UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GABRIEL MARTINS MACHADO CHRISTO
JOÃO VITOR DE FREITAS BARBOSA
VINÍCIUS LIMA MEDEIROS

UM SISTEMA PARA GERENCIAMENTO DE ATAS DE TRABALHOS DE CONCLUSÃO DE CURSO

RIO DE JANEIRO
2024

GABRIEL MARTINS MACHADO CHRISTO
JOÃO VITOR DE FREITAS BARBOSA
VINÍCIUS LIMA MEDEIROS

UM SISTEMA PARA GERENCIAMENTO DE ATAS DE TRABALHOS DE CONCLUSÃO DE CURSO

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

Orientador: Prof. Gabriel P. Silva

RIO DE JANEIRO
2024

Christo, Gabriel Martins Machado
C556s Um sistema para gerenciamento de atas de Trabalhos de Conclusão de Curso / Gabriel Martins Machado Christo, João Vitor de Freitas Barbosa e Vinícius Lima Medeiros. – 2024.

83 f.

Orientador: Gabriel Pereira da Silva.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação)- Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2024.

1. Engenharia de software. 2. Design patterns. 3. Metodologia ágil. 4. Componentes. 5. Conteinerização. I. Barbosa, João Vitor de Freitas. II. Medeiros, Vinícius Lima. III. Silva, Gabriel Pereira da (Orient.). IV. Universidade Federal do Rio de Janeiro, Instituto de Computação. V. Título.

GABRIEL MARTINS MACHADO CHRISTO
JOÃO VITOR DE FREITAS BARBOSA
VINÍCIUS LIMA MEDEIROS

UM SISTEMA PARA GERENCIAMENTO DE ATAS DE TRABALHOS DE CONCLUSÃO DE CURSO

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

Aprovado em 01 de julho de 2024

BANCA EXAMINADORA:

Gabriel P. Silva
D. Sc. (UFRJ)

João Carlos P. Silva
D. Sc. (UFRJ)

Juliana B. S. França
D. Sc. (UFRJ)

Dedicamos este trabalho às pessoas que tornaram possível sua realização. Aos nossos professores, que com paciência, sabedoria e dedicação nos guiaram ao longo desta jornada acadêmica, transmitindo não apenas conhecimento, mas também inspiração e apoio. Suas orientações foram fundamentais para o desenvolvimento deste trabalho e para o nosso crescimento como profissionais e seres humanos.

Além disso, expressamos nossa profunda gratidão aos colegas de turma, cuja colaboração e troca de ideias enriqueceram nossa experiência acadêmica. Juntos, enfrentamos desafios e celebramos conquistas, criando laços que esperamos que perdurem para além desta etapa de nossas vidas.

Por fim, dedicamos este trabalho às futuras gerações de estudantes e pesquisadores, na esperança de que nossos esforços possam contribuir para o avanço do conhecimento e para um mundo mais justo e sustentável.

AGRADECIMENTOS

Gostaríamos de expressar nossa sincera gratidão à Universidade Federal do Rio de Janeiro (UFRJ) pelo suporte institucional fornecido durante este período de estudo. Agradecemos aos departamentos, bibliotecários e demais funcionários que tornaram possível nossa jornada acadêmica, oferecendo recursos e assistência sempre que necessário.

Além disso, extendemos nossos agradecimentos aos profissionais e instituições externas que contribuíram indiretamente para este trabalho, seja por meio de entrevistas, acesso a dados ou outras formas de colaboração. Suas contribuições foram inestimáveis para o desenvolvimento deste projeto.

Por fim, não poderíamos deixar de agradecer aos nossos familiares e amigos, cujo apoio incondicional foi nosso porto seguro nos momentos desafiadores. Seu amor e encorajamento foram essenciais para enfrentar os obstáculos e celebrar as vitórias ao longo desta jornada.

"As invenções são, sobretudo, o resultado de um trabalho teimoso, em que não deve haver lugar para o esmorecimento."

Santos Dumont

RESUMO

Este trabalho propõe o desenvolvimento de um sistema de gerenciamento de atas de Trabalhos de Conclusão de Curso, implementado com o uso de conceitos de *design patterns* (padrões de projeto) e a metodologia ágil XP (Xtreme Programming). O principal objetivo do sistema é facilitar o processo de elaboração, revisão, aprovação e arquivamento das atas na universidade, otimizando a comunicação entre os diferentes envolvidos, como orientadores, alunos e membros da banca examinadora.

Para alcançar esse objetivo, o projeto utilizou técnicas de desenvolvimento ágil do XP, que incluem programação em pares, testes automatizados, integração contínua e iterações curtas. Essa abordagem flexível e iterativa permitiu adaptar o *software* às necessidades em constante evolução dos usuários, garantindo a entrega de valor de forma incremental. Além disso, foram aplicados padrões de projeto para garantir uma arquitetura de *software* robusta, modular e de fácil manutenção, promovendo a reutilização de código e a escalabilidade do sistema. A utilização de componentes e a separação dos contêineres foi fundamental para agilizar o desenvolvimento e garantir a consistência da interface do usuário.

A autenticação foi implementada utilizando o método JWT (JSON Web Tokens), proporcionando uma camada adicional de segurança ao sistema. O *software* foi desenvolvido utilizando tecnologias modernas e recentes, como linguagens de programação atualizadas, *frameworks* de desenvolvimento web e bancos de dados eficientes. A conteinerização foi empregada para facilitar a implantação do projeto em diferentes ambientes e tornar a manutenção mais fácil para as próximas gerações.

Acreditamos que esse conjunto de práticas garantiu um produto final compatível com os padrões atuais de tecnologia, de modo a oferecer uma experiência de usuário aprimorada. Espera-se que a implementação desse sistema traga benefícios significativos para a universidade, simplificando e agilizando os processos administrativos relacionados aos Trabalhos de Conclusão de Curso, além de melhorar a eficiência e a transparência em todo o ciclo de vida desses trabalhos acadêmicos.

Palavras-chave: engenharia de software; design patterns; metodologia ágil; componentes; conteinerização.

ABSTRACT

This work proposes the development of a management system for Final Project minutes, implemented using design patterns concepts and the agile methodology XP (Xtreme Programming). The main objective of the system is to facilitate the process of drafting, reviewing, approving, and archiving the minutes at the university, optimizing communication among the different stakeholders, such as advisors, students, and examination board members.

To achieve this goal, the project utilized agile development techniques from XP, including pair programming, automated testing, continuous integration, and short iterations. This flexible and iterative approach allowed the software to adapt to the constantly evolving needs of users, ensuring the incremental delivery of value.

Additionally, design patterns were applied to ensure a robust, modular, and easily maintainable software architecture, promoting code reuse and system scalability. The use of components and container separation was fundamental to speeding up development and ensuring the consistency of the user interface.

Authentication was implemented using the JWT (JSON Web Tokens) method, providing an additional layer of security to the system. The software was developed using modern and up-to-date technologies, such as current programming languages, web development frameworks, and efficient databases. Containerization was employed to facilitate project deployment in different environments and to make maintenance easier for future generations.

We believe that this set of practices ensured a final product compatible with current technology standards, offering an enhanced user experience. It is expected that the implementation of this system will bring significant benefits to the university, simplifying and speeding up the administrative processes related to Final Projects, as well as improving efficiency and transparency throughout the lifecycle of these academic works.

Keywords: software engineering; design patterns; agile methodology; componentization; containerization.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

O gerenciamento eficiente das atas de Trabalho de Conclusão de Curso (TCC) é fundamental para garantir a qualidade e a transparência dos processos acadêmicos em uma universidade. A motivação para o desenvolvimento deste sistema surgiu da necessidade de aprimorar a organização, a comunicação e a acessibilidade das informações relacionadas às atas de TCC. Atualmente, a universidade enfrenta desafios na gestão desses documentos, que podem levar a atrasos, falta de padronização e dificuldades na comunicação entre os envolvidos no processo.

Com o objetivo de aprimorar essa gestão, este trabalho propõe o desenvolvimento de um sistema específico para essa finalidade. O sistema foi implementado utilizando conceitos de *design patterns* e a metodologia ágil Xtreme Programming (XP), visando facilitar a elaboração, revisão, aprovação e arquivamento das atas, além de otimizar a comunicação entre orientadores, alunos e membros da banca examinadora.

Para alcançar esse objetivo, foram aplicadas técnicas de desenvolvimento ágil do XP, incluindo programação em pares, testes automatizados, integração contínua e iterações curtas. Essa abordagem permite adaptar o sistema às necessidades em constante evolução dos usuários, garantindo a entrega de valor de forma incremental.

Além disso, foram utilizados *design patterns* para garantir uma arquitetura de *software* modular e de fácil manutenção, promovendo a reutilização de código e a escalabilidade do sistema. A autenticação foi implementada utilizando o método JWT (JSON Web Tokens), proporcionando uma camada adicional de segurança ao sistema.

O sistema foi desenvolvido utilizando tecnologias modernas, como linguagens de programação atualizadas, *frameworks* de desenvolvimento web e bancos de dados eficientes. A conteinerização foi empregada para facilitar a implantação do projeto em diferentes ambientes e tornar a manutenção mais fácil para as próximas gerações.

Espera-se que a implementação desse sistema traga benefícios para a universidade, simplificando e agilizando os processos administrativos relacionados aos TCCs, além de melhorar a eficiência e a transparência em todo o ciclo de vida desses trabalhos acadêmicos.

Neste contexto, este trabalho se propõe a explorar os detalhes da implementação do sistema, destacando os benefícios que sua utilização trará para a universidade, tais como a simplificação dos processos administrativos relacionados aos TCCs, a melhoria da eficiência e a transparência em todo o ciclo de vida desses trabalhos acadêmicos.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, abordaremos dois temas centrais para a compreensão e o desenvolvimento de um sistema de gerenciamento de atas de defesa de TCC. Primeiramente, discutiremos o conceito de gerenciamento de atas de defesa de TCC, destacando sua importância, elementos-chave e os benefícios de um sistema automatizado. Em seguida, realizaremos um estudo sobre as tecnologias que podem ser aplicadas no desenvolvimento de um sistema automatizado para o gerenciamento dessas atas, analisando suas características e vantagens.

2.1 CONCEITOS DE GERENCIAMENTO DE ATAS DE DEFESA DE TCC

O gerenciamento de atas de defesa de Trabalho de Conclusão de Curso (TCC) é um processo essencial nas instituições acadêmicas, assegurando a organização, documentação e verificação das etapas finais do percurso acadêmico dos alunos. Abordaremos os conceitos fundamentais que orientam a criação, manutenção e utilização das atas de defesa de TCC, destacando sua importância e os elementos-chave envolvidos nesse processo.

2.1.1 Importância do Gerenciamento de Atas de Defesa de TCC

As atas de defesa de TCC são documentos que registram todos os detalhes de uma defesa de TCC, incluindo informações sobre os participantes, data, horário, local, resultados e observações dos examinadores. A gestão eficiente dessas atas é importante por diversos motivos. Primeiro, as atas servem como documentação oficial, garantindo que todas as informações relevantes estejam documentadas de forma precisa e acessível. Além disso, um sistema bem estruturado de gerenciamento de atas contribui para a transparência e controle do processo, permitindo que todas as partes interessadas, como alunos, orientadores e membros da banca, acompanhem o status e os resultados das defesas. Por fim, com as atas devidamente organizadas, é possível realizar análises estatísticas e consultas rápidas sobre o desempenho dos alunos, a carga de trabalho dos orientadores e a eficiência do processo de defesa de TCC.

2.1.2 Elementos-Chave do Gerenciamento de Atas de Defesa de TCC

O gerenciamento de atas de defesa de TCC envolve vários componentes e processos que garantem a eficiência e a precisão da documentação. Entre os principais elementos estão:

- **Cadastro de Informações**: Inclui a entrada de dados detalhados sobre a defesa, como título do trabalho, palavras-chave, data, horário, local, e a composição da

banca examinadora.

- **Autenticação e Autorização**: Garantir que apenas usuários autorizados possam acessar e modificar as atas é fundamental para manter a integridade dos dados. Isso envolve a implementação de sistemas de autenticação robustos.
- **Edição e Atualização**: O sistema deve permitir a edição de informações, assegurando que qualquer alteração seja devidamente registrada e auditável.
- **Geração de Documentos**: A capacidade de gerar PDFs das atas facilita a distribuição e arquivamento dos documentos oficiais.
- **Análise de Dados**: Ferramentas para visualizar e analisar dados das defesas ajudam na tomada de decisões e na melhoria contínua dos processos acadêmicos.

2.1.3 Exemplo de Sistema de Gerenciamento de Atas Existente

Um exemplo de sistema de gerenciamento de trabalhos de conclusão de curso é o Sistema Integrado de Processos (SIP). Este sistema é utilizado para a gestão de TCCs pela Universidade Federal de Lavras (UFLA) e possui funcionalidades que visam otimizar o processo de documentação e consulta de informações. Embora não tenhamos acesso à documentação completa do SIP, algumas características principais podem ser observadas com base nas informações gerais acessíveis ao público.(UFLA, 2024)

O Sistema SIP oferece funcionalidades como a criação, armazenamento e visualização dos dados de trabalhos de conclusão de curso e autenticação de usuários, abrangendo discentes, docentes, colaboradores, técnico-administrativos e o público em geral. Entre as vantagens do Sistema SIP estão a interface para criação e edição de atas e os recursos de segurança para autenticação de usuários. No entanto, a falta de acesso completo à documentação impede uma análise detalhada das funcionalidades completas.

Para o público em geral, o sistema oferece acesso aos dados completos dos TCCs, funcionando como um repositório. Isso inclui informações detalhadas sobre os trabalhos, permitindo que qualquer pessoa possa consultar e obter dados relevantes sobre os TCCs defendidos na instituição.

2.1.4 Benefícios de um Sistema Automatizado

A implementação de um sistema automatizado para o gerenciamento de atas de defesa de TCC oferece diversos benefícios em comparação com métodos manuais. Primeiramente, a eficiência é significativamente aumentada, pois o sistema reduz o tempo e o esforço necessários para criar, editar e consultar atas de defesa. Além disso, a precisão é aprimorada, minimizando erros humanos e garantindo que todas as informações sejam registradas de forma correta e consistente.

A acessibilidade é outro benefício que vale a pena citar, já que facilita o acesso remoto às atas, permitindo que os participantes do processo possam consultar e gerenciar informações de qualquer lugar. Por fim, sistemas automatizados podem incorporar medidas de segurança avançadas para proteger os dados sensíveis dos usuários e das instituições.

2.2 PADRÕES DE PROJETO E SUA APLICAÇÃO EM DESENVOLVIMENTO DE *SOFTWARE*

No universo dinâmico do desenvolvimento de *software*, os programadores se deparam com desafios recorrentes que exigem soluções robustas e reutilizáveis. É nesse contexto que os padrões de projeto emergem como ferramentas valiosas para a criação de *software* de alta qualidade.

2.2.1 O que são padrões de projeto?

Padrões de projeto representam soluções abstratas para problemas comuns de design de *software*. Eles fornecem modelos comprovados para estruturar classes, objetos e suas interações, otimizando a organização, a flexibilidade e a manutenabilidade do código.(GURU, 2024)

2.2.1.1 Tipos de padrões de projeto

Os padrões de projeto, com visto na Figura 1, são categorizados em três grupos principais, cada um abordando diferentes aspectos do desenvolvimento de *software*:

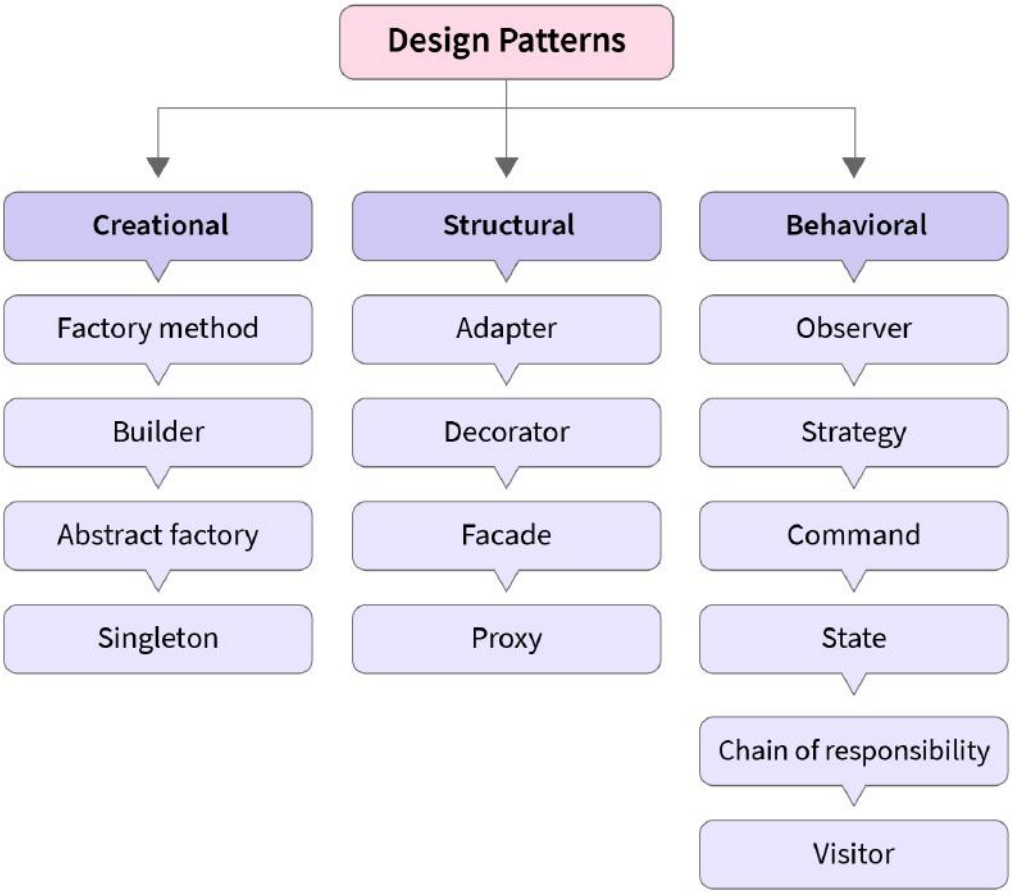
1. **Padrões Criacionais:** Focam na criação de objetos de forma eficiente e flexível, otimizando a inicialização e a composição de classes. Exemplos: Factory Method, Singleton, Builder.

2. **Padrões Estruturais:** Concentram-se na organização e na interconexão de classes, promovendo a modularidade e a reutilização de código. Exemplos: Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Proxy.

3. **Padrões Comportamentais:** Definem a comunicação e a colaboração entre objetos, encapsulando responsabilidades e promovendo a flexibilidade do código. Exemplos: Observer, Strategy, Template Method, Chain of Responsibility, Command, Iterator, Mediator, Memento, State, Visitor.

A utilização de padrões de projeto oferece diversos benefícios para o desenvolvimento de software. Primeiramente, há uma melhoria na qualidade do código, aumentando a legibilidade, a modularidade, a flexibilidade e a manutenibilidade, o que facilita a compreensão, modificação e reutilização do código. Além disso, a reutilização de soluções

Figura 1 – Tipos de padrões de projeto



Fonte: scaler.com

comprovadas fornece soluções testadas e validadas para problemas recorrentes, economizando tempo e esforço. Outro benefício é a maior comunicabilidade entre desenvolvedores, estabelecendo uma linguagem comum entre os programadores e facilitando a colaboração e a compreensão mútua do código. Por fim, a promoção de boas práticas de design incentiva a adoção de princípios de design sólidos, como modularidade, coesão, baixo acoplamento e alta reutilização.

Para ilustrar a aplicação prática dos padrões de projeto, considere os seguintes exemplos: o Padrão Observer (Observador) é utilizado em sistemas de notificação, permitindo que diversos objetos sejam notificados sobre mudanças no estado de um objeto central. O Padrão Strategy (Estratégia) é empregado em algoritmos com comportamentos variáveis, encapsulando diferentes estratégias em classes intercambiáveis. Já o Padrão Template Method (Método Modelo) define um esqueleto de um algoritmo, com pontos de extensão para customização por subclasses, promovendo a flexibilidade sem comprometer a estrutura geral.

Os padrões de projeto representam um conjunto de ferramentas para o desenvolvi-

mento de *software* de qualidade. Ao dominar e aplicar esses padrões de forma consciente, os programadores podem otimizar seus projetos, aprimorar a legibilidade e a manutenibilidade do código, e contribuir para a construção de soluções robustas e escaláveis.

2.3 METODOLOGIA ÁGIL XTREME PROGRAMMING (XP) E SUAS PRÁTICAS

No dinâmico mundo do desenvolvimento de *software*, metodologias ágeis como o Extreme Programming (XP) surgem como bússolas para auxiliar na navegação por projetos complexos e em constante mudança. O XP, com suas práticas focadas na colaboração, na adaptabilidade e no *feedback* contínuo, oferece um caminho promissor para o desenvolvimento de *software* de alta qualidade.(AGILEALLIANCE, 2024)

2.3.1 História e valores fundamentais

O XP, idealizado por Kent Beck no final da década de 1990, se destaca como uma das metodologias ágeis mais influentes da atualidade. Como podemos ver na Figura 2, sua filosofia central reside em quatro valores fundamentais (AMOEBOIDS, 2024):

Comunicação: Incentiva uma comunicação aberta, transparente e constante entre todos os membros da equipe, promovendo o alinhamento e a resolução conjunta de problemas.

Simplicidade: Prioriza a simplicidade no design e na implementação do *software*, evitando soluções complexas e desnecessárias que dificultem o desenvolvimento e a manutenção.

Feedback: Valoriza o *feedback* contínuo do cliente e dos membros da equipe, permitindo identificar falhas precocemente e ajustar o curso do projeto de forma ágil.

Coragem: Enfatiza a importância da coragem para experimentar novas ideias, assumir riscos calculados e aprender com os erros, impulsionando a inovação e a adaptabilidade.

Respeito: Cada membro da equipe deve respeitar os outros, o trabalho que fazem e os processos que foram estabelecidos. Isso cria um ambiente colaborativo e de apoio.

2.3.2 Boas práticas

Para concretizar seus valores fundamentais, o XP se baseia em um conjunto de práticas interligadas que guiam o desenvolvimento do software. Primeiramente, o planejamento relativo adota um planejamento incremental e adaptável, dividindo o projeto em pequenas iterações com entregas frequentes. Isso permite ajustes constantes de acordo com o *feedback* e as mudanças no contexto do projeto, facilitando a adaptação rápida às necessidades emergentes do cliente e a incorporação de melhorias contínuas.

Além disso, o design orientado por testes (TDD) enfatiza a escrita de testes unitários automatizados antes da implementação do código. Essa prática garante a qualidade e a

Figura 2 – Extreme Programming - Ilustração

Fonte: amoeboids.com

confiabilidade do software desde as etapas iniciais. Ao escrever testes antes do código, os desenvolvedores asseguram que cada funcionalidade atenda aos requisitos especificados e que eventuais erros sejam identificados e corrigidos rapidamente, melhorando a estabilidade do sistema ao longo do tempo.

Outra prática fundamental é a programação em par, que promove a colaboração em tempo real entre dois programadores trabalhando em um mesmo computador. Essa abordagem otimiza a comunicação, a resolução de problemas e o compartilhamento de conhecimento. Trabalhando juntos, os programadores podem discutir soluções, identificar erros e aprender um com o outro, resultando em um código de melhor qualidade.

Podemos citar também a integração contínua que realiza integrações frequentes do código de todos os membros da equipe em um repositório central automatizado. Isso permite a detecção precoce de falhas e a entrega contínua de novas versões do software. Com integrações frequentes, é possível identificar e resolver conflitos de código rapidamente, garantindo que a base de código permaneça consistente e funcional.

A refatoração também é incentivada, promovendo a refatoração regular do código para eliminar duplicações, melhorar a legibilidade e otimizar o design. Essa prática garante a qualidade e a sustentabilidade do código ao longo do tempo. Através da refatoração, o código é constantemente melhorado, facilitando a manutenção futura e reduzindo a complexidade desnecessária.

A liberação de clientes envolve o cliente ativamente no processo de desenvolvimento, permitindo que ele forneça *feedback* contínuo e participe da validação das funcionalidades do software. A interação constante com o cliente garante que o produto final atenda às

suas necessidades e expectativas, aumentando a satisfação do cliente e a probabilidade de sucesso do projeto.

Além disso, a metáfora do sistema utiliza uma metáfora compartilhada por toda a equipe para representar o sistema em desenvolvimento. Isso facilita a comunicação, a compreensão mútua e a coesão do código. Uma metáfora clara e compreensível ajuda a equipe a manter uma visão consistente do sistema, facilitando a tomada de decisões e a resolução de problemas.

As histórias de usuário são empregadas como forma de descrever as funcionalidades desejadas pelo cliente, priorizando as mais importantes para o negócio. Cada história de usuário representa uma pequena funcionalidade que pode ser desenvolvida, testada e entregue independentemente, permitindo um desenvolvimento ágil e focado nas necessidades do cliente.

A prática de sentar-se com o cliente promove a comunicação direta entre a equipe de desenvolvimento e o cliente, facilitando o entendimento das necessidades e a resolução de dúvidas. A comunicação direta reduz a chance de mal-entendidos e garante que a equipe de desenvolvimento esteja sempre alinhada com os objetivos do cliente.

Por fim, o teste de aceitação realiza testes de aceitação com o cliente para validar se o software atende às suas expectativas e requisitos, garantindo a entrega de um produto final que atenda às suas necessidades. Os testes de aceitação são realizados com base em critérios definidos pelo cliente, assegurando que o software seja funcional e atenda às especificações acordadas.

2.3.3 Benefícios

Adotar o XP em um projeto de software oferece diversos benefícios. Um dos principais é a maior qualidade do software, já que o foco em testes, na refatoração e no feedback contínuo garante a entrega de um software robusto, confiável e livre de falhas. Esse enfoque sistemático na qualidade desde as etapas iniciais até a finalização do projeto assegura que o produto final seja de alta qualidade.

Além disso, o XP proporciona maior agilidade e flexibilidade. O planejamento incremental e adaptável permite que a equipe se ajuste rapidamente às mudanças nos requisitos e no contexto do projeto. Isso é essencial em um ambiente dinâmico de desenvolvimento de software, onde os requisitos podem evoluir rapidamente. Essa capacidade de adaptação garante que o projeto possa responder eficazmente a novas demandas e desafios.

Outro benefício significativo é a maior satisfação do cliente. O envolvimento ativo do cliente no processo de desenvolvimento garante que o software atenda às suas expectativas e necessidades reais. Isso, combinado com a maior produtividade da equipe, que resulta da colaboração, comunicação e *feedback* contínuo, otimiza o trabalho da equipe, aumentando sua eficiência. Por fim, a maior motivação da equipe é alcançada através de um

ambiente de trabalho colaborativo, autonomia e valorização do *feedback*, proporcionando um ambiente de trabalho mais motivador e engajador para todos os membros da equipe.

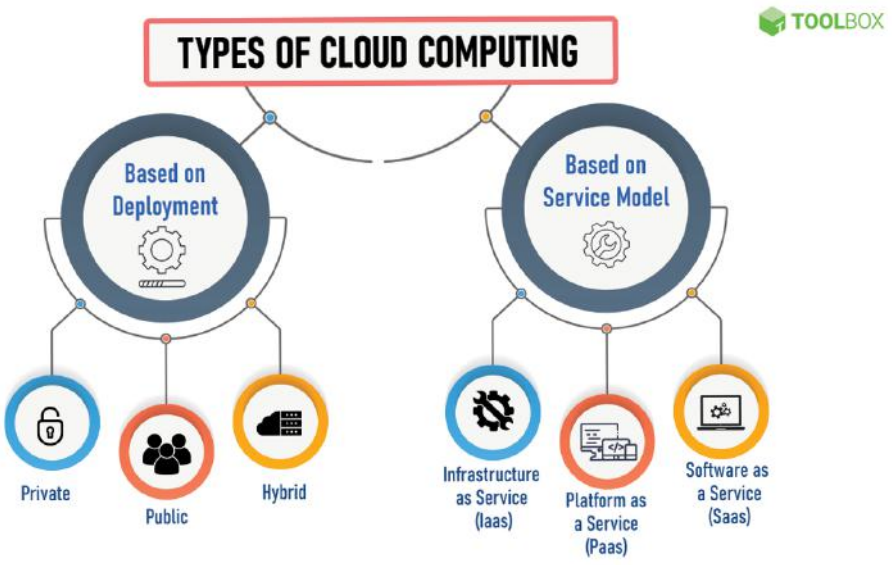
2.4 TECNOLOGIAS MODERNAS PARA DESENVOLVIMENTO DE *SOFTWARE*

O cenário do desenvolvimento de *software* está em constante evolução, impulsionado por avanços tecnológicos que abrem um leque de possibilidades para a criação de soluções inovadoras e eficientes. As empresas que desejam se manter competitivas nesse mercado dinâmico precisam estar atentas às novas tendências e adotar as tecnologias mais recentes para otimizar seus processos e aprimorar seus produtos. Podemos elucidar algumas dessas tecnologias da seguinte maneira:

Nuvem Computacional

A computação em nuvem refere-se ao fornecimento de serviços de computação, incluindo armazenamento, processamento, e rede, através da internet. Esse modelo permite que indivíduos e organizações acessem e utilizem recursos computacionais de maneira flexível e escalável, sem a necessidade de investir em infraestrutura física própria, o que poderia aumentar custos operacionais. A computação em nuvem oferece várias vantagens, como redução de custos, facilidade de manutenção, acesso remoto e escalabilidade quase ilimitada, tornando-se uma solução popular para diversas aplicações, desde armazenamento de dados até execução de aplicativos complexos.(CLOUD.GOOGLE, 2023)

Figura 3 – Nuvem computacional



Fonte: images.spiceworks.com

Podemos separar a computação em nuvem em dois modelos, como podemos ver na Figura 3. Com relação ao modelo de implantação (*based on deployment*), a computação

em nuvem pode ser classificada em três categorias principais: nuvem pública, nuvem privada e nuvem híbrida. A nuvem pública é gerenciada por provedores de serviços externos e é disponibilizada para o público em geral, oferecendo recursos escaláveis sob demanda. A nuvem privada é dedicada a uma única organização, proporcionando maior controle sobre os dados e a infraestrutura, ideal para empresas que precisam de alto nível de segurança e conformidade. A nuvem híbrida combina elementos das nuvens pública e privada, permitindo que dados e aplicativos sejam compartilhados entre elas, proporcionando flexibilidade e otimização dos recursos.

Baseado no modelo de serviço (*based on service model*), a computação em nuvem é categorizada em três tipos principais: Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS). IaaS oferece recursos de infraestrutura, como servidores e armazenamento, que podem ser provisionados e gerenciados pela internet. PaaS fornece uma plataforma que permite aos desenvolvedores criar, testar e implantar aplicativos sem se preocupar com a gestão da infraestrutura subjacente. SaaS oferece software e aplicativos através da internet, acessíveis via navegador, eliminando a necessidade de instalação e manutenção de software local. Esses modelos de serviço permitem que as empresas escolham o nível de controle e gerenciamento que desejam, alinhando-se às suas necessidades e capacidades específicas.

Microsserviços

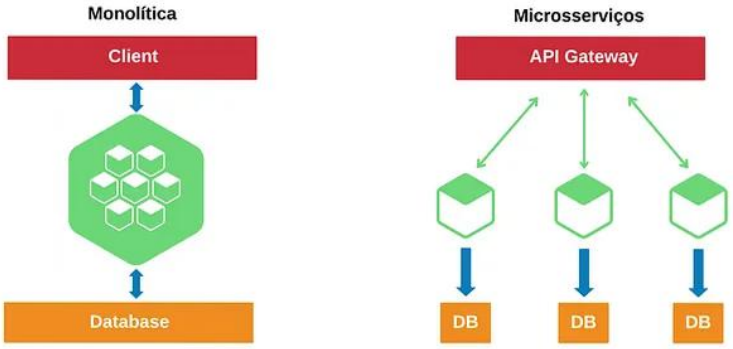
A arquitetura de microsserviços é um estilo de desenvolvimento de *software* onde uma aplicação é composta por um conjunto de serviços pequenos, independentes e altamente coesos. Cada um desses serviços executa um processo único e é projetado para cumprir uma responsabilidade específica.(AWS.AMAZON, 2023)

A modularidade e a independência são características centrais dessa abordagem. Cada microsserviço é um módulo separado que implementa uma funcionalidade específica. Eles podem ser desenvolvidos, implantados e escalados de forma independente. Isso permite que as equipes trabalhem em diferentes serviços ao mesmo tempo sem se atrapalharem.

A comunicação entre os microsserviços é feita através de *APIs* (Interfaces de Programação de Aplicações), geralmente utilizando protocolos como HTTP/REST, gRPC ou mensagens assíncronas. Essa abordagem de comunicação possibilita a integração e a colaboração entre os diversos serviços que compõem a aplicação.

Como podemos ver na Figura 4, uma das principais vantagens da arquitetura de microsserviços é a descentralização. Em termos de dados, cada microsserviço pode ter seu próprio banco de dados, adequado às suas necessidades específicas, evitando a dependência de um grande banco de dados monolítico. No que diz respeito à governança, diferentes equipes podem usar diferentes tecnologias e linguagens de programação para desenvolver seus microsserviços, de acordo com a adequação à tarefa.(LEARN.MICROSOFT, 2023)

Figura 4 – Arquitetura de Microsserviços



Fonte: micro.medium.com

A escalabilidade é outra grande vantagem dessa arquitetura. Cada serviço pode ser escalado de forma independente, permitindo que os recursos sejam alocados eficientemente onde são mais necessários. Isso resulta em um uso mais eficiente dos recursos e na capacidade de atender a demandas variáveis com mais flexibilidade.

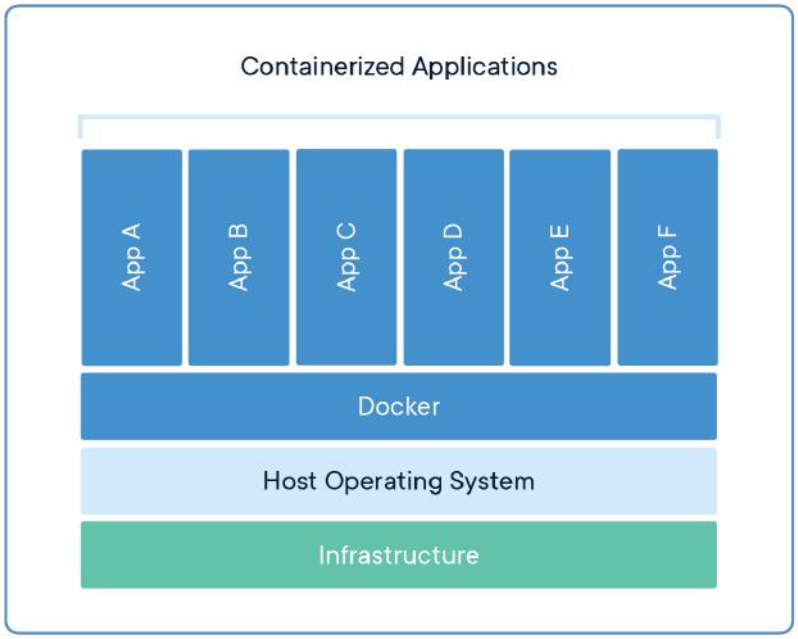
Contêineres

A tecnologia de contêineres, como Docker e Kubernetes, está revolucionando a forma como os *softwares* são implantados e gerenciados. Os contêineres encapsulam um aplicativo e seus dependentes em um pacote isolado, permitindo que sejam facilmente implantados em diferentes ambientes, sejam eles na nuvem, em servidores locais ou em dispositivos embarcados. Essa tecnologia oferece portabilidade, previsibilidade e escalabilidade para aplicações, otimizando o processo de desenvolvimento e implantação.(DOCS.GITHUB, 2024)

Os contêineres são uma abstração leve e portátil que envolve um aplicativo, juntamente com todas as suas dependências, como bibliotecas e configurações necessárias para executá-lo. Como ilustrado na Figura 5, eles são executados em um ambiente isolado, garantindo consistência e previsibilidade independentemente do ambiente de implantação. Isso significa que os desenvolvedores podem criar e testar aplicativos em seus próprios ambientes de desenvolvimento e, em seguida, implantá-los facilmente em servidores de produção ou em nuvens públicas.

A portabilidade oferecida pelos contêineres é especialmente valiosa em um mundo onde as empresas buscam cada vez mais ambientes de implantação híbridos e *multicloud*. Com os contêineres, os aplicativos podem ser executados de maneira consistente em diferentes plataformas de nuvem, facilitando a migração e evitando o bloqueio de fornecedor. Isso

Figura 5 – Docker - container



Fonte: docker.com

proporciona às empresas flexibilidade para escolher a melhor plataforma de nuvem para suas necessidades específicas.

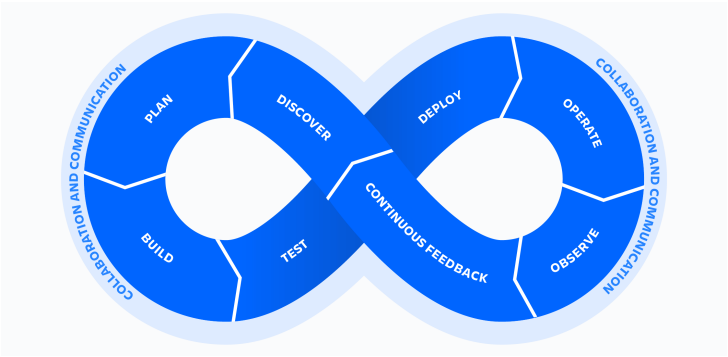
DevOps

A abordagem DevOps combina práticas de desenvolvimento e operação de *software*, promovendo colaboração, automação e entrega contínua de valor. Essa cultura de trabalho permite que as empresas reduzam o tempo de lançamento de novos recursos, melhorem a qualidade do *software* e aumentem a satisfação dos clientes. DevOps utiliza ferramentas e práticas de automação para integrar o desenvolvimento, o teste e a implantação de *software*, otimizando o fluxo de trabalho e agilizando a entrega de novos produtos.(ABOUT.GITLAB, 2024)

No cerne do DevOps está a colaboração entre equipes de desenvolvimento e operações, que tradicionalmente operam de forma separada. Ao unir essas equipes, o DevOps promove uma abordagem holística para o desenvolvimento de *software*, onde o foco é na entrega de valor ao cliente de forma rápida e eficiente. Isso é alcançado por meio de comunicação e compartilhamento contínuos de conhecimento e responsabilidades entre as equipes, como exibido na Figura 6.

A automação desempenha um papel fundamental no DevOps, permitindo que as empresas eliminem tarefas manuais repetitivas e propensas a erros. Ferramentas de automação são usadas para configurar ambientes de desenvolvimento, testar o código, realizar implantações e monitorar o desempenho do aplicativo em produção. Isso não apenas ace-

Figura 6 – Ciclo de vida do DevOps



Fonte: atlassian.com

lera o processo de desenvolvimento, mas também aumenta a consistência e confiabilidade do *software*.

Desenvolvimento Low-Code/No-Code

As plataformas de desenvolvimento Low-Code/No-Code permitem que pessoas com pouca ou nenhuma experiência em programação criem aplicativos robustos e funcionais. Essas ferramentas visuais oferecem interfaces intuitivas e recursos de arrastar e soltar, democratizando o desenvolvimento de *software* e permitindo que as empresas agilizem a criação de soluções para suas necessidades específicas.(POWERAPPS.MICROSOFT, 2024)

As tecnologias modernas para desenvolvimento de *software* oferecem um vasto leque de ferramentas e recursos para que as empresas criem soluções inovadoras, eficientes e escaláveis. Ao adotar essas tecnologias e práticas, as empresas podem se manter competitivas no mercado, aprimorar a experiência do cliente e impulsionar o crescimento do negócio. É fundamental que as empresas estejam atentas às novas tendências e invistam na capacitação de seus profissionais para que possam aproveitar ao máximo o potencial das tecnologias modernas e se destacar em um mercado em constante evolução.

Ao longo da jornada de desenvolvimento deste trabalho, nos dedicamos a explorar e discutir as diversas tecnologias modernas para o desenvolvimento do nosso *software*, buscando compreender suas características, funcionalidades e potenciais aplicações. Através de pesquisas abrangentes e análises criteriosas, avaliamos cada tecnologia em relação às necessidades específicas do nosso projeto, ponderando seus prós e contras e considerando os recursos disponíveis e as habilidades da equipe.

Com base em um entendimento mais aprofundado das tecnologias e do contexto do nosso trabalho, selecionamos as ferramentas e práticas mais adequadas para o desenvolvimento do nosso projeto. A escolha foi guiada por critérios como eficiência, escalabilidade,

flexibilidade, facilidade de uso e compatibilidade com o ambiente de desenvolvimento existente.

2.5 SEGURANÇA DA INFORMAÇÃO: AUTENTICAÇÃO COM JWT

2.5.1 Motivação

A segurança da informação se torna cada vez mais crucial para proteger dados confidenciais e garantir a integridade das transações online. Entre as diversas técnicas de autenticação disponíveis, o JSON Web Token (JWT) se destaca como uma solução robusta e versátil para gerenciar o acesso a sistemas e recursos.(IMPERVA, 2024)

2.5.2 Definição e princípios

O que é JSON Web Token (JWT)?

O JWT é um formato de *token* compacto e autocontido que serve para transmitir informações de autenticação entre diferentes partes de um sistema(JWT, 2024). Conforme mostrado na Figura 7, ele é composto por três seções principais:

Cabeçalho (Header): Contém metadados sobre o *token*, como o algoritmo de assinatura e o tipo de *token*.

Payload (Carga útil): Armazena as informações do usuário autenticado, como nome, ID e permissões.

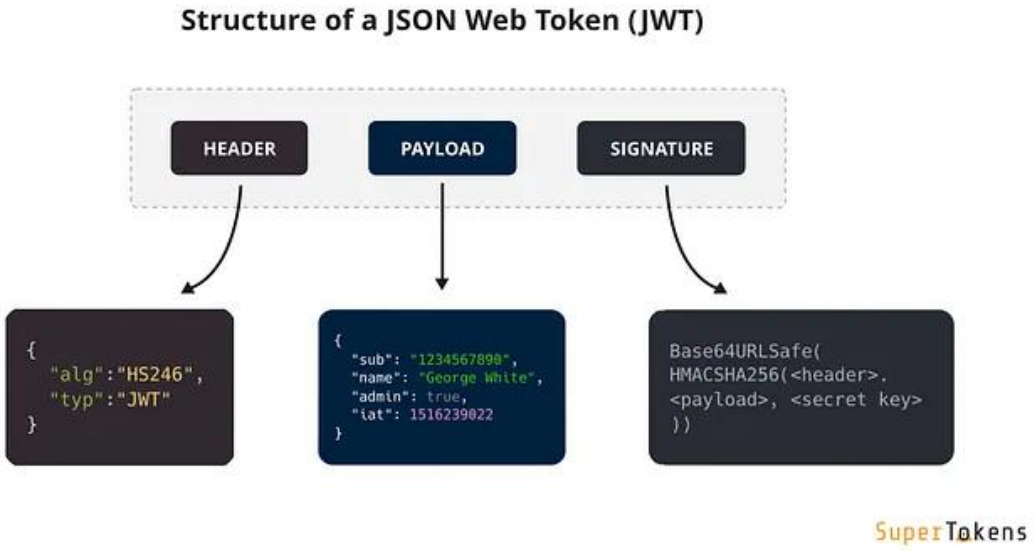
Assinatura (Signature): Garante a integridade e autenticidade do *token*, utilizando criptografia digital para verificar se o *token* foi criado por um emissor confiável e não foi modificado durante a transmissão.

2.5.3 Vantagens da Autenticação com JWT

A utilização de JWT para autenticação oferece diversas vantagens em comparação a métodos tradicionais. Em termos de segurança, a assinatura digital garante a integridade e autenticidade do token, protegendo contra falsificações e interceptações. A eficiência também é um ponto forte, já que o formato compacto do JWT otimiza a transmissão de dados, reduzindo o uso de largura de banda e diminuindo o tempo de resposta.

Outra vantagem significativa é a descentralização. O JWT permite a autenticação sem a necessidade de um servidor central, tornando o sistema mais escalável e resiliente. A flexibilidade do JWT é notável, pois pode ser utilizado em diversos cenários de autenticação, como APIs, *web applications* e *mobile applications*. Além disso, a simplicidade na implementação é um benefício adicional, já que o JWT pode ser facilmente integrado a diferentes plataformas e linguagens de programação.

Figura 7 – Representação da estrutura do JWT



Fonte: supertokens.com

2.5.4 Aplicações Práticas da Autenticação com JWT

O JWT encontra diversas aplicações práticas na área de segurança da informação. Uma dessas aplicações é a autenticação de usuários em APIs, onde o JWT permite que os usuários se autentiquem em APIs RESTful, protegendo o acesso a recursos e funcionalidades. Além disso, o JWT pode ser utilizado para implementar sistemas de Single Sign-On (SSO), permitindo que os usuários se autentiquem em diferentes aplicações com um único login.

