novo protótipo para a versão web foi recriado e validado, assegurando que atendesse aos requisitos já existentes O trabalho de Ferreira et al (2018), faz uma comparação entre frameworks, apontando o React como uma boa escolha, outro ponto a ser considerado foi que os desenvolvedores possuíam experiência com essa tecnologia.

Com a caracterização da interface e a definição do *framework* concluídas, era iniciado o desenvolvimento em si do front-end da aplicação. A ênfase foi colocada na criação de componentes reutilizáveis, que otimizou o tempo de codificação e diminuiu significativamente o retrabalho. Além disso, a integração com o *back-end* foi feita por meio de comunicação via API Rest. O sistema já está com autenticação, cadastro de usuário e Fórum integrados, faltando apenas a parte de agendamento e horários de monitorias. Os testes de Validação e Verificação (V&V) foram feitos nas funcionalidades já implementadas e os resultados obtidos demonstram que o sistema apresentou uma boa usabilidade, No entanto, algumas mensagens de feedback precisam ser aprimoradas.

Uma das principais dificuldades enfrentadas no decorrer do projeto foi o consumo da API REST e a integração do *front-end* com o *back-end*. Esse desafio envolveu a configuração de requisições assíncronas, Tratamento de respostas e erros, e a garantia de que a comunicação entre as camadas da aplicação ocorresse de forma fluida e segura. A partir dessas dificuldades, uma lição aprendida crucial foi a importância dos testes de integração entre o *front-end* e o *back-end* desde as fases iniciais do desenvolvimento, nesse contexto, a utilização da extensão Thunder Client foi fundamental, ela permitiu realizar requisições independentes ao *back-end* e validar as respostas da API de forma isolada, agilizando a

identificação e correção de problemas de comunicação antes mesmo da implementação completa do *front-end*.

É importante ressaltar as limitações do sistema em sua versão atual. Embora as funcionalidades de autenticação, cadastro e fórum de discussões estejam implementadas e testadas, o protótipo não contempla a totalidade dos requisitos levantados. Funcionalidades como agendamento de monitorias e perfis detalhados de alunos e monitores foram projetadas, mas não implementadas, devido às restrições de tempo e escopo do trabalho. Essas limitações abrem caminho para futuros aprimoramentos.

Para trabalhos futuros, propõe-se a integração dos módulos de agendamentos, horários de monitoria e tela de perfil do aluno, que são requisitos já previstos no sistema. Além disso, sugere-se a implementação de um sistema de notificações para informar os usuários sobre respostas às suas perguntas, bem como sobre a aceitação ou o cancelamento de agendamentos, com o objetivo de mantê-los sempre atualizados quanto às suas interações na plataforma.

REFERÊNCIAS

OLIVEIRA, Heron Rodrigues. **Amo (ambiente de monitoria online)**: desenvolvimento do front-end de uma aplicação para o auxílio da monitoria acadêmica. 2024. Trabalho de Conclusão de Curso (Graduação em Engenharia de Software) – Universidade Federal do Ceará, Russas, 2024. Disponível em: http://repositorio.ufc.br/handle/riufc/79576. Acesso em: 06 jan. 2024.

LOPES, Lorena Silveira. **Desenvolvimento de uma aplicação web para gerenciamento de TCC**. 2021. 45 f. Trabalho de Conclusão de Curso (Graduação em Engenharia da Computação) – Universidade Federal de Ouro Preto, João Monlevade, 2021.

SANTANA, Rafael Barroso de. **Health clinic**: aplicação web na saúde, ampliando o acesso e a comunicação. 2024. Trabalho de Conclusão de Curso (Graduação em Análise e Desenvolvimento de Sistemas) – Instituto Federal da Paraíba, Cajazeiras, 2024.

COSTA, Bianca Maria de Melo *et al*. Tecnologia digital como ferramenta na monitoria acadêmica do curso de Odontologia em tempos de pandemia COVID-19. **Revista da ABENO**, Brasília, DF, v. 21, n. 1, p. 1-8, 2021. Disponível em: https://revabeno.emnuvens.com.br/revabeno/article/view/1187/1005. Acesso em: 04 dez. 2024.

GARCIA, Luciane Terra dos Santos; SILVA FILHO, Luiz Gomes da; SILVA, Maria Verônica Gomes da. Monitoria e avaliação formativa em nível universitário: desafios e conquistas. **Perspectiva**, Florianópolis, v. 31, n. 3, p. 973-1003, set./dez. 2013.

MATTOS, Renata Pontin de; ANTONELLI, Humberto Lidio; SALGADO, André de Lima. Accessibility and usability evaluation of rich internet applications. *In*: INTERNATIONAL WEB FOR ALL CONFERENCE (W4A), 13., 2016, Montreal. **Proceedings** [...]. Montreal: ACM, 2016. p. 1-4. Disponível em: https://doi.org/10.1145/2976796.2988221. Acesso em: 20 jan. 2025.

GONÇALVES, Mariana Fiuza *et al*. A importância da monitoria acadêmica no ensino superior. **Práticas Educativas, Memórias e Oralidades - REMO**, Iguatu, CE, v. 3, n. 1, p. e313757, jan./abr. 2021. Disponível em: https://doi.org/10.47149/pemo.v3i1.3757. Acesso em: 12 nov. 2024.

SOUZA, Cícera Saraiva de; OLIVEIRA, Luciana Sena de Souza; ABRANTES, Kennia Sibelly Marques de. A importância da monitoria no âmbito do ensino superior: um relato de experiência. *In*: CONGRESSO NACIONAL DE EDUCAÇÃO (CONEDU), 3., 2016, Natal. **Anais** [...]. Natal: Realize Editora, 2016. Disponível em: http://www.editorarealize.com.br/editora/anais/conedu/2016/trabalho_ev056_md1_sa11_id4955.pdf. Acesso em: 20 set. 2024.

CONALLEN, Jim. Modeling Web Application Architectures with UML. **Communications of the ACM**, New York, v. 42, n. 10, p. 63-70, out. 1999.

AXIOS. **Axios**: promise based HTTP client for the browser and node.js. [S. l.: s. n., 20--]. Disponível em: https://axios-http.com/. Acesso em: 13 maio 2025.

LIU, K. *et al*. Design and development of management information system for research project process based on front-end and back-end separation. *In*: INTERNATIONAL CONFERENCE ON COMPUTING INTELLIGENCE AND INFORMATION SYSTEM (CIIS), 2017, Nanjing. **Proceedings** [...]. Nanjing: IEEE, 2017. p. 338-342.

MARK, Thomas. **React in action**. New York: Simon and Schuster, 2018.

DANIELSSON, William. **React Native Application Development**: Development of a cross-platform application for monitoring of truck tests. 2016. 52 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Linköping University, Linköping, 2016. Disponível em: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-129671. Acesso em: 12 dez. 2024.

MICROSOFT. **Visual Studio Code Documentation**. Redmond: Microsoft, [2025]. Disponível em: https://code.visualstudio.com/docs. Acesso em: 12 jun. 2025.

KUMAR, A.; SINGH, R. K. Comparative analysis of angularjs and reactjs. **International Journal of Latest Trends in Engineering and Technology**, London, v. 7, n. 4, p. 225-227, out. 2016. Disponível em: https://doi.org/10.21172/1.74.032. Acesso em: 07 set. 2024.

ABREU, Lucas; BATISTA, Esteic Janaína Santos; PACHER, Gabriel A.; CASTRO JÚNIOR, Amaury. Plataforma de monitoria online. *In*: CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (CBIE), 5., 2016, Uberlândia. **Anais** [...]. Uberlândia: Sociedade Brasileira de Computação, 2016. p. 371-380. Disponível em: https://doi.org/10.5753/cbie.wcbie.2016.371. Acesso em: 4 jan. 2025.

FERREIRA, Haline Kelly; ZUCHI, Jederson Donizete. Análise comparativa entre frameworks frontend baseados em javascript para aplicações web. **Revista Interface Tecnológica**, Lins, v. 15, n. 2, p. 111-123, jul./dez. 2018. Disponível em: https://revista.fatectq.edu.br/interfacetecnologica/pt_BR/article/view/502/302. Acesso em: 19 dez. 2024.

GOWDA, P.; GOWDA, A. N. Best Practices in REST API Design for Enhanced Scalability and Security. **Journal of Artificial Intelligence, Machine Learning and Data Science**, [S. l.], v. 2, n. 1, p. 827-830, jan. 2024. Disponível em: https://doi.org/10.51483/IJCSP.2.1.2024.827-830. Acesso em: 20 nov. 2024.

COSENTINO, V.; CÁNOVAS IZQUIERDO, J. L.; CABOT, J. A systematic mapping study of development with GitHub. **IEEE Access**, New York, v. 5, p. 7173-7192, 2017. Disponível em: https://doi.org/10.1109/ACCESS.2017.2682323. Acesso em: 17 mar. 2025.

CUNHA, Bruna Carolina Rodrigues da. Desenvolvimento full-stack com Javascript: uma visão geral e prática. *In*: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO (CSBC), 43., 2024, Brasília. **Anais** [...]. Brasília: Sociedade Brasileira de Computação, 2024. Disponível em: https://doi.org/10.5753/csbc.2024.233917. Acesso em: 22 set. 2024.

KURIAN, Gethyl George. How Virtual DOM and Diffing Works in React. **Medium**, [S. l.], 22 jan. 2024. Disponível em: https://medium.com/@gethylgeorge/how-virtual-dom-and-diffing-works-in-react-6fc805f9f84e. Acesso em: 18 fev. 2025.

THUNDER CLIENT. **Thunder Client**: Lightweight Rest API Client for VS Code. [S. l.: s. n., 2024]. Disponível em: https://www.thunderclient.com. Acesso em: 16 mar. 2025.

SOMMERVILLE, Ian. **Software engineering**. 9. ed. Boston: Pearson Education, 2011.

PAULA FILHO, Wilson de Pádua. **Engenharia de software**. Rio de Janeiro: LTC, 2003.

PRESSMAN, Roger S. **Engenharia de software**: uma abordagem profissional. 5. ed. Rio de Janeiro: McGraw-Hill, 2002.