PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE SOFTWARE PARA AGENDAMENTOS DE TRABALHO CONCLUSÃO DE CURSO

VITOR MANOEL ALVES DE SOUZA
2025

VITOR MANOEL ALVES SOUZA

DESENVOLVIMENTO DE SOFTWARE PARA AGENDAMENTOS DE TRABALHO CONCLUSÃO DE CURSO

Trabalho de Conclusão de Curso apresentado à Escola Politécnica e de Artes, da Pontifícia Universidade de Goiás, como parte dos requisitos para obtenção do título de Bacharel em Engenharia da Computação
Orientador (a):
Profª. Dr. Solange da Silva.
Banca examinadora:
Prof. Dr. Vicente Paulo de Camargo.
Prof. Ma. Miriam Sandra Rosa Gusmão.

VITOR MANOEL ALVES DE SOUZA

DESENVOLVIMENTO DE SOFTWARE PARA AGENDAMENTOS DE TRABALHO CONCLUSÃO DE CURSO

Trabalho de Conclusão de Curso aprovado em sua forma final pela Escola Politécnica e de Artes, da Pontifícia Universidade Católica de Goiás, para obtenção do título de Bacharel em Engenharia da Computação, em:10/06/2025_.

Orientador (a): Profª. Dr. Solange da Silva

Prof. Dr. Vicente Paulo de Camargo

Profa. Ma. Miriam Sandra Rosa Gusmão

GOIÂNIA
2025

Este trabalho é dedicado à minha família, Adaildo, Marceli, Thiago e a minha noiva Luana.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me conceder força, saúde e sabedoria ao longo dessa jornada.

Aos meus pais, pelo amor incondicional, pelos valores que me ensinaram e por sempre acreditarem no meu potencial, mesmo nos momentos mais desafiadores.

Ao meu irmão Thiago, pelo companheirismo, pelas palavras de incentivo e por ser um exemplo de perseverança em minha vida.

À minha noiva Luana, pelo apoio constante, pela paciência nos momentos de ausência e por compartilhar comigo cada conquista com amor e entusiasmo.

À minha orientadora, professora Solange, pela dedicação, pelas orientações valiosas e por me guiar com firmeza e gentileza durante todo o desenvolvimento deste trabalho. Sua contribuição foi essencial para a realização deste projeto.

A todos que, de alguma forma, fizeram parte deste percurso, o meu sincero muito obrigado.

RESUMO

O objetivo geral deste trabalho foi o de desenvolver uma aplicação para automatizar o processo de agendamento de TCC da Pontifícia Universidade Católica de Goiás, proporcionando uma interface acessível e intuitiva para professores e administradores, garantindo eficiência na gestão de horários e salas, evitando conflitos ou sobreposição de eventos. Este estudo permitiu constatar que o desenvolvimento web é altamente versátil, permitindo a criação de soluções eficientes e adaptáveis, por meio de um amplo conjunto de tecnologias e ferramentas. O projeto possibilitou a vivência de todas as etapas do desenvolvimento de um software, desde o levantamento de requisitos, passando pelo *design* e implementação, até a resolução de desafios técnicos, resultando em um produto funcional e alinhado às necessidades institucionais. Dessa forma, conclui-se que a aplicação desenvolvida atingiu os objetivos propostos, ao automatizar o processo de agendamento de apresentações acadêmicas, promovendo maior organização, controle e eficiência. A pesquisa demonstrou a viabilidade da integração entre tecnologias modernas e demandas institucionais.

Palavras-chaves: *Software.* Desenvolvimento de aplicação web. *Web.* Engenharia de Requisitos.

ABSTRACT

The overall objective of this work was to develop an application to automate the TCC scheduling process at a university, providing an accessible and intuitive interface for professors and administrators, ensuring efficient management of schedules and rooms, avoiding conflicts or overlapping events. This study demonstrated that web development is highly versatile, allowing the creation of efficient and adaptable solutions through a wide range of technologies and tools. The project made it possible to experience all stages of software development, from requirements gathering, through design and implementation, to solving technical challenges, resulting in a functional product aligned with institutional needs. Thus, it is concluded that the developed application achieved the proposed objectives by automating the process of scheduling academic presentations, promoting greater organization, control and efficiency. The research demonstrated the feasibility of integrating modern technologies and institutional demands.

Keywords**:** *Software. Web application development. Web. Requirements engineering.*

LISTA DE FIGURAS

Figura 1 - Diagrama de casos de uso do sistema de agendamento. 17
Figura 2 – Ilustração do fluxo de uma API. 19
Figura 3 - *Static Site.* Site Estático. 22
Figura 4 – *Dynamic Site*. 23
Quadro 1 – Autenticação. 29
Quadro 2 – Inserção de professores na plataforma. 30
Quadro 3 – Edição de professor. 31
Quadro 4 – Remoção de professor. 32
Quadro 5 - Criação de formulário. 33
Quadro 6 – Edição de formulário. 34
Quadro 7 – Preenchimento dos formulários. 34
Quadro 8 – Edição de respostas. 35
Quadro 9 – Gerar arquivo com as respostas. 35
Figura 5 – Cliente-Servidor. 36
Figura 6 – Tela de login. 45
Figura 7 – Tela de resposta ao envio de e-mail. 46
Figura 8 – Tela de início. 47
Figura 9 – Tela de preenchimento formulário. 48
Figura 10 – Tela da lista de respostas. 49
Figura 11 – Tela de visualização da resposta. 50
Figura 12 – Tela de criação de usuários. 50
Figura 13 – Tela de gerenciamento de usuários. 51

LISTA DE SIGLAS

AMD	*Advanced Micro Devices*
API	*Application Programming Interface*
CRUD	*Create Read Update Delete*
CSS	*Cascading Style Sheets*
ER	Engenharia de Requisitos
IDE	Ambiente de Desenvolvimento Integrado
JSON	JavaScript *Object Notation*
JWT	JSON *Web Token*
ORM	*Object-Relational Mapping*
PROPS	*Properties*
RAM	*Random Access Memory*
RF	Requisitos Funcionais
SQL	*Structured Query Language*
SSG	*Server Side Generate*
SSR	*Server Side Rendering*
TCC	Trabalho de Conclusão de Curso
UI	*User Interface*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
UUID	Identificador Universalmente Único
UX	*User Experience*
VSCODE	Visual Studio Code
W3C	*World Wide Web Consortium*
WEB	*World Wide Web*

SUMÁRIO

DESENVOLVIMENTO DE SOFTWARE PARA AGENDAMENTOS DE TRABALHO CONCLUSÃO DE CURSO 1
AGRADECIMENTOS 5
RESUMO 6
ABSTRACT 7
LISTA DE FIGURAS 8
SUMÁRIO 10
1. INTRODUÇÃO 12
2 REFERENCIAL TEÓRICO 13
Este capítulo é constituído de 2 partes: uma de conceitos e definições e outras de trabalhos relacionados. 13
2.1 Conceitos e Definições 13
13
2.1.1 Requisitos Funcionais 14
2.1.2 Linguagem de modelagem unificada ou Unified Modeling Language (UML) 15
2.1.3 Diagrama de Casos de Uso 15
2.1.4 Word Wide Web (WEB) 18
2.1.5 Application Programming Interface (API) 18
2.1.7 Usabilidade e Experiência do Usuário (UX) 20
2.1.8 Modelos de aplicações Web 22
22
2.1.9 Aplicação Web 22
Single Page Applications (SPA) 23
Progressive Web Apps (PWA) 24
2.2 Trabalhos Relacionados 24
2.2.1 Modelagem e Prototipação de um Sistema de Gestão Acadêmica. 24
2.2.2 Desenvolvimento de um aplicativo para agendar tarefas acadêmicas em dispositivos móveis. 24
3. PROCEDIMENTOS METODOLÓGICOS 26
4. DESENVOLVIMENTO DO SOFTWARE 28
4.1 Requisitos Funcionais - RF 28
35
5 DESENVOLVIMENTO DO SOFTWARE 36
5.1 Arquitetura Cliente-Servidor 36
5.2 Tecnologias Utilizadas no Frontend 37
5.2.1 Next.js 37
5.2.2 TypeScript 37
5.2.3 Shadcn UI 38
5.2.4 Zod 39
5.2.5 React Hook Form 39
5.2.6 Outras Tecnologias Complementares 40
5.3 Backend 41
5.3.1 Nest.js 41

5.3.2 Prisma ORM 42
5.3.3 PostgreSQL 43
5.3.4 SendGrid 43
Para o envio de emails transacionais, especialmente os links de autenticação, foi integrado o serviço SendGrid, uma plataforma especializada em comunicação por e-mail, conforme a documentação Sendgrid: 43
5.3.5 Outras Tecnologias e Bibliotecas 44
5.4 Telas da aplicação 44
6 CONCLUSÃO 52

1. INTRODUÇÃO

A automatização de tarefas manuais tem se mostrado uma ferramenta poderosa para otimizar processos e aumentar a eficiência. Segundo o SEBRAE (2023), a implementação de soluções automatizadas pode reduzir erros, agilizar respostas e melhorar a produtividade, ao permitir que as equipes se concentrem em atividades de maior valor estratégico, eliminando a necessidade de retrabalho e reduzindo o tempo de execução das tarefas

Nos últimos anos, o avanço da tecnologia da informação tem proporcionado soluções inovadoras para diversos setores, incluindo a educação. Faculdades e universidades, além de lidarem com a gestão de seus currículos e processos acadêmicos, enfrentam desafios logísticos no que diz respeito à organização de espaços e horários. Um desses desafios envolve o agendamento de eventos importantes, como defesas de trabalhos de conclusão de curso (TCCs), reuniões de comitês e utilização de salas de aula para atividades acadêmicas e extracurriculares.

Justifica-se estudar este tema, pois muitas instituições de ensino superior ainda utilizam métodos manuais ou sistemas descentralizados para gerir esses agendamentos, o que pode resultar em conflitos de horários, superlotação de recursos e uma considerável perda de tempo tanto para o corpo docente quanto para os funcionários administrativos. Diante dessa realidade, a automação do processo de agendamento surge como uma solução capaz de otimizar o uso dos espaços, minimizar erros e proporcionar maior eficiência operacional para a universidade.

Diante deste contexto, este trabalho visa responder a seguinte questão de pesquisa: **Como automatizar um agendamento de TCC utilizando ferramentas da Web?**

O objetivo geral deste trabalho é desenvolver uma aplicação para automatizar o processo de agendamento de TCC da Pontifícia Universidade Católica de Goiás, proporcionando uma interface acessível e intuitiva para professores e administradores, garantindo eficiência na gestão de horários e salas, evitando conflitos ou sobreposição de eventos.

Os objetivos específicos são:

- Aprofundar o conhecimento sobre Desenvolvimento de *software*.
- Fazer o levantamento de requisitos com o *Stakeholders*.
- Planejar e desenvolver a aplicação.
- Realizar testes e validação do aplicativo na faculdade parceira.
- Usar a tecnologia como meio de automatizar tarefas manuais.

Espera-se que os resultados deste trabalho possam contribuir:

- Automatizando o sistema de agendamento de TCC da universidade, facilitando este procedimento
- Evitando redundâncias nas informações.
- Facilitando a verificação das informações tanto para professor quanto para alunos.

Quanto aos aspectos metodológicos, a natureza desta pesquisa é um resumo de assunto. Quanto aos seus objetivos é uma pesquisa exploratória e descritiva. Em relação aos procedimentos técnicos, é uma pesquisa bibliográfica e experimental.

Esta monografia está estruturada da seguinte maneira: neste capítulo estão apresentados o contexto do trabalho, questão de pesquisa, objetivo geral e específicos, além do resultado esperado. O capítulo 2 traz o referencial teórico, no qual são apresentados conceitos e definições sobre o tema e trabalhos correlatos. No capítulo 3 é mostrado os procedimentos metodológicos para atingir o objetivo geral. O capítulo 4 apresenta o processo de preparação dos para o desenvolvimento da aplicação. O capítulo 5 apresenta as etapas do desenvolvimento do *Software*, bem como as ferramentas utilizadas para tal. Finalmente, o capítulo 6 traz as considerações finais e sugestões de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Este capítulo é constituído de 2 partes: uma de conceitos e definições e outra de trabalhos relacionados.

2.1 Conceitos e Definições

Esta parte traz conceitos e definições utilizados neste presente trabalho.

2.1.1 Requisitos Funcionais

Os requisitos funcionais descrevem o comportamento e as funcionalidades que um sistema deve implementar para atender às necessidades dos usuários e alcançar seus objetivos (Sommerville, 2011). Esses requisitos são essenciais para guiar o desenvolvimento e garantir que o sistema atenda às expectativas dos *stakeholders* (Larman, 2004). No contexto desta aplicação, os requisitos funcionais foram elaborados para definir as principais operações e interações entre os usuários (professores e administrador) e o sistema de agendamento de apresentações de TCC.

De acordo com Pressman (2015), requisitos funcionais devem ser claros e detalhados, descrevendo as funcionalidades do sistema em termos de entradas, processos e saídas. Para esta aplicação de agendamento, os principais requisitos funcionais incluem:

- **Autenticação e Gestão de Acesso**: Como primeiro passo, a aplicação deve garantir que somente usuários autorizados possam acessar suas respectivas áreas. O administrador tem privilégios elevados, permitindo-lhe gerenciar professores e definir permissões, enquanto os professores acessam para agendar apresentações de TCC. Isso está de acordo com a importância da segurança em sistemas de informação, conforme sugerido por Sommerville (2011).
- **Inserção e Gestão de Professores**: O administrador deve poder criar perfis de professores, inserindo nome, matrícula, cargo (professor ou administrador) e e-mail. Para autenticar na aplicação o professor precisa colocar o e-mail e vai receber o link de acesso no mesmo.
- **Agendamento de Apresentações**: Professores devem ter a possibilidade de agendar suas apresentações através do formulário.
- **Consulta e Cancelamento de Apresentações**: Os professores devem ter acesso à sua própria lista de agendamentos, com funcionalidades de edição e cancelamento, desde que dentro dos prazos estipulados. Segundo Sommerville (2011), oferecer flexibilidade ao usuário para gerenciar suas interações com o sistema é uma característica importante de sistemas bem projetados.

- **O administrador deve ser capaz de gerar um relatório no formato de planilha:** O administrador deve ser capaz de gerar uma planilha com as respostas do formulário em um formato esperado.

Portanto, os requisitos funcionais desta aplicação foram desenhados para otimizar o processo de agendamento, proporcionando uma interface clara e funcionalidades adequadas para atender às necessidades específicas da instituição, dos professores e do administrador.

2.1.2 Linguagem de modelagem unificada ou Unified Modeling Language (UML)

A Linguagem de Modelagem Unificada é uma linguagem padrão de modelagem para representar de forma visual o design e a arquitetura de sistemas de software. De acordo com Booch (2005), é uma linguagem que tem o seu vocabulário formado por regras e essas focam na representação conceitual e física de um sistema, ou seja, uma linguagem para visualização.

Segundo Sommerville (2011), a UML é uma linguagem essencial para apoiar práticas de desenvolvimento orientado a objetos, fornecendo uma base para a documentação, análise e comunicação entre membros de equipes multidisciplinares.

2.1.3 Diagrama de Casos de Uso

O Diagrama de Casos de Uso é um diagrama da UML que possui a finalidade de representar as principais funcionalidades de um sistema sob a perspectiva do usuário comum. Ele demonstra, de forma visual, como os diferentes usuários (ou outros sistemas externos) interagem com as funcionalidades oferecidas pela aplicação, possibilitando uma compreensão geral do comportamento do sistema sem a necessidade de entrar em detalhes técnicos ou de implementação.

De acordo com Pressman (2010), esse tipo de diagrama é especialmente útil nas fases iniciais do desenvolvimento de software, pois auxilia na identificação e na organização dos requisitos funcionais do sistema. Além disso, contribui significativamente para a comunicação entre os membros da equipe de desenvolvimento e os stakeholders, garantindo que todos compreendam o escopo das funcionalidades previstas.

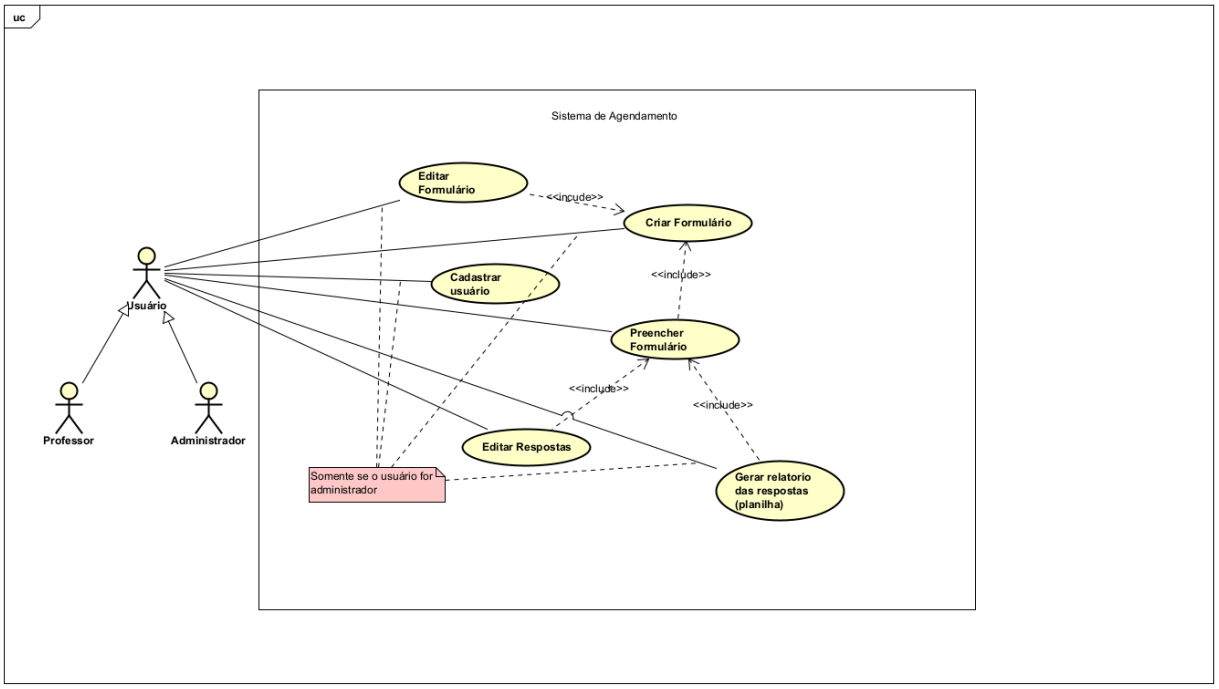
Os principais elementos que compõem um Diagrama de Casos de Uso são:

- **Ator**: Representa qualquer entidade externa ao sistema que interage com ele, podendo ser um usuário humano, outro sistema ou até mesmo um hardware.
- **Caso de uso**: Define uma funcionalidade específica que o sistema executa em resposta a uma solicitação de um ator.
- **Associações**: São as conexões que demonstram quais atores estão envolvidos em quais casos de uso.
- **Relacionamentos**: São utilizados para estruturar os casos de uso, sendo os mais comuns a inclusão (<<include>>), a extensão (<<extend>>) e a generalização. Esses relacionamentos possibilitam a reutilização de funcionalidades e a organização do diagrama de forma mais eficiente.

A adoção dos diagramas de casos de uso no processo de desenvolvimento de software é motivada pela clareza e sua capacidade de representar o escopo funcional do sistema de maneira visível e simples. Isso torna esse tipo de diagrama uma ferramenta essencial para a modelagem de sistemas.

A Figura 1. representa o Diagrama de Casos de Uso do Sistema de agendamento proposto. No Diagrama é possível identificar dois tipos de usuário, ambos generalizados como o ator usuário.

Figura 1: Diagrama de casos de uso do sistema de agendamento.



Fonte: Autoria própria.

Desta forma, tanto os professores quanto os administradores compartilham certas funcionalidades, no entanto, possuem responsabilidades específicas.

- **Cadastrar usuário**: funcionalidade exclusiva do administrador, responsável por adicionar novos usuários ao sistema.
- **Criar Formulário**: funcionalidade central para configurar formulários utilizados no processo de agendamento. Inclui os casos de uso "Editar Formulário" e "Preencher Formulário".
- **Editar Formulário**: permite modificar formulários existentes, e está incluído no caso de uso "Criar Formulário".
- **Preencher Formulário**: utilizado pelos professores para inserir dados nos formulários disponíveis, sendo incluído tanto em "Criar Formulário" quanto em "Editar Respostas".

- **Editar Respostas**: permite alterar respostas já preenchidas. Está diretamente ligado aos usuários e também inclui o caso de uso "Preencher Formulário".

- **Gerar relatório das respostas (planilha)**: funcionalidade para exportar os dados inseridos em formulários em formato de planilha. Inclui o caso de uso "Preencher Formulário".

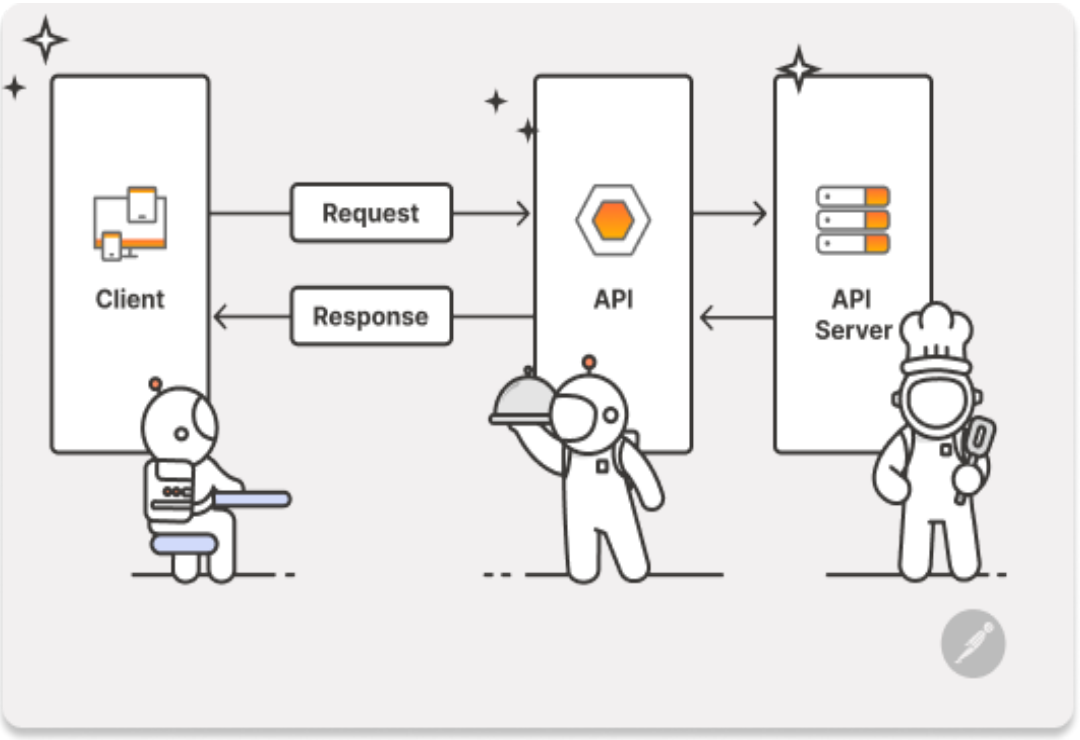
2.1.4 *Word Wide Web (WEB)*

Como descrito pela MDN (s.d.), a Web, ou World Wide Web, é um sistema de páginas publicadas e interconectadas, acessadas por meio da Internet. É importante destacar que a Web não é sinônimo de Internet, mas sim uma de suas aplicações mais populares, permitindo a navegação por documentos e recursos interligados por hiperlinks.

2.1.5 *Application Programming Interface (API)*

Conforme a definição encontrada no site *Red Hat* (2023), APIs são ferramentas que são utilizadas para integrar sistemas, sem a necessidade de saber como o serviço foi implementado na prática. Elas possuem um protocolo e uma documentação bem definida, na qual, garante que o interessado possa estruturar de forma precisa sua solicitação e receba uma resposta da API, na Figura 2 é mostrado, de forma ilustrativa, como é o funcionamento de uma API, no qual o manda uma requisição e a API faz a comunicação com o banco de dados, entregando o conteúdo como um “garçom”.

Figura 2: Ilustração do fluxo de uma API.



Fonte: Postman, s.d.

2.1.6 *Representational State Transfer* (REST API)

As APIs possuem uma comunicação via *Hypertext Transfer Protocol* (HTTP), que é um protocolo que em sua utilidade serve para transmissão de documentos de hipermídia. De acordo com Patni (2017) APIs com regras baseadas em REST, possuem algumas características que melhoram uma API, algumas delas são:

- **Escalabilidade:** simplificada entre componentes é uma característica essencial no REST, facilitando o crescimento de sistemas distribuídos e permitindo que novos componentes sejam adicionados sem comprometer o desempenho.
- **Modificabilidade**: A estrutura REST permite a modificação de componentes de maneira independente, promovendo a separação de responsabilidades e reduzindo riscos de falhas quando alterações são realizadas em partes do sistema.

- **Confiabilidade**: A restrição de não manter estado (*stateless*) facilita a recuperação em caso de falha, pois cada solicitação inclui todas as

informações necessárias para seu processamento.

- Dos verbos HTTPs, temos: GET, POST, PUT, DELETE e PATCH, que através destes verbos é possível fazer um *Create, Read, Update, Delete* (CRUD). O verbo GET é utilizado para buscar informações há um recurso específico, no POST e usado para enviar alguma informação que irá causar mudança em algum estado, já no método PUT e usado para atualizar um estado específico através de um identificador e o *DELETE* é usado para a remoção de algum recurso.

2.1.7 Usabilidade e Experiência do Usuário (UX)

A usabilidade e a experiência do usuário (UX) são componentes cruciais no desenvolvimento de sistemas web, especialmente em plataformas que visam atender a diferentes perfis de usuários, como administradores e professores em uma aplicação de agendamento de TCCs. A usabilidade, segundo Nielsen (1993), refere-se à facilidade com que um usuário pode interagir com uma interface para alcançar seus objetivos de maneira eficiente e satisfatória. No contexto deste trabalho, garantir uma alta usabilidade implica oferecer uma interface que seja intuitiva, de fácil aprendizado e que minimize a ocorrência de erros durante o processo de agendamento.

Nielsen (1993) define cinco componentes principais da usabilidade:

1. **Facilidade de Aprendizado**: O sistema deve permitir que os usuários aprendam a utilizá-lo rapidamente, sem a necessidade de tutoriais extensos. No caso do sistema de agendamento de TCCs, os professores e administradores devem conseguir navegar pela plataforma com facilidade e sem a necessidade de assistência técnica constante.
2. **Eficiência**: Após a fase de aprendizado, os usuários devem ser capazes de realizar suas tarefas de maneira rápida. O sistema de agendamento proposto facilita a escolha de horários e salas em um fluxo contínuo, minimizando o número de cliques e etapas necessárias para a conclusão de uma reserva.
3. **Memorabilidade**: Mesmo que os usuários não utilizem o sistema frequentemente, ele deve ser intuitivo o suficiente para que os processos

possam ser facilmente relembrados após um período de inatividade. O uso de uma interface simples e clara ajuda a garantir que os professores consigam reutilizar a plataforma sem a necessidade de reaprendizagem.

4. **Prevenção de Erros**: O sistema deve ser projetado para evitar que os usuários cometam erros. Neste caso, a prevenção de conflitos de horários na reserva de salas é essencial. A interface oferece avisos imediatos caso um professor tente agendar um horário já reservado, promovendo uma experiência mais segura e eficiente.
5. **Satisfação**: A interface deve ser agradável para os usuários. A experiência subjetiva do usuário ao utilizar o sistema de agendamento é de grande importância para o sucesso da aplicação, uma vez que interfaces agradáveis tendem a aumentar a satisfação e engajamento dos usuários.

Além da usabilidade, a Experiência do Usuário (UX) também desempenha um papel central. Enquanto a usabilidade se concentra na facilidade de uso, a UX foca nas emoções e percepções do usuário ao interagir com o sistema (Garrett, 2011). Uma boa UX abrange fatores como a clareza de *feedback* visual, a resposta rápida às interações dos usuários e a personalização da interface. No caso da aplicação desenvolvida, ao finalizar um agendamento, o professor recebe *feedback* visual imediato, confirmando a ação, o que aumenta sua confiança no sistema.

A responsividade também é um aspecto crítico da UX moderna, já que muitos usuários acessam sistemas a partir de dispositivos móveis. O sistema desenvolvido é responsivo, garantindo que os professores possam realizar agendamentos de TCCs tanto em computadores quanto em smartphones e tablets, sem perder a funcionalidade ou a usabilidade da plataforma.

Por fim, um sistema com boa UX deve ser acessível. Segundo o W3C (*World Wide Web Consortium*), acessibilidade implica que pessoas com deficiência possam utilizar e interagir com o sistema de forma equivalente a outros usuários. Isso envolve garantir compatibilidade com leitores de tela e uma navegação facilitada para todos. Portanto, ao aplicar os princípios de usabilidade e UX ao desenvolvimento da aplicação de agendamento de TCCs, garantimos uma interface intuitiva, eficiente e acessível, promovendo uma experiência de uso agradável e satisfatória para todos os usuários envolvidos.

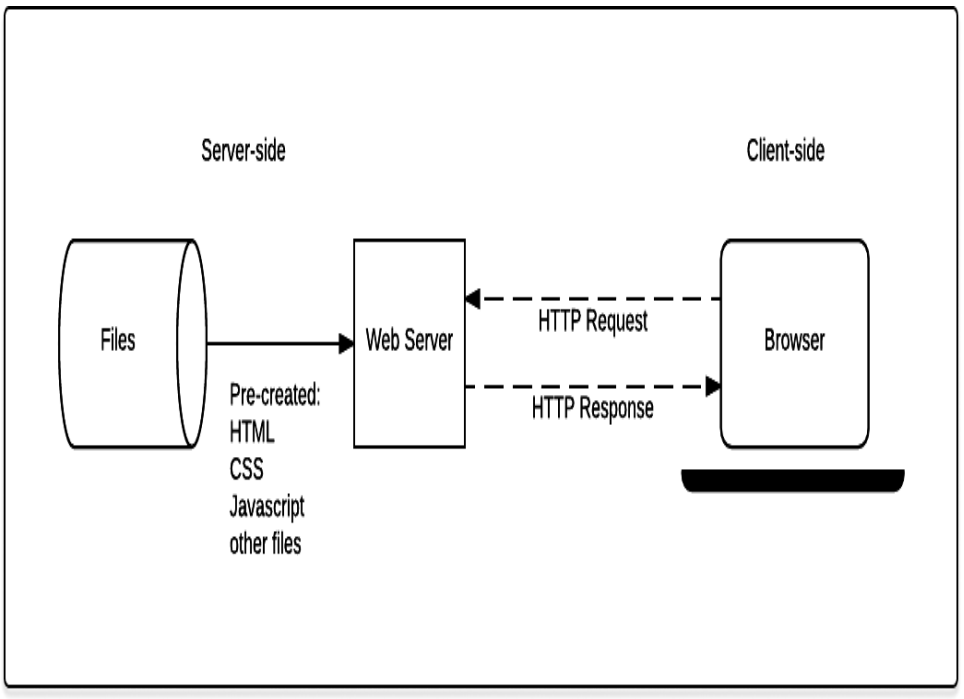
2.1.8 Modelos de aplicações Web

A Web possui diversas formas e estruturas de se estruturar uma aplicação, como: Web dinâmico, Web estáticos, aplicação de Página Única, *Progressive Web* Apps, Lojas *Onlines* e dentre outros.

2.1.9 Aplicação Web

Os *Softwares* desenvolvidos para web possuem como principal característica: ser executado no navegador da Web. Possui uma arquitetura cliente-servidor, na qual, há scripts no cliente e no servidor (AMAZON), no qual o servidor é uma API, como mostrado na Figura 3, onde os arquivos são gerados de forma estatica.

Figura 3 - *Static Site.* Site Estático

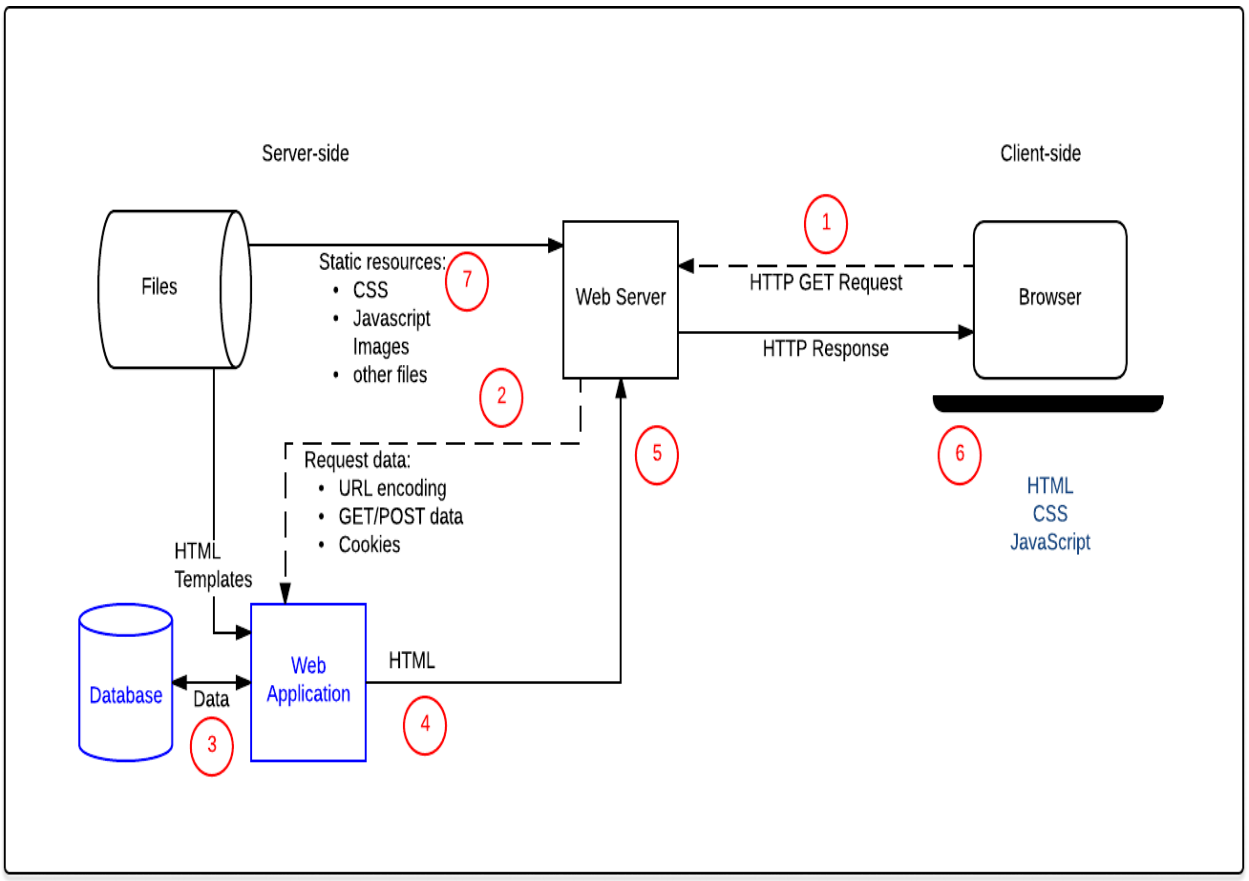


Fonte: MDN, s.d.

Há também sites dinâmicos capazes de gerar e entregar conteúdo com base na URL e nos dados de solicitação fornecidos, em vez de retornar sempre o mesmo conteúdo estático para uma URL específica. Isso significa que o conteúdo pode ser personalizado ou atualizado dinamicamente de acordo com as interações do usuário ou as variáveis passadas (MDN).

O desenvolvimento web evoluiu significativamente e atualmente os sistemas são baseados em *frameworks* e padrões que oferecem alta interatividade, responsividade e eficiência conforme identificado na Figura 4.

Figura 4 - *Dynamic Sites*



Fonte: MDN, s.d.

Para a aplicação foi escolhido a anatomia de sites dinâmicos, pois o conteúdo vai ser gerado de acordo com a URL, além disso, os parâmetros vão ser utilizados para salvar filtros da própria aplicação.

Single Page Applications **(SPA)**

Single Page Applications (SPA) são aplicativos web que carregam uma única página de *HyperText Markup Language* (HTML), que segundo MDN (s.d.) são elementos que formam diferentes partes do conteúdo de uma página. SPAs formam dinamicamente o conteúdo conforme o usuário interage com a aplicação, sem a necessidade de recarregar a página completa. Isso proporciona uma experiência mais rápida e fluida.

***Progressive Web Apps* (PWA)**

Progressive Web Apps (PWAs) são aplicativos que utilizam tecnologias web modernas para proporcionar uma experiência semelhante à de aplicativos nativos.

2.3.4 Front-End

O frontend é a camada de uma aplicação responsável pela interface e interação com o usuário. Refere-se a todas as partes visíveis de um sistema web, como páginas, botões, menus, formulários e elementos gráficos.
É desenvolvido utilizando tecnologias como HTML (HyperText Markup Language), CSS (Cascading Style Sheets) e JavaScript, além de frameworks e bibliotecas como React, Vue.js e Angular.

Essa camada se preocupa principalmente com aspectos como design, usabilidade, responsividade e experiência do usuário (UX).

2.3.5 Back-End

O backend representa a camada da aplicação que opera no lado do servidor. É responsável por realizar o processamento de dados, regras de negócio, armazenamento em banco de dados e gerenciamento da comunicação entre o cliente (frontend) e o servidor.

O backend trabalha com bancos de dados relacionais (MySQL, PostgreSQL) ou não relacionais (MongoDB), além de realizar a exposição de APIs RESTful ou GraphQL, que fornecem os dados ao frontend.

2.2 Trabalhos Relacionados

Existem alguns trabalhos na área de desenvolvimento Web que merecem destaque e que colaboraram para o escopo deste trabalho.

2.2.1 Modelagem e Prototipação de um Sistema de Gestão Acadêmica.

O trabalho de Aldeir e Leal (2013), teve como objetivo geral o desenvolvimento de um sistema web para gerenciar informações acadêmicas e automatizar processos administrativos de instituições de ensino superior.

Para isso, os autores realizaram uma análise detalhada de requisitos,

utilizando diagramas UML para modelagem funcional, e criaram um sistema que permitia o registro e consulta de dados acadêmicos por alunos, professores e administradores.

Como conclusão, o sistema contribuiu para a simplificação da gestão acadêmica, promovendo eficiência no acesso e atualização de informações e centralizando processos antes dispersos. Além disso, a prototipação iterativa revelou-se essencial para adequar o sistema às necessidades reais dos usuários e minimizar falhas durante o desenvolvimento.

2.3 Desenvolvimento de um aplicativo para agendar tarefas acadêmicas em dispositivos móveis.

O trabalho proposto por Andrews (2015), teve como objetivo desenvolver uma aplicação móvel que ajudasse os estudantes a organizar suas atividades acadêmicas, como provas, trabalhos e eventos. A aplicação utilizou o SQLite para armazenamento local de dados e ofereceu funcionalidades como agendamento personalizado, notificações e lembretes.

O desenvolvimento foi orientado por modelagem em UML, detalhando fluxos de casos de uso e regras de negócio, como a exigência de cadastro do usuário para habilitar as funcionalidades.

Os autores concluíram que a ferramenta foi eficaz para melhorar a organização acadêmica dos estudantes e destacou a relevância de notificações automáticas. Contudo, sugeriram melhorias na usabilidade e integração com sistemas externos para ampliar seu impacto.

2.3.1 Docker

O Docker é uma plataforma de código aberto que permite o desenvolvimento, empacotamento e execução de aplicações em ambientes isolados chamados **containers**. De acordo com sua documentação oficial (DOCKER), o Docker automatiza a implantação de aplicações dentro desses containers, que são unidades leves e portáteis, contendo tudo o que a aplicação precisa para funcionar: código, runtime, bibliotecas, variáveis de ambiente e configurações.

A arquitetura do Docker é composta por três principais elementos:

- **Docker Engine**: é o motor de runtime que gerencia a criação, execução e supervisão dos containers. Ele opera em uma arquitetura cliente-servidor, onde o cliente Docker se comunica com o daemon Docker (dockered).

- **Imagens Docker**: são modelos somente leitura que definem o que será executado no container. Elas são construídas a partir de um arquivo chamado **Dockerfile**, que contém instruções para configuração do ambiente.

- **Docker Hub ou Registries**: repositórios centralizados que armazenam e distribuem imagens Docker, permitindo o versionamento e compartilhamento entre desenvolvedores e equipes.

Neste trabalho, o Docker foi utilizado para subir e gerenciar um container com o banco de dados PostgreSQ**L**, proporcionando um ambiente de desenvolvimento isolado e facilmente replicável. Com o uso da imagem oficial do PostgreSQL disponível no Docker Hub, foi possível iniciar o banco de dados com apenas alguns comandos, garantindo agilidade na configuração, persistência dos dados via volumes e facilidade de manutenção do ambiente durante o desenvolvimento do sistema.

3. PROCEDIMENTOS METODOLÓGICOS

Esta pesquisa segundo sua natureza é um resumo de assunto e segundo Wazlawick (2020), um resumo de assunto sintetiza os principais aspectos de um tema, buscando a clareza e objetividade ao descrever o foco de um trabalho.

Quanto aos procedimentos técnicos, esta é uma pesquisa bibliográfica e experimental. Segundo Gil (2022), a pesquisa bibliográfica é baseada em materiais já existentes, seja livros ou artigos publicados na internet. Isso inclui algumas etapas como:

- Escolha do tema: Esta é a primeira etapa que foi a escolha do presente tema deste trabalho.
- Levantamento bibliográfico: Nesta etapa foi feito estudos direcionados ao tema e ferramentas utilizadas no trabalho.
- Formulação do problema**: Como automatizar um agendamento de TCC utilizando ferramentas da Web?**.

Segundo GIL (2017), a pesquisa experimental envolve o controle rigoroso dos fatores que influenciam um fenômeno (Z) para entender quais são essenciais para sua ocorrência. Isso é feito ao manipular e isolar fatores, como A, B, C e D, para observar como influenciam Z. Se um único fator, como C, for necessário e suficiente para Z ocorrer, então ele pode ser considerado sua causa. A pesquisa experimental é realizada ao executar as seguintes etapas, conforme as orientações de Gil (2017)

1) Formulação do problema: **Como automatizar um agendamento de TCC utilizando ferramentas da Web?**
2) Definição do plano experimental: Foi planejada a estrutura lógica e de testes para verificar quais métodos serão implementados. Em um ambiente para testes, posteriormente um de produção. Incluindo tecnologias que fornecem uma escalabilidade maior e uma eficiência de alto nível, com compatibilidade de ambientes móveis, sendo desenvolvida na Web.
3) Determinação do ambiente: Foi conduzido em um ambiente de Homologação, com máquinas virtuais para hospedar o que for necessário. O ambiente de desenvolvimento foi alternado em duas máquinas, um *Notebook* Dell G15, com processador i5-13450HX 2.40 GHz e 16gb *Random Access Memory* (RAM) e outro com um *Advanced Micro Devices* (AMD) Ryzen 5 500 com 12gb de RAM, o

primeiro notebook e configurado com o sistema operacional Windows e o segundo com o Ubuntu Debian (Linux). Nas máquinas que possuem Sistemas Operacionais (SO) Windows utilizarei o WSL um subsistema Linux dentro do *windows*. Para o *deploy* da aplicação será utilizado a Vercel, que é uma empresa que fornece ferramentas e serviços em nuvem para desenvolvedores. Para edição de código local foi usado o *Visual Studio Code* (VSCODE), que possui uma ampla quantidade de extensões que auxiliam no desenvolvimento do *Software*.

4) Coleta dos dados: Foi feito testes locais para testar a conexão do Banco de Dados e os *endpoints,* foi usado o Postman para enviar requisições para simular a aplicação comunicando com o *Backend.*
será utilizado bibliotecas para testes, tanto no *frontend* e *Backend.*
Teste Cross-Browser e Responsividade: Foram realizados testes para garantir a funcionalidade da aplicação em diferentes navegadores e dispositivos, como em dispositivos móveis para verificar também a responsividade.

5) Análise e interpretação dos dados: Será gerado relatórios gerados após os testes para identificar falhas em funções específicas ou na comunicação entre o *back-end.* Os fluxos principais, como o agendamento de apresentações e respostas ao formulário, serão testados em cenários simulando o fluxo real. Para desempenho, a aplicação vai ser submetida a cenários de alta carga de requisições. Para os testes de segurança será efetuada requisições sem autenticação.

6) Redação do relatório: escrita do TCC.

4. DESENVOLVIMENTO DO SOFTWARE

Este capítulo descreve toda estrutura do sistema, que por sua vez é um software para agendamento de TCCs, utilizando tecnologias e conceitos da Web para sua construção. Vai ser descrito requisitos funcionais e suas descrições, que são a base do projeto, bem como o Diagrama de classes, mostrando a estrutura do banco de dados da aplicação, usando conceitos da UML.

4.1 Requisitos Funcionais - RF

Como partes dos requisitos**,** tem se os requisitos funcionais ou RFs, após os levantamentos dos requisitos e gerado as seguintes tabelas com uma estrutura de:

- **Código identificador:** Número que identifica qual o requisito apresentado.
- **Ator:** É quem vai fazer a ação no sistema ou o próprio sistema com os usuários.
- **Condições de entrada e pré-requisitos:** Condições de entrada e pré-requisitos são as quais o ator ou sistema devem estar antes de executar a ação.
- **Saídas e pós-requisitos:** Saídas e pós-requisitos: é a saída gerada após a etapa anterior, o comportamento esperado no sistema.

RF01: Autenticação

O requisito RF01 é o qual permite que o administrador da plataforma efetue login, conforme o Quadro 1.

Quadro 1 - Autenticação.

IDENTIFICADOR	NOME	ATOR	ENTRADA E PRÉ-REQUISITOS	SAÍDAS E POS-REQUISITOS
1	AUTENTICAÇÃO	ADMINISTRADOR E PROFESSORES	O USUÁRIO DEVE FORNECER EMAIL VÁLIDO	ACESSO À PLATAFORMA SE O EMAIL FOR UM EMAIL VÁLIDO

Fonte: Autoria própria.

O requisito RF01, é utilizado para representar a autenticação do Administrador e professores no *Software*, representando a interação com a página de autenticação, além disso, representa o seu estado de saída que é o acesso a aplicação através de um email válido.

RF02: Inserção de Professores

O requisito RF02, mostra a inserção de novos professores na plataforma, conforme o Quadro 2.

Quadro 2 - Inserção de professores na plataforma

IDENTIFICADOR	NOME	ATOR	ENTRADA E PRÉ-REQUISITOS	SAÍDAS E POS-REQUISITOS
2	INSERÇÃO DE PROFESSORES	ADMINISTRADOR	NOME, MATRíCULA E E-MAIL, O ADMINISTRADOR DEVE ESTAR LOGADO	PROFESSOR CADASTRADO COM SUCESSO NA PLATAFORMA

Fonte: Autoria própria.

O requisito RF02, representa a ação de adição de novos professores na plataforma, no qual o administrador deve estar *logado* e fornecer um email, matrícula, cargo e nome, e no qual a saída e o professor por meio de um link enviado no email.

RF03: Edição de Professores

O requisito RF03, deve permitir ao administrador editar os dados dos professores, conforme mostra o Quadro 3.

Quadro 3 - Edição de professor.

IDENTIFICADOR	NOME	ATOR	ENTRADA E PRÉ-REQUISITOS	SAÍDAS E POS-REQUISITOS
3	EDIÇÃO DE PROFESSORES	ADMINISTRADOR	DADOS DO PROFESSOR A SEREM EDITADOS. O ADMINISTRADOR DEVE ESTAR LOGADO	DADOS DO PROFESSOR ATUALIZADOS NO SISTEMA

Fonte: Autoria própria.

O requisito RF03, permite que o administrador possa editar os dados dos professores cadastrados na plataforma, gerando novos dados para o professor/usuário editado.

RF04: Remoção de Professores

O requisito RF04 deve permitir que o administrador remova acesso de um ou mais professores da plataforma, como mostrado no Quadro 4.

Quadro 4 - Remoção de professor

IDENTIFICADOR	NOME	ATOR	ENTRADA E PRÉ-REQUISITOS	SAÍDAS E POS-REQUISITOS
4	REMOÇÃO DE PROFESSORES	ADMINISTRADOR	SELEÇÃO DO PROFESSOR A SER REMOVIDO. O ADMINISTRADOR DEVE ESTAR LOGADO	DADOS DO PROFESSOR ATUALIZADOS NO SISTEMA

Fonte: Autoria própria.

RF04 representa a função de remoção de um usuário através do administrador que deve estar com sua sessão ativa, que por sua vez, possui como saída a remoção de um usuário qualquer.

RF05: Criação do formulário

Requisito RF05 é o que permite ao professor agendar a apresentação do TCC, através do formulário, conforme mostrado no Quadro 5.

Quadro 5 - Criação do formulário.

IDENTIFICADOR	NOME	ATOR	ENTRADA E PRÉ-REQUISITOS	SAÍDAS E POS-REQUISITOS
5	CRIAÇÃO DE FORMULÁRIO	ADMINISTRADOR	DEVE DISPOR DADOS E PERGUNTAS DO FORMULÁRIO. O ADMINISTRADOR DEVE ESTAR LOGADO	NOVO FORMULÁRIO CRIADO COM SUCESSO

Fonte: Autoria própria.

Após o requisito RF05, deve ser possível o administrador criar um formulário, com as questões e sua respectiva ordem.

RF06: Edição de formulário.

O requisito RF06 é o qual efetua a edição do formulário, por meio das novas informações ou alterações efetuadas, conforme mostrado no Quadro 6.

Quadro 6 - Edição de formulário.

IDENTIFICADOR	NOME	ATOR	ENTRADA E PRÉ-REQUISITOS	SAÍDAS E POS-REQUISITOS
6	EDIÇÃO DE FORMULÁRIO	ADMINISTRADOR	DEVE FORNECER OS NOVOS DADOS E PERGUNTAS DO FORMULÁRIO. O ADMINISTRADOR DEVE ESTAR LOGADO	FORMULÁRIO EDITADO

Fonte: Autoria própria.

RF07: Preenchimento dos formulários.

Requisito RF07, permite ao professor ver sua lista de apresentações agendadas, conforme ilustra o Quadro 7.

Quadro 7 - Preenchimento dos formulários.

IDENTIFICADOR	NOME	ATOR	ENTRADA E PRÉ-REQUISITOS	SAÍDAS E POS-REQUISITOS
7	PREENCHIMENTO DO FORMULÁRIO	ADMINISTRADOR E PROFESSORES	DEVE RESPONDER O FORMULÁRIO. O ADMINISTRADOR OU PROFESSOR DEVE ESTAR LOGADO	FORMULÁRIO PREENCHIDO

Fonte: Autoria própria.

RF08: Edição das respostas.

Requisito RF08, permite que o usuário (professor/administrador) edite ou exclua a resposta de um ou mais formulário, conforme mostrado no Quadro 8.

Quadro 8 - Edição de resposta.

IDENTIFICADOR	NOME	ATOR	ENTRADA E PRÉ-REQUISITOS	SAÍDAS E POS-REQUISITOS
8	EDIÇÃO E EXCLUSÃO DAS RESPOSTAS	ADMINISTRADOR E PROFESSORES	DEVE SELECIONAR A RESPOSTA A SER EDITADA. O ADMINISTRADOR OU PROFESSOR DEVE ESTAR LOGADO	RESPOSTA ATUALIZADA

Fonte: Autoria própria.

RF09: Gerar arquivo com respostas.

Requisito RF09, permite que o administrador baixe um arquivo no formato csv, baseado nas respostas do formulário selecionado, conforme mostrado no Quadro 9.

Quadro 9 - Gerar arquivo com as respostas.

IDENTIFICADOR	NOME	ATOR	ENTRADA E PRÉ-REQUISITOS	SAÍDAS E POS-REQUISITOS
9	RELATÓRIO	ADMINISTRADOR	DEVE SELECIONAR O FORMULÁRIO. O ADMINISTRADOR OU PROFESSOR DEVE ESTAR LOGADO	GERAR OU ATUALIZAR UM ARQUIVO CSV OU XLSX COM OS DADOS NECESSÁRIOS BASEADOS NAS RESPOSTAS DO FORMULÁRIO.

Fonte: Autoria própria.

5 DESENVOLVIMENTO DO SOFTWARE

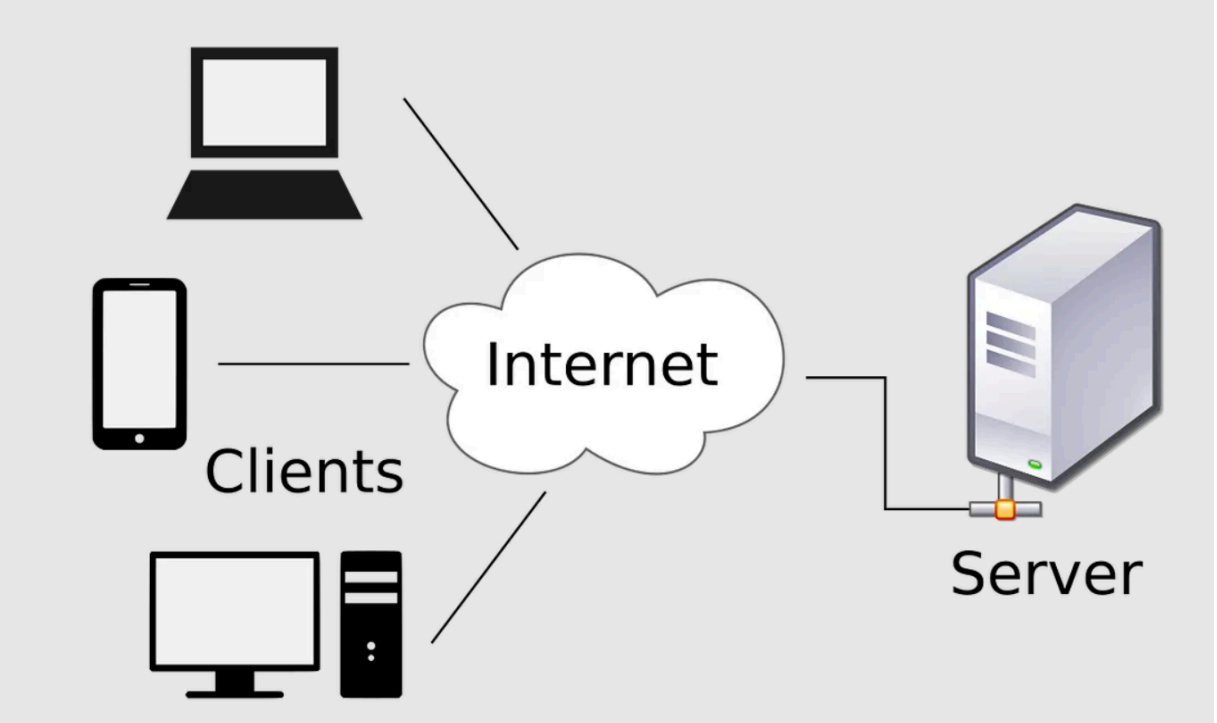
Este capítulo apresenta os detalhes técnicos da implementação do software desenvolvido, descrevendo as tecnologias utilizadas, a arquitetura do sistema, os principais componentes e como eles se integram para formar a solução completa. O sistema foi desenvolvido utilizando uma arquitetura moderna de aplicação web, com separação entre *frontend* e *backend*

5.1 Arquitetura Cliente-Servidor

A aplicação segue a arquitetura cliente-servidor, na qual o cliente é a aplicação web e o servidor é o *backend*. O cliente é responsável por interagir com o usuário, enviando requisições para o servidor, que processa os dados e retorna as respostas.

A comunicação entre cliente e servidor ocorre via HTTP/HTTPS, utilizando *APIs RESTful* para a troca de informações. O *backend* gerencia a lógica de negócios e o acesso ao banco de dados, garantindo a integridade e segurança dos dados. A estrutura geral do sistema pode ser visualizada no diagrama na Figura 6:

Figura 5 - Cliente-Servidor.



Fonte: Medium, 2021.

5.2 Tecnologias Utilizadas no *Frontend*

Para o desenvolvimento da camada de apresentação do sistema, foi realizada uma seleção de tecnologias modernas que pudessem proporcionar uma experiência de usuário fluida e responsiva, além de facilitar o processo de desenvolvimento e manutenção do código. A seguir, são apresentadas as principais tecnologias utilizadas no *frontend* da aplicação:

5.2.1 Next.js

O Next.js foi escolhido como *framework* principal de desenvolvimento *frontend* devido às suas características avançadas de renderização e otimização. Trata-se de um *framework* React que oferece múltiplas estratégias de renderização:

- ***Server-Side Rendering* (SSR)**: O conteúdo é gerado no servidor a cada requisição, o que melhora significativamente o tempo de carregamento inicial da página, além de fornecer conteúdo dinamicamente atualizado.
- ***Static Site Generation* (SSG)**: Páginas são pré-renderizadas durante o processo de build, resultando em carregamentos extremamente rápidos e redução da carga nos servidores.

Segundo a documentação oficial do Next.js (VERCEL, 2024), o *framework* também oferece um sistema de roteamento baseado em arquivos, que simplifica a estruturação da aplicação, eliminando a necessidade de configurações complexas de rotas. Além disso, inclui recursos como otimizações automáticas de imagens, divisão de código (*code splitting*), pré-busca inteligente (*prefetching*) e suporte nativo a API Routes, permitindo a criação de *endpoints* de API no mesmo projeto do *frontend*.

5.2.2 TypeScript

O TypeScript foi adotado como linguagem de programação para adicionar tipagem estática ao JavaScript, trazendo os seguintes benefícios para o desenvolvimento do sistema:

- **Detecção precoce de erros**: A verificação de tipos em tempo de desenvolvimento permite identificar e corrigir erros antes mesmo da execução

do código, reduzindo significativamente bugs relacionados a tipos incorretos de dados.

- **Melhor documentação do código**: A definição explícita de tipos serve como documentação integrada, facilitando a compreensão das interfaces e contratos entre diferentes partes do sistema.

- **Refatoração segura**: O sistema de tipos do TypeScript permite realizar refatorações complexas com maior confiança, pois o compilador identifica automaticamente usos incorretos após alterações nas estruturas de dados.

- **Autocompletar avançado**: Editores e IDEs podem oferecer sugestões de código mais precisas e contextualizadas, aumentando a produtividade dos desenvolvedores.

- A integração do TypeScript com o ecossistema React e Next.js é robusta, permitindo a tipagem de *props*, estados, *hooks* e contextos, o que contribui para um código mais seguro e manutenível.

5.2.3 Shadcn UI

Para a construção da interface de usuário, foi utilizada a biblioteca Shadcn UI, que oferece um conjunto de componentes acessíveis, customizáveis e com design consistente. O Shadcn UI não é instalado como uma dependência tradicional, mas sim como um conjunto de componentes que são incorporados diretamente ao código da aplicação (SHADCN UI,2025), permitindo:

- **Customização avançada**: Os componentes podem ser modificados livremente para atender às necessidades específicas do projeto, sem as limitações de design impostas por bibliotecas tradicionais.

- **Consistência visual**: Apesar da flexibilidade, os componentes seguem princípios de design consistentes, facilitando a criação de interfaces coerentes e profissionais.

- **Acessibilidade incorporada**: Os componentes são construídos seguindo as melhores práticas de acessibilidade, garantindo que a aplicação possa ser utilizada por pessoas com diferentes necessidades.

- A biblioteca inclui componentes essenciais como botões, campos de formulário, modais, menus *dropdown*, tabelas, entre outros, todos implementados com foco na usabilidade e compatibilidade com diferentes dispositivos.

5.2.4 Zod

O Zod foi implementado como biblioteca de validação de dados para garantir a integridade das informações manipuladas pela aplicação (ZOD,2025). Trata-se de uma biblioteca de validação de esquemas com suporte nativo a TypeScript, que permite:

- **Validação declarativa**: Os esquemas de validação são definidos de forma declarativa e reutilizável, facilitando a manutenção e evolução das regras de validação.

- **Inferência de tipos**: O Zod é capaz de inferir automaticamente tipos TypeScript a partir dos esquemas de validação, eliminando a necessidade de duplicação de definições.

- **Validação em *runtime***: Além da verificação estática proporcionada pelo TypeScript, o Zod realiza validações em tempo de execução, garantindo que dados externos (como entradas de usuário ou respostas de API) estejam em conformidade com as estruturas esperadas.

- **Mensagens de erro customizáveis**: A biblioteca permite a definição de mensagens de erro personalizadas para cada regra de validação, melhorando a experiência do usuário ao fornecer feedback preciso sobre problemas de validação.

A integração do Zod com o TypeScript proporciona um sistema de validação robusto que mantém a consistência entre os tipos estáticos e as validações em runtime, reduzindo significativamente a possibilidade de bugs relacionados a dados inválidos.

5.2.5 *React Hook Form*

Para o gerenciamento de formulários, foi utilizado o React Hook Form, uma

biblioteca que proporciona uma solução eficiente e performática para criar e validar formulários no React. Entre suas principais características, destacam-se:

- **Renderizações mínimas**: A biblioteca foi projetada para minimizar o número de re-renderização, utilizando referências não controladas (*uncontrolled components*) e registrando campos por meio de refs.
- **Performance otimizada**: O gerenciamento interno de estados é altamente otimizado, resultando em formulários que respondem rapidamente mesmo com muitos campos.
- **Integração perfeita com bibliotecas de validação**: O React Hook Form se integra facilmente com o Zod através do resolver, permitindo reutilizar os esquemas de validação em todo o sistema.
- **Gerenciamento simplificado de estados complexos**: A biblioteca facilita o tratamento de formulários aninhados, *arrays* de campos, dependências entre campos e outras estruturas complexas comuns em aplicações empresariais.
- **Suporte a campos controlados e não controlados**: Oferece flexibilidade para trabalhar com diferentes abordagens de gerenciamento de estado dos campos, dependendo das necessidades específicas de cada caso.
- A combinação do React Hook Form com o Zod proporcionou um sistema de formulários robusto, performático e fortemente tipado, melhorando significativamente a experiência do usuário e reduzindo a complexidade do código relacionado a formulários.

5.2.6 Outras Tecnologias Complementares

Além das principais bibliotecas e *frameworks* mencionados, o *frontend* do sistema também utiliza um conjunto de tecnologias complementares que contribuem para a qualidade e eficiência do desenvolvimento:

- **TailwindCSS**: Framework de CSS utilitário que permite a criação de interfaces responsivas e consistentes através de classes predefinidas, acelerando o desenvolvimento visual sem comprometer a flexibilidade.

- **Axios**: Cliente HTTP baseado em *Promises*, utilizado para realizar requisições à API do *backend* de forma simplificada, com suporte a *interceptors*, transformação de resposta e gerenciamento de erros.

- **Date-fns**: Biblioteca moderna para manipulação de datas em JavaScript, utilizada para formatação, cálculos e manipulação de informações temporais.

- **React Context API**: Utilizada para gerenciamento de estado global da aplicação, compartilhando dados como informações do usuário autenticado, preferências de tema e configurações do sistema.

- **react-query**: Biblioteca para gerenciamento de estado do servidor, fornecendo *hooks* para busca, cache e atualização de dados remotos de forma eficiente.

- A escolha e integração dessas tecnologias resultou em uma arquitetura *frontend* robusta, manutenível e com excelente experiência de usuário, alinhada com os requisitos funcionais e não-funcionais estabelecidos para o sistema.

5.3 *Backend*

A camada de *backend* do sistema foi implementada utilizando um conjunto de tecnologias modernas e robustas, selecionadas com base em critérios como manutenibilidade, escalabilidade, segurança e produtividade no desenvolvimento. A seguir, são apresentadas em detalhes as principais tecnologias que compõem a infraestrutura do servidor:

5.3.1 Nest.js

O Nest.js foi adotado como framework principal para o desenvolvimento do *backend* devido à sua arquitetura modular e escalável. Trata-se de um *framework* para Node.js inspirado no Angular, que implementa conceitos avançados de arquitetura de *software*:

- **Arquitetura modular**: O Nest.js promove a organização do código em módulos coesos e com baixo acoplamento, facilitando a manutenção e o desenvolvimento paralelo por diferentes equipes.

- **Injeção de dependências**: O *framework* oferece um sistema robusto de injeção de dependências que facilita a implementação de código testável e desacoplado, permitindo a substituição de implementações concretas por *mocks* durante os testes.
- **Decoradores e metadados**: Utilizando recursos avançados do TypeScript, o Nest.js permite definir rotas, *middlewares*, *pipes* e outros componentes de forma declarativa, reduzindo o código *boilerplate*.
- **Suporte a múltiplos protocolos de transporte**: Além do HTTP tradicional (Express ou Fastify), o *framework* suporta *WebSockets*, gRPC e microserviços baseados em mensageria, facilitando a implementação de comunicação em tempo real e arquiteturas distribuídas.
- ***Interceptores*, *Pipes* e *Guards***: O Nest.js oferece abstrações para transformação de dados, validação, controle de acesso e manipulação de requisições/respostas, facilitando a implementação de preocupações transversais de forma modular.

A arquitetura fortemente opinativa do Nest.js contribuiu significativamente para a consistência do código e para a produtividade da equipe de desenvolvimento, estabelecendo padrões claros para a implementação de diferentes componentes do sistema.

5.3.2 Prisma ORM

O Prisma foi selecionado como ORM (*Object-Relational Mapping*) para abstração da camada de acesso a dados, oferecendo uma experiência de desenvolvimento moderna e *type-safe*:

- ***Schema* declarativo**: O Prisma utiliza uma linguagem de definição de esquema própria (Prisma *Schema*) que permite modelar o banco de dados de forma declarativa e intuitiva, facilitando o entendimento da estrutura de dados.
- **Migrações automatizadas**: A partir do *schema* definido, o Prisma gera automaticamente migrações SQL que podem ser versionadas e aplicadas de forma controlada, facilitando a evolução do banco de dados.
- **Cliente fortemente tipado**: O Prisma *Client*, gerado a partir do *schema*, oferece uma API completamente tipada para acesso ao banco de dados, com autocompletar e verificação estática de queries.

A adoção do Prisma simplificou significativamente a interação com o banco de dados e aumentou a confiabilidade do código relacionado a operações de persistência, além de proporcionar uma experiência de desenvolvimento mais fluida com *feedback* imediato sobre erros em consultas.

5.3.3 PostgreSQL

Para a persistência de dados, foi escolhido o PostgreSQL, um sistema de gerenciamento de banco de dados relacional avançado, *open-source* e com amplo suporte na comunidade:

- **Tipos de dados avançados**: Além dos tipos básicos, o PostgreSQL oferece suporte a tipos complexos como *arrays*, JSON, JSONB, UUID, tipos geométricos e tipos personalizados, que foram utilizados para modelar entidades complexas do domínio.
- **Extensibilidade**: O PostgreSQL permite a criação de extensões que adicionam funcionalidades específicas, como PostGIS para dados geoespaciais ou pgcrypto para funções criptográficas avançadas.
- **Replicação e alta disponibilidade**: O PostgreSQL oferece recursos nativos de replicação que permitem implementar estratégias de alta disponibilidade e balanceamento de carga nas consultas.

A robustez, confiabilidade e capacidade de lidar com cargas de trabalho complexas do PostgreSQL o tornaram uma escolha ideal para o armazenamento dos dados do sistema, garantindo integridade e performance mesmo com o crescimento da base de dados.

5.3.4 SendGrid

Para o envio de emails transacionais, especialmente os links de autenticação, foi integrado o serviço SendGrid, uma plataforma especializada em comunicação por e-mail, conforme a documentação Sendgrid:

- **API REST simplificada**: O SendGrid oferece uma API REST bem documentada e bibliotecas cliente para diversas linguagens, facilitando a integração com o *backend* Node.js.
- **Templates HTML personalizados**: A plataforma permite a criação e gerenciamento de templates HTML responsivos para emails, que foram

utilizados para padronizar a comunicação com os usuários.

- **Escalabilidade**: O SendGrid é capaz de lidar com volumes elevados de envios, garantindo que o sistema possa escalar sem preocupações com infraestrutura de email.

A integração com o SendGrid proporcionou uma comunicação confiável por e-mail, essencial para o processo de autenticação baseado em links temporários, além de simplificar o envio de outras notificações transacionais importantes do sistema.

5.3.5 Outras Tecnologias e Bibliotecas

Além das tecnologias principais, o *backend* do sistema também utiliza um conjunto de bibliotecas e ferramentas complementares:

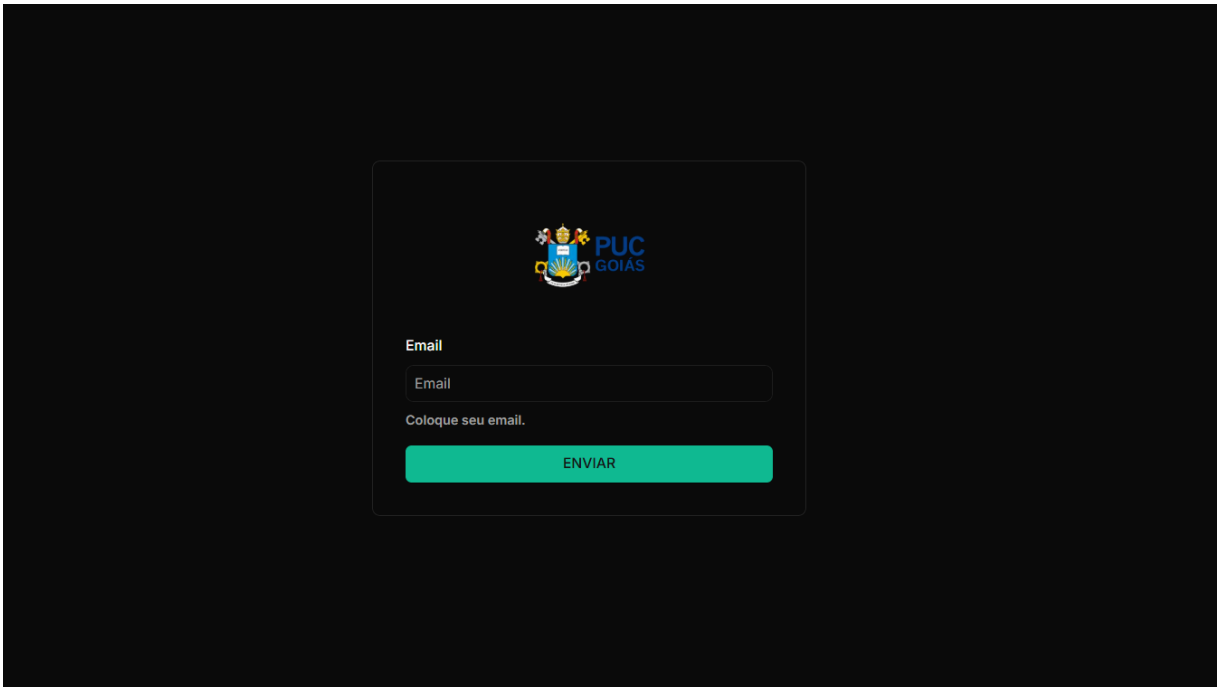
- **JSON Web Token (JWT)**: Implementado para autenticação e autorização, permitindo o controle de acesso seguro aos recursos da API através de tokens assinados.
- ***bcrypt***: Biblioteca utilizada para o *hash* seguro de senhas, garantindo que informações sensíveis nunca sejam armazenadas em texto plano no banco de dados.
- ***Swagger*/OpenAPI**: Utilizado para documentação automática da API, gerando uma interface interativa que facilita o entendimento e teste dos *endpoints* disponíveis.

A combinação dessas tecnologias resultou em um *backend* robusto, seguro e manutenível, capaz de atender aos requisitos funcionais e não-funcionais estabelecidos para o sistema, com especial atenção à segurança, performance e escalabilidade.

5.4 Telas da aplicação

Nesta seção, são apresentados alguns prints das principais telas do sistema, com o objetivo de ilustrar o funcionamento das funcionalidades implementadas e demonstrar a aplicação prática das tecnologias descritas anteriormente. A Figura 6 mostra a tela real de login da aplicação, contendo o *input* do e-mail e o botão de enviar o link de acesso.

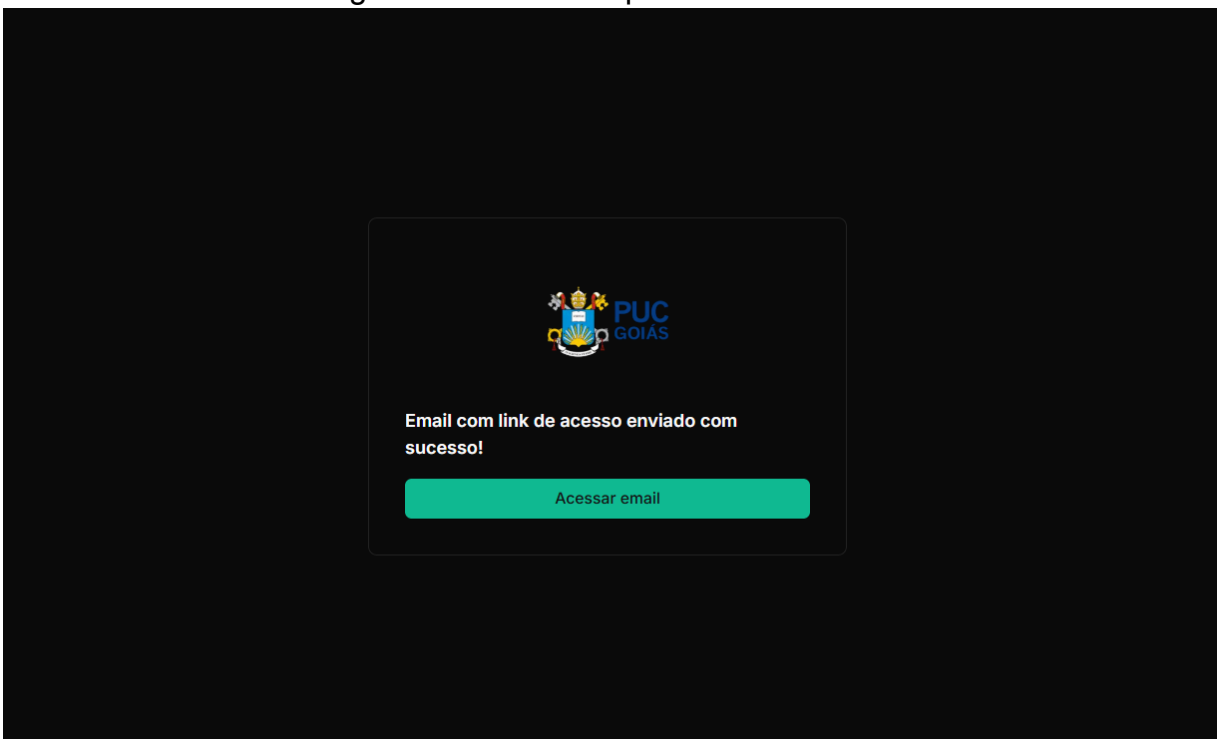
Figura 6 - Tela de login.



Fonte: Autoria própria

Na Figura 7 é mostrado a tela de *feedback* para o usuário após ele preencher o input com e-mail e clicar no botão de enviar, contendo uma mensagem no canto superior com a mensagem de resposta da ação, indicando que o link de acesso do usuário foi enviado com sucesso.

Figura 7 - Tela de resposta ao envio de e-mail.



Fonte: Autoria própria.

A Figura 8 é a tela inicial da aplicação após o *login,* contendo os formulários disponíveis e um conjunto de botões que possibilita a alteração de visualização entre formulários e as respostas do usuário, além do botão de atualizar a planilha que possui as respostas.

Figura 8 - Tela de início.

Fonte: Autoria própria.

Na Figura 9 mostra a tela de preenchimento de um formulário qualquer criado pelo professor administrador.

Figura 9: Tela de preenchimento do formulário.

Fonte: Autoria própria.

A Figura 10 é a tela de visualização da lista de respostas fornecidas pelo professor em determinados formulários.

Figura 10: Tela da lista de respostas.

Fonte: Autoria própria.

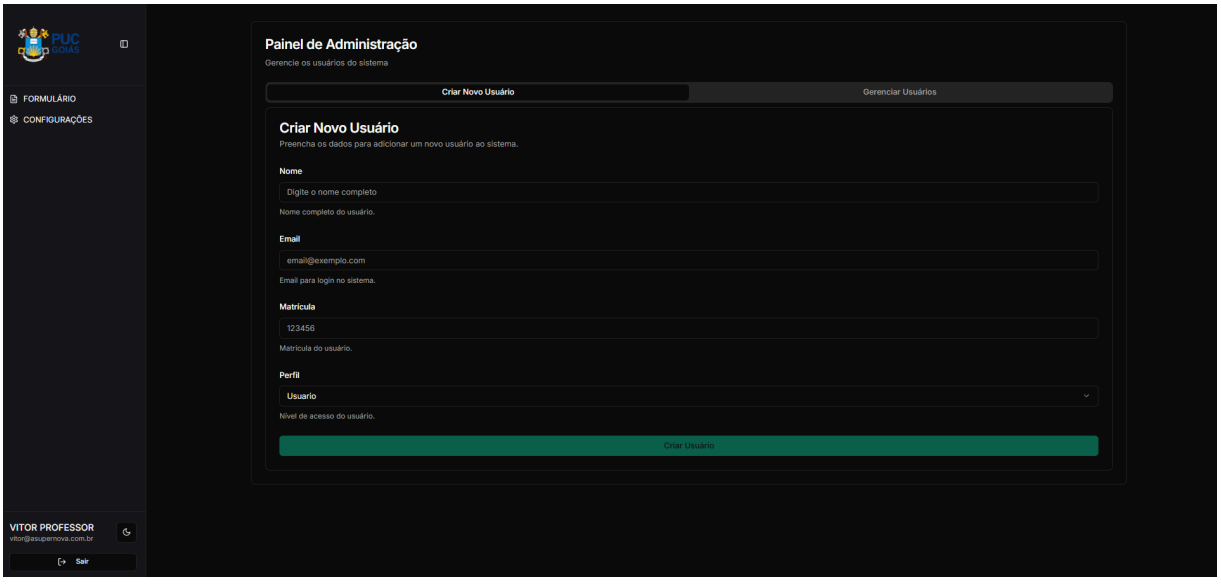
Na Figura 11 é a visualização da resposta selecionada previamente pelo usuário, resultado da ação do botão.

Figura 11 - Tela de visualização da resposta.

Fonte: Autoria própria.

A Figura 13 apresenta a página destinada à criação de novos usuários no sistema, acessível exclusivamente por usuários com perfil de administrador.

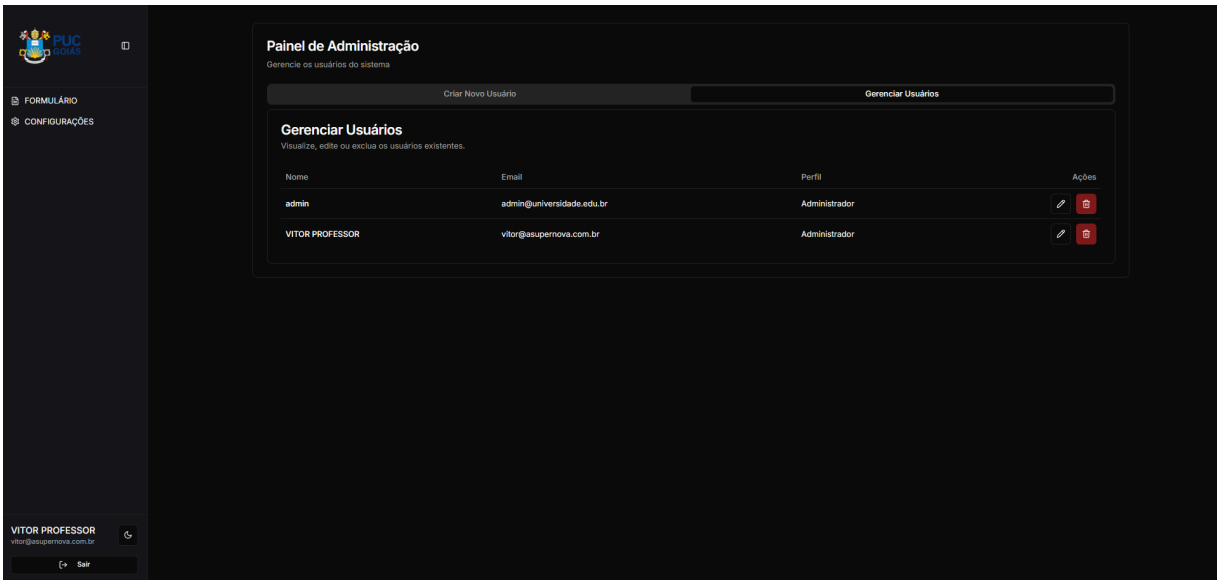
Figura 12 - Tela de criação de usuários.



Fonte: Autoria Própria.

A Figura 13 apresenta a página de gerenciamento dos usuários, disponibilizando recursos como exclusão e edição de usuários.

Figura 13 - Tela de gerenciamento de usuários.



Fonte: Autoria própria.

A construção do sistema foi guiada por decisões técnicas criteriosas, fundamentadas em princípios de modularidade, escalabilidade, segurança e usabilidade. A combinação de tecnologias modernas no frontend e no backend proporcionou uma base sólida para o desenvolvimento de uma aplicação eficiente, intuitiva e com excelente desempenho.

O uso de frameworks e bibliotecas como Next.js, TypeScript, React Hook Form, Nest.js e Prisma contribuiu para a organização e clareza do código, além de facilitar a manutenção e a evolução contínua do sistema. A aplicação de validações robustas, a adoção de padrões de projeto e a atenção à experiência do usuário reforçam a maturidade da solução desenvolvida.

6 CONCLUSÃO

O objetivo geral deste trabalho foi o de desenvolver uma aplicação para automatizar o processo de agendamento de TCC da Pontifícia Universidade Católica de Goiás, proporcionando uma interface acessível e intuitiva para professores e administradores, garantindo eficiência na gestão de horários e salas, evitando conflitos ou sobreposição de eventos.

Durante o desenvolvimento, desafios importantes foram enfrentados, como a estruturação e modelagem do banco de dados, bem como a elaboração de diagramas de casos de uso, com o obje Este estudo permitiu constatar que o desenvolvimento web é altamente versátil, permitindo a criação de soluções eficientes e adaptáveis, por meio de um amplo conjunto de tecnologias e ferramentas. O projeto possibilitou a vivência de todas as etapas do desenvolvimento de um software, desde o levantamento de requisitos, passando pelo design e implementação, até a resolução de desafios técnicos, resultando em um produto funcional e alinhado às necessidades institucionais.

Na etapa inicial, a captação dos requisitos mostrou-se essencial para o direcionamento das demais fases. Essa atividade foi conduzida por meio de reuniões com o *stakeholder*, visando à compreensão do problema e à definição precisa das funcionalidades do sistema.

Dessa forma, conclui-se que a aplicação desenvolvida atingiu os objetivos propostos, ao automatizar o processo de agendamento de apresentações acadêmicas, promovendo maior organização, controle e eficiência. A pesquisa demonstrou a viabilidade da integração entre tecnologias modernas e demandas institucionais. Espera-se que este trabalho sirva de base para futuras melhorias, como a ampliação de funcionalidades, o aperfeiçoamento da interface e a realização de testes em ambientes reais.

Para continuidade deste trabalho os seguintes trabalhos futuros:

- Melhorar o gerenciamento das salas que possuem disponibilidade de projetores.
- Ampliação de funcionalidades, o aperfeiçoamento da interface.

- Implementar conexão com uma planilha em nuvem para facilitar a consistência dos dados

REFERÊNCIAS

AWS, O QUE é uma aplicação Web? – Explicação sobre aplicações Web — AWS. Disponível em: https://aws.amazon.com/pt/what-is/web-application/. Acesso em: 19 nov. 2024.

AXIOS. ***Introdução | Axios Docs*.** Disponível em: https://axios-http.com/ptbr/docs/intro. Acesso em: 19 nov. 2024.

BENTO, A.; LUSTOSA, L. ***Universidade Federal do Pará Campus Universitário de Marabá Faculdade de Computação Bacharelado em Sistemas de Informação*.** [S.l.: s.n.]. Disponível em: https://repositorio.unifesspa.edu.br/bitstream/123456789/153/1/TCC_%20Sistema%20web%20para%20gest%C3%A3o%20acad%C3%AAmica.pdf. Acesso em: 19 nov. 2024.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. ***The unified modeling language user guide***. Boston: Addison-Wesley Professional, 2005.

DO, A. et al. ***Desenvolvimento de um aplicativo para agendar tarefas acadêmicas em dispositivos móveis*.** [S.l.: s.n.]. Disponível em: https://riu.ufam.edu.br/bitstream/prefix/4841/1/PIB-E-0192-2014.pdf. Acesso em: 19 nov. 2024.

FOWLER, M. ***UML distilled: a brief guide to the standard object modeling language***. 3. ed. Boston: Addison Wesley, 2004.

GARRETT, J. J. ***The elements of user experience: user-centered design for the web and beyond*.** Berkeley: New Riders, 2011.

GILLEANES, T. A.; GUEDES, C. ***UML: uma abordagem prática*.** São Paulo: Novatec, 2008.

GUI, K. ***Introdução à arquitetura cliente-servidor*.** Disponível em: https://medium.com/@kaisergui258/introdu%C3%A7%C3%A3o-a-arquitetura-cliente-servidor-dfae4c0218bd. Acesso em: 12 maio 2025.

LARMAN, C. ***Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo*.** Porto Alegre: Bookman, 2004.

MARCUS, R. ***User experience: 5 princípios de usabilidade para seu projeto*.** Disponível em: https://hawk.digital/ux-design-entenda-os-5-principios-de-usabilidade-para-seu-projeto. Acesso em: 26 nov. 2024.

MOZILLA. ***Client-server overview*.** Disponível em: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Client-Server_overview. Acesso em: 19 nov. 2024.

NESTJS, DOCUMENTATION | NestJS - A progressive Node.js framework. Disponível em: https://docs.nestjs.com/. Acesso em: 15 nov. 2024.

NESTJS. **Documentation | NestJS - A progressive Node.js framework.** Disponível em: https://docs.nestjs.com. Acesso em: 05 Jan. 2025.

NIELSEN, J. ***Usability engineering*.** Boston: Academic Press, 1993.

NPM. ***npm | build amazing things*.** Disponível em: https://www.npmjs.com/. Acesso em: 19 nov. 2024.

PATNI, S. ***Pro RESTful APIs: design, build and integrate with REST, JSON, XML and JAX-RS*.** Santa Clara, CA: Apress, 2017.

POSTGRESQL. ***Documentation***. Disponível em: https://www.postgresql.org/docs/. Acesso em: 19 nov. 2024.

POSTMAN, A BEGINNER'S guide to APIs | Postman. Disponível em: https://www.postman.com/what-is-an-api/. Acesso em: 15 nov. 2024.

PRESSMAN, R. S. ***Engenharia de software*. 8. ed**. São Paulo: McGraw-Hill, 2015.

PRISMA. ***Prisma documentation: concepts, guides, and reference*.** Disponível em: https://www.prisma.io/docs/. Acesso em: 19 nov. 2024.

REDHAT, O QUE é API? Disponível em: https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces. Acesso em: 9 nov. 2024.

SEBRAE, AUTOMATIZAÇÃO de processos: rumo à eficiência e inovação. Disponível em: https://www.sebraepr.com.br/comunidade/artigo/automatizacao-de-processos-rumo-a-eficiencia-e-inovacao. Acesso em: 5 nov. 2024.

SOMMERVILLE, I. ***Engenharia de software*. 9. ed.** São Paulo: Pearson, 2011.

TAILWIND CSS. ***Documentation - Tailwind CSS*.** Disponível em: https://tailwindcss.com/docs. Acesso em: 19 nov. 2024.

TANSTACK. ***Overview | TanStack Query Docs*.** Disponível em: https://tanstack.com/query/latest/docs/framework/react/overview. Acesso em: 19 nov. 2024.

TWILIO. ***SendGrid for developers documentation | SendGrid Docs | Twilio***. Disponível em: https://www.twilio.com/docs/sendgrid/for-developers. Acesso em: 19 nov. 2024.

VERCEL. ***Getting started | Next.js*.** Disponível em: https://nextjs.org/docs. Acesso em: 19 nov. 2024.

VERCEL, RENDERING: Server-side rendering (SSR) | Next.js. Disponível em:https://nextjs.org/docs/pages/building-your-application/rendering/server-side-rendering. Acesso em: 19 nov. 2024.

W3C. ***Web content accessibility guidelines (WCAG)*** *2.1*. Disponível em: https://www.w3.org/TR/WCAG21/. Acesso em: 19 nov. 2024.

WORLD Wide Web. Disponível em: https://developer.mozilla.org/en-US/docs/Glossary/World_Wide_Web. Acesso em: 12 nov. 2024.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 • Setor Universitário
Caixa Postal 86 • CEP 74605-010
Goiânia • Goiás • Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br • reitoria@pucgoias.edu.br

RESOLUÇÃO nº 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O estudante Vitor Manoel Alves de Souza do Curso de Engenharia da Computação, matrícula 20201003300030, telefone: 62 999864177 e-mailvitoralvesbvgs@gmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado DESENVOLVIMENTO DE SOFTWARE PARA AGENDAMENTOS DE TRABALHO CONCLUSÃO DE CURSO, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 21 de Junho de 2025.

Assinatura do autor: Vitor Manoel A. de Souza

Nome completo do autor: Documento assinado digitalmente gov.br VITOR MANOEL ALVES DE SOUZA Data: 22/06/2025 17:46:56-0300

Assinatura do professor–orientador: Documento assinado digitalmente gov.br SOLANGE DA SILVA Data: 23/06/2025 09:33:45-0300 Verifique em https://validar.iti.gov.br

Nome completo do professor-orientador: Solange da Silva