Outra aplicação é a autorização de acesso, onde o JWT pode armazenar informações sobre as permissões do usuário, permitindo controlar o acesso a recursos específicos dentro de um sistema. O JWT também pode ser utilizado para a validação de *tokens* de acesso, verificando se o usuário está autenticado e se possui as permissões necessárias para acessar um recurso.

2.5.5 Considerações e Desafios

Embora o JWT ofereça diversos benefícios, é importante considerar alguns aspectos para garantir sua utilização segura e eficaz. A segurança das chaves criptográficas utilizadas para assinar e verificar os *tokens* é essencial, e elas devem ser armazenadas de forma segura para evitar acessos não autorizados. Além disso, a validade dos *tokens* deve ser definida para evitar que sejam usados após a expiração da sessão do usuário.

A implementação de mecanismos de revogação de *tokens* é fundamental para invalidar *tokens* em caso de necessidade, como perda ou roubo de credenciais. Também é importante

definir políticas para o gerenciamento de *tokens*, incluindo seu armazenamento, rotação e expiração.

O JSON Web Token se consolida como uma ferramenta poderosa para autenticação e autorização em sistemas web modernos. Sua combinação de segurança, eficiência e flexibilidade o torna ideal para diversos cenários de autenticação, desde APIs RESTful até sistemas de SSO complexos. Ao implementar o JWT de forma consciente e segura, as empresas podem garantir a proteção de seus dados e aprimorar a experiência do usuário em suas aplicações digitais.

3 METODOLOGIA

Neste capítulo, elucidaremos as metodologias empregadas no desenvolvimento do sistema de gerenciamento de atas de defesa de TCC. A escolha das metodologias adequadas é fundamental para garantir a eficiência e a qualidade do desenvolvimento de software. Descreveremos detalhadamente os métodos e abordagens que utilizamos ao longo do projeto, incluindo as práticas de desenvolvimento, os processos de gerenciamento de projeto e as técnicas de teste e validação.

Além disso, apresentaremos as tecnologias escolhidas para o desenvolvimento do sistema. A justificativa detalhada para a seleção de cada tecnologia, com base em suas características e vantagens específicas, será abordada no próximo capítulo. A combinação das metodologias de desenvolvimento e das tecnologias adotadas visa assegurar que o sistema atenda aos requisitos funcionais e não funcionais estabelecidos, proporcionando uma solução robusta, segura e escalável para o gerenciamento das atas de defesa de TCC.

3.1 DESCRIÇÃO DA ABORDAGEM METODOLÓGICA

A abordagem metodológica adotada neste trabalho baseia-se em uma combinação de princípios da metodologia ágil Xtreme Programming (XP) e técnicas de desenvolvimento de software utilizando design patterns. A escolha por uma metodologia ágil visa proporcionar flexibilidade e adaptabilidade ao processo de desenvolvimento, permitindo uma resposta rápida às mudanças e demandas dos usuário. A metodologia XP enfatiza a colaboração entre os membros da equipe, a comunicação constante e a entrega contínua de incrementos de software funcional. Para garantir a qualidade do produto final, serão aplicadas práticas como programação em pares, onde dois desenvolvedores trabalham em conjunto em um mesmo código, revisões de código e testes automatizados.

Além disso, a utilização de padrões de projeto durante o desenvolvimento do *software* proporcionou uma estrutura sólida e modular, facilitando a manutenção e promovendo a reutilização de código. Os padrões de projeto são soluções comprovadas para problemas recorrentes no desenvolvimento de software, permitindo uma abordagem consistente e eficiente para a implementação das funcionalidades necessárias para o gerenciamento das atas de defesa de Trabalho de Conclusão de Curso (TCC).

A metodologia adotada prevê iterações curtas e frequentes, nas quais novas funcionalidades são implementadas e entregues em intervalos regulares. Isso possibilita uma rápida validação das soluções propostas pelos usuários e permite ajustes ao longo do processo de desenvolvimento. A integração contínua será utilizada para garantir a estabilidade e consistência do código, realizando testes automatizados de forma regular e integrando as alterações feitas pela equipe de desenvolvimento de forma contínua ao repositório principal

do projeto.

A combinação dos princípios ágeis da metodologia XP com a utilização de design patterns proporcionou um processo de desenvolvimento eficiente e adaptável, visando atender às necessidades em constante evolução dos usuários e garantir a qualidade e robustez do software de gerenciamento das atas de TCC. A justificativa detalhada para a seleção das tecnologias utilizadas será abordada no próximo capítulo.

3.2 DETALHAMENTO DAS TÉCNICAS DE DESENVOLVIMENTO UTILIZADAS

3.2.1 Programação em Pares

Como parte do desenvolvimento do nosso programa, adotamos a prática de programação em pares. Durante essa etapa, dois membros da equipe trabalhavam em conjunto em um mesmo código, alternando entre os papéis de "motorista"e "navegador". Essa abordagem promoveu uma colaboração mais estreita entre os desenvolvedores, possibilitando a troca de conhecimentos e ideias em tempo real. Ao programar em pares, pudemos identificar e corrigir erros mais rapidamente, além de garantir uma revisão contínua do código, resultando em um produto final de maior qualidade.

3.2.2 Testes Automatizados

Outra técnica fundamental que utilizamos foi a implementação de testes automatizados. Desenvolvemos uma ampla gama de testes unitários e de integração para garantir a robustez e a confiabilidade do nosso software. Esses testes foram executados de forma automatizada em cada etapa do desenvolvimento, permitindo-nos detectar e corrigir falhas rapidamente. A utilização de testes automatizados também nos proporcionou uma maior segurança ao realizar alterações no código, uma vez que nos permitiu verificar se as novas funcionalidades não afetavam negativamente o funcionamento do sistema.

3.2.3 Integração Contínua

A integração contínua foi uma das peças-chave do nosso processo de desenvolvimento ágil. Configuramos um robusto pipeline de integração contínua, utilizando ferramentas como Github Actions e Docker. Esse pipeline foi projetado para compilar, testar e implantar automaticamente o código sempre que uma alteração era feita no repositório principal do projeto. Com a integração contínua, conseguimos garantir uma sincronização constante e sem interrupções entre as contribuições individuais da equipe, além de acelerar o ciclo de entrega do software. Essa abordagem nos permitiu detectar e corrigir erros de forma rápida e eficiente, garantindo a estabilidade e a confiabilidade do nosso sistema.

3.2.4 Iterações Curtas

Para garantir uma entrega contínua e incremental do nosso programa, optamos por adotar iterações curtas. Dividimos o desenvolvimento em ciclos curtos de trabalho, geralmente de duas a quatro semanas, nos quais implementávamos e entregávamos novas funcionalidades de forma iterativa. Essa abordagem nos permitiu obter feedback dos usuários mais rapidamente e ajustar nossa direção conforme necessário. Além disso, as iterações curtas nos ajudaram a manter o foco nas prioridades do projeto e a responder de forma ágil às mudanças nos requisitos. Ao final de cada iteração, realizávamos uma revisão do progresso e planejávamos as próximas etapas do desenvolvimento, garantindo assim um progresso contínuo e consistente do nosso programa.

3.3 UTILIZAÇÃO DE DESIGN PATTERNS NA ARQUITETURA DE SOFTWARE

Desde o início do projeto, focamos na definição clara dos requisitos e no planejamento detalhado das fases de desenvolvimento. Este planejamento minucioso incluiu a elaboração de diagramas UML, que foram fundamentais para a modelagem dos componentes principais, como ThesisCommitteeMeetingMinute, ExaminationBoardMember, Member e UserInfo. A partir desses diagramas, conseguimos estabelecer uma visão compartilhada entre a equipe, o que facilitou a implementação coerente e alinhada com os objetivos do sistema.

No que diz respeito à arquitetura de software, optamos por uma abordagem baseada em microserviços, visando promover a escalabilidade e a modularidade do sistema. Essa arquitetura permitiu a divisão do sistema em componentes independentes, cada um responsável por uma parte específica da funcionalidade, facilitando o desenvolvimento, o teste e a manutenção. Além disso, a utilização de design patterns foi fundamental para garantir a coesão e a flexibilidade dos microserviços, permitindo uma fácil integração e evolução do sistema ao longo do tempo.

Para garantir a qualidade e a estabilidade do código, implementamos práticas de integração contínua e entrega contínua (CI/CD), utilizando ferramentas como Github Actions e Docker. Isso nos permitiu automatizar o processo de compilação, teste e implantação do software, garantindo que as alterações fossem integradas de forma suave e sem interrupções no fluxo de trabalho da equipe. Além disso, adotamos uma abordagem de DevOps, promovendo a colaboração entre os desenvolvedores e os operadores de sistemas para garantir uma entrega rápida e confiável do software.

A abordagem que adotamos combinou os princípios ágeis da metodologia XP com técnicas modernas de desenvolvimento de software e arquitetura baseada em microserviços. A utilização de design patterns e práticas de integração contínua nos permitiu desenvolver um software robusto, escalável e de alta qualidade, capaz de atender às necessidades em constante evolução dos usuários.

3.4 TECNOLOGIAS ESCOLHIDAS PARA O DESENVOLVIMENTO DO SOFTWARE

3.4.1 Frontend

- React
- Next.js
- Chakra UI

3.4.2 Backend

- Spring Framework
- Spring Security (para JWT)
- Spring Data (para o banco de dados)
- Spring Boot

3.4.3 Banco de Dados

- PostgreSQL

3.4.4 Ferramentas de Desenvolvimento e Implantação

- Docker (para containerização)
- Docker Hub
- GitHub (para controle de versão e integração contínua)
- GitHub Actions (para automação de integração contínua)
- Spring Test (para testes automatizados)
- Test-Driven Development (TDD)

No próximo capítulo deste trabalho, exploraremos em detalhes as razões fundamentais por trás da seleção de cada tecnologia mencionada e sua importância no desenvolvimento do software. Faremos uma análise abrangente de como cada uma das tecnologias, desde o frontend até as ferramentas de desenvolvimento e implantação, contribui para a eficácia e eficiência do projeto. Além disso, apresentaremos uma explicação detalhada de cada tecnologia, abordando suas características principais, vantagens e aplicações específicas no contexto do nosso sistema. Essa análise detalhada será crucial para fornecer uma compreensão do ambiente tecnológico escolhido e justificar as decisões tomadas durante o processo de desenvolvimento do software.

4 DESENVOLVIMENTO DO SOFTWARE

4.1 FRONTEND

4.1.1 Descrição da escolha do React com Next.js e Chakra UI

A escolha do React em conjunto com Next.js e Chakra UI para o desenvolvimento do software foi baseada em uma série de critérios. Inicialmente, o React foi selecionado devido à sua popularidade e eficiência no desenvolvimento de interfaces de usuário dinâmicas e responsivas. Sua abordagem baseada em componentes permite uma organização modular do código, facilitando a manutenção e escalabilidade do software ao longo do tempo(NATIVE, 2024).

Além disso, a integração do Next.js oferece diversas vantagens, como renderização do lado do servidor (SSR) e geração de páginas estáticas (SSG), o que melhora significativamente o desempenho e a experiência do usuário, especialmente em aplicações de larga escala. Essa combinação garante uma rápida renderização inicial das páginas, resultando em tempos de carregamento mais curtos e uma melhor classificação nos mecanismos de busca, fatores cruciais para o sucesso de qualquer software na web atual.(NEXTJS, 2024)

A escolha do Chakra UI como *framework* de design foi motivada pela sua capacidade de fornecer uma base sólida e consistente para a interface do usuário. Com uma ampla gama de componentes pré-construídos e estilizados, o Chakra UI simplifica o processo de criação de interfaces atraentes e acessíveis, permitindo que os desenvolvedores foquem mais na lógica do sistema do que na estilização.

Além disso, o Chakra UI oferece suporte a temas personalizáveis e uma sintaxe baseada em estilo, o que facilita a criação de designs únicos e adaptáveis às necessidades específicas do software. Sua documentação abrangente e comunidade ativa também são aspectos que contribuem para uma curva de aprendizado suave e um desenvolvimento mais eficiente.(UI, 2024)

Figura 8 – React - Chakra UI

Fonte: s3.amazonaws.com

A escolha do React com Next.js e Chakra UI para o desenvolvimento do software foi motivada pela combinação de sua eficiência, desempenho e facilidade de uso. Essas tecnologias proporcionam uma base sólida para a criação de interfaces modernas e responsivas, garantindo uma experiência de usuário excepcional e um produto final de alta qualidade.

4.2 BACKEND

4.2.1 Discussão sobre a escolha do Spring Boot

A decisão de adotar o Spring Boot para o desenvolvimento do sistema foi fundamentada em sua robustez, versatilidade e eficiência. O Spring Boot, um *framework* baseado em Java, oferece uma ampla gama de recursos e funcionalidades que simplificam o desenvolvimento do sistema. Sua escolha pode ser motivada pela popularidade do Java no ambiente corporativo, pela extensa comunidade de desenvolvedores e pela vasta documentação disponível.

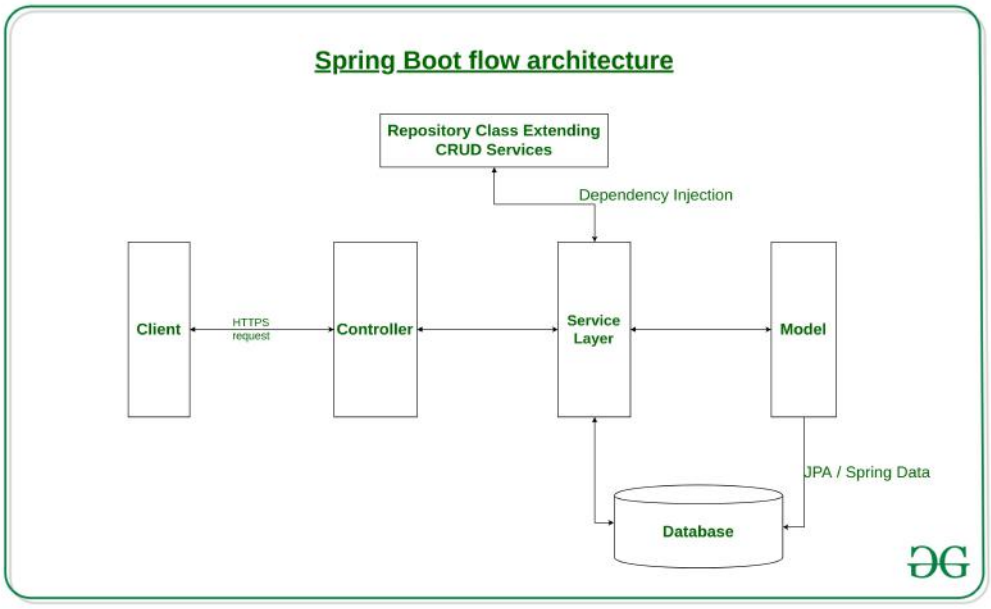
Uma das principais vantagens do Spring Boot é sua capacidade de facilitar a configuração e o desenvolvimento de aplicativos, graças ao seu modelo de convenção sobre configuração. Isso permite aos desenvolvedores concentrarem-se mais na lógica de negócios do que na configuração detalhada do ambiente de desenvolvimento. Além disso, o Spring Boot oferece integração com uma variedade de bibliotecas e *frameworks*, tornando-o uma escolha flexível para uma ampla gama de projetos e requisitos.(SPRING.IO, 2024)

Outra vantagem do Spring Boot é a sua arquitetura baseada em microsserviços. Como podemos ver na Figura 9, com o Spring Boot, é possível modularizar o código em diferentes serviços independentes, o que facilita a escalabilidade e a manutenção do sistema. Além disso, o Spring Boot oferece suporte a padrões como RESTful APIs e Spring Data, simplificando o desenvolvimento de aplicativos com uma arquitetura de software moderna e orientada a serviços.

No entanto, a adoção do Spring Boot também apresenta desafios. Um dos principais desafios é a curva de aprendizado inicial, especialmente para desenvolvedores, que como nós, não estávamos familiarizados com o ecossistema Spring. O Spring Boot possui uma ampla gama de funcionalidades e conceitos, o que pode tornar a sua compreensão e utilização inicial um pouco complexas.

Além disso, a complexidade de gerenciar múltiplos microsserviços pode se tornar um desafio operacional à medida que o sistema cresce. Foi necessário um planejamento da arquitetura e da infraestrutura para garantir uma integração e operação coesa entre os serviços. Por fim, a escolha do Spring Boot para o desenvolvimento do projeto foi motivada por suas vantagens em termos de eficiência, versatilidade e suporte à arquitetura de microsserviços.

Figura 9 – Spring boot



Fonte: media.geeksforgeeks.org

4.2.2 Utilização do Spring Data para integração com o banco de dados PostgreSQL

Foi feito o uso do Spring Data para integração com o banco de dados PostgreSQL e essa escolha foi fundamentada em sua capacidade de simplificar o desenvolvimento do software e proporcionar uma camada de abstração sobre o acesso aos dados. O Spring Data oferece suporte a uma variedade de bancos de dados, incluindo o PostgreSQL, e sua integração com o Spring Framework torna-o uma escolha natural para o projeto. A escolha do PostgreSQL como banco de dados foi motivada por sua robustez, confiabilidade e recursos avançados de SQL, tornando-o adequado para uma ampla gama de sistemas (MKYONG, 2023).

O uso do Sprint Data tem muitas vantagens quando feito da maneira correta, sendo que uma das principais é sua abordagem baseada em interfaces e anotações, que simplifica o desenvolvimento de consultas e operações de CRUD (Create, Read, Update, Delete) no banco de dados. Com o Spring Data, conseguimos escrever menos código *boilerplate* e se concentrar mais na lógica de negócios do sistema. Além disso, o Spring Data oferece suporte a recursos avançados, como consultas nativas, paginação, ordenação e mapeamento objeto-relacional (ORM), o que facilitou o desenvolvimento de consultas complexas e otimizadas.

Podemos citar também sua integração transparente com o Spring Framework e outros módulos do Spring, como o Spring Boot e o Spring MVC. Isso permite aos desenvolvedores criar aplicativos coesos e modularizados, com uma arquitetura consistente e fácil de man-

ter. Além disso, o Spring Data oferece suporte a transações declarativas e gerenciamento de contexto de persistência, garantindo consistência e integridade nos dados do sistema.

A utilização do Spring Data para integração com o banco de dados PostgreSQL ofereceu uma série de vantagens em termos de produtividade, consistência e manutenção do código. No entanto, é importante citar os desafios associados, como a necessidade ocasional de consultas personalizadas e o impacto potencial no desempenho do sistema.

4.2.3 Implementação do Spring Security para autenticação com JWT

A escolha da implementação do Spring Security para autenticação com JWT (JSON Web Tokens) foi motivada pela necessidade de garantir a segurança e a integridade dos dados em um sistema web moderno. O Spring Security é um *framework* amplamente utilizado para lidar com aspectos de segurança em aplicativos Java, oferecendo uma ampla gama de recursos para autenticação e autorização. A utilização de JWT como método de autenticação complementou essa escolha, proporcionando uma maneira segura e eficiente de transmitir informações de autenticação entre cliente e servidor.(DOCS.SPRING.IO, 2023)

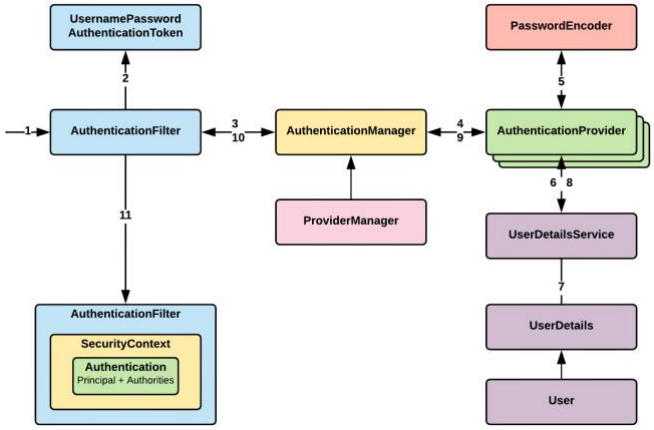
A escalabilidade e flexibilidade destacaram-se como principais vantagens dessa abordagem. O Spring Security facilitou a configuração de uma variedade de métodos de autenticação, desde formulários até *token* e *OAuth*, adaptando-se aos requisitos específicos da nossa aplicação. Ao adotarmos JWT, os *tokens* de autenticação são gerados pelo servidor e esses por sua vez possuem informações detalhadas sobre o usuário autenticado, como suas permissões e papel. Essa estratégia simplificou o processo de autenticação permitindo a implementação de uma política de segurança mais sólida.

A compatibilidade com a arquitetura de microsserviços foi outra vantagem significativa do uso do Spring Security com JWT. Devido à natureza autocontida e autoexplicativa dos *tokens* JWT, não foi necessário armazenar informações de sessão no servidor. Isso simplificou tanto a escalabilidade quanto a manutenção. Além disso, os *tokens* JWT podem ser facilmente verificados e validados por diferentes serviços em um ambiente de microsserviços, garantindo a integridade e a segurança dos dados em toda a arquitetura, como podemos ver na Figura 10, onde a camada de autenticação garante essa integridade.

No entanto, a implementação do Spring Security com JWT também apresentou desafios, especialmente em relação à segurança e à gestão dos *tokens*. Foi preciso garantir a proteção adequada dos *tokens* JWT contra ataques de falsificação e adulteração, implementando medidas de segurança, como a assinatura digital e a validação de *tokens* expirados.

A implementação do Spring Security para autenticação com JWT ofereceu uma solução bem robusta, garantindo a segurança do sistema. Por fim, o Spring Security com JWT pôde fornecer uma camada confiável de proteção para dados sensíveis e usuários autenticados.

Figura 10 – Spring security



Fonte: hermes.dio.me

4.2.4 Arquitetura de microsserviços, quando aplicável

A arquitetura de microsserviços é um estilo de desenvolvimento de software que estrutura uma aplicação como uma coleção de serviços independentes, cada um executando um processo de negócios específico e comunicando-se através de protocolos leves, como HTTP.

Os microsserviços permitem escalar partes específicas da aplicação de acordo com a demanda, o que é especialmente útil em cenários onde certas funcionalidades são mais requisitadas do que outras. Cada serviço pode ser dimensionado de forma independente, resultando em melhor desempenho global. Com a arquitetura de microsserviços, cada serviço é independente, o que facilita a manutenção e atualização do sistema. Mudanças em um serviço não afetam os outros, possibilitando o desenvolvimento e a implantação contínua de novas funcionalidades sem interrupções.

A falha em um microsserviço não compromete todo o sistema, pois outros serviços podem continuar operando normalmente. Além disso, a arquitetura distribuída permite implementar estratégias de recuperação de falhas, como o uso de replicação e balanceamento de carga. Cada microsserviço pode ser desenvolvido e mantido utilizando tecnologias diferentes, adequadas às necessidades específicas da funcionalidade que ele oferece. Isso permite escolher a melhor linguagem, framework e banco de dados para cada serviço, sem ficar preso a uma única tecnologia.

Os microsserviços facilitam a evolução gradual do sistema, permitindo que diferentes partes da aplicação sejam atualizadas e substituídas independentemente umas das outras. Isso é especialmente útil em projetos grandes e complexos, onde diferentes equipes podem trabalhar em serviços distintos sem interferências.

É importante ressaltar que a adoção de microsserviços também traz desafios, como a complexidade na gestão de várias partes distribuídas do sistema, o *overhead* na comu-

nicação entre serviços e a necessidade de uma infraestrutura adequada para suportar a escalabilidade e a resiliência. Foi crucial avaliar cuidadosamente as necessidades e requisitos do projeto, considerando fatores como escopo da aplicação e escalabilidade esperada. Em muitos casos, a abordagem monolítica pode ser mais adequada, especialmente em projetos menores e menos complexos.

4.3 DOCKER E CONTEINERIZAÇÃO

4.3.1 Explicação sobre o uso do Docker para criar e gerenciar contêineres

Docker desponta como uma ferramenta poderosa para criar, distribuir e gerenciar contêineres de maneira simplificada e altamente eficaz.

O Docker é uma plataforma de código aberto que possibilita o empacotamento, distribuição e execução de aplicativos dentro de contêineres. Estes contêineres são unidades de software leves que incluem tudo o que é necessário para executar um aplicativo - código, runtime, bibliotecas, dependências e configurações. Essa abordagem garante consistência e portabilidade em diferentes ambientes.

Para alcançar seus objetivos, o Docker utiliza tecnologias de virtualização a nível de sistema operacional. Ele se vale de recursos do kernel do sistema operacional, como namespaces e cgroups, para criar ambientes isolados. Dessa forma, cada contêiner possui seu próprio ambiente isolado, evitando conflitos entre dependências e garantindo que as aplicações sejam executadas de forma segura e confiável, sem interferência externa.

4.3.2 Benefícios obtidos com a conteinerização

Um dos principais benefícios obtidos com a conteinerização foi a **portabilidade**. Os contêineres encapsulam não apenas o aplicativo, mas também suas dependências e ambientes de execução, tornando-os independentes da infraestrutura subjacente. Isso significa que um contêiner Docker pode ser executado de forma consistente em qualquer ambiente que suporte Docker, seja um ambiente de desenvolvimento local, servidores bare-metal, máquinas virtuais ou na nuvem. Isso simplifica a implantação de aplicativos em diferentes ambientes e reduz as diferenças entre os ambientes de desenvolvimento, teste e produção.

Outro benefício crucial da conteinerização foi o **isolamento e segurança** que ela oferece. Os contêineres oferecem isolamento entre aplicativos e seus ambientes, garantindo que cada aplicativo execute suas operações sem interferir uns com os outros. Como os contêineres compartilham recursos do sistema operacional hospedeiro, como o kernel, eles são mais leves que máquinas virtuais e oferecem um nível de isolamento semelhante. Isso ajuda a evitar conflitos entre dependências e a proteger contra vazamentos de dados e violações de segurança.

Além disso, a conteinerização proporciona uma **utilização mais eficiente dos recursos do sistema**. Ao contrário das máquinas virtuais, que exigem a virtualização

completa de um sistema operacional, os contêineres compartilham o kernel do sistema operacional hospedeiro. Isso resulta em uma utilização mais eficiente dos recursos do sistema, tornando-os ideais para ambientes de desenvolvimento, teste e produção.

A **escalabilidade** é outro ponto forte da conteinerização. Ela facilita a escalabilidade horizontal e vertical dos sistemas. Os contêineres podem ser rapidamente criados e destruídos conforme necessário para lidar com variações na demanda de recursos, permitindo que os sistemas sejam dimensionados de forma eficiente para atender às necessidades do usuário. Além disso, ferramentas de orquestração de contêineres, como Kubernetes, ajudam a automatizar o dimensionamento e a distribuição de contêineres em diferentes hosts e ambientes.

Finalmente, a conteinerização simplifica a implantação e atualização de sistemas. Os contêineres encapsulam todas as dependências e configurações necessárias para executar o sistema, garantindo uma implantação consistente e sem tempo de inatividade. Além disso, as atualizações de sistema podem ser facilmente gerenciadas por meio de imagens Docker e ferramentas de automação, facilitando a manutenção e o gerenciamento contínuo do sistema.

4.3.3 Como os contêineres foram configurados para o ambiente de desenvolvimento e produção

Os contêineres são configurados para o ambiente de desenvolvimento e produção por meio de diferentes abordagens e ferramentas que visam garantir consistência, portabilidade e eficiência operacional. Aqui estão algumas das principais práticas e ferramentas utilizadas para configurar contêineres em ambos os ambientes:

Dockerfile

Um Dockerfile é um arquivo de texto que contém instruções para a criação de uma imagem Docker. Ele define o ambiente do contêiner, incluindo as dependências, configurações e comandos necessários para executar o aplicativo. Por meio do Dockerfile, é possível especificar a base da imagem, instalar pacotes e bibliotecas, configurar variáveis de ambiente e copiar arquivos para dentro do contêiner.

Imagens Docker

As imagens Docker são pacotes de software que contêm todo o necessário para executar um aplicativo, incluindo o código, as dependências e as configurações. Essas imagens são criadas com base nas instruções definidas em um Dockerfile e podem ser distribuídas e compartilhadas por meio de repositórios, como o Docker Hub. No ambiente de desenvolvimento, as imagens podem ser construídas localmente a partir de um Dockerfile ou

baixadas de um repositório. No ambiente de produção, as imagens são frequentemente implantadas em um registro privado ou em um serviço de nuvem.

Orquestração de Contêineres

Em ambientes de produção, é comum usar ferramentas de orquestração de contêineres, como Kubernetes, Docker Swarm ou Amazon ECS, para gerenciar e escalar contêineres de forma automatizada. Essas ferramentas fornecem recursos para implantar, gerenciar e monitorar contêineres em clusters de hosts, garantindo alta disponibilidade, escalabilidade e recuperação de falhas. Elas também oferecem recursos avançados, como balanceamento de carga, auto-escalonamento e atualizações sem tempo de inatividade.

Variáveis de Ambiente e Configurações Externas

Os contêineres podem ser configurados para aceitar configurações externas por meio de variáveis de ambiente, arquivos de configuração ou volumes montados. Isso permite que as configurações do aplicativo sejam gerenciadas de forma flexível e separada do código-fonte, facilitando a implantação em diferentes ambientes sem a necessidade de modificar o contêiner. Por exemplo, é comum configurar as conexões com banco de dados, endereços de serviço e chaves de API por meio de variáveis de ambiente.

Pipeline de Integração Contínua e Implantação Contínua (CI/CD)

Para automatizar o processo de construção, teste e implantação de contêineres, é comum usar pipelines de CI/CD. Esses pipelines permitem integrar alterações de código, construir imagens Docker, executar testes automatizados e implantar automaticamente as alterações em ambientes de desenvolvimento, teste e produção. Ferramentas populares para CI/CD incluem Jenkins, GitLab CI/CD e GitHub Actions.

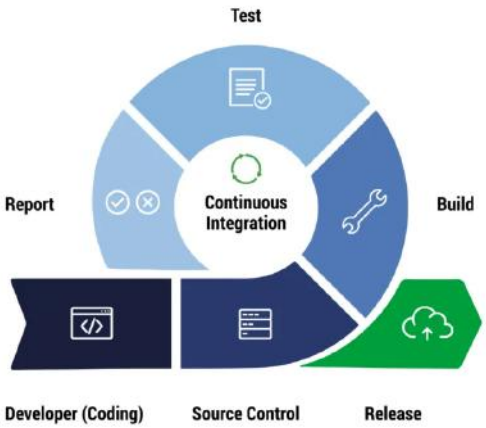
4.4 GITHUB ACTIONS E INTEGRAÇÃO CONTÍNUA

A Integração Contínua (CI) é uma prática fundamental no desenvolvimento de software moderno, permitindo que equipes de desenvolvimento integrem e testem continuamente o código-fonte à medida que é desenvolvido. Essa abordagem visa detectar e corrigir problemas de integração o mais cedo possível, garantindo a qualidade do software e facilitando a entrega contínua de novas funcionalidades aos usuários. O GitHub Actions é uma ferramenta de automação de CI amplamente utilizada que desempenha um papel crucial nesse processo.

Conceito

A Integração Contínua é uma prática na qual os desenvolvedores integram seu código ao repositório compartilhado várias vezes ao dia. Isso é complementado pela execução automática de testes de unidade, integração e aceitação, garantindo que o código integrado não quebre funcionalidades existentes. Como mostrado na Figura 11, a CI promove a detecção precoce de bugs e conflitos de integração, reduzindo assim o tempo e os esforços necessários para resolver problemas de software.

Figura 11 – Integração Contínua



Fonte: tierpoint.com

Papel do GitHub Actions na Integração Contínua

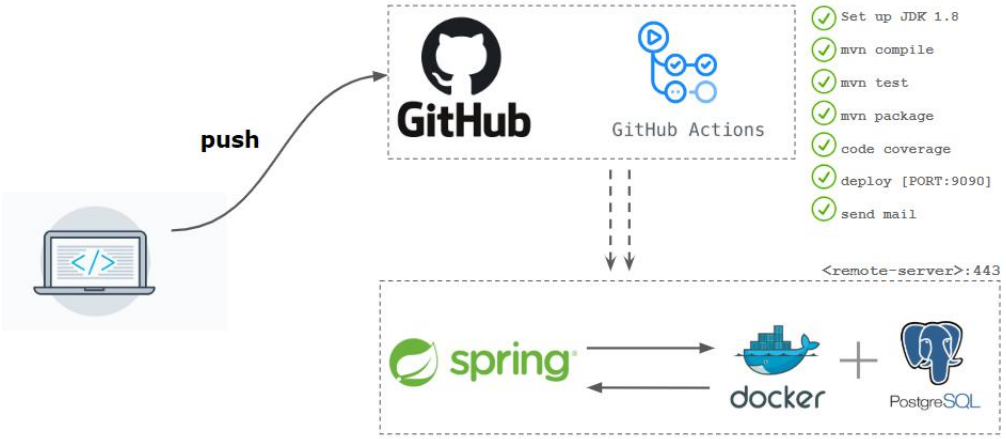
O GitHub Actions é uma ferramenta de automação de código aberto que permite configurar pipelines de CI/CD de forma flexível e escalável. Ele automatiza o processo de integração, teste e implantação de software, ajudando as equipes a implementar a Integração Contínua de forma eficaz. Com o GitHub Actions, podemos definir e executar pipelines que automatizam tarefas como compilação de código, execução de testes automatizados, análise de qualidade de código e implantação em ambientes de desenvolvimento, teste e produção.

Funcionalidades e Vantagens do GitHub Actions

O GitHub Actions oferece uma variedade de funcionalidades que o tornam uma escolha popular para a implementação de CI/CD. Ele suporta integração com uma ampla gama de ferramentas e frameworks de desenvolvimento, permitindo a construção de pipelines

personalizados que atendam às necessidades específicas de cada projeto. Além disso, o GitHub Actions possui uma interface de usuário intuitiva e extensível, facilitando a configuração e monitoramento de pipelines de CI/CD. Como podemos ver na figura 12, a capacidade de escalabilidade horizontal também permite lidar com grandes volumes de compilações e execuções de testes em ambientes distribuídos, já que todo o fluxo é feito de maneira automática, possibilitando um feedback quase que imediato para o desenvolvedor.

Figura 12 – GitHub Actions



Fonte: 8grams.tech

Estudos de Caso e Exemplos de Uso

Vários estudos de caso e exemplos de uso demonstram a eficácia do GitHub Actions na implementação bem-sucedida de práticas de Integração Contínua. Empresas de diversos setores, desde *startups* até grandes corporações, utilizam o GitHub Actions para automatizar seus processos de desenvolvimento e entrega de software. Exemplos de casos de uso incluem a integração contínua de microsserviços em arquiteturas de software distribuídas, a automação de testes de regressão em aplicativos web e móveis, e a implantação automatizada de atualizações de software em ambientes de produção.

Em resumo, o GitHub Actions desempenha um papel fundamental na implementação bem-sucedida de práticas de Integração Contínua, permitindo que equipes de desenvolvimento automatizem o processo de integração, teste e implantação de software. Sua flexibilidade, escalabilidade e ampla adoção na comunidade de desenvolvimento de software o tornam uma escolha popular para organizações que buscam melhorar a qualidade, velocidade e confiabilidade de seus processos de desenvolvimento de software.

4.4.1 Configuração do GitHub Actions para integração contínua

A configuração do GitHub Actions para integração contínua envolve uma série de etapas e práticas que visam automatizar o processo de construção, teste e implantação de software.

1. **Criação do Workflow:**

   O primeiro passo para configurar o GitHub Actions para integração contínua é criar um workflow em um repositório no GitHub. O workflow é definido em um arquivo YAML, que especifica as etapas do processo de CI/CD. Esse arquivo deve ser colocado na pasta `.github/workflows` do repositório.

2. **Configuração dos Jobs:**

   Um workflow pode conter múltiplos jobs, onde cada job representa um conjunto de etapas a serem executadas em um ambiente de execução isolado. Cada job pode ser configurado para rodar em diferentes sistemas operacionais e utilizar diversas versões de ferramentas de desenvolvimento, garantindo flexibilidade e compatibilidade.

3. **Definição das Etapas:**

   Dentro de cada job, definimos uma série de etapas específicas que serão executadas sequencialmente. Essas etapas podem incluir a configuração do ambiente, a instalação de dependências, a execução de testes e a implantação de artefatos. As etapas são definidas utilizando uma sintaxe declarativa em YAML, permitindo a inclusão de comandos shell, scripts personalizados e ações reutilizáveis.

4. **Configuração de Triggers:**

   Os triggers são eventos que acionam a execução de um workflow no GitHub Actions. Podemos configurar os triggers para iniciar um workflow sempre que ocorrerem eventos específicos, como commits no repositório, pull requests abertos, ou em horários programados. Isso garante que o código seja testado e integrado continuamente à medida que é desenvolvido, reduzindo o tempo necessário para detectar e corrigir problemas de integração.

5. **Configuração de Ambientes de Teste:**

   A integração contínua envolve não apenas a compilação e teste do código-fonte, mas também a execução de testes automatizados em diferentes ambientes de teste, como testes de unidade, integração e aceitação. No GitHub Actions, esses ambientes de teste podem ser configurados como etapas adicionais no workflow, utilizando ferramentas de automação de testes compatíveis.

6. **Implantação Automatizada:**

Além dos testes automatizados, o GitHub Actions pode ser configurado para realizar implantações automatizadas de artefatos de software em ambientes de produção. Isso geralmente envolve o uso de ferramentas de implantação como Ansible, Chef ou Kubernetes, integradas ao GitHub Actions por meio de ações personalizadas. A implantação automatizada garante que as alterações de código sejam entregues rapidamente aos usuários finais, com um mínimo de intervenção manual.

7. **Monitoramento e Relatórios:**

   Por fim, o GitHub Actions fornece recursos avançados de monitoramento e geração de relatórios que permitem acompanhar o progresso dos workflows e analisar métricas de desempenho, como tempo de construção, taxa de sucesso dos testes e qualidade do código. Essas informações são essenciais para identificar áreas de melhoria no processo de desenvolvimento e garantir a qualidade e confiabilidade contínuas do software.

Dessa forma, a configuração do GitHub Actions para integração contínua envolve a criação e configuração de workflows, definição de triggers, configuração de ambientes de teste, implantação automatizada e monitoramento contínuo do processo de desenvolvimento. Essas práticas garantem que o código seja testado e integrado continuamente, resultando em um processo de desenvolvimento mais eficiente e confiável.

4.4.2 Descrição dos pipelines de CI/CD

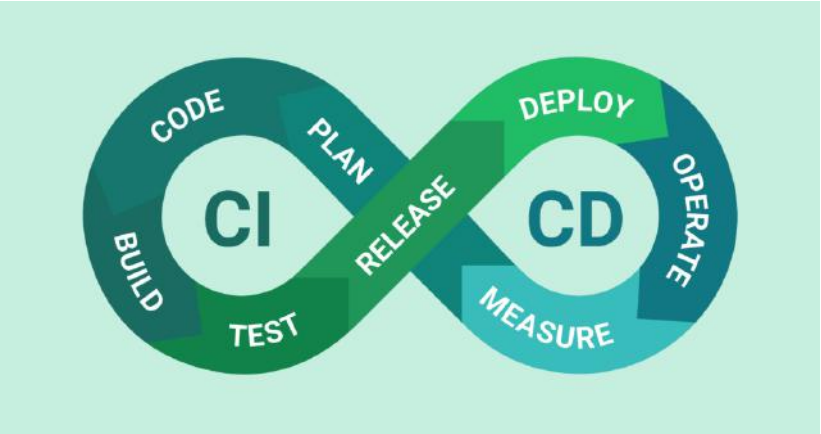
Um pipeline de CI/CD é uma representação estruturada do fluxo de trabalho de integração contínua e entrega contínua em um projeto de desenvolvimento de software. Ele descreve as etapas necessárias para construir, testar e implantar um sistema de forma automatizada e consistente.

Como representado na Figura 13, cada pipeline é composto por uma série de etapas, que representam diferentes fases do processo de construção e implantação do aplicativo. Essas etapas podem incluir a compilação de código-fonte, execução de testes de unidade, integração de código, execução de testes de integração, construção de artefatos de implantação, implantação em ambientes de teste e produção, entre outros.

Dentro de cada etapa, existem passos individuais que descrevem as tarefas específicas a serem executadas. Esses passos podem incluir comandos de shell, invocações de ferramentas de construção, execução de testes automatizados, implantação de artefatos, entre outros. Cada passo é uma unidade de trabalho executada sequencialmente dentro da etapa.

No nosso projeto, utilizamos o GitHub Actions para gerenciar o pipeline de CI/CD. A seguir, na Figura 14, apresentamos um exemplo do arquivo de workflow que automatiza a construção e implantação da nossa aplicação:

Figura 13 – Pipelines no CI/CD



Fonte: media.licdn.com

Os pipelines de CI/CD podem ser acionados por vários eventos, como commits em um repositório de controle de versão, pull requests abertos, agendamento cron, aprovações manuais, entre outros. No exemplo da Figura 14, o gatilho é configurado para iniciar o workflow sempre que houver um push na branch deploy.

Além disso, os pipelines podem ser configurados com parâmetros que permitem a personalização do processo de construção e implantação. Os parâmetros podem incluir a seleção do ambiente de implantação (desenvolvimento, teste, produção), a versão do software a ser implantada, as credenciais de acesso aos serviços externos, entre outros.

Após a conclusão de cada execução do pipeline, é comum enviar notificações para as partes interessadas, informando sobre o status da execução. Isso ajuda a manter todas as partes envolvidas informadas sobre o progresso do processo de desenvolvimento.

4.4.3 Integração do GitHub Actions com o GitHub e como isso agilizou o processo de desenvolvimento

A integração entre o GitHub Actions e o GitHub foi altamente benéfica para o nosso processo de desenvolvimento de software. O GitHub é uma plataforma de hospedagem de código-fonte baseada em Git, amplamente utilizada para colaborar em projetos de código aberto e privados. Ao integrarmos o GitHub Actions com o GitHub, conseguimos automatizar tarefas de CI/CD diretamente a partir de eventos e mudanças no repositório do GitHub, resultando em um processo de desenvolvimento mais ágil e eficiente.

A integração entre o GitHub Actions e o GitHub foi alcançada por meio de webhooks, que são mecanismos de comunicação que permitem que o GitHub notifique o GitHub Actions sempre que ocorrerem eventos específicos no repositório, como commits, pull requests ou lançamentos. O GitHub Actions então inicia automaticamente workflows de CI/CD em resposta a esses eventos, realizando tarefas como compilação de código,

Figura 14 – Github Actions workflow

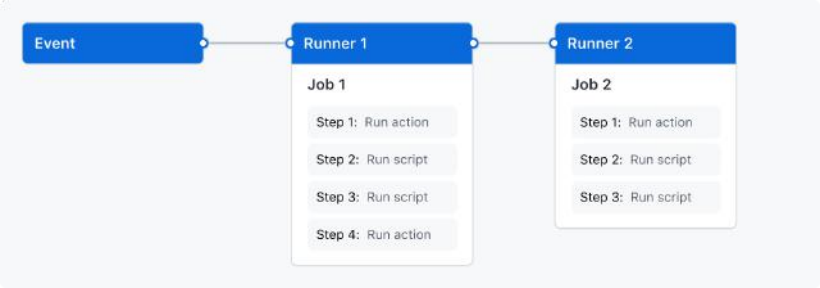
Workflow file for this run
.github/workflows/prod.yml at 59d732b

```yaml
name: Deploy Application

on:
    push:
        branches: [deploy]

jobs:
    build:
        runs-on: ubuntu-latest
        steps:
            - name: Checkout code
              uses: actions/checkout@v3
            #- name: Setup Java
            #  uses: actions/setup-java@v3
            #  with:
            #    distribution: 'temurin'
            #    java-version: '17'
            #- name: Build project
            #  run: mvn clean install -DskipTests -Dmaven.repo.local=/root/.m2
            - name: Login Docker Hub
              run: docker login -u ${{secrets.DOCKER_USERNAME}} -p ${{secrets.DOCKER_PASSWORD}}
            - name: Build server docker image
              run: docker build -t tccufrj/tcc:server ./server/
            - name: Build client docker image
              run: docker build -t tccufrj/tcc:client ./client/
            - name: Push server image docker
              run: docker push tccufrj/tcc:server
            - name: Push client image docker
              run: docker push tccufrj/tcc:client
            - name: Upload application
              uses: appleboy/ssh-action@master
              with:
                host: ${{ secrets.SSH_URL }}
                username: ${{ secrets.SSH_USERNAME }}
                password: ${{ secrets.SSH_PASSWORD }}
                port: ${{ secrets.SSH_PORT }}
                script: |
                  docker login -u ${{secrets.DOCKER_USERNAME}} -p ${{secrets.DOCKER_PASSWORD}}
                  cd /home/tcc_project
                  docker compose down
                  docker compose pull
                  docker compose up -d
```

execução de testes automatizados e implantação em ambientes de teste e produção. Esse fluxo pode ser visto de maneira mais específica na Figura 15

Essa integração trouxe várias vantagens que agilizaram nosso processo de desenvolvimento de software, sendo uma delas a automação de tarefas repetitivas. Ao integrar o GitHub Actions com o GitHub, automatizamos tarefas como compilar e testar o código

Figura 15 – GitHub Actions - GitHub



Fonte: 8grams.tech

sempre que uma alteração ocorre no repositório. Isso eliminou a necessidade de intervenção manual e acelerou o ciclo de feedback, permitindo que identificássemos e corrigíssemos problemas mais rapidamente.

Com a integração contínua entre o GitHub Actions e o GitHub, recebemos feedback imediato sobre a integridade e qualidade do código. Se uma alteração introduzir um erro ou quebrar uma funcionalidade existente, o GitHub Actions nos notifica imediatamente, permitindo que corrijamos o problema antes que ele se torne um obstáculo para outros membros da equipe.

A integração do GitHub Actions com o GitHub também facilitou a implantação automatizada de novas versões do software em ambientes de teste e produção. Os artefatos de construção são automaticamente implantados em servidores de teste para validação e, posteriormente, promovidos para ambientes de produção com confiança, graças à garantia de qualidade fornecida pelos testes automatizados.

Além disso, ao integrar o GitHub Actions com o GitHub, todas as mudanças no código-fonte e nos processos de construção e implantação são registradas e rastreadas de forma automática. Isso fornece uma trilha de auditoria completa que ajuda a identificar quem fez quais alterações, quando e por quê, facilitando a resolução de problemas e a conformidade com os requisitos regulatórios.

Como resultado, a integração entre o GitHub Actions e o GitHub foi uma prática altamente eficaz que agilizou nosso processo de desenvolvimento de software, automatizando tarefas repetitivas, fornecendo feedback imediato, facilitando a implantação de novas versões e fornecendo rastreabilidade e auditoria completas. Essa integração foi essencial para adotarmos práticas modernas de desenvolvimento ágil e entrega contínua.

4.5 TESTES

4.5.1 Discussão sobre a abordagem de Test-Driven Development (TDD)

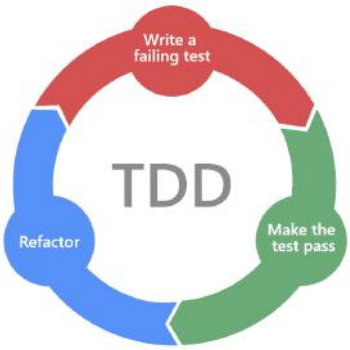
O Test-Driven Development (TDD) é uma metodologia de desenvolvimento de software ágil que enfatiza a escrita de testes unitários automatizados antes de cada trecho de código.

Essa abordagem contrasta com o desenvolvimento tradicional, onde os testes geralmente são escritos após a implementação do código, ou nem sempre são escritos.

No TDD, o processo de desenvolvimento segue um ciclo de três etapas, tal como ilustrado na Figura 16:

1. **Vermelho:** Escrever um teste unitário automatizado que falha porque o código ainda não foi implementado.
2. **Verde:** Implementar o código mínimo necessário para que o teste passe.
3. **Refatorar:** Reorganizar o código para melhorar sua legibilidade, manutenibilidade e design sem alterar sua funcionalidade.

Figura 16 – Desenvolvimento Orientado a Testes - TDD



Fonte: marsner.com

Benefícios do TDD

O TDD (Test-Driven Development) traz diversos benefícios para o desenvolvimento de software. Primeiramente, ele melhora a qualidade do código, pois o foco em testes unitários garante que o código esteja livre de bugs e atenda aos requisitos específicos. Além disso, o TDD contribui para um design mais robusto, já que o processo de refatoração constante resulta em um código mais modular, flexível e fácil de manter.

Outro benefício significativo é a maior confiança no código. A existência de testes unitários automatizados aumenta a confiança na base de código, facilitando a introdução de novas funcionalidades e correções de bugs. Além disso, o TDD reduz o retrabalho, pois a identificação e correção de erros em um estágio inicial do desenvolvimento minimizam o retrabalho e o tempo de depuração subsequente.

Por fim, o TDD promove um desenvolvimento mais focado. Ele ajuda a manter o foco nos requisitos específicos do software, evitando a implementação de código desnecessário. Isso torna o processo de desenvolvimento mais eficiente e direcionado, resultando em um produto final de maior qualidade.

Desafios do TDD

A adoção da metodologia TDD (Test-Driven Development) apresenta alguns desafios, pois há uma curva de aprendizado que deve ser levada em consideração. Aprender e adotar o TDD pode exigir tempo e esforço, especialmente para equipes que não estão familiarizadas com a abordagem. Além disso, o tempo de desenvolvimento inicial pode ser maior. O ciclo de escrita de testes, implementação e refatoração pode aumentar o tempo inicial de desenvolvimento, especialmente para funcionalidades complexas.

Outro desafio significativo é a disciplina. O TDD exige disciplina e comprometimento da equipe para escrever testes antes de cada trecho de código e realizar as refatorações necessárias. Além disso, escrever bons testes é essencial para o sucesso do TDD. Escrever testes unitários eficazes requer habilidade e experiência para garantir que os testes estejam bem estruturados e relevantes.

Aplicabilidade do TDD

O Test-Driven Development foi uma metodologia poderosa que trouxe diversos benefícios para o desenvolvimento do sistema, como código de alta qualidade, design robusto e maior confiança na base de código. No entanto, avaliamos os desafios da metodologia e consideramos sua aplicabilidade ao contexto específico do projeto antes de adotá-la.

Combinamos o TDD com outras práticas de desenvolvimento ágil para criar um processo de desenvolvimento mais completo e eficiente. Utilizamos ferramentas e *frameworks* específicos para TDD que facilitaram a escrita e execução de testes unitários. A comunicação e a colaboração entre os membros da equipe foram cruciais para o sucesso da implementação do TDD.

O TDD ofereceu uma abordagem disciplinada e focada para o desenvolvimento de software, proporcionando benefícios significativos em termos de qualidade, confiabilidade e manutenibilidade do código. Ao avaliar cuidadosamente os benefícios e desafios da metodologia, tomamos decisões informadas sobre se o TDD era a escolha certa para nossos projetos.

4.5.2 Utilização do Spring Test para escrever e executar testes automatizados

O uso do Spring Test garantiu a qualidade e confiabilidade do software por meio de testes automatizados. O Spring Test ofereceu uma solução robusta para as nossas necessidades, fornecendo ferramentas e recursos que simplificaram a escrita e execução de testes automatizados em nosso sistema Spring.

Uma das principais vantagens do Spring Test foi sua integração perfeita com o Spring Framework, amplamente utilizado no desenvolvimento de aplicativos Java. Isso permitiu que aproveitássemos os recursos do Spring, como injeção de dependência e contexto de

sistema, ao escrever testes automatizados. Além disso, o Spring Test se integrou facilmente com outras bibliotecas de teste populares, como Mockito, tornando-o uma escolha versátil para diversos cenários de teste.

Ao utilizarmos o Spring Test, pudemos escrever testes de unidade, testes de integração e testes de ponta a ponta para diferentes camadas e componentes do sistema. Isso incluiu testes para controladores, serviços, repositórios e camadas de acesso a dados, garantindo que cada parte do sistema fosse testada de forma abrangente e eficaz.

Outra motivação para utilizarmos o Spring Test foi sua capacidade de simplificar a configuração e execução de testes automatizados. O Spring Test ofereceu uma variedade de recursos e anotações que facilitaram a escrita de testes, como @SpringBootTest para carregar o contexto de aplicativo do Spring durante os testes de integração, @MockBean e @Autowired para gerenciar dependências, e @Transactional para garantir testes transacionais seguros.

Por fim, a utilização do Spring Test para escrever e executar testes automatizados proporcionou maior confiabilidade e estabilidade ao código desenvolvido, garantindo que todas as partes do sistema fossem testadas de forma abrangente e eficaz em um ambiente de desenvolvimento ágil.

4.5.3 Visão geral do processo

No desenvolvimento do nosso programa, seguimos um fluxo de trabalho bem definido para garantir a eficiência e a qualidade do processo. O fluxo começava com o *code commit*, onde os membros da equipe realizavam alterações e melhorias no código-fonte do programa. Após as modificações serem feitas localmente, elas eram enviadas para o repositório do GitHub por meio de um commit.

Assim que o código era enviado para o GitHub, o GitHub Actions entrava em ação. O GitHub Actions, configurado para monitorar o repositório do GitHub, detectava automaticamente qualquer alteração no código-fonte e iniciava o processo de integração contínua. Esse processo incluía etapas como compilação do código, execução de testes automatizados e análise estática do código.

Após a conclusão bem-sucedida do processo de integração contínua, o GitHub Actions iniciava a construção de uma imagem Docker. Essa imagem continha o ambiente de execução necessário para o programa, juntamente com o código-fonte atualizado. O GitHub Actions então enviava essa imagem para o Docker Hub, um registro centralizado de imagens Docker.

Uma vez que a imagem Docker estava disponível no Docker Hub, ela poderia ser implantada em qualquer ambiente de execução compatível com Docker. Ao adotar esse fluxo, como representado na Figura 12, garantimos uma implantação consistente e controlada do nosso programa, mantendo a integridade do código-fonte e a sincronização entre os diferentes ambientes de execução.

Assim sendo, o fluxo que utilizamos no desenvolvimento do nosso programa seguia uma sequência de *code commit*, GitHub, GitHub Actions, Docker e Docker Hub, garantindo um processo de desenvolvimento eficiente e automatizado, desde a escrita do código até a implantação do software.

5 RESULTADOS E DISCUSSÃO

Neste capítulo, apresentaremos os resultados obtidos durante o desenvolvimento do sistema de gerenciamento de ata de defesa de TCC. O objetivo deste capítulo é exibir e discutir as principais telas e funcionalidades implementadas no sistema, destacando como cada componente contribui para a gestão eficiente das defesas de TCC.

Inicialmente, apresentaremos duas histórias de usuário que ilustram os diferentes fluxos de interação com a plataforma, fornecendo um contexto prático de como o sistema é utilizado. Em seguida, descreveremos as telas em detalhes, abordando suas características e funcionalidades. Cada tela será acompanhada por uma imagem ilustrativa que permitirá uma melhor visualização e compreensão das interfaces desenvolvidas. Posteriormente, discutiremos os aspectos técnicos e funcionais, ressaltando os pontos fortes e identificando áreas que podem ser aprimoradas.

Buscamos demonstrar a eficácia do sistema em atender às necessidades dos usuários e em facilitar o processo de gerenciamento de atas de defesa de TCC. Além disso, a discussão proporcionará uma reflexão sobre o desenvolvimento do projeto, possibilitando sugestões para futuras melhorias e expansões do sistema.

5.1 HISTÓRIAS DE USUÁRIO

Para ilustrar a utilização do sistema, apresentamos duas histórias de usuário que demonstram diferentes fluxos de interação com o sistema.

5.1.1 História de Usuário 1: Criação e Visualização de Ata de Defesa

Maria é uma professora que precisa gerenciar as atas de defesa de TCC dos seus alunos. Primeiramente, ela faz o login no sistema utilizando seu *e-mail* e senha. Após autenticar-se com sucesso, Maria é redirecionada para o *dashboard*. No *dashboard*, ela seleciona a opção para criar uma nova ata de defesa. Ela preenche os campos, como número da ata, título do trabalho, palavras-chave, membros da banca examinadora, data, horário e local da defesa, resultado e nota atribuída. Após preencher todas as informações, Maria salva a ata.

Em seguida, Maria visualiza a ata recém-criada, acessando a seção de visualização de atas de defesa no *dashboard*. Ela localiza a ata específica usando os filtros de busca disponíveis. Ao encontrar a ata, Maria clica no card correspondente para abrir os detalhes completos. Satisfeita com as informações registradas, ela opta por gerar um PDF da ata, utilizando a funcionalidade disponível na tela de visualização. O PDF é gerado e salvo, permitindo que Maria o compartilhe e arquive conforme necessário.

5.1.2 História de Usuário 2: Visualização de Declaração de Participação em Bancas

João é um professor que frequentemente participa como membro de bancas de defesa de TCC. Para acessar sua declaração de participação, ele faz o login no sistema utilizando seu *e-mail* e senha. Após autenticar-se com sucesso, João é redirecionado para o *dashboard*. No *dashboard*, ele navega até a seção de declarações de participação.

Nessa seção, João pode visualizar um resumo das bancas nas quais ele participou, incluindo informações sobre orientações, coorientações e participações em bancas. Além disso, ele pode ver os detalhes das últimas cinco atividades em que esteve envolvido. João decide gerar um PDF da sua declaração de participação, utilizando a funcionalidade disponível na tela. O PDF é gerado, contendo todas as informações detalhadas sobre sua participação, permitindo que João utilize o documento para fins administrativos ou comprobatórios.

5.2 APRESENTAÇÃO DAS TELAS

5.2.1 Login

Figura 17 – Tela de login

A tela de login é a porta de entrada do sistema de gerenciamento de ata de defesa de TCC, sendo uma das interfaces mais importantes para garantir a segurança e a autenticação dos usuários. Conforme mostrado na Figura 17, esta tela foi projetada para ser simples e intuitiva, facilitando o acesso dos usuários ao sistema.

Descrição da Tela

O campo de e-mail é onde o usuário deve inserir seu endereço de e-mail registrado, sendo fundamental para identificar o usuário que está tentando acessar o sistema. Outro

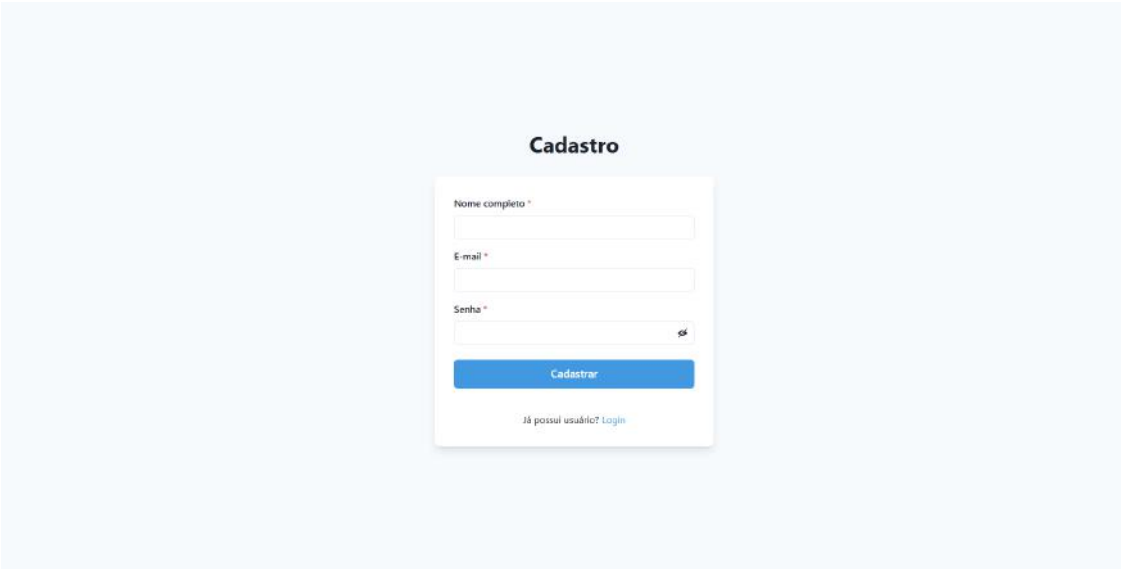
componente é o campo de senha, um campo de entrada de texto com caracteres ocultos, onde o usuário deve inserir sua senha, garantindo que apenas usuários autenticados possam acessar o sistema.

Além disso, há um botão "Esqueci a Senha", um link clicável que redireciona o usuário para a tela de recuperação de senha. Esta funcionalidade é crucial para usuários que não se lembram de sua senha, permitindo que redefinam a senha por meio de um processo de verificação via e-mail. O botão "Finalizar Cadastro"leva o usuário à tela de confirmação de e-mail e posteriormente à finalização de cadastro, sendo útil para novos usuários que já iniciaram o processo de cadastro, mas ainda não concluíram.

Por fim, o botão "Cadastre-se"redireciona o usuário para a tela de registro de novos usuários, uma funcionalidade essencial para permitir que novos usuários criem uma conta no sistema. Esses componentes juntos garantem que a tela de login seja funcional e segura, facilitando a autenticação e o acesso dos usuários ao sistema.

5.2.2 Cadastro

Figura 18 – Tela de cadastro



Como podemos ver na Figura 18, a tela de cadastro possui três campos para criação da conta. O campo de Nome Completo é um campo de entrada de texto onde o usuário deve inserir seu nome completo, sendo essencial para identificar o usuário de forma adequada dentro do sistema.

Outro componente é o campo de E-mail, onde o usuário deve inserir seu endereço de e-mail. Este campo é utilizado tanto para o login quanto para a comunicação e recuperação de senha, sendo uma informação crítica para a criação da conta. O campo de Senha é um campo de entrada de texto com caracteres ocultos, onde o usuário deve inserir sua senha, garantindo que a conta do usuário seja protegida por uma credencial segura.

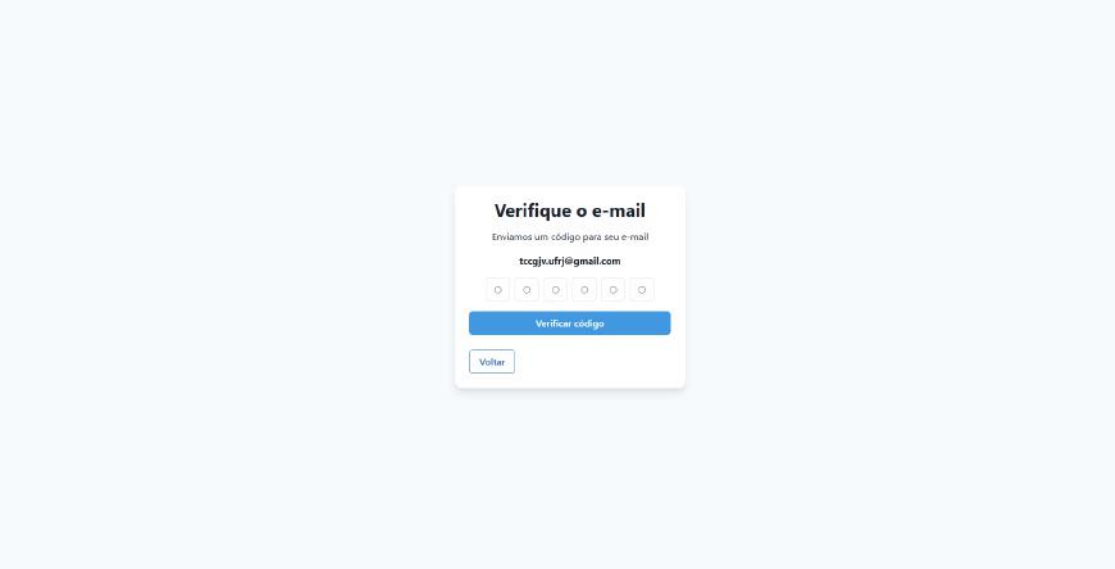
Por fim, o botão "Cadastrar"é o ponto de ação final que permite a conclusão do processo de cadastro. Ao ser clicado, este botão submete as informações preenchidas ao servidor.

A tela de cadastro é um ponto de entrada para novos usuários, garantindo que eles possam se registrar no sistema de maneira rápida e eficiente. A simplicidade dos campos requeridos (nome completo, e-mail e senha) facilita o processo de registro, minimizando barreiras e incentivando novos usuários a se inscreverem.

Além disso, a coleta de informações básicas, como o e-mail e a senha, permite que o sistema implemente medidas de segurança adequadas, como a verificação de e-mail e a proteção por senha. Estas medidas são essenciais para garantir que apenas usuários autorizados possam acessar as funcionalidades internas do sistema.

5.2.3 Verificar e-mail

Figura 19 – Tela de verificar e-mail



A tela de verificação de e-mail possui os seguintes componentes, como apresentado na Figura 19. Um desses componentes é o campo de código de verificação, um campo de entrada de texto onde o usuário deve inserir o código enviado para o seu endereço de e-mail registrado. Este código é um conjunto de números gerados automaticamente pelo sistema. Outro componente importante é o botão "Verificar código", que ao ser clicado, submete o código de verificação inserido pelo usuário. Se o código for válido, o sistema confirma a verificação do e-mail e permite que o usuário prossiga com o próximo passo, seja ele a conclusão do cadastro, redefinição de senha ou acesso ao sistema.

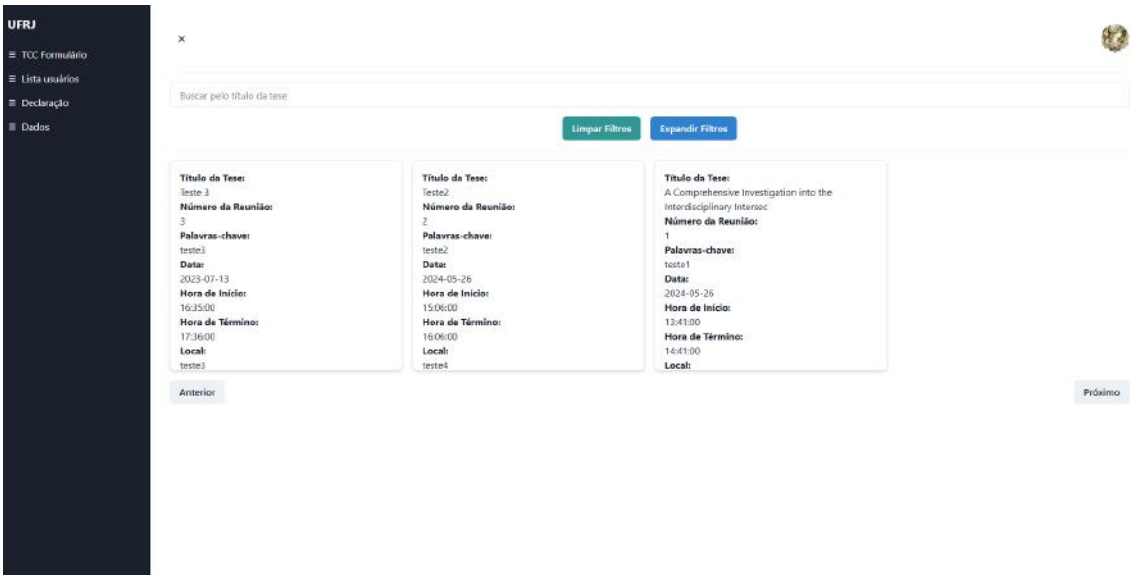
A tela de verificação de e-mail aparece em três situações principais: criação de conta, recuperação de senha e finalização de cadastro. Após o usuário preencher o formulário de cadastro, ele é redirecionado para a tela de verificação de e-mail para confirmar que

o endereço de e-mail fornecido é válido e de sua propriedade. Quando o usuário solicita a recuperação de senha, um código de verificação é enviado para seu e-mail, e ele deve inserir este código na tela de verificação para prosseguir com a redefinição da senha. Caso o usuário tenha iniciado o processo de cadastro, mas não o tenha concluído anteriormente, ele será direcionado para a tela de verificação de e-mail ao tentar finalizar o cadastro.

A verificação de e-mail é um passo crucial para garantir a segurança do sistema, prevenindo o uso de endereços de e-mail falsos ou não autorizados. Este processo ajuda a proteger as contas dos usuários e a integridade dos dados no sistema.

5.2.4 Dashboard

Figura 20 – Tela inicial do sistema - dashboard



A tela de *dashboard* é a interface principal do sistema de gerenciamento de ata de defesa de TCC, acessível imediatamente após o login. Ela serve como um hub central, permitindo ao usuário acessar todas as funcionalidades essenciais do sistema de forma eficiente e intuitiva. A tela de *dashboard* possui componentes e funcionalidades que ajudam o usuário a navegar com facilidade pelo sistema, como representado na Figura 20.

Através da navegação principal, o usuário pode acessar diferentes seções do sistema, como criação, edição e visualização de atas de defesa, dados do perfil, e informações de contato dos usuários.

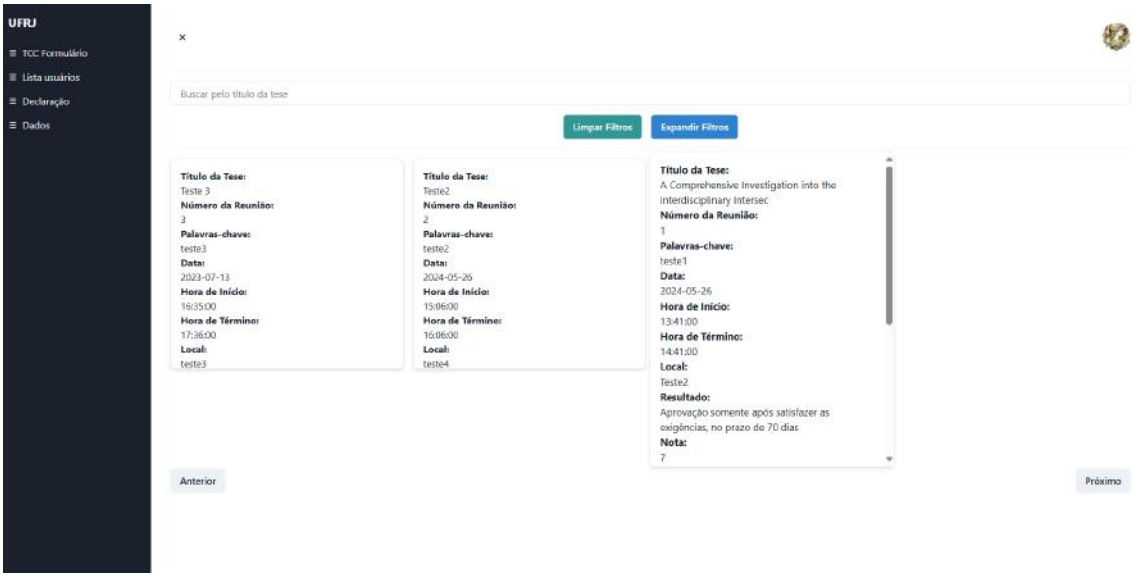
A seção de Atas de Defesa permite ao usuário criar novas atas de defesa, editar atas existentes e visualizar atas já registradas, sendo crucial para a gestão e documentação das defesas de TCC. A seção de Perfil permite ao usuário visualizar e editar seus dados pessoais e de perfil, garantindo que suas informações estejam sempre atualizadas e corretas. Na seção de Contatos, o usuário pode acessar os dados de contato de outros usuários do

sistema, como orientadores e membros da banca, facilitando a comunicação e colaboração entre os envolvidos nas defesas de TCC.

A seção de Declarações oferece aos orientadores e membros da banca a possibilidade de emitir declarações de participação, necessárias para documentar oficialmente sua contribuição nas defesas de TCC. A seção de Gráficos exibe análises e gráficos de dados relevantes, permitindo aos usuários visualizar estatísticas e informações importantes sobre as defesas de TCC e o desempenho do sistema.

No *dashboard*, o usuário pode visualizar as últimas atas de defesa registradas. Ao passar o mouse sobre esses itens, eles são expandidos, permitindo que o usuário veja detalhes adicionais das atas diretamente na tela inicial, como ilustrado nas Figuras 21 e 22.

Figura 21 – Interação do usuário com as últimas atas registradas



A tela de *dashboard* também inclui a funcionalidade de expandir e contrair filtros para a busca de atas já existentes. Isso permite que os usuários realizem buscas detalhadas e encontrem rapidamente as atas específicas de que precisam, utilizando diferentes critérios de filtragem, como mostrado na Figura 23.

5.2.5 Visualização do perfil

A tela de visualização dos dados do perfil é uma interface onde o usuário pode conferir todas as informações pessoais que foram registradas no sistema de gerenciamento de ata de defesa de TCC. Esta tela é essencial para garantir que os dados do usuário estejam corretos e atualizados, além de permitir acesso fácil para alterações quando necessário. Conforme mostrado na Figura 24, a tela de visualização dos dados do perfil possui os seguintes componentes:

Figura 22 – Interação do usuário com as últimas atas registradas

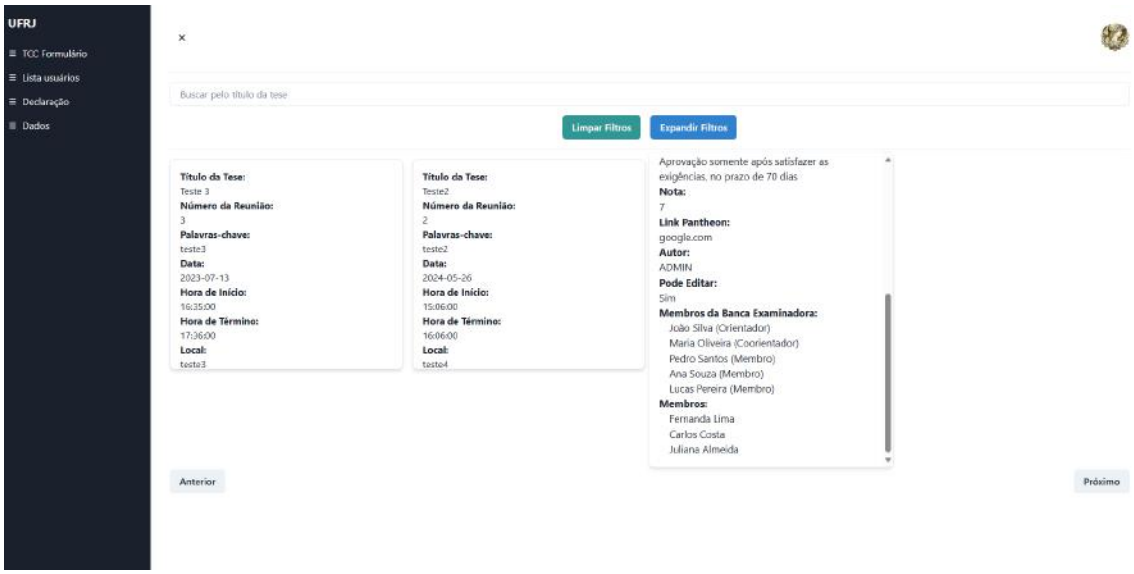
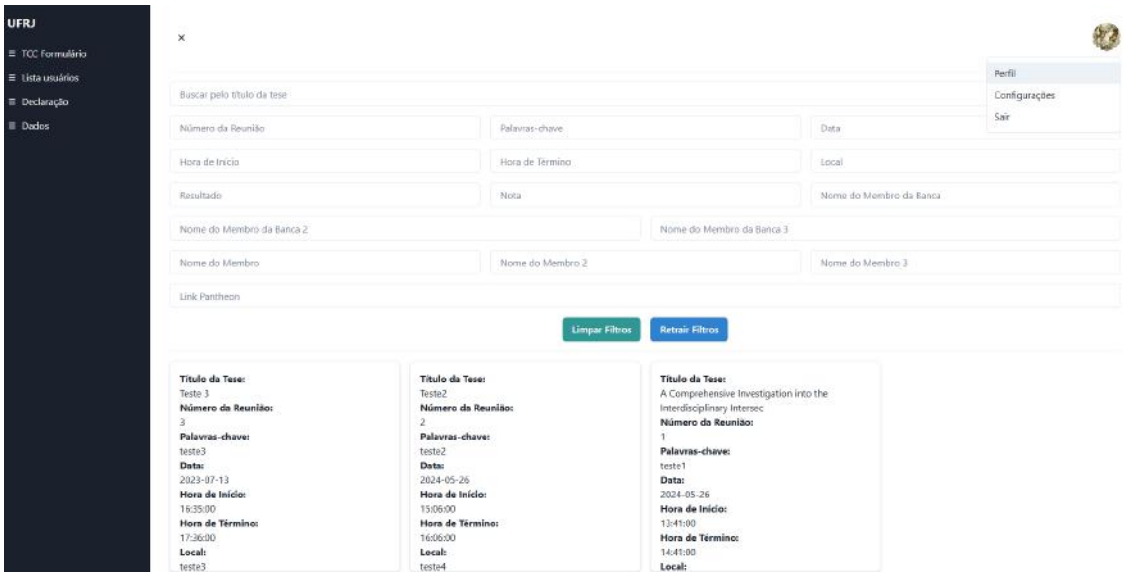
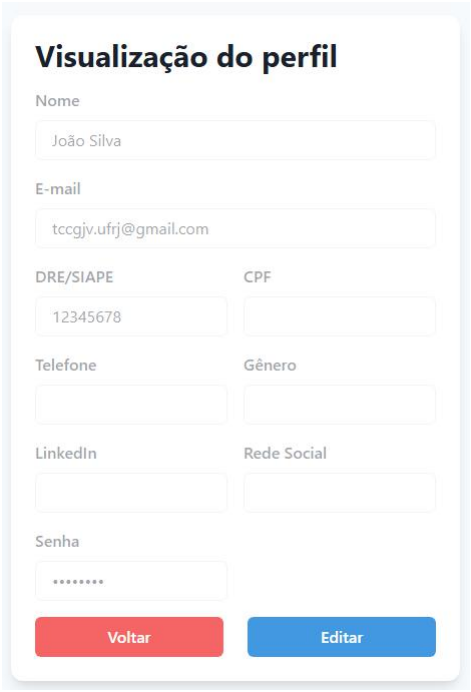


Figura 23 – Tela inicial do sistema - filtros de pesquisa



- **Nome Completo:** Exibe o nome completo do usuário registrado no sistema.
- **E-mail:** Mostra o endereço de e-mail utilizado pelo usuário para acessar o sistema.
- **DRE:** Exibe o número de matrícula ou registro acadêmico do usuário.
- **CPF:** Mostra o CPF do usuário, utilizado para identificação pessoal no Brasil.
- **Telefone:** Exibe o número de telefone de contato do usuário.
- **Gênero:** Mostra a informação de gênero do usuário, conforme registrado no sistema.
- **LinkedIn:** Exibe o link para o perfil do usuário no LinkedIn, caso tenha sido fornecido.

Figura 24 – Tela de visualizar dados do usuários



- **Rede Social:** Mostra um link para outra rede social que o usuário tenha cadastrado.

Além dessas informações, a tela inclui dois botões importantes. O botão "Voltar"redireciona o usuário de volta para a tela inicial (dashboard), permitindo uma navegação fácil e rápida entre as funcionalidades do sistema. O botão "Editar"leva o usuário para a tela de edição dos dados do perfil, onde ele pode atualizar qualquer uma das informações mostradas.

5.2.6 Edição dos dados do perfil

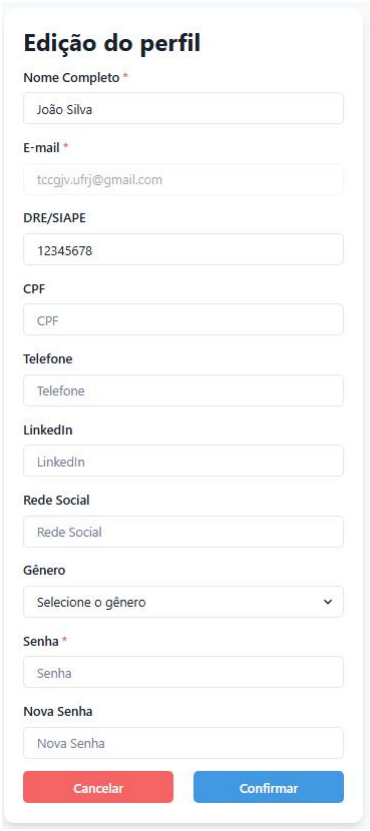
A tela de edição dos dados do usuário é permite que os usuários atualizem suas informações pessoais e de acesso de forma segura e eficiente. Esta tela é projetada para ser intuitiva, garantindo que os usuários possam facilmente modificar seus dados quando necessário. A tela de edição dos dados do usuário possui os seguintes componentes, conforme visualizado na Figura 25.

A tela de edição dos dados do usuário possui campos iguais aos da tela de visualização, mas que podem ser modificados, exceto o campo de e-mail.

A tela inclui campos de senha que têm funcionalidades específicas. O campo de senha deve ser sempre preenchido para confirmar qualquer alteração nos dados do perfil, garantindo que somente o usuário autenticado possa modificar suas informações. O campo de nova senha é opcional e deve ser preenchido apenas se o usuário desejar alterar sua senha atual.

Além desses campos, a tela possui dois botões para a interação do usuário. O botão "Confirmar"submete as alterações feitas nos campos de dados do perfil, sendo necessário

Figura 25 – Tela de editar dados do usuários



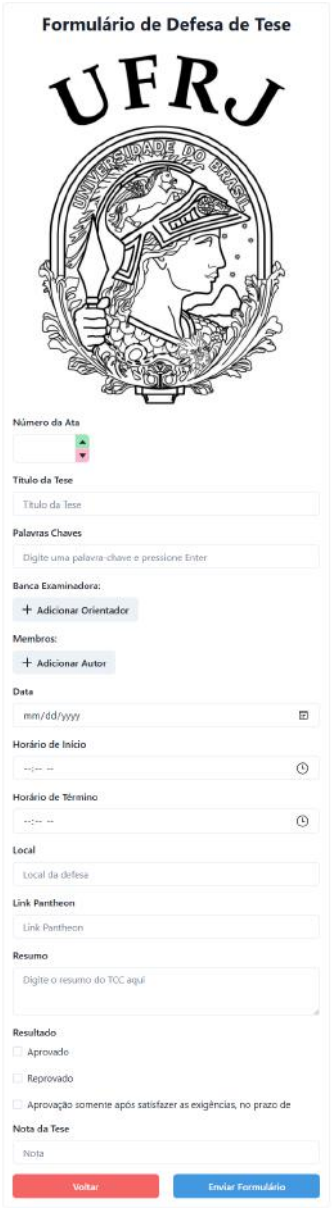
preencher a senha atual para confirmar e salvar as mudanças. O botão "Cancelar"permite ao usuário cancelar as alterações e retornar à tela inicial do sistema sem salvar as mudanças.

5.2.7 Criação de ata de defesa

A tela de criação de ata de defesa permite aos usuários registrar todos os detalhes necessários sobre uma defesa de TCC, garantindo que todas as informações relevantes sejam documentadas de forma precisa e organizada. A tela pode ser acessada a partir da tela inicial (dashboard), os usuários podem navegar até a criação de ata de defesa através do menu lateral pelo botão "TCC Formulário". A tela de criação de ata de defesa possui os seguintes campos, como podemos ver na Figura 26:

- **Número da Ata:** Um campo de entrada de texto para o usuário inserir o número único da ata de defesa.
- **Título do Trabalho:** Um campo de entrada de texto para o usuário inserir o título do trabalho defendido.
- **Palavras-chave:** Um campo de entrada de texto para o usuário inserir as palavras-chave relevantes ao trabalho, facilitando a indexação e a busca.

Figura 26 – Tela de criar ata de defesa



- **Banca Examinadora:** Um campo para selecionar os membros da banca examinadora, sendo o primeiro obrigatoriamente o orientador.
- **Membros da Defesa:** Um campo para selecionar os membros da defesa, com um mínimo de um membro.
- **Data:** Um campo de seleção de data para o usuário escolher a data da defesa.
- **Horário de Início:** Um campo de seleção de hora para o usuário definir o horário de início da defesa.
- **Horário de Término:** Um campo de seleção de hora para o usuário definir o horário de término da defesa.

- **Local:** Um campo de entrada de texto para o usuário inserir o local onde a defesa ocorreu.
- **Link Pantheon:** Um campo de entrada de texto para o usuário inserir o link para o trabalho no repositório Pantheon.
- **Resultado:** Um campo de seleção para o usuário escolher o resultado da defesa.
- **Nota:** Um campo de entrada de texto para o usuário inserir a nota atribuída à defesa.

5.2.8 Visualização de ata de defesa

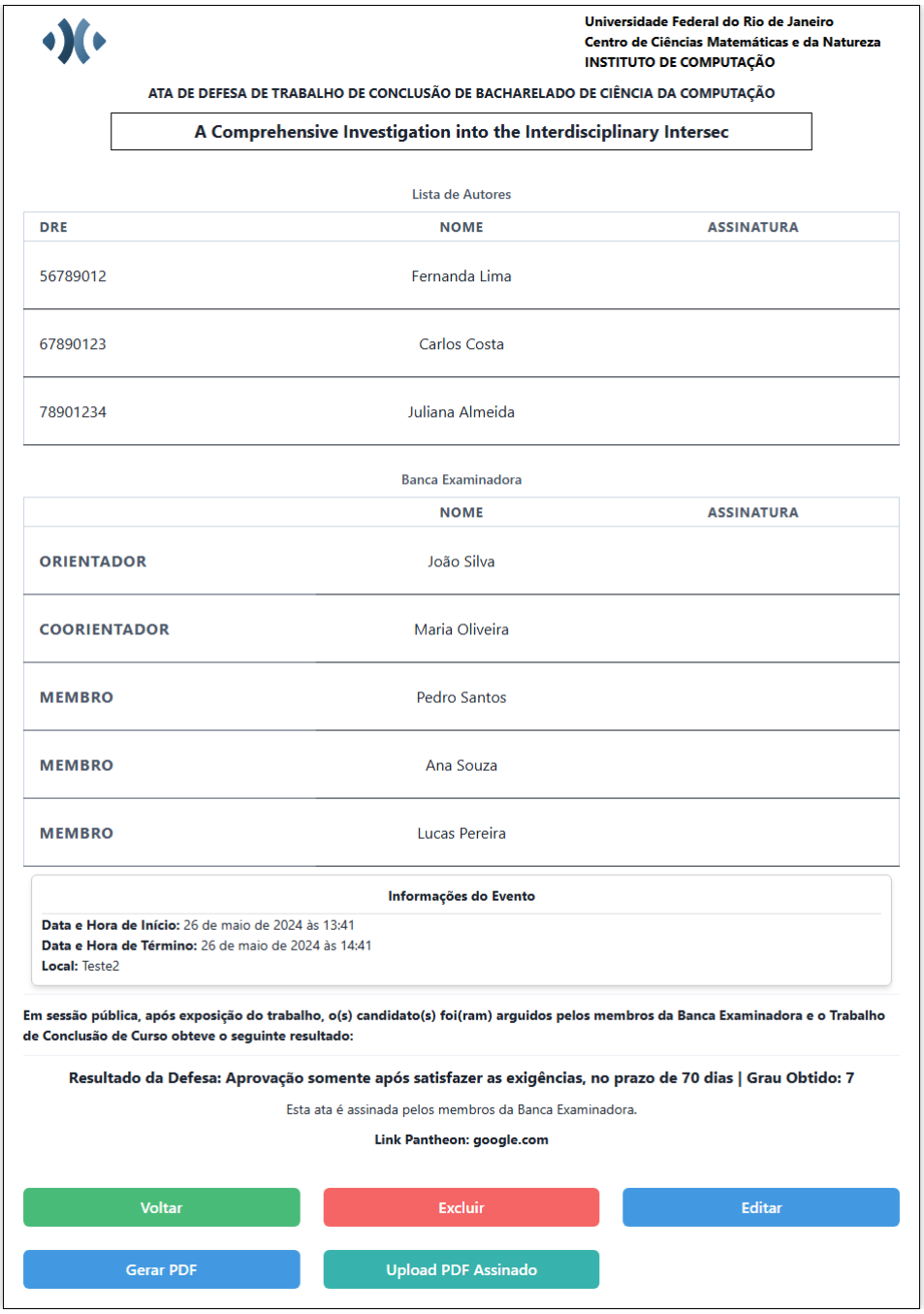
A tela de visualização de ata de defesa permite aos usuários acessarem e revisarem todas as informações registradas sobre uma defesa de TCC. Esta tela é projetada para ser informativa e funcional, facilitando a consulta e o gerenciamento das atas de defesa. A tela de visualização pode ser acessada a partir do *dashboard* (tela inicial) utilizando os filtros de busca disponíveis. Ao localizar a ata desejada, o usuário pode clicar no card correspondente para visualizar os detalhes completos da ata de defesa. Como mostrado na Figura 27, a tela de visualização de ata de defesa exibe os seguintes dados:

- **Título do Trabalho:** O título do trabalho defendido.
- **Banca Examinadora:** A lista dos membros da banca examinadora, destacando o orientador.
- **Membros da Defesa:** A lista dos membros presentes na defesa.
- **Data:** A data da defesa.
- **Horário de Início:** O horário de início da defesa.
- **Horário de Término:** O horário de término da defesa.
- **Local:** O local onde a defesa ocorreu.
- **Link Pantheon:** O link para o trabalho no repositório Pantheon.
- **Resultado:** O resultado da defesa.
- **Nota:** A nota atribuída à defesa.

A tela inclui várias opções de interação:

- **Voltar:** Disponível para todos os usuários, este botão permite retornar à tela anterior (dashboard).

Figura 27 – Tela de visualizar dados da ata de defesa



- **Gerar PDF:** Disponível para todos os usuários, este botão permite gerar um PDF da ata de defesa visualizada.
- **Excluir:** Disponível apenas para usuários que são membros da banca dessa ata de defesa, para o usuário que criou a ata ou para administradores do sistema. Este botão permite excluir a ata de defesa.
- **Editar:** Disponível apenas para usuários que são membros da banca dessa ata de defesa, para o usuário que criou a ata ou para administradores do sistema. Este botão permite editar os dados da ata de defesa.

- **Upload PDF Assinado:** Disponível apenas para usuários que são membros da banca dessa ata de defesa, para o usuário que criou a ata ou para administradores do sistema. Este botão permite fazer o upload do PDF assinado da ata de defesa.
- **Download PDF Assinado:** Disponível para todos os usuários, mas somente se o upload do PDF assinado já tiver sido realizado. Este botão permite baixar o PDF assinado da ata de defesa.

A tela de visualização de ata de defesa é crucial para a consulta e gestão das informações registradas sobre as defesas de TCC. Ela permite que todos os dados relevantes sobre a defesa sejam visualizados de maneira estruturada e acessível. Com o botão "Voltar", a navegação entre as telas do sistema se torna mais fácil, melhorando a usabilidade geral.

A funcionalidade de gerar e baixar PDFs, sejam eles assinados ou não, é essencial para a documentação e o arquivamento das atas de defesa, assegurando que os registros estejam sempre disponíveis para consulta futura. As opções de exclusão, edição e upload de PDF assinado são restritas a membros da banca, ao criador da ata e a administradores. Isso garante que apenas usuários autorizados possam modificar ou gerenciar esses registros críticos, aumentando a segurança e a integridade dos dados.

A opção de download do PDF assinado, disponível para todos os usuários uma vez que o upload tenha sido realizado, garante que as informações oficiais estejam acessíveis quando necessário, promovendo a transparência e a acessibilidade das informações no sistema.

5.2.9 Edição de ata de defesa

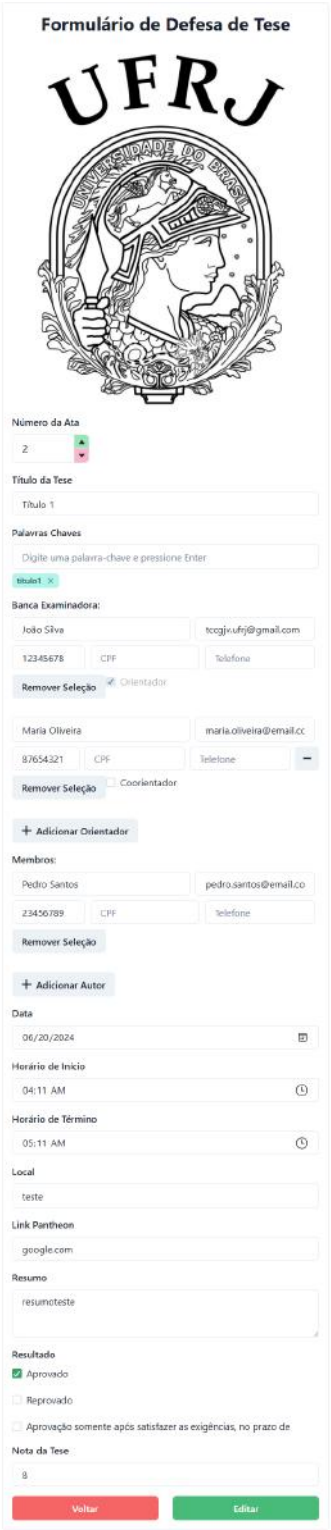
A tela de edição de ata de defesa funciona de maneira similar à tela de criação de ata de defesa, com a diferença de que os dados já vêm pré-preenchidos com as informações da ata de defesa selecionada, como pode ser visto na Figura 28. Isso permite ao usuário editar os dados existentes conforme necessário. Assim, o usuário pode atualizar informações como número da ata, título do trabalho, palavras-chave, membros da banca examinadora, membros da defesa, data, horário de início e término, local, link Pantheon, resultado e nota, mantendo a documentação precisa e atualizada.

5.2.10 Tela da listagem de usuários

A tela que lista usuários exibe uma lista completa de todos os usuários registrados no sistema. Nesta tela, como indicado na Figura 29, é possível visualizar os dados de contato de cada usuário, facilitando a comunicação e a colaboração entre os membros do sistema.

Somente administradores e professores têm acesso a essa tela, garantindo que apenas usuários com permissões adequadas possam visualizar e gerenciar as informações dos usuários. Os administradores têm acesso a uma coluna especial que permite tornar um

Figura 28 – Tela de editar dados da ata de defesa



usuário professor ou revogar esse status, proporcionando controle no gerenciamento de permissões dentro do sistema, como mostrado na Figura 30.

Ao acessar a tela, os usuários autorizados podem ver informações como nome, e-mail, telefone e DRE. Esta funcionalidade é particularmente útil para coordenadores, orientadores e membros da banca, que frequentemente precisam se comunicar com outros

Figura 29 – Lista de usuários

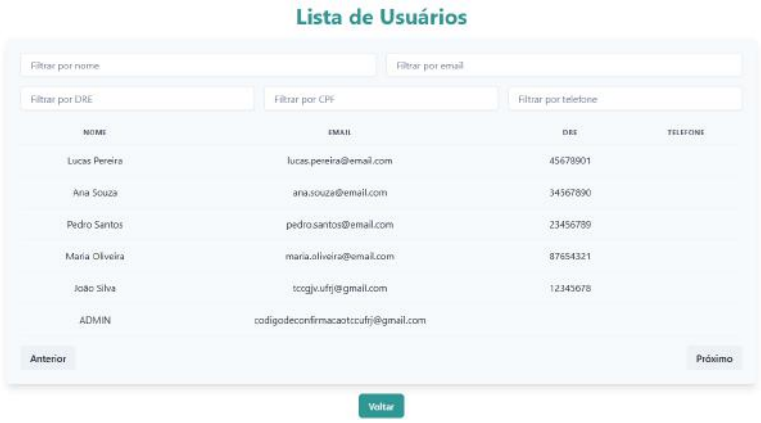


Figura 30 – Lista de usuários pela visão de um admin

participantes do processo de defesa de TCC.

A tela é projetada para ser fácil de usar, permitindo que os usuários encontrem rapidamente os contatos de que precisam. Para facilitar ainda mais a navegação, a tela possui filtros de busca que permitem localizar usuários específicos com base em critérios como nome, e-mail ou outros dados de contato. Essas opções de filtragem e pesquisa melhoram a eficiência da tela, tornando mais rápido e fácil encontrar as informações de contato desejadas.

5.2.11 Tela de declaração

A tela de declaração fornece informações sobre as atividades de orientação, coorientação e participação em bancas do usuário logado. Esta tela é projetada para ser informativa, facilitando a geração de declarações oficiais.

De acordo com o que é visto na Figura 31, ao acessar a tela, o usuário vê um texto que resume quantas orientações, coorientações ou participações em bancas o mesmo possui. Além disso, os dados das últimas cinco atividades são listados na tela, fornecendo detalhes sobre cada uma delas.

Figura 31 – Declaração de participação como orientador, coorientador ou banca

A tela também inclui uma funcionalidade para gerar um PDF da declaração, permitindo que o usuário crie uma versão imprimível e oficial das informações apresentadas. Esta funcionalidade é especialmente útil para documentar formalmente a participação em atividades acadêmicas e pode ser utilizada para fins administrativos, como comprovação de envolvimento em bancas de defesa de TCC.

5.2.12 Tela de gráficos dos dados do sistema

A tela de gráficos dos dados do sistema permite a visualização e análise das informações coletadas no sistema de gerenciamento de ata de defesa de TCC. Esta tela oferece uma representação visual dos dados, facilitando a compreensão e a análise das tendências e padrões. Como podemos ver nas Figuras 32, 33 e 34, temos três abas de gráficos.

A tela também possui um filtro de data inicial e final, permitindo que os usuários selecionem um período específico para a análise dos dados. Este filtro é utilizado para focar em intervalos de tempo específicos e obter informações mais relevantes e contextualizadas.

- Gráfico de Distribuição de Notas das Atas de Defesa: Este gráfico mostra como as notas atribuídas nas defesas de TCC são distribuídas, permitindo identificar padrões de avaliação e a qualidade das defesas ao longo do tempo.
- Gráfico de Distribuição de Resultados: Este gráfico apresenta a distribuição dos resultados das defesas de TCC (aprovado, reprovado, etc.), fornecendo uma visão sobre o desempenho geral dos alunos.
- Gráfico de Número de Defesas por Período: Este gráfico exibe o número de defesas realizadas em diferentes períodos (semestre, ano, etc.), ajudando a identificar picos e tendências sazonais nas defesas de TCC.
- Gráfico de Número de Orientações/Bancas por Professor: Este gráfico mostra quantas orientações e participações em bancas cada professor teve, permitindo avaliar a

Figura 32 – Distribuição de notas e de resultados

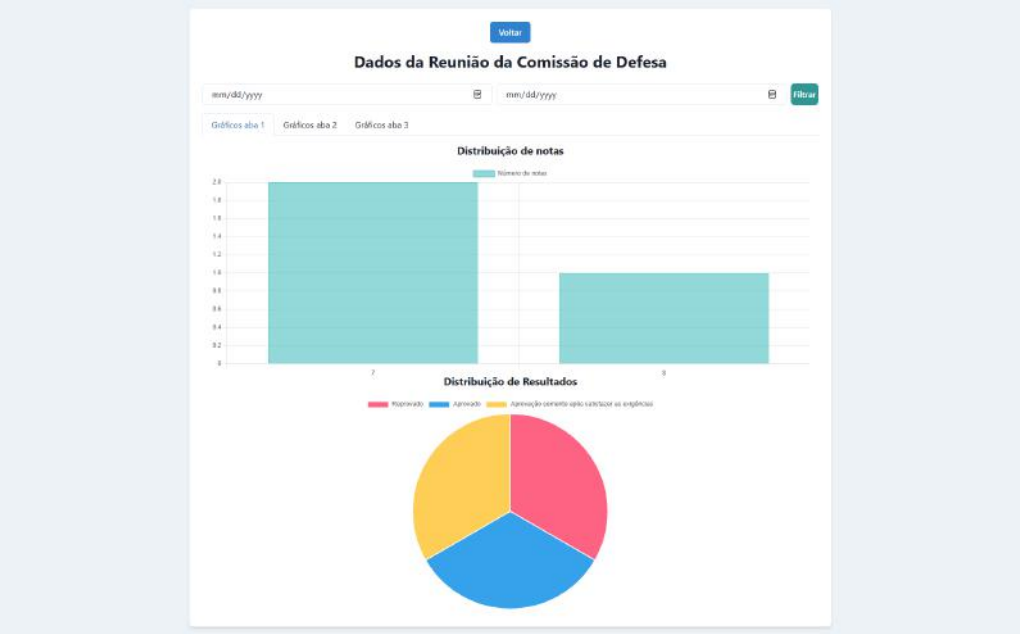


Figura 33 – Distribuição por número de defesas por período e orientações/bancas por professor

distribuição de trabalho entre os docentes e identificar aqueles com maior ou menor carga de orientação.

- Gráfico de Número de Membros da Ata por Gênero: Este gráfico apresenta a distribuição de gênero entre os membros das atas de defesa, ajudando a analisar a diversidade e a representação de gênero.

Esses gráficos são interativos e permitem aos usuários obter conhecimentos valiosos so-

Figura 34 – Distribuição por gênero

bre o funcionamento e o desempenho do sistema de defesa de TCC. A visualização gráfica dos dados facilita a identificação de padrões, tendências e áreas que podem necessitar de atenção ou melhorias. A tela de gráficos dos dados do sistema é uma ferramenta para a análise visual e a compreensão das informações geradas pelo sistema de gerenciamento de ata de defesa de TCC. Ela permite uma análise dos dados, apoiando a tomada de decisões informadas e a melhoria do processo de defesa de TCC.

5.3 DIAGRAMA DE CLASSES

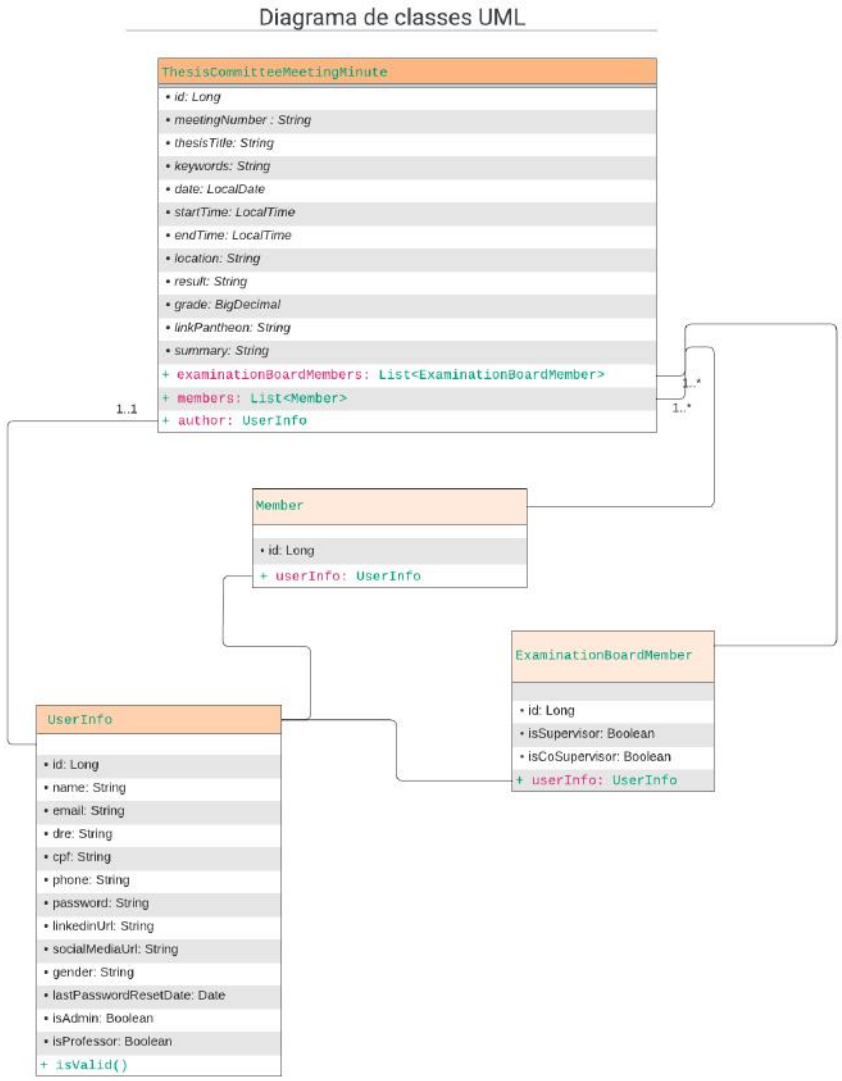
O diagrama de classes UML apresentado na Figura 35, ilustra a estrutura do nosso sistema de gerenciamento de atas de trabalho de conclusão de curso. O diagrama mostra quatro classes principais: ThesisCommitteeMeetingMinute, Member, UserInfo e ExaminationBoardMember, e as relações entre elas.

A classe ThesisCommitteeMeetingMinute é a principal e contém informações detalhadas sobre as defesas de cada trabalho, como o número do trabalho, título do trabalho, palavras-chave, data, hora de início e término, local, resultado, nota, link para o Pantheon, e um resumo. Além desses atributos, esta classe mantém listas de membros da banca (examinationBoardMembers) e membros (members), além de uma referência ao autor da ata (author) no sistema, que é do tipo UserInfo.

A classe UserInfo armazena informações pessoais e de autenticação de um usuário, incluindo campos como nome, email, DRE, CPF, telefone, senha, links de redes sociais, gênero, data da última redefinição de senha, e flags indicando se o usuário é administrador ou professor. Também possui um método isValid(), que verifica a validade das informações do usuário.

A classe Member representa um membro participante e contém um atributo userInfo que se refere a uma instância da classe UserInfo. Isso permite associar informações deta-

Figura 35 – Diagrama de classes UML



lhadas de um usuário específico a um membro.

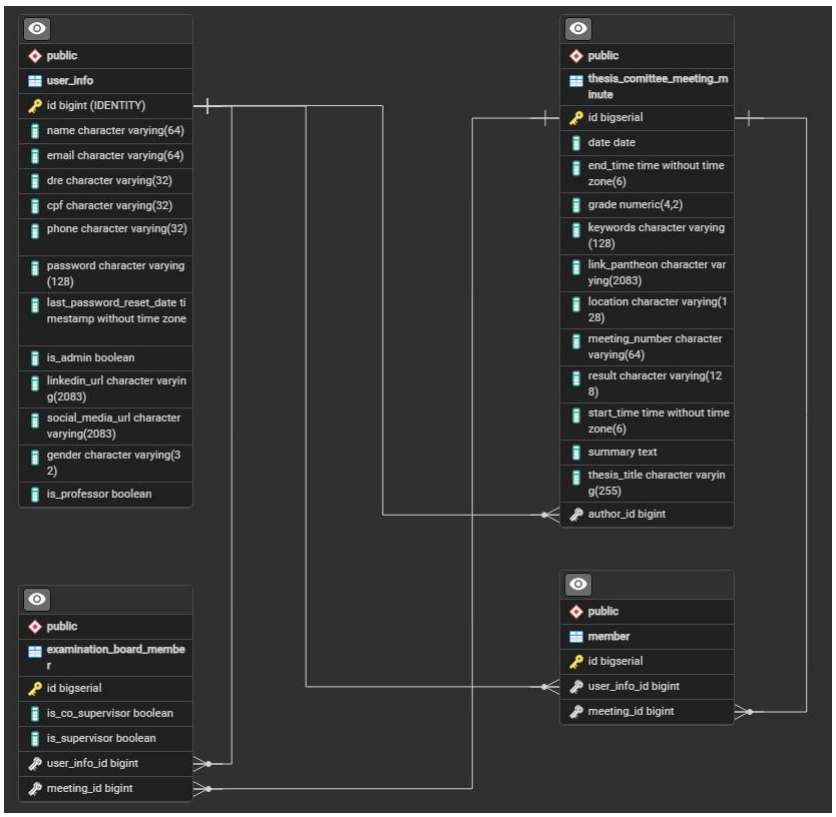
A classe ExaminationBoardMember representa os membros da banca examinadora, com atributos que indicam se o membro é supervisor ou co-supervisor. Semelhante à classe Member, ExaminationBoardMember também contém uma referência a um UserInfo, associando informações detalhadas do usuário ao membro da banca.

As relações entre as classes são expressas no diagrama por meio de linhas conectando os diferentes elementos. A classe ThesisCommitteeMeetingMinute possui uma relação de composição com ExaminationBoardMember e Member, indicando que uma ata de trabalho pode ter múltiplos membros e membros da banca associada a ela. Ambas as classes Member e ExaminationBoardMember têm uma relação de associação com UserInfo, indicando que cada membro e membro da banca estão associados a informações de um usuário específico.

Na Figura 36 podemos visualizar a estrutura do nosso banco de dados projetado no

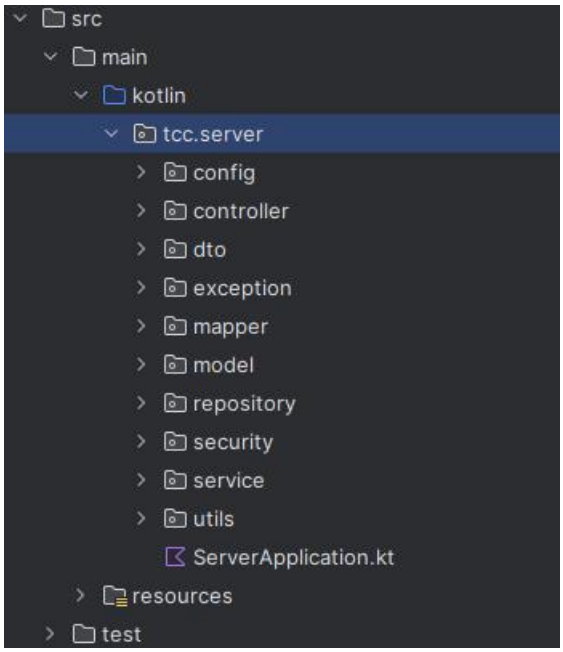
PostgreSQL, que segue o mesmo padrão da Figura 35

Figura 36 – Estrutura do banco de dados



5.4 ESTRUTURA BACK-END

Figura 37 – Back-end



Na Figura 37 temos a estrutura *back-end* do nosso projeto. A seguir, faremos uma breve explicação sobre o significado de cada item.

Configuração (config)

- Este pacote contém classes de configuração do Spring, que são responsáveis por definir o comportamento do sistema. Isso inclui configurações de segurança, configurações de banco de dados, e outras configurações específicas da aplicação.

Controladores (controller)

- As classes dentro deste pacote são responsáveis por lidar com as requisições HTTP recebidas. Cada controlador mapeia endpoints específicos e define como as solicitações devem ser processadas, interagindo com os serviços para obter ou manipular dados.

Objetos de Transferência de Dados (dto)

- O pacote dto contém classes que são usadas para transferir dados entre as camadas do sistema. DTOs são usados para encapsular dados e fornecer uma forma segura de transferir informações entre o cliente e o servidor.

Exceções (exception)

- Este pacote contém classes relacionadas ao tratamento de exceções. Aqui são definidos exceções personalizadas e como elas devem ser tratadas quando ocorrem, garantindo que o sistema possa lidar com erros.

Mapeadores (mapper)

- As classes dentro deste pacote são responsáveis por converter entidades do modelo em DTOs e vice-versa. Mapeadores ajudam a separar a lógica de negócios dos detalhes de implementação da transferência de dados.

Modelos (model)

- Este pacote contém as classes de modelo que representam as entidades do banco de dados. Cada modelo é uma representação direta de uma tabela no banco de dados, e contém anotações do JPA para mapeamento ORM (Object-Relational Mapping).

Repositórios (repository)

- O pacote repository contém interfaces que estendem JpaRepository ou CrudRepository do Spring Data. Essas interfaces são responsáveis por fornecer métodos para realizar operações CRUD (Create, Read, Update, Delete) nas entidades do modelo.

Segurança (security)

- As classes dentro deste pacote são responsáveis pela configuração e gestão da segurança da aplicação. Isso inclui a configuração de autenticação e autorização, geralmente utilizando Spring Security, e a implementação de JWT para autenticação baseada em *tokens*.

Serviços (service)

- O pacote service contém a lógica de negócios da aplicação. As classes de serviço interagem com os repositórios para realizar operações de negócios e são chamadas pelos controladores para processar as requisições dos usuários.

Utilitários (utils)

- Este pacote contém classes utilitárias que fornecem funcionalidades auxiliares que podem ser utilizadas em diversas partes do sistema. Isso inclui funções de formatação, validação, ou outras operações comuns.

Classe Principal (ServerApplication.kt)

- Esta é a classe principal do sistema Spring Boot. Ela contém o método main que inicializa a aplicação, configurando todos os componentes definidos nos pacotes acima.

A estrutura do *back-end* do nosso projeto seguiu uma organização modular clara, o que facilitou a manutenção e escalabilidade da aplicação. Cada pacote teve uma responsabilidade específica, seguindo os princípios de separação de preocupações e facilitando o desenvolvimento colaborativo. Esta organização não apenas melhorou a legibilidade do código, mas também assegurou que cada componente do sistema pudesse ser desenvolvido, testado e mantido de forma independente.

5.5 ESTRUTURA FRONT-END

Na Figura 38 temos a estrutura *front-end* do nosso projeto. A seguir, faremos uma breve explicação sobre o significado de cada item.

Diretório "app"

Arquivos:

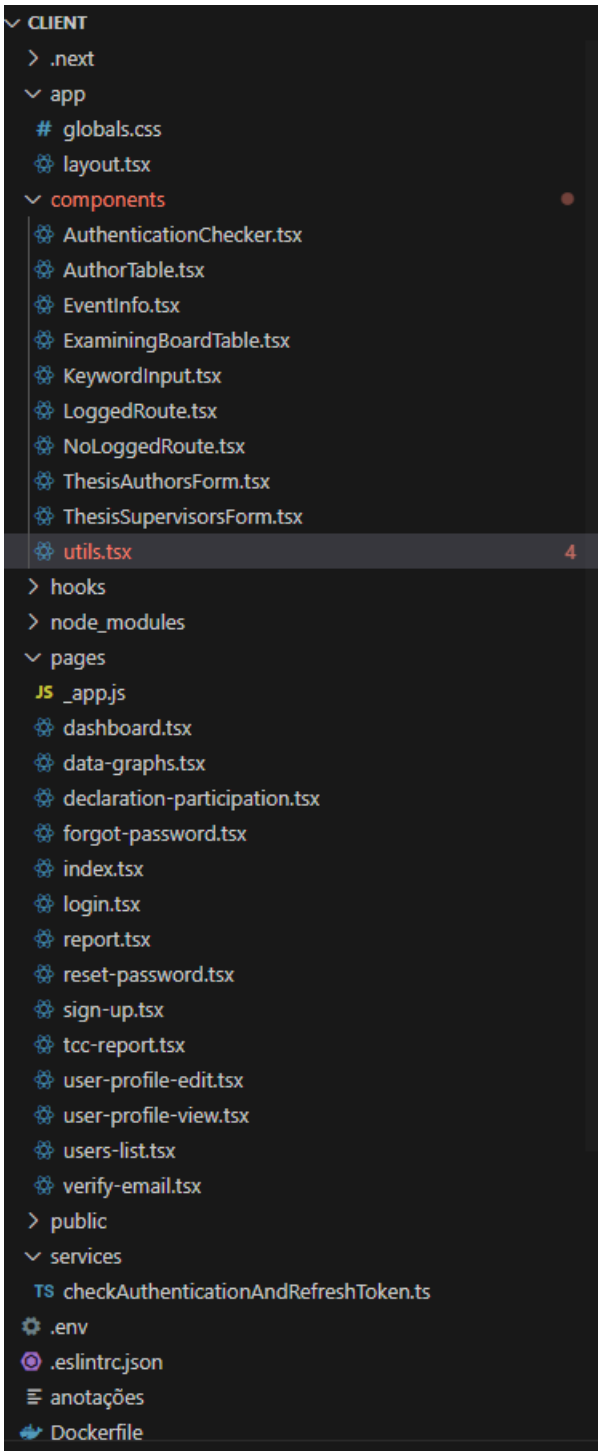
- globals.css: Arquivo de estilos globais aplicado a todo o sistema.
- layout.tsx: Componente de layout que define a estrutura básica das páginas do sistema, incluindo elementos comuns como cabeçalhos e rodapés.

Diretório "components"

Contém componentes reutilizáveis que podem ser utilizados em várias páginas do sistema.

Componentes:

Figura 38 – Front-end



- AuthenticationChecker.tsx: Componente para verificar a autenticação do usuário.
- AuthorTable.tsx: Tabela para exibir autores.
- EventInfo.tsx: Componente para exibir informações sobre eventos.
- ExaminingBoardTable.tsx: Tabela para exibir membros da banca examinadora.
- KeywordInput.tsx: Componente para entrada de palavras-chave.

- LoggedRoute.tsx: Rota protegida que requer autenticação.
- NoLoggedRoute.tsx: Rota acessível apenas para usuários não autenticados.
- ThesisAuthorsForm.tsx: Formulário de preenchimento de autores da ata de defesa.
- ThesisSupervisorsForm.tsx: Formulário de preenchimento de membros da banca da ata de defesa.
- utils.tsx: Arquivo utilitário para funções auxiliares.

Diretório "hooks"

Contém hooks personalizados que encapsulam lógica de estado e efeitos reutilizáveis. Em nosso contexto, na pasta *hooks* do projeto, encontra-se o arquivo withAuth.tsx. Este arquivo implementa um High-Order Component (HOC) que lida com a autenticação de usuários. A principal função deste HOC é verificar se um usuário está autenticado antes de permitir o acesso a determinados componentes.

Quando um componente é envolvido pelo withAuth, ele ganha uma camada de segurança que verifica a autenticação do usuário. Se o usuário não estiver autenticado, ele será redirecionado automaticamente para a página de login. Caso contrário, o componente original será renderizado normalmente, permitindo o acesso apenas a usuários autenticados.

Essa abordagem garante que certas partes da aplicação, especialmente aquelas que contêm informações sensíveis ou funcionalidades restritas, sejam acessíveis somente por usuários que tenham passado pelo processo de autenticação. Dessa forma, o withAuth contribui significativamente para a segurança e integridade da aplicação, proporcionando uma maneira eficiente de controlar o acesso com base na autenticação do usuário.

Diretório "pages"

Diretório específico do Next.js que contém todos os componentes de página. Cada arquivo aqui representa uma rota no sistema.

Páginas:

- app.js: Componente personalizado do App que inicializa as páginas do Next.js.
- dashboard.tsx: Página do *dashboard* principal.
- data-graphs.tsx: Página para exibição de gráficos de dados.
- declaration-participation.tsx: Página para declarações de participação.
- forgot-password.tsx: Página para recuperação de senha.
- index.tsx: Página inicial.
- login.tsx: Página de login.

- report.tsx: Página para visualização da ata de defesa.
- reset-password.tsx: Página para redefinir senha.
- sign-up.tsx: Página de cadastro.
- tcc-report.tsx: Página para criação e edição da ata de defesa.
- user-profile-edit.tsx: Página de edição do perfil do usuário.
- user-profile-view.tsx: Página de visualização do perfil do usuário.
- users-list.tsx: Página de listagem de usuários.
- verify-email.tsx: Página de verificação de e-mail.

Diretório "public"

Contém arquivos estáticos como imagens, ícones e fontes que podem ser servidos diretamente.

Diretório "services"

Arquivos:

- checkAuthenticationAndRefreshToken.ts: Serviço para verificar e atualizar *tokens* de autenticação.

Arquivos de Configuração

- .env: Arquivo de variáveis de ambiente.
- .eslintrc.json: Configuração do ESLint para padronização e qualidade do código.
- Dockerfile: Arquivo de configuração para a criação de contêineres Docker.

A estrutura modular, com componentes reutilizáveis e páginas bem definidas, segue as melhores práticas de desenvolvimento, facilitando a manutenção e a escalabilidade do projeto.

Separação de Preocupações

A separação entre componentes de UI (*components*), lógica de estado (*hooks*), e serviços (*services*) promoveu a clareza e a modularidade, tornando o código mais fácil de entender e de gerenciar.

A existência de componentes reutilizáveis e *hooks* personalizados mostra um design orientado à reutilização de código, reduzindo a duplicação e melhorando a consistência do sistema.

Gerenciamento de Estado e Autenticação:

O uso de componentes como AuthenticationChecker.tsx e serviços para gerenciamento de *tokens* mostra um cuidado com a segurança e o gerenciamento de autenticação, essencial para sistemas que lidam com dados sensíveis.

Next.js e Chakra UI

O uso de Next.js facilitou a criação de rotas e a renderização do lado do servidor, enquanto o Chakra UI ofereceu uma estilização consistente e um design responsivo, melhorando a experiência do usuário.

A presença de arquivos de configuração como .env e .eslintrc.json mostrou nosso compromisso com boas práticas de desenvolvimento, incluindo a gestão de configurações sensíveis e a padronização de código. O Dockerfile permitiu a containerização, facilitando a implantação em diversos ambientes.

A estrutura do *front-end* do projeto foi organizada e seguiu as melhores práticas de desenvolvimento com React, Next.js e Chakra UI. A modularidade, separação de preocupações e reutilização de código foram bem abordadas, facilitando a manutenção, escalabilidade e desenvolvimento contínuo do sistema.

5.6 TESTES AUTOMATIZADOS

Implementamos uma série de testes automatizados utilizando o *framework* Spring Test. O objetivo desses testes foi garantir a funcionalidade e a integridade do sistema, que gerencia atas de trabalhos conclusão de curso. A seguir vamos ilustrar alguns testes realizados que dizem respeito ao gerenciamento das atas e de seus PDFs.

ThesisCommitteeMeetingControllerTest

- Certificamos a criação correta de atas em diversos cenários, como mostrado na Figura 39.

UploadControllerTest

Como exibido na Figura 40, temos alguns testes que dizem respeito ao PDF

- testUploadPDF(): Verificamos a funcionalidade de *upload* de arquivos PDF.
- testCheckPDF(): Validamos se um arquivo PDF específico estava presente no sistema.
- testDeletePDF(): Testamos a funcionalidade de exclusão de arquivos PDF

Esses testes foram desenvolvidos com o uso do Spring Test, que fornece um ambiente de teste abrangente para aplicações Java baseadas no Spring Framework. A automação dos testes foi crucial para manter a qualidade e a confiabilidade do software, permitindo identificar rapidamente quaisquer falhas durante o desenvolvimento.

Figura 39 – Testes automatizados

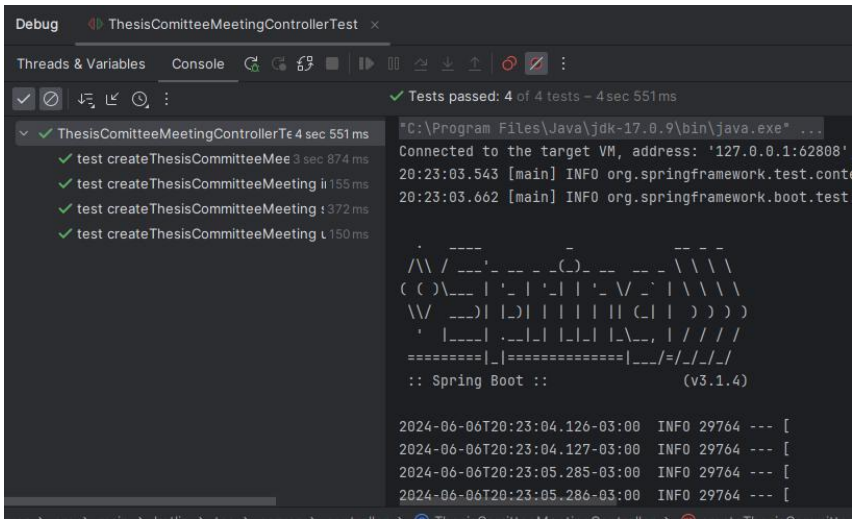
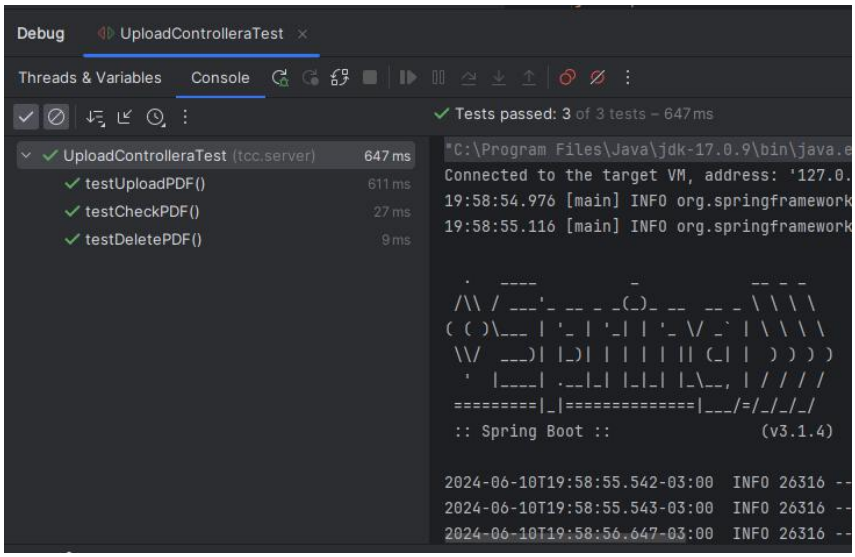


Figura 40 – Testes automatizados



A abordagem automatizada não só aumentou a eficiência, mas também garantiu que todas as funcionalidades críticas fossem validadas consistentemente, contribuindo para a estabilidade do sistema.

6 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo desenvolver um sistema de gerenciamento de ata de defesa de TCC, proporcionando uma solução prática e eficiente para a organização e documentação das defesas de TCC. Ao longo do desenvolvimento deste projeto, abordamos diversos aspectos cruciais para a criação de um sistema robusto e funcional.

6.1 RECAPITULAÇÃO DOS PRINCIPAIS PONTOS ABORDADOS NO TRABALHO

Iniciamos o projeto com uma análise detalhada dos requisitos, identificando as necessidades específicas dos professores e da instituição acadêmica em si. Em seguida, projetamos a arquitetura do sistema utilizando UML (Unified Modeling Language) para garantir uma estrutura clara e coerente. Desenvolvemos o sistema utilizando tecnologias modernas e eficientes, como:

- Frontend: Utilizamos React para a criação de uma interface de usuário interativa e responsiva, Next.js para renderização do lado do servidor e roteamento eficiente, e Chakra UI para a estilização e design de componentes.
- Backend: Implementamos o servidor com o Spring Framework, usando Spring Boot para facilitar a configuração e o desenvolvimento rápido do sistema, Spring Security para autenticação segura com JWT (JSON Web Tokens), e Spring Data para interagir com o banco de dados. As classes do backend foram escritas em Kotlin, aproveitando seus recursos modernos e seguros para aumentar a produtividade e a qualidade do código.
- Banco de Dados: Utilizamos PostgreSQL para armazenamento de dados, oferecendo uma solução robusta e escalável.
- Ferramentas de Desenvolvimento e Implantação: Docker foi utilizado para containerização, garantindo que o ambiente de desenvolvimento fosse consistente em todas as máquinas. GitHub foi usado para controle de versão e integração contínua. Para garantir a qualidade do código, utilizamos Spring Test para testes automatizados e seguimos a abordagem de Test-Driven Development (TDD).

6.2 REFLEXÕES SOBRE AS CONTRIBUIÇÕES DO SISTEMA

O sistema de gerenciamento de ata de defesa de TCC desenvolvido neste trabalho oferece várias contribuições significativas:

- Eficiência e Organização: O sistema facilita a criação, edição e visualização das atas de defesa, centralizando todas as informações em uma plataforma única e acessível.
- Segurança e Autenticidade: Com mecanismos de autenticação robustos, o sistema garante que apenas usuários autorizados possam acessar e modificar as informações sensíveis.
- Facilidade de Uso: A interface amigável e intuitiva melhora a experiência do usuário, tornando o sistema acessível a todos os membros envolvidos no processo de defesa de TCC.
- Transparência e Documentação: A possibilidade de gerar e baixar PDFs das atas de defesa e declarações assegura uma documentação transparente e oficial, essencial para fins administrativos e acadêmicos.

6.3 LIMITAÇÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Apesar das contribuições positivas, o sistema ainda apresenta algumas limitações que podem ser abordadas em trabalhos futuros:

- Integração com Outras Plataformas: A integração com sistemas de gestão acadêmica existentes poderia proporcionar uma experiência ainda mais integrada e eficiente para os usuários.
- Melhorias na Interface de Usuário: Embora a interface atual seja funcional, há sempre espaço para melhorias em termos de design e usabilidade, tornando a interação ainda mais intuitiva.
- Funcionalidades Adicionais: A inclusão de funcionalidades como notificações automáticas, relatórios avançados e suporte a múltiplos idiomas poderia expandir significativamente as capacidades do sistema.
- Testes e Validação: Realizar testes mais extensivos com um grupo maior de usuários finais ajudaria a identificar problemas não previstos e a refinar o sistema para atender melhor às necessidades dos usuários.

O trabalho demonstrou a viabilidade e a importância de um sistema de gerenciamento de ata de defesa de TCC, destacando suas contribuições significativas para a eficiência e organização do processo de defesa de TCC. As limitações identificadas e as sugestões para trabalhos futuros oferecem um caminho para o aprimoramento contínuo deste sistema, visando sempre atender melhor às necessidades da UFRJ e dos usuários.

6.4 APRENDIZADO E DIFICULDADES

Durante o desenvolvimento do nosso sistema para gerenciamento de atas de TCC, passamos por um processo de aprendizado e superação de desafios. Inicialmente, nos deparamos com a necessidade de escolher tecnologias adequadas que atendessem às demandas do projeto, o que nos levou a uma análise das opções disponíveis.

Optamos por utilizar tecnologias modernas tanto no *front-end* quanto no *back-end*, visando garantir uma experiência de usuário fluida e um desempenho robusto do sistema. Durante a implementação, encontramos dificuldades relacionadas à integração das diferentes ferramentas e *frameworks*, especialmente no que diz respeito à comunicação entre o *front-end* e o *back-end.*

A colaboração em equipe foi essencial para o sucesso do trabalho como um todo. A comunicação constante e a divisão clara de tarefas permitiram que cada membro contribuísse de forma significativa. Enfrentamos dificuldades, como a necessidade de sincronizar diferentes ambientes de desenvolvimento e garantir que todos estivessem alinhados quanto às melhores práticas e padrões de código.

Além disso, a utilização de ferramentas de automação e integração contínua foi crucial para manter a qualidade e a consistência do nosso código ao longo do desenvolvimento. Aprendemos a importância de manter um fluxo de trabalho organizado e eficiente, o que nos permitiu iterar rapidamente e corrigir problemas de forma ágil.

REFERÊNCIAS

ABOUT.GITLAB. What is devops? **about.gitlab**, 2024. Disponível em: https://about.gitlab.com/topics/devops/. Acesso em: 30 março.2024.

AGILEALLIANCE. What is extreme programming? **Agilealliance**, 2024. Disponível em: https://www.agilealliance.org/glossary/xp/#:~:text=Extreme%20Programming%20(XP)%20is%20an,engineering%20practices%20for%20software%20development. Acesso em: 20 outubro.2023.

AMOEBOIDS. Extreme programming: What is it exactly? **Amoeboids**, 2024. Disponível em: https://amoeboids.com/blog/extreme-programming/. Acesso em: 20 outubro.2023.

AWS.AMAZON. O que são microsserviços? **aws.amazon**, 2023. Disponível em: https://aws.amazon.com/pt/microservices/. Acesso em: 18 março.2024.

CLOUD.GOOGLE. O que é a computação em nuvem? **cloud.google**, 2023. Disponível em: https://cloud.google.com/learn/what-is-cloud-computing?hl=pt-br. Acesso em: 16 março.2024.

DOCS.GITHUB. Introdução aos contêineres de desenvolvimento. **docs.github**, 2024. Disponível em: https://docs.github.com/pt/codespaces/setting-up-your-project-for-codespaces/adding-a-dev-container-configuration/introduction-to-dev-containers. Acesso em: 26 março.2024.

DOCS.SPRING.IO. Documentation. **docs.spring.io**, 2023. Disponível em: https://docs.spring.io/spring-security/reference/index.html. Acesso em: 13 março.2024.

GURU, R. What's a design pattern? **Refactoring Guru**, 2024. Disponível em: https://refactoring.guru/design-patterns/what-is-pattern. Acesso em: 20 outubro.2023.

IMPERVA. Information security: The ultimate guide. **Imperva**, 2024. Disponível em: https://www.imperva.com/learn/data-security/information-security-infosec/. Acesso em: 22 outubro.2023.

JWT. What is json web token? **JWT.IO**, 2024. Disponível em: https://jwt.io/introduction#:~:text=JSON%20Web%20Token%20(JWT)%20is,because%20it%20is%20digitally%20signed. Acesso em: 21 outubro.2023.

LEARN.MICROSOFT. O que são microsserviços? **learn.microsoft**, 2023. Disponível em: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices. Acesso em: 18 março.2024.

MKYONG. Documentation. **mkyong**, 2023. Disponível em: https://mkyong.com/spring-boot/spring-boot-spring-data-jpa-postgresql/. Acesso em: 19 fevereiro.2024.

NATIVE, R. Documentation. **React Native**, 2024. Disponível em: https://reactnative.dev/docs/getting-started. Acesso em: 16 janeiro.2024.

NEXTJS. Documentation. **Nextjs**, 2024. Disponível em: https://nextjs.org/docs. Acesso em: 16 janeiro.2024.

POWERAPPS.MICROSOFT. Low-code vs. no-code app development. **powerapps.microsoft**, 2024. Disponível em: https://powerapps.microsoft.com/en-us/low-code-no-code-development-platforms/. Acesso em: 30 março.2024.

SPRING.IO. Documentation. **Spring.io**, 2024. Disponível em: https://spring.io/projects/spring-boot. Acesso em: 9 janeiro.2024.

UFLA. Sistema integrado de processos. 2024. Disponível em: http://www.sip.prg.ufla.br/. Acesso em: 30 junho.2024.

UI, C. Documentation. **Chakra UI**, 2024. Disponível em: https://v2.chakra-ui.com/getting-started. Acesso em: 15 janeiro.2024.