UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ENGENHARIA MECÂNICA

ALANA INGRID FERNANDES COSTA DA SILVA

DESENVOLVIMENTO DE APLICAÇÃO WEB PARA SUPORTE AO DESENVOLVIMENTO DE PRODUTOS

Recife
2023

ALANA INGRID FERNANDES COSTA DA SILVA

DESENVOLVIMENTO DE APLICAÇÃO WEB PARA SUPORTE AO DESENVOLVIMENTO DE PRODUTOS

Trabalho de Conclusão de Curso de Graduação em Engenharia de Mecânica do Centro de Tecnologia e Geociências, da Universidade Federal Pernambuco como requisito para a obtenção do título de Engenheira Mecânica.
Orientador: Prof. Francisco Fernando R. Pereira

Recife
2023

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Silva, Alana Ingrid Fernandes Costa da.
Desenvolvimento de aplicação web para suporte ao desenvolvimento de produtos / Alana Ingrid Fernandes Costa da Silva. - Recife, 2023.
61 p. : il., tab.

Orientador(a): Francisco Fernando Roberto Pereira
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Engenharia Mecânica - Bacharelado, 2023.

1. Processo de Desenvolvimento de Produto. 2. Engenharia. 3. Aplicação Web. 4. Produto Mínimo Viável. I. Pereira, Francisco Fernando Roberto. (Orientação). II. Título.

620 CDD (22.ed.)

Universidade Federal de Pernambuco
Departamento de Engenharia Mecânica Centro de Tecnologia e Geociências- CTG/EEP

DEMEC

ATA DE SESSÃO DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO – TCC2

Ao 25.° dia do mês de setembro do ano de dois mil e vinte e três, às 13:30 horas, de forma virtual através da plataforma google meet, reuniu-se a banca examinadora para a sessão pública de defesa do Trabalho de Conclusão de Curso em Engenharia Mecânica da Universidade Federal de Pernambuco, intitulado **Desenvolvimento de Aplicação Web para Suporte ao Desenvolvimento de Produtos**, elaborado pela aluna **Alana Ingrid Fernandes Costa da Silva**, matrícula 20180015310, composta pelos avaliadores Prof. **Francisco Fernando Roberto Pereira** (orientador), Profa. **Janaína Moreira de Meneses** (avaliadora) e Profa. **Marcele Elisa Fontana** (avaliadora). Após a exposição oral do trabalho, a candidata foi arguida pelos componentes da banca que em seguida reuniram-se e deliberaram pela sua aprovação, atribuindo-lhe a média 9,5, julgando-a apta(X) / inapta( ) à conclusão do curso de Engenharia Mecânica. Para constar, redigi a presente ata aprovada por todos os presentes, que vai assinada pelos membros da banca.

Orientador:	Prof. Francisco Fernando Roberto Pereira	Nota: 9,5
Assinatura	gov.br Documento assinado digitalmente FRANCISCO FERNANDO ROBERTO PEREIRA Data: 25/09/2023 14:30:45-0300 Verifique em https://validar.iti.gov.br	
Avaliadora Interna:	Profa. Janaína Moreira de Meneses	Nota: 9,5
Assinatura	gov.br Documento assinado digitalmente JANAINA MOREIRA DE MENESES Data: 26/09/2023 16:05:14-0300 Verifique em https://validar.iti.gov.br	
Avaliadora Interna:	Profa. Marcele Elisa Fontana	Nota: 9,5
Assinatura	gov.br Documento assinado digitalmente MARCELE ELISA FONTANA Data: 25/09/2023 14:35:40-0300 Verifique em https://validar.iti.gov.br	

Recife, 25 de setembro de 2023.

Prof. Marcus Costa de Araújo
Coordenador de Trabalho de Conclusão de curso - TCC
Curso de Graduação em Engenharia Mecânica – CTG/EEP-UFPE

AGRADECIMENTOS

Aos meus familiares, cujo apoio incondicional e compreensão foram pilares fundamentais nos momentos mais desafiadores e que entenderam minha ausência enquanto me dedicava a este trabalho e à minha formação.

Aos amigos que estiveram ao meu lado, demonstrando uma amizade verdadeira e oferecendo apoio constante ao longo de toda a jornada dedicada a este projeto.

Agradeço profundamente ao meu orientador, que conduziu este trabalho com paciência e dedicação, compartilhando generosamente seu conhecimento e expertise.

Aos estimados professores, cujos conselhos, assistência e paciência foram essenciais para orientar e enriquecer meu processo de aprendizado.

Às pessoas com as quais tive a honra de conviver ao longo desses anos de curso, que me inspiraram, incentivaram e, sem dúvida, contribuíram para minha formação acadêmica.

Aos meus colegas de curso, com os quais compartilhei intensamente os desafios e as conquistas dos últimos anos, pelo companheirismo e pela rica troca de experiências que enriqueceram não apenas minha formação acadêmica, mas também minha jornada pessoal.

RESUMO

Este projeto se concentra no desenvolvimento de uma aplicação web que visa oferecer suporte ao Processo de Desenvolvimento de Produtos (PDP), uma atividade fundamental na engenharia que tem evoluído ao longo das últimas décadas e se tornou vital para a competitividade das empresas. O PDP é um processo complexo que envolve diversas etapas, desde a identificação de oportunidades até o lançamento do produto final. Gerenciar eficientemente todas essas fases é crucial para o sucesso das empresas na área de engenharia. O objetivo principal deste projeto é conceber e analisar um Produto Mínimo Viável (MVP) de uma aplicação web que ofereça suporte essencial ao desenvolvimento de produtos durante as etapas críticas do PDP. Esta aplicação é projetada para ser uma ferramenta abrangente que simplifica e melhora a gestão de projetos, promovendo a eficiência, colaboração e inovação. A conclusão do projeto destaca o sucesso na criação da aplicação web, que atende às metas estabelecidas. A integração eficaz do PDP no projeto, a clara definição das etapas e a compreensão aprofundada das necessidades dos usuários resultaram em uma interface intuitiva. Embora o MVP tenha incorporado simplificações devido a limitações de tempo e recursos, ele mantém sua integridade e robustez. A aplicação web contribui significativamente para melhorar a competitividade das empresas na área de engenharia, fornecendo uma base sólida para futuros desenvolvimentos e melhorias. Isso permite que as empresas gerenciem projetos de forma eficiente e desenvolvam produtos de alta qualidade, mantendo-se competitivas no mercado.

Palavras-chave: Processo de Desenvolvimento de Produto; Engenharia; Aplicação Web; Produto Mínimo Viável

ABSTRACT

This project focuses on the development of a web application aimed at providing support for the Product Development Process (PDP), a fundamental activity in engineering that has evolved over the past decades and has become crucial for companies' competitiveness. The PDP is a complex process that encompasses various stages, from identifying opportunities to the final product launch. Efficiently managing all these phases is crucial for the success of companies in the field of engineering. The main objective of this project is to conceive and analyze a Minimum Viable Product (MVP) of a web application that provides essential support for product development during the critical stages of the PDP. This application is designed to be a comprehensive tool that simplifies and enhances project management, promoting efficiency, collaboration, and innovation. The project's conclusion highlights the success in creating the web application, which meets the established goals. The effective integration of the PDP into the project, clear definition of stages, and a deep understanding of user needs have resulted in an intuitive interface. Although the MVP has incorporated simplifications due to time and resource limitations, it maintains its integrity and robustness. The web application significantly contributes to improving companies' competitiveness in the field of engineering, providing a solid foundation for future developments and enhancements. This enables companies to efficiently manage projects and develop high-quality products while remaining competitive in the market.

Keywords: Product Development Process; Engineering; Web Application; Minimum Viable Product

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Interface de Programação de Aplicativos
CSS	Cascading Style Sheets
FMEA	Failure Mode and Effects Analysis
HTML	Hypertext Markup Language
JVM	Máquina Virtual Java
MVP	Produto Mínimo Viável
PDP	Processo de Desenvolvimento de Produto
W3C	World Wide Web Consortium

SUMÁRIO

1 INTRODUÇÃO

Nas últimas décadas, o PDP (Processo de Desenvolvimento de Produto) tem evoluído consideravelmente, em especial no âmbito da engenharia, tornando-se crucial para a competitividade e sobrevivência de empresas e indústrias (Cervo; Bervian; Silva, 2008). Uma metodologia de desenvolvimento de produtos cada vez mais séria e eficaz é necessária para reduzir riscos e intervalos que compõem essa atividade (Montgomery; Porter, 1991).

No contexto da gestão de projetos, a necessidade de desenvolver ferramentas eficazes para agilizar esse processo torna-se evidente. Processos mal definidos e tarefas desorganizadas representam um risco significativo para o sucesso do projeto. Diante desse cenário, há a necessidade premente de criar uma ferramenta capaz de oferecer suporte ao desenvolvimento de produtos, abrangendo todas as etapas do ciclo de vida do projeto.

A engenharia mecânica, que lida com a criação de produtos e exige um processo de desenvolvimento bem definido, ilustra a importância da abordagem de gerenciamento do desenvolvimento de produtos (Ulrich; Eppinger, 2015). Através da aplicação das práticas do PDP, é possível reduzir custos, otimizar o tempo de criação do projeto e aprimorar a qualidade do produto (Cooper, 2011). Portanto, é imperativo o desenvolvimento de um software projetado para apoiar o PDP, essencial tanto para o gerenciamento eficaz de projetos de produtos quanto para a competitividade das empresas no setor de engenharia mecânica.

O sucesso no desenvolvimento de produtos frequentemente depende da integração de equipes multidisciplinares. A colaboração entre profissionais de diferentes áreas proporciona uma visão abrangente do projeto, resultando em um processo mais eficiente e na identificação precoce de problemas, (Johansson, 2019). Essa abordagem também estimula a inovação e a criatividade, reduzindo custos e o tempo necessário para concluir o projeto. A integração entre profissionais de diferentes áreas é essencial para obter produtos de alta qualidade, minimizar falhas e alcançar o sucesso no PDP (Johansson, 2019; Martins; Lambe, 2013).

Assim, o desenvolvimento de um software voltado para as necessidades do PDP atende às demandas da indústria atual, beneficiando as atividades de gerenciamento de projetos e a busca por produtos de alta qualidade e desempenho. A gestão de projetos de desenvolvimento de produtos envolve uma cadeia crítica de informações, e a implementação de um software especializado desempenha um papel fundamental nesse processo (Nadler; Tushman, 1995).

O MVP (Produto Mínimo Viável), elaborado com base nas especificações e requisitos originados dos propósitos delineados, incorpora as funcionalidades essenciais para avaliar os principais alvos da aplicação, a qual se concentra no suporte ao gerenciamento de projetos no âmbito do PDP. Adotando as melhores práticas de programação, essa abordagem culminou em uma aplicação web que atende de maneira abrangente aos objetivos

estabelecidos, oferecendo uma solução eficaz para as demandas identificadas.

1.1 OBJETIVOS

O objetivo deste trabalho foi desenvolver uma aplicação web que forneça suporte ao gerenciamento de projetos de produtos, seguindo as atividades descritas no PDP. Para alcançar esse objetivo geral, serão estabelecidos os seguintes objetivos específicos:

- Descrever e integrar as etapas do PDP na plataforma.
- Levantar os requisitos e necessidades dos usuários para criar uma interface gráfica intuitiva.
- Desenvolver uma base sólida para a aplicação, identificando as entidades do sistema e estabelecendo suas interações e relações para garantir o correto funcionamento do sistema.
- Criar uma interface gráfica atraente e intuitiva para os usuários, considerando a usabilidade, navegabilidade e organização das informações.
- Criar um MVP (Produto Mínimo Viável) eficiente, escalável e de fácil manutenção, selecionando a arquitetura, ferramentas e linguagens de programação adequadas, integrando componentes e utilizando APIs para aprimorar a colaboração e produtividade da equipe de desenvolvimento.
- Verificar o funcionamento correto da plataforma e validar as soluções implementadas.

A realização deste projeto de desenvolvimento de uma aplicação web tem como objetivo principal atender às necessidades práticas e gerenciais das empresas, além de abordar questões sociais relevantes e contribuir para os Objetivos de Desenvolvimento Sustentável (ODS).

Do ponto de vista prático, o projeto visa simplificar e otimizar o processo de gerenciamento de projetos de desenvolvimento de produtos. Isso pode resultar em maior eficiência operacional, redução de custos e um aumento na qualidade dos produtos desenvolvidos. Essa melhoria direta nas operações das empresas é fundamental para sua competitividade no mercado.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 PROCESSO DE DESENVOLVIMENTO DE PRODUTO

2.1.1 Fundamentos do Processo

O PDP (Processo de Desenvolvimento de Produto) é um processo iterativo, que envolve diversas etapas, desde a concepção da ideia até o lançamento do produto no mercado. As principais etapas do PDP são: identificação de oportunidades, geração de ideias, seleção de ideias, desenvolvimento do conceito, projeto, prototipagem, teste e validação, produção e lançamento (Rozenfeld, 2023). Essas etapas podem variar de acordo com a complexidade do produto e as características do mercado.

O PDP é fundamental para a engenharia, pois é por meio dele que as empresas criam, aprimoram e lançam novos produtos no mercado (Mital; Desai; Subramanian, 2014). Através desse processo, as empresas buscam atender às necessidades e expectativas dos consumidores, oferecendo soluções inovadoras e competitivas. Um processo de desenvolvimento de produtos bem executado é crucial para o sucesso do produto no mercado. A falta de um processo estruturado pode levar a atrasos no lançamento, aumento de custos, insatisfação do cliente e perda de mercado para a concorrência. Por isso, as empresas buscam implementar processos eficazes de PDP, que possam garantir a qualidade do produto e a satisfação do cliente (Kotler; Keller, 2019).

2.1.2 Fases do PDP

O PDP é um processo iterativo que envolve diversas etapas, desde a concepção da ideia até o lançamento do produto no mercado. Cada fase do processo é importante para garantir que o produto atenda às necessidades do mercado e dos clientes, e para minimizar riscos e incertezas ao longo do processo.

A Figura 1 abaixo apresenta um modelo simplificado do Processo de Desenvolvimento de Produto:

Figura 1 – Modelo simplificado do Processo de Desenvolvimento de Produto.

Fonte: O autor (2023).

A seguir, são apresentadas as etapas-chave do PDP relacionadas ao modelo simplificado na Figura 1:

- **Identificação de oportunidades e geração de ideias**: Na primeira etapa do PDP, a empresa deve realizar pesquisas de mercado para identificar tendências, necessidades

e oportunidades de inovação. Isso é conhecido como identificação de oportunidades. Em seguida, na fase de geração de ideias, a empresa deve buscar ideias criativas que possam atender às necessidades e expectativas dos clientes. Essa fase pode envolver *brainstorming*, análise de tendências e *benchmarking*.

• **Seleção de ideias**: Após a geração de ideias, a empresa deve selecionar aquelas que têm maior potencial de sucesso. Nesta fase, são realizadas análises de viabilidade técnica, financeira e de mercado para selecionar as ideias mais promissoras.

• **Desenvolvimento do conceito**: Na fase de desenvolvimento do conceito, a empresa deve elaborar uma descrição detalhada do produto, definindo suas características, funcionalidades e benefícios. Nesta fase, é importante considerar as necessidades e expectativas dos clientes e as características do mercado.

• **Projeto**: Na fase de projeto, a empresa deve desenvolver o projeto detalhado do produto, definindo suas especificações técnicas, materiais e processos de fabricação. Nesta fase, a engenharia desempenha um papel fundamental, garantindo que o produto seja viável e atenda às especificações definidas.

• **Prototipagem**: Na fase de prototipagem, são produzidos protótipos do produto para testes e validações. Nesta fase, são realizados testes de funcionalidade, ergonomia, estética e outros aspectos relevantes para a qualidade do produto.

• **Teste e validação**: Na fase de teste e validação, o produto é submetido a testes mais rigorosos para verificar se atende aos requisitos do mercado e do cliente. Nesta fase, são realizados testes de durabilidade, segurança, conformidade com normas e regulamentos, entre outros.

• **Produção**: Após a validação do produto, a fase de produção é iniciada. Nesta etapa, são definidos os processos de fabricação e as especificações técnicas do produto. É importante garantir que a produção seja capaz de atender à demanda prevista e que os custos estejam dentro do orçamento definido.

• **Lançamento**: Finalmente, na fase de lançamento, o produto é introduzido no mercado. É importante que a empresa tenha uma estratégia de lançamento bem definida, que inclua ações de marketing, comunicação e distribuição. É fundamental que a empresa esteja preparada para enfrentar a concorrência e oferecer suporte técnico e pós-venda aos clientes.

2.1.3 Metodologias

Para que o processo seja eficiente, é necessário utilizar metodologias, métodos e ferramentas adequadas. Nesse contexto, as metodologias são conjuntos de etapas que orientam o processo de desenvolvimento, enquanto os métodos e as ferramentas são técnicas específicas que auxiliam em cada etapa do processo (Cooper, 2011).

Existem diversas metodologias de PDP, como a *Stage-Gate®*, a *Design Thinking* e a *agile* (Wheelwright e Clark, 2011). Cada uma dessas metodologias tem suas próprias características e vantagens, mas todas têm em comum o objetivo de garantir a qualidade do produto e a satisfação do cliente.

2.1.3.1 Design Thinking

Design Thinking é uma metodologia que busca resolver problemas complexos de maneira inovadora, criativa e centrada no usuário. Essa abordagem tem origem no design, mas tem sido aplicada em diversas áreas, incluindo engenharia, negócios, educação e saúde (Plattner; Meinel; Leifer, 2013).

O processo de *Design Thinking* é dividido em cinco fases interdependentes: Empatia, Definição, Ideação, Prototipação e Teste/Experimentação (Figura 2). A primeira fase, Empatia, envolve a compreensão profunda dos usuários e suas necessidades, desejos e comportamentos. Nessa fase, os designers devem imergir no universo do usuário e entender suas experiências, desafios e expectativas, de forma a identificar as oportunidades de criação de valor (Kelley; Kelley, 2013; Martin, 2009).

Na fase de Definição, as informações obtidas na fase anterior são analisadas e sintetizadas, e um ponto de vista é definido para guiar o restante do processo. Nessa etapa, os designers devem transformar as informações coletadas em *insights* e definir o desafio a ser resolvido. O objetivo é estabelecer um foco claro e direcionar os esforços criativos (Brown, 2008; Martin, 2009).

Na fase de Ideação, as ideias são geradas livremente e de forma colaborativa, sem julgamento. Nessa etapa, os designers devem criar um ambiente criativo e estimulante, onde todas as ideias são valorizadas e incentivadas. O objetivo é gerar uma grande quantidade de ideias e conceitos inovadores, que possam ser refinados posteriormente (Brown, 2008; Kelley; Kelley, 2013).

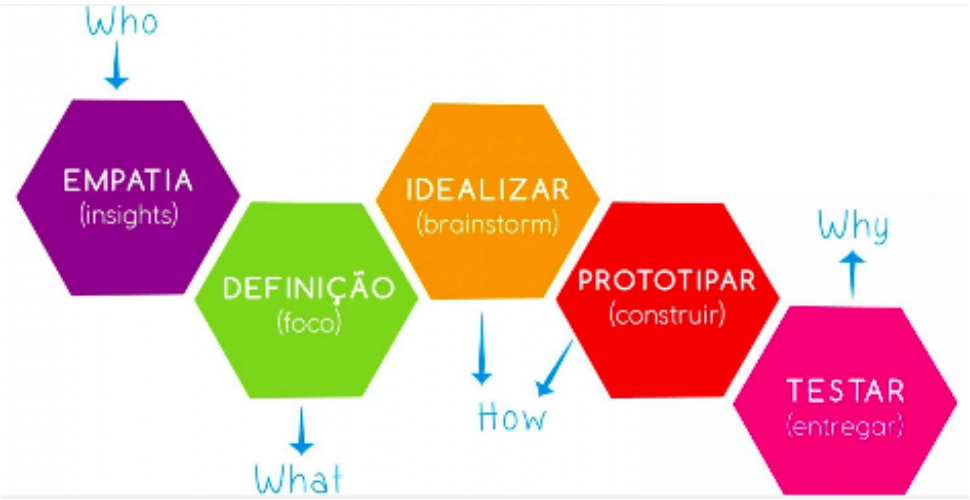
Na fase de Prototipação, os conceitos são transformados em soluções tangíveis e experimentais. Nessa etapa, os designers devem criar modelos e protótipos que possam ser testados e avaliados pelos usuários. O objetivo é transformar as ideias abstratas em soluções concretas, que possam ser refinadas e melhoradas (Plattner; Meinel; Leifer, 2013; Martin, 2009).

Por fim, na fase de Teste/Experimentação, as soluções são testadas e refinadas com base no *feedback* dos usuários. Nessa etapa, os designers devem avaliar o desempenho dos

protótipos e coletar *feedback* dos usuários, de forma a identificar pontos de melhoria. O objetivo é garantir que a solução criada atenda às necessidades dos usuários e resolva o desafio definido na fase de Definição (Brown, 2009; Martin, 2009).

O *Design Thinking* tem como principais vantagens a resolução de problemas de forma criativa e centrada no usuário, a promoção da colaboração e da inovação, além de fornecer um processo estruturado e iterativo para desenvolvimento de soluções. Essa abordagem tem sido utilizada com sucesso em diversos contextos, como no desenvolvimento de produtos, serviços e estratégias de negócios s (Brown, 2009; Plattner; Meinel; Leifer, 2013).

Figura 2 – Fases do processo de *Design Thinking*



Fonte: Adaptado de CDL Bom Despacho (2023).

2.1.3.2 Lean Product Development

O *Lean Product Development* é uma metodologia que tem como objetivo eliminar desperdícios e aumentar a eficiência do processo de desenvolvimento de produtos. Utilizando ferramentas de melhoria contínua, como o *Kaizen* e *Poka-Yoke*, busca simplificar o processo de desenvolvimento, reduzir a complexidade e garantir que somente etapas essenciais sejam realizadas. Além disso, enfatiza a colaboração de toda a equipe de desenvolvimento, incluindo marketing, vendas e produção, para garantir que as necessidades dos clientes sejam atendidas durante todo o processo (Womack; Jones, 2003; Ward; Sobek II, 2016; Liker, 2004). O *Lean Product Development* busca reduzir custos e melhorar a qualidade do produto, tornando o processo mais eficiente. Esta metodologia pode ser aplicada em diversos tipos de empresas e setores, auxiliando no processo de desenvolvimento de novos produtos (Martin, 2009).

Figura 3 – Ciclo lean detalhado.

Fonte: 49 Educação (2023).

2.1.3.3 Metodologia Ágil

A metodologia *agile* é uma abordagem de desenvolvimento de *software* que tem como objetivo entregar *software* de qualidade em um curto espaço de tempo, com a colaboração constante do cliente e a capacidade de responder rapidamente a mudanças no mercado ou nos requisitos do projeto. A metodologia é baseada em quatro valores fundamentais: indivíduos e interações, *software* funcionando, colaboração com o cliente e resposta a mudanças, e em doze princípios que guiam o trabalho da equipe de desenvolvimento (Agile Manifesto, 2001).

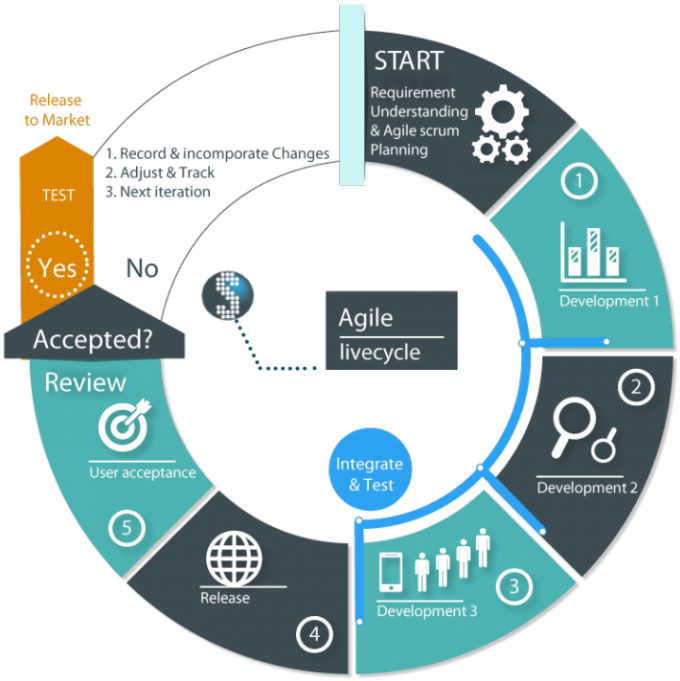
A metodologia *agile* promove uma cultura colaborativa e uma abordagem iterativa e incremental para o desenvolvimento de *software*, com foco na entrega de valor ao cliente em um ritmo constante. Essa abordagem se baseia em ciclos curtos de desenvolvimento, chamados de iterações ou *sprints*, geralmente com duração de duas a quatro semanas. Cada iteração é focada em um conjunto específico de funcionalidades, e a equipe trabalha para implementá-las em um *software* funcionando e testado. O cliente é envolvido ativamente em todo o processo de desenvolvimento, revisando e dando *feedback* constante sobre o produto em desenvolvimento (Larman, 2003).

Essa metodologia enfatiza a importância de ter uma equipe multifuncional, com membros de diferentes áreas, como desenvolvimento, testes, design e negócios, trabalhando juntos em uma colaboração contínua (Schwaber; Sutherland, 2017). A equipe deve ser auto-organizada e ter a capacidade de tomar decisões rapidamente, garantindo a entrega de valor ao cliente em cada iteração.

As metodologias ágeis, como *Scrum*, *Kanban*, XP e *Lean*, têm suas próprias práticas e processos específicos, mas compartilham os mesmos princípios e valores fundamentais da metodologia *agile* (Schwaber; Sutherland, 2017). Cada metodologia tem suas próprias vantagens e desafios, e a escolha da metodologia mais adequada deve levar em consideração as necessidades e características do projeto, da equipe de desenvolvimento e da organização como um todo.

A adoção da metodologia *agile* pode trazer várias vantagens, como maior satisfação do cliente, entrega de *software* funcionando em um curto espaço de tempo, flexibilidade para lidar com mudanças nos requisitos e processos mais eficientes e eficazes de desenvolvimento de *software*.

Figura 4 – Ciclo de desenvolvimento da metodologia *agile* (Ágil).



Fonte: Metodologia Ágil e Scrum (2016).

2.1.4 Ferramentas

O PDP é uma atividade complexa e multidisciplinar que envolve várias etapas. É fundamental utilizar ferramentas adequadas para garantir a eficiência e a qualidade do processo. O CAD é uma ferramenta comumente utilizada no PDP para projetar produtos em um ambiente virtual, o que reduz tempo e custos. A análise FMEA é uma técnica que identifica potenciais falhas do produto, avalia sua gravidade e probabilidade, e toma medidas para evitá-las (Rodrigues; Echeveste, 2008). A análise de valor, análise de risco e análise de custo-benefício são outros métodos que podem ser utilizados no PDP para reduzir custos, gerenciar riscos e avaliar se os benefícios do produto são maiores do que os

custos envolvidos no processo de desenvolvimento. A escolha das ferramentas adequadas para o PDP deve levar em consideração as características do projeto, do mercado e da empresa (Rodrigues; Echeveste, 2008). O uso integrado dessas técnicas pode reduzir riscos, aumentar eficiência e reduzir custos do processo de desenvolvimento de produtos.

2.2 DESENVOLVIMENTO DE APLICAÇÕES WEB

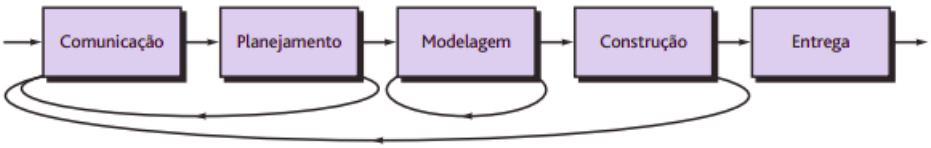
2.2.1 Processos de software

O *software*, como apontado por Pressman (2016), consiste em uma combinação de instruções, representadas por programas de computador, que, quando executados, fornecem as funcionalidades desejadas. Além disso, essa composição engloba estruturas de dados que capacitam os programas a manipular informações de forma apropriada, bem como informações descritivas, tanto em formato impresso quanto virtual, que explicam o funcionamento e a utilização dos programas.

No âmbito da engenharia de *software*, os profissionais enfrentam desafios que se enquadram em sete categorias distintas. O *software* de sistema, por exemplo, é constituído por programas que processam informações complexas, atendendo a outros *software*s. Em contrapartida, o *software* de aplicação compreende programas independentes, destinados a resolver necessidades específicas de negócios. A categoria de *software* de engenharia/-científico concentra-se em programas voltados para cálculos de grande escala em áreas como astronomia e biologia molecular. Por sua vez, o *software* embarcado é direcionado para produtos ou sistemas que controlam funções específicas. Já o *software* para linha de produtos é projetado para atender a diferentes clientes em mercados específicos. As aplicações *web* e aplicativos móveis estão orientadas para redes e dispositivos móveis. Por fim, o *software* de inteligência artificial emprega algoritmos não numéricos para resolver problemas complexos, como reconhecimento de padrões e sistemas especialistas (Pressman, 2016).

O desenvolvimento de um *software* é um processo ágil e adaptativo que busca garantir a entrega de um produto de alta qualidade, alinhado com os requisitos definidos no início do projeto, seguindo uma abordagem típica da engenharia de *software*. Esse processo abrange várias etapas, começando pela compreensão do problema por meio da análise e definição de requisitos, onde são identificadas as necessidades que motivaram a criação do produto. Em seguida, ocorre o planejamento da solução, com a modelagem do sistema desejado e a definição das especificações para o *software*. Na sequência, a execução do plano se dá por meio da geração do código necessário para criar o MVP (Produto Mínimo Viável). Finalmente, a validação do resultado ocorre com a realização de testes para assegurar a qualidade do produto. Essa abordagem se revela iterativa e essencial no contexto do desenvolvimento de *software* (Pressman, 2016).

Figura 5 – Fluxo de processo.



Fonte: Adaptado de Pressman (2016).

2.2.2 Design e Arquitetura de Software

Segundo Martin (2017), o objetivo da arquitetura de *software* é minimizar os recursos humanos necessários para construir e manter o sistema necessário. A arquitetura de *software* é um campo de estudo da engenharia de *software* que trata da organização estrutural de sistemas de *software*. Ela é composta por um conjunto de decisões estratégicas que definem a estrutura, comportamento e propriedades do sistema. Segundo Shaw e Garlan (1996), é "a organização fundamental de um sistema, incorporando seus componentes, suas relações com o ambiente e os princípios que governam seu design e evolução".

A arquitetura de *software* é importante pois fornece um modelo abstrato do sistema, que permite aos desenvolvedores entender como os diferentes componentes do sistema se relacionam e interagem entre si. Isso facilita o processo de desenvolvimento, uma vez que os desenvolvedores podem trabalhar em componentes específicos do sistema sem ter que entender todo o sistema como um todo.

Além disso, uma boa arquitetura pode facilitar a manutenção e evolução do sistema, uma vez que ela permite a adição e remoção de componentes sem afetar outros componentes do sistema. Isso pode reduzir o tempo e o custo necessários para a implementação de novas funcionalidades ou correção de *bugs*. Em termos simples, a arquitetura de *software* trata da organização dos componentes do sistema, como eles são estruturados e interagem entre si. Essa representação possibilita analisar a efetividade da aplicação, verificar se atende aos requisitos e considerar possíveis mudanças para reduzir riscos. Vale ressaltar que a arquitetura não é um *software* operacional, mas enfatiza o papel dos componentes de *software*, que podem ser módulos do programa ou classes no código (Pressman, 2016).

Uma arquitetura bem projetada pode trazer vários benefícios, tais como:

- Escalabilidade: Uma boa arquitetura de *software* pode permitir que a aplicação seja escalável, ou seja, que possa crescer e lidar com um grande volume de usuários e de dados;
- Manutenibilidade: Pode facilitar a manutenção da aplicação, tornando mais fácil adicionar novas funcionalidades, corrigir erros e realizar atualizações;
- Reusabilidade: Pode promover a reutilização de código, permitindo que componentes de *software* possam ser reaproveitados em diferentes partes da aplicação ou em outras aplicações;

- Performance: Pode influenciar o desempenho da aplicação, permitindo que as páginas sejam carregadas mais rapidamente e que a aplicação seja mais responsiva;
- Segurança: Contribuir para a segurança da aplicação, permitindo que sejam implementados mecanismos de autenticação, autorização e proteção contra ataques;

Existem vários padrões de arquitetura de *software*, que são modelos comuns que descrevem a estrutura e o comportamento de um sistema. Esses padrões ajudam os desenvolvedores de *software* a selecionar as melhores soluções para seus sistemas, permitindo a criação de soluções que são eficazes. Dentro de uma arquitetura de *software*, podem ser encontrados diversos padrões que lidam com questões importantes, como concorrência, persistência e distribuição (Pressman, 2016).

As aplicações *web* seguem uma arquitetura cliente-servidor, geralmente estruturadas em várias camadas. Essa estrutura inclui uma camada de visualização ou interface do usuário, uma camada controladora (que direciona o fluxo de dados de acordo com as regras de negócio definidas) e uma camada de conteúdo ou modelo, que também pode conter as regras de negócio. A camada de dados fica no servidor, e as regras de negócio podem ser implementadas usando uma linguagem de programação. O projeto arquitetural de uma aplicação *web* também é influenciado pela estrutura do conteúdo que precisa ser acessado pelo cliente, podendo ser linear ou não linear (Pressman, 2016).

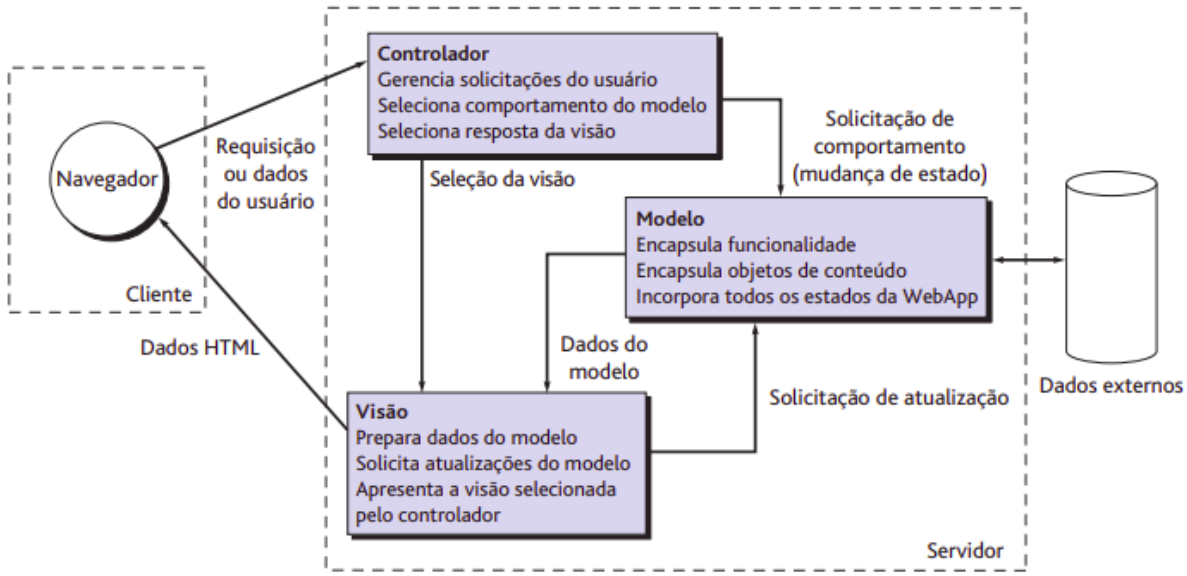
Entre os padrões de arquitetura mais comuns, como o Modelo-Visão-Controlador (MVC), o Modelo-Visão-Presenter (MVP) e o Modelo-Visão-ViewModel (MVVM), o padrão MVC é amplamente utilizado em aplicações *web*. Ele é composto por três componentes principais que desempenham papéis específicos na estrutura da aplicação.

O Modelo representa os dados e as regras de negócio, enquanto a Visão é responsável pela apresentação dos dados e interação com o usuário. O terceiro componente, o Controlador, gerencia as ações do usuário e a comunicação entre a Visão e o Modelo.

Em aplicações *web*, é comum a organização em camadas para separar diferentes funcionalidades. Uma abordagem recomendada é a arquitetura de três camadas, que mantém a interface de navegação e o comportamento da aplicação independentes. Essa separação simplifica a implementação e promove a reutilização de código (Pressman, 2016).

No contexto de *WebApps*, o Modelo-Visão-Controlador (MVC) é um modelo de infraestrutura sugerido. Sua proposta fundamental é separar a interface do usuário da funcionalidade e do conteúdo de informações da aplicação *web*. Essa divisão é crucial para melhorar a organização e a manutenção do *software* (Pressman, 2016).

Figura 6 – Arquitetura MVC.



Fonte: Pressman (2016).

2.2.3 Projeto de aplicação web

O projeto de um *software web* envolve atividades técnicas e não técnicas, como definir a aparência e a compreensão da aplicação, criar a interface gráfica e estabelecer a arquitetura geral. Além disso, o desenvolvimento das funções, o planejamento da navegação e a realização de testes são cruciais para garantir a qualidade do *software*, de acordo com as percepções dos usuários.

O projeto de *WebApps* é composto por seis etapas principais, guiadas pelas informações obtidas durante a modelagem de requisitos. O projeto de conteúdo se baseia no modelo de conteúdo desenvolvido durante a análise. O projeto estético define o layout visível para o usuário. Já o projeto da arquitetura foca na estrutura geral de hipermídia de todos os objetos de conteúdo e funções. A interface é projetada para estabelecer mecanismos de layout e interação com o usuário, enquanto o projeto de navegação determina como o usuário percorrerá a estrutura de hipermídia. Por fim, o projeto de componentes aborda a estrutura interna detalhada dos elementos funcionais da *WebApp* (Pressman, 2016).

A qualidade de uma aplicação *web* é definida considerando aspectos como funcionalidade, usabilidade, confiabilidade, eficiência, manutenibilidade, segurança, escalabilidade e desempenho. Para alcançar esses atributos, um bom *software web* deve apresentar as seguintes características: simplicidade, consistência, identidade, robustez, facilidade de navegação e apelo visual (Pressman, 2016).

2.2.3.1 Recursos Tecnológicos

Os recursos tecnológicos são fundamentais para o desenvolvimento do projeto, compreendendo as linguagens de programação empregadas, os *frameworks* que proporcionam abstrações e facilitam o desenvolvimento, bem como as ferramentas como IDE e de testes utilizadas para garantir a qualidade do *software*. O *framework* é a base dos padrões utilizados no projeto, elevando a abstração dos componentes e facilitando a implementação de funcionalidades complexas (Pressman, 2016). No contexto das linguagens e *frameworks*, podemos dividir entre *back-end*, responsável pela lógica do sistema, e *front-end*, que engloba a interface com o usuário. Esses recursos desempenham um papel crucial para alcançar os objetivos do projeto e garantir um produto de alta qualidade.

2.2.3.1.1 *Front-end*

No desenvolvimento de aplicações *web*, HTML e CSS são duas das tecnologias mais fundamentais. HTML (Hypertext Markup Language) é a linguagem de marcação usada para estruturar e exibir conteúdo em uma página *web*. Já o CSS (Cascading Style Sheets) é uma linguagem de folha de estilo que define a apresentação visual de uma página *web*.

O HTML é uma linguagem de marcação de hipertexto que permite definir e estruturar o conteúdo de uma página *web* em diferentes tipos de elementos, como cabeçalhos, parágrafos, imagens, links, tabelas e formulários. O HTML é uma linguagem padronizada pela W3C (World Wide Web Consortium) e está em constante evolução, sendo a versão mais recente a HTML5 (Castro; Hyslop, 2014).

O CSS é uma linguagem de folha de estilo que permite controlar a apresentação visual de uma página *web*, definindo propriedades como cores, fontes, espaçamento, bordas e posicionamento dos elementos HTML. Ele também é padronizado pela W3C e a versão mais recente é o CSS3 (Castro; Hyslop, 2014).

Em linguagens de programação, o JavaScript é uma linguagem fundamental no contexto de aplicações *web*, desempenhando um papel essencial na criação de interatividade e dinamismo nas páginas. Ele é amplamente utilizado para realizar ações no lado do cliente, como validação de formulários, manipulação de elementos da página, animações, requisições assíncronas a servidores, dentre outros. No desenvolvimento de aplicações *web* modernas, o JavaScript é frequentemente combinado com tecnologias como HTML e CSS para criar interfaces de usuário atraentes e responsivas. Além disso, *frameworks* e bibliotecas JavaScript, como React, Angular e Vue.js, são amplamente utilizados para simplificar e agilizar o desenvolvimento de aplicações complexas.

No contexto do desenvolvimento *web*, os *frameworks* são ferramentas que desempenham um papel fundamental ao facilitar a criação de aplicações, fornecendo um conjunto de bibliotecas e recursos pré-definidos. Eles agilizam a construção de interfaces de usuário, a comunicação com servidores e a manipulação de dados.

No entanto, é crucial selecionar o *framework* mais adequado para cada projeto, considerando suas necessidades específicas. Na atualidade, diversos *frameworks* são amplamente utilizados no desenvolvimento *front-end*, proporcionando soluções eficazes e produtivas. Alguns dos mais populares são o React e o Next.js. Criado pelo Facebook, o React é uma biblioteca *front-end* JavaScript de código aberto que permite a criação de interfaces de usuário declarativas e reativas. Ele utiliza a linguagem JavaScript e é baseado em componentes, permitindo a reutilização de código em diferentes partes da aplicação. O React também possui um sistema de gerenciamento de estados, facilitando a manipulação de dados em tempo real (Banks, 2017). Com relação ao Next.js, ele é um *framework* de desenvolvimento *web* baseado em React que permite a criação de aplicações *web full-stack* de forma eficiente. Ele é projetado para simplificar e acelerar o processo de desenvolvimento, fornecendo recursos avançados e otimizações integradas para criar aplicativos React poderosos e de alto desempenho. Também é responsável por abstrair e configurar ferramentas necessárias para utilizar o React (Vercel, 2023).

2.2.3.1.2 *Back-end*

Uma aplicação *back-end* está relacionada com as partes do programa que engloba sua regra de negócio, seu funcionamento e que não pode ser acessados por um usuário. O código pode ser composto por uma ou mais linguagens de programação e o termo *back-end* é conhecido também como a camada de acesso. Ela suporta os serviços do *front-end*, se conectando com ele e interagindo com os recursos da interface (Techtarget, 2023). Portanto, refere-se à parte integral e não visível de uma aplicação, responsável por abrigar a lógica subjacente e os componentes de processamento de dados. Em outras palavras, é a espinha dorsal da aplicação, onde ocorre a manipulação e o gerenciamento dos dados, a implementação das regras de negócios e a interação com bancos de dados e sistemas externos.

Além disso, no âmbito do desenvolvimento, surge uma definição crucial denominada API (Interface de Programação de Aplicativos). Essa nomenclatura abarca um conjunto meticuloso de regras e protocolos que promovem uma interligação harmoniosa entre sistemas variados, aplicativos ou módulos (Amazon, 2023). A API, nesse contexto, é uma parte específica do *back-end* que expõe *endpoints* (pontos de acesso) para permitir que outras aplicações ou sistemas interajam com a funcionalidade do *back-end* de forma controlada e padronizada. Portanto, uma API é um componente do *back-end* que possibilita a comunicação e a troca de dados entre diferentes sistemas.

No contexto do desenvolvimento de *software*, a escolha das linguagens de programação e *frameworks* para o *back-end* desempenha um papel crucial na eficiência, escalabilidade e funcionalidade geral de uma aplicação. As principais opções disponíveis no cenário atual são o Node.js, Spring Boot e o Quarkus.

O Node.js é um ambiente de tempo de execução de código aberto, construído com base na *engine* V8 do Google Chrome, que permite a execução de código JavaScript fora de um navegador *web*. Diferentemente do JavaScript tradicional, que é principalmente usado para criar interações em páginas *web*, o Node.js possibilita a execução do JavaScript em servidores, tornando-o uma tecnologia essencial para o desenvolvimento de aplicações *back-end* escaláveis e orientadas a evento (Node.js, 2023).

O Spring Boot é um reconhecido *framework* Java de código aberto, projetado para produzir aplicações de qualidade que operam na JVM (Máquina Virtual Java). Ele apresenta autoconfiguração e simplifica o desenvolvimento ágil de aplicações *web* e microsserviços (IBM, 2023).

As abordagens tradicionais do Java foram feitas para aplicativos grandes e lentos em um cenário sem nuvem, *containers* ou Kubernetes. Os *frameworks* Java precisaram se adaptar a essa nova realidade. Com isso, o Quarkus foi desenvolvido para que programadores Java possam criar aplicativos modernos, nativos da nuvem. É um *framework* Java otimizado para Kubernetes, GraalVM e HotSpot, feito com as melhores bibliotecas e padrões Java. O objetivo é fazer do Java a principal plataforma em ambientes Kubernetes e *serverless*, oferecendo aos desenvolvedores uma base para diferentes tipos de aplicativos distribuídos (Quarkus, 2023).

3 METODOLOGIA

3.1 PROPOSTA DO TRABALHO

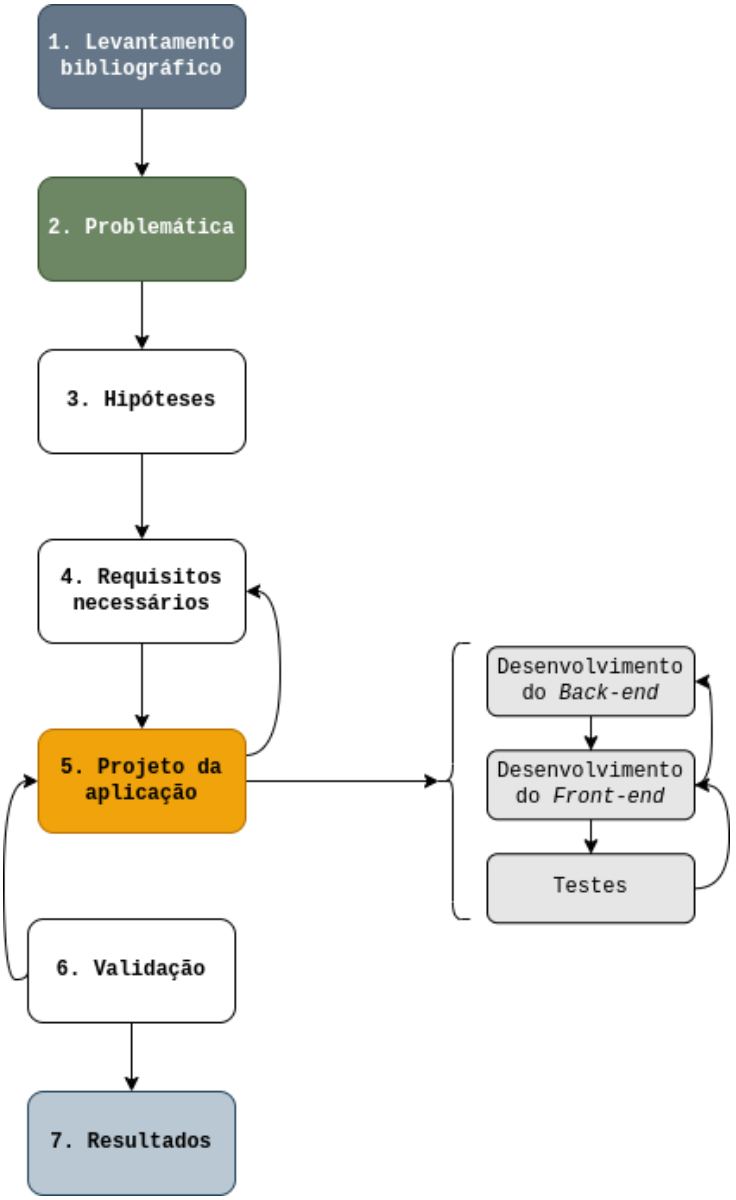
O método científico utilizado será o hipotético-dedutivo. Portanto, o trabalho começará com uma hipótese ou uma suposição sobre como a aplicação *web* pode melhorar o processo de desenvolvimento de produto PDP, que será então testada e validada por meio de análises, experimentos e coleta de dados.

O trabalho seguirá as seguintes etapas ilustradas na Figura 7:

- 1. Realização de um levantamento bibliográfico abrangente sobre o processo de desenvolvimento de produtos, incluindo uma investigação detalhada de suas principais fases, metodologias, métodos e ferramentas utilizados. Além disso, houve o estudo a partir de artigos e livros sobre o desenvolvimento de *software*, com foco especial na aplicação *web*. Isso envolveu a análise das linguagens de programação mais adequadas para o desenvolvimento da aplicação, bem como a investigação da arquitetura de *software* mais apropriada para garantir a eficiência, escalabilidade e manutenibilidade do sistema.
- 2. Identificação dos principais problemas e desafios no processo de desenvolvimento de produtos, realizada por meio de análises de artigos acadêmicos e estudos de casos. Esses desafios fundamentaram a proposta da aplicação web como solução para melhorar a eficiência e segurança no processo de desenvolvimento de produtos.
- 3. Definição precisa dos objetivos do TCC, delineando os resultados esperados e as metas a serem alcançadas ao longo do desenvolvimento do projeto.
- 4. Realização de uma análise detalhada dos requisitos necessários para a aplicação *web*, seguida pela definição das especificações de projeto que nortearam o desenvolvimento do *software*. Essa etapa envolveu a identificação das funcionalidades e características-chave da aplicação.
- 5. Realização do desenvolvimento da aplicação *web*, de acordo com as especificações pré-definidas. Nessa etapa, foram implementadas as funcionalidades, organização do código-fonte tanto no *front-end* quanto no *back-end*, testes e adoção das melhores práticas de programação para assegurar a qualidade e eficiência do *software*.
- 6. Validação da aplicação *web* por meio de testes e avaliações sistemáticas para garantir que ela atenda aos requisitos definidos e às necessidades dos usuários. Essa etapa envolveu a detecção e correção de eventuais erros, bem como a obtenção de *feedback* dos usuários para possíveis melhorias.

- 7. Análise dos resultados obtidos, comparando o processo de desenvolvimento de produto com e sem a utilização da aplicação *web*.

Figura 7 – Fluxograma dos procedimentos utilizados nesse trabalho.



Fonte: O autor (2023).

3.2 DESENVOLVIMENTO DO PROJETO

Com base nos objetivos do projeto, que abrangem a elaboração de um MVP para um aplicativo com base em definições e requisitos pré-estabelecidos, o processo foi iniciado com a identificação dos requisitos utilizando uma metodologia específica de trabalho. Posteriormente, foi conduzida a modelagem do sistema, incluindo a definição das entidades que farão parte da aplicação.

3.2.1 Metodologia de Desenvolvimento da Aplicação Web

Para desenvolver o MVP do produto de forma eficiente e flexível, adotou-se uma abordagem ágil. Foi seguida uma metodologia ágil simplificada, na qual iniciou-se a identificação das principais funcionalidades consideradas essenciais para o MVP. Posteriormente, essas funcionalidades foram priorizadas com base no valor que cada uma agregaria ao produto.

Ao longo do processo, foram empregadas iterações curtas, implementando-se uma funcionalidade de cada vez. Tal abordagem permitiu a concentração na qualidade e na entrega incremental. A cada iteração, revisões foram realizadas para avaliar o progresso e ajustar prioridades, quando necessário.

Ao final do processo, o MVP foi desenvolvido de maneira ágil e eficaz, resultando na entrega de um produto funcional que atendeu às principais necessidades dos usuários. A abordagem ágil simplificada desempenhou um papel fundamental, mantendo o foco nas prioridades e garantindo uma entrega de valor em um projeto desenvolvido de maneira independente.

3.2.2 Análise dos requisitos e especificação

3.2.2.1 Integração do PDP

A integração das etapas do PDP na plataforma é um requisito fundamental para garantir que o *software* atenda aos objetivos propostos. Para tanto, é necessário definir a maneira como cada uma das etapas do PDP será mapeada e incorporada ao fluxo de trabalho da aplicação.

Nesse sentido, o requisito da interface da aplicação é que seja projetada de forma a possibilitar uma visualização clara e acessível das atividades que compõem o PDP. Além disso, uma página informativa é necessária para proporcionar aos usuários uma compreensão abrangente dos propósitos da plataforma, juntamente com definições das etapas do PDP.

Isso assegura não apenas que os usuários tenham acesso fácil às funcionalidades do PDP, mas também compreendam a relevância e sequência das etapas, promovendo assim uma utilização mais eficaz da plataforma e uma execução mais eficiente do PDP.

3.2.2.2 Definição de Requisitos

A definição dos requisitos da aplicação é um passo crucial para criar uma interface gráfica intuitiva e funcional. Isso envolve a identificação das necessidades dos usuários em relação ao uso da plataforma, bem como a determinação das funcionalidades essenciais que devem ser incorporadas ao sistema. Os requisitos podem ser agrupados em duas categorias principais: requisitos funcionais e requisitos não funcionais.

Tabela 1 – Requisitos Funcionais e Não Funcionais

FUNCIONAL	**NÃO-FUNCIONAL**
Autenticação de Usuário Simplificada	Usabilidade e Interface Intuitiva
Acesso a Definições das Fases do PDP	Desempenho e Escalabilidade
Gestão Versátil de Múltiplos Projetos	Segurança e Autenticação
Administração de Fases do PDP por Projeto	Compatibilidade com Navegadores
Criação Personalizada de Atividades	Backup e Recuperação de Dados
Visualização Intuitiva de Progresso	

3.2.2.2.1 Requisitos Funcionais

Os requisitos funcionais são uma categoria de requisitos de *software* que descrevem as funcionalidades e as ações que um sistema ou aplicação deve ser capaz de executar para atender às necessidades dos usuários e atingir seus objetivos.

Os requisitos funcionais definidos para a aplicação englobam diversas funcionalidades essenciais. Primeiramente, a Autenticação de Usuário Simplificada oferece aos usuários a capacidade de acessar a plataforma de forma ágil e conveniente, utilizando suas contas pessoais. Em seguida, o Acesso a Definições das Fases do PDP viabiliza a obtenção de informações detalhadas sobre as definições das fases do PDP, permitindo uma compreensão completa de cada etapa do processo.

Além disso, a Gestão Versátil de Múltiplos Projetos possibilita aos usuários criar e gerenciar vários projetos distintos na plataforma, garantindo flexibilidade para lidar com diferentes cenários. A Administração de Fases do PDP por Projeto facilita a gestão das fases do PDP em cada projeto, permitindo que os usuários organizem e acompanhem o fluxo de desenvolvimento de forma específica.

Adicionalmente, a Criação Personalizada de Atividades capacita os usuários a criar atividades personalizadas para cada fase do PDP, proporcionando uma abordagem adaptada às necessidades de cada projeto. Por fim, a Visualização Intuitiva de Progresso oferece uma interface que possibilita uma análise rápida e informada do estado do projeto, proporcionando uma visualização compreensível do progresso das atividades e etapas.

3.2.2.2.2 Requisitos Não Funcionais

Os requisitos não funcionais são características, propriedades ou restrições que se aplicam a um sistema ou aplicação, mas não dizem respeito diretamente às funcionalidades específicas que o sistema deve realizar. Em vez disso, eles descrevem como o sistema deve operar, performar, se comportar e ser executado em termos de qualidade, confiabilidade, segurança, usabilidade e outros aspectos. Enquanto os requisitos funcionais se concentram no "o que"o sistema faz, os requisitos não funcionais se concentram no "como"ele faz.

Os requisitos não funcionais definidos para o sistema desempenham um papel crítico na garantia da qualidade e desempenho globais do sistema. Primeiramente, a Usabilidade e Interface Intuitiva estabelecem que a interface da aplicação deve ser intuitiva, simplificando a navegação e a execução das tarefas relacionadas ao PDP.

Além disso, os requisitos de Desempenho e Escalabilidade enfatizam a necessidade de a aplicação ser capaz de gerenciar um volume significativo de projetos, fases, etapas e atividades sem prejudicar o desempenho. Isso é fundamental para permitir a análise e validação do MVP de forma eficaz.

Em relação à Segurança e Autenticação, é fundamental incorporar mecanismos de autenticação e autorização para garantir o acesso seguro e individualizado dos usuários à aplicação.

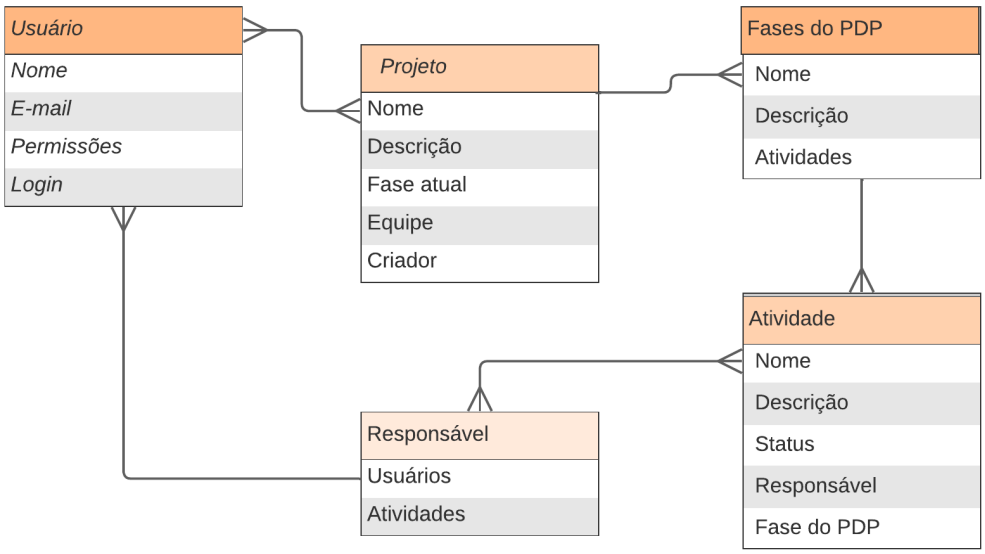
A Compatibilidade com Navegadores estabelece que a aplicação deve ser compatível com os principais navegadores da web, com o objetivo de proporcionar uma experiência consistente para os usuários. Isso é especialmente relevante para validar o MVP de maneira abrangente.

Por fim, o requisito de Backup e Recuperação de Dados destaca a importância de implementar mecanismos regulares e eficientes de backup para assegurar a recuperação de dados, permitindo também a realização de testes do MVP.

3.2.2.3 Definição das Entidades e Relações

A base sólida da aplicação será construída por meio da definição das entidades que comporão o sistema e suas interações. As entidades, como projetos, atividades e fases do PDP, precisam ser identificadas e suas relações estabelecidas para garantir o correto funcionamento da plataforma. Nesse contexto, foram determinadas as seguintes entidades e suas relações:

Figura 8 – Entidades e relacionamentos da aplicação.



Fonte: O autor (2023).

Portanto, o sistema será constituído por cinco entidades-chave: Usuário, Projeto, Fases do PDP, Atividade e Responsável. No diagrama, as relações foram modeladas para representar diversas conexões entre essas entidades, como interações "de muitos para muitos"(várias entidades relacionadas a várias), "de um para muitos"(uma entidade relacionada a várias) e outras dinâmicas de relacionamento. Essas relações são cruciais para capturar as complexas formas como as diferentes entidades se conectam e colaboram dentro do sistema.

3.2.2.4 Desenvolvimento do Front-end

O desenvolvimento do *front-end* é crucial para a experiência dos usuários. O requisito é criar uma interface atraente e intuitiva, facilitando usabilidade, navegação e organização das informações. Por isso, escolheu-se o Next.js, um *framework* React que se destaca para a criação de um MVP. Isso se deve à eficiência no desenvolvimento, desempenho aprimorado (incluindo SSR), integração com APIs, Vercel e documentação completa.

3.2.2.5 Desenvolvimento do Back-end

A etapa de desenvolvimento do *back-end* desempenha um papel fundamental no projeto. Diante das opções disponíveis no mercado em termos de linguagens de programação e *frameworks* para essa finalidade, optou-se pela utilização da linguagem JavaScript,

com foco no ambiente de execução Node.js. Essa escolha foi guiada pela facilidade de integração entre o Vercel e a linguagem JavaScript.

Dado que o *front-end* foi projetado usando o *framework* Next.js, foi decidido incorporar o *back-end* diretamente na própria aplicação Next.js. No Next.js, qualquer arquivo localizado na pasta app/api é mapeado para /api e é tratado como um ponto de extremidade de API em vez de uma página. Esses *bundles* são exclusivamente para o lado do servidor e não afetam o tamanho do *bundle* no lado do cliente (Vercel, 2023). Vale ressaltar que o Vercel é amplamente reconhecido por sua capacidade de facilitar o processo de implantação de aplicativos desenvolvidos com Next.js. Portanto, devido à simplicidade do *back-end*, optou-se por uma integração perfeitamente coesa entre o *back-end* e o *front-end*, simplificando assim o processo de hospedagem de ambos os componentes.

A escolha viabiliza a implementação da aplicação de maneira coesa na plataforma Vercel, permitindo sua hospedagem exclusivamente lá. Essa abordagem integrada e a hospedagem no Vercel foram consideradas ideais para testar o MVP, oferecendo uma solução eficaz para esta fase inicial do projeto.

3.2.2.6 Desenvolvimento do MVP da Aplicação

O desenvolvimento eficiente, escalável e de fácil manutenção do MVP é essencial no projeto. Isso envolve escolher a arquitetura certa, linguagens de programação e ferramentas adequadas. Integrar componentes e utilizar APIs populares para aprimorar a colaboração e produtividade da equipe é crucial para um MVP funcional.

Nesse contexto, como mencionado anteriormente, escolheu-se JavaScript, Node.js e o *framework* Next.js. Quanto à arquitetura, o Next.js segue a abordagem JAMstack, que separa *front-end* e *back-end*, permitindo um desenvolvimento *front-end* eficiente e independente de APIs de *back-end*. A estrutura suporta CSS comum, Scss, Sass pré-compilados, CSS-in-JS e JSX estilizado. Para o desenvolvimento, utilizou-se o Visual Studio Code com extensões específicas para facilitar a programação nas linguagens escolhidas.

4 RESULTADOS

4.1 BANCO DE DADOS

A Vercel oferece uma ampla variedade de produtos de armazenamento gerenciados e sem servidor, todos integrados ao Next.js. Entre essas opções, o Vercel Postgres se destaca como um banco de dados PostgreSQL sem servidor, estrategicamente projetado para se integrar harmoniosamente às Vercel Functions e ao *framework* de *front-end*. Essa escolha se revela sólida e confiável, especialmente quando se lida com dados complexos e transacionais, bem como com formatos de dados diversos e tipos personalizados, como *JSON*, *arrays* ou conteúdo gerado pelo usuário.

Nesse contexto, optou-se pelo PostgreSQL como o banco de dados ideal para o projeto Next.js, decisão que pode ser justificada por várias razões. Uma delas é a natureza das entidades presentes no projeto, que possuem colunas específicas e relacionamentos intrincados. Além disso, os requisitos da aplicação se alinham de maneira notável com as características oferecidas pelo PostgreSQL. Essa seleção estratégica proporciona uma base sólida para o desenvolvimento do MVP.

Para futuras melhorias do projeto, uma adição complementar ao PostgreSQL seria o Vercel Blob. O Vercel Blob oferece um armazenamento otimizado para imagens, vídeos e outros tipos de arquivos, tornando-o uma excelente opção para o gerenciamento de *uploads* de arquivos em cada fase e atividade do processo. Essa combinação de Vercel Postgres e Vercel Blob proporcionaria uma base sólida e escalável para a aplicação, atendendo tanto às necessidades atuais quanto às futuras expansões.

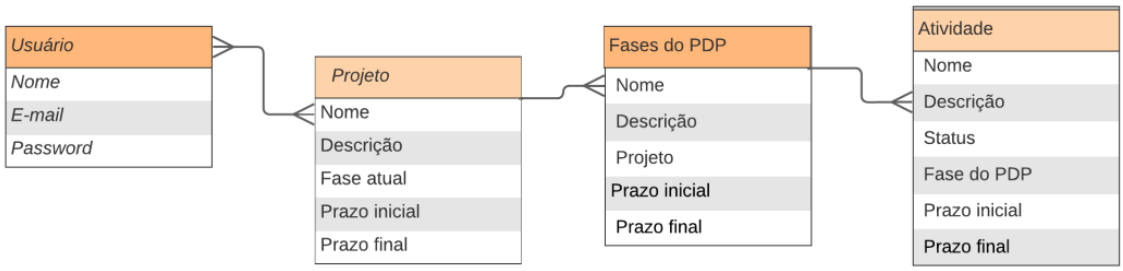
Na etapa de configuração do banco de dados, seguiu-se a documentação oficial da plataforma Vercel. Inicialmente, foi implementado um banco de dados PostgreSQL, para o qual foi instalada a dependência '@vercel/postgres' no projeto. Em seguida, criou-se o banco de dados diretamente em nosso projeto hospedado na plataforma Vercel e o mesmo foi configurado.

Para estabelecer a conexão com o banco de dados, foi utilizado o SDK Vercel Postgres, o qual requer algumas credenciais. Ao vincular o banco de dados ao projeto, essas credenciais foram geradas automaticamente e disponibilizadas como variáveis de ambiente.

Assim que o banco de dados PostgreSQL foi configurado e conectado ao projeto, importou-se essas variáveis de ambiente para o ambiente local de desenvolvimento, permitindo o acesso ao banco de dados.

Com essa infraestrutura estabelecida, procedeu-se à criação das tabelas do banco de dados de acordo com as entidades e relacionamentos definidos no escopo do projeto. No entanto, vale ressaltar que, para simplificar a implementação inicial e permitir testes eficientes do MVP, realizou-se modificações nos relacionamentos e estrutura das tabelas, como visto abaixo.

Figura 9 – Entidades e relacionamentos da aplicação modificado.



Fonte: O autor (2023).

A criação das tabelas foi realizada por meio de uma API desenvolvida internamente dentro do projeto Next.js, aproveitando as funcionalidades do Next.js e SQL. Os detalhes desse processo serão discutidos na próxima seção.

4.2 DESIGN DAS TELAS DO FRONT-END

O desenvolvimento do *front-end* da aplicação se iniciou com a criação de um repositório no GitHub para gerenciar o projeto por meio de versionamento. Seguindo a documentação oficial do Next.js, foi estabelecido, dentro do repositório, uma aplicação Next.js usando a ferramenta npx (*Node Package eXecute*), que é um executor de pacotes NPM. Isso permitiu inicializar o projeto com um *template* predefinido.

Durante esse processo, a linguagem de programação escolhida foi o JavaScript. O Next.js recomenda o uso do App Router para aplicações mais recentes. Com isso em mente, foi adotada essa abordagem, a qual é um sistema de roteamento que se fundamenta na estrutura dos arquivos. Adicionalmente, optou-se pelo uso do ESLint para aprimorar a qualidade do código. A decisão foi tomada de não utilizar o Tailwind CSS, entre outras especificações detalhadas (Figura 10):

Figura 10 – Especificação do projeto inicial.

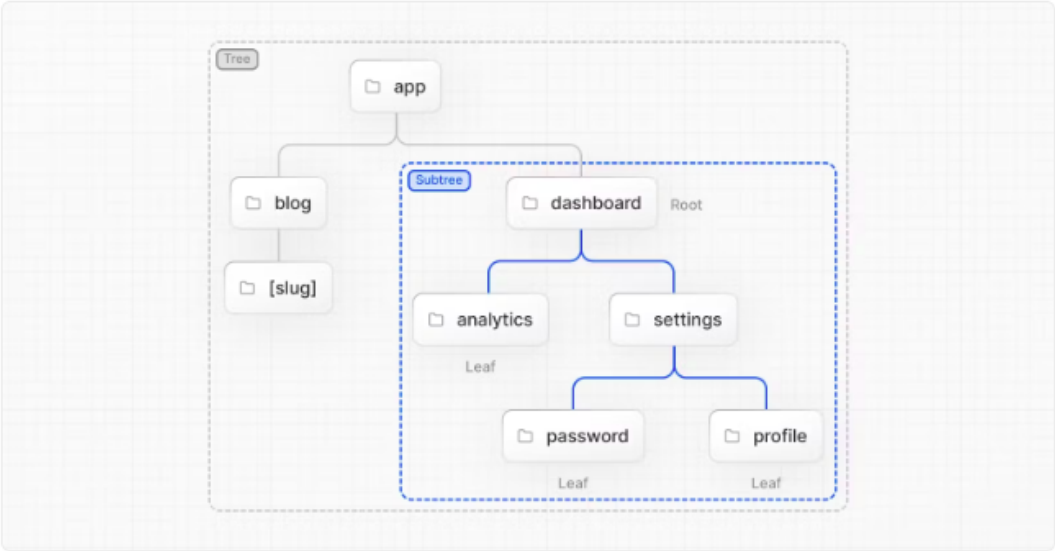
```
What is your project named? meu-tcc
Would you like to use TypeScript? No
Would you like to use ESLint? Yes
Would you like to use Tailwind CSS? No
Would you like to use 'src/' directory? No
Would you like to use App Router? (recommended) Yes
Would you like to customize the default import alias? No
```

Fonte: Vercel (2023).

A base da aplicação é o sistema de roteamento, o qual é composto por uma estrutura de árvore com subárvores, raízes e folhas. Os arquivos que estão dentro da pasta /app são

Componentes do Servidor React. Além disso, uma rota é um trajeto singular de pastas aninhadas, que segue a hierarquia do sistema de arquivos, partindo da pasta raiz até chegar a uma pasta de nível inferior, que por sua vez inclui um arquivo page.js. Além disso, os arquivos são empregados para criar a interface do usuário que é apresentada para um segmento de rota específico (Vercel, 2023).

Figura 11 – Estrutura hierarquica da aplicação.

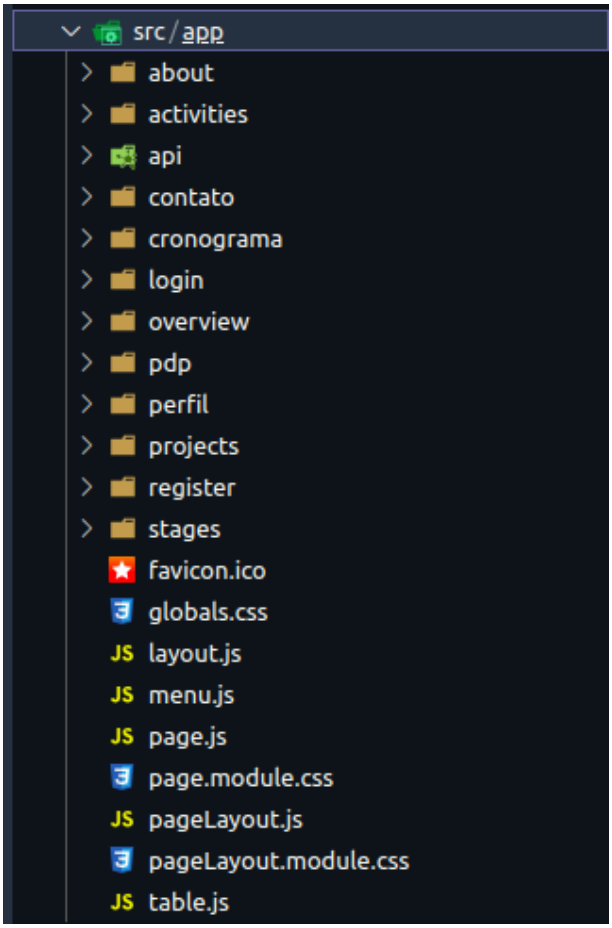


Fonte: Vercel (2023).

Com este objetivo em mente, foram criadas rotas com nomes estratégicos que correspondem às entidades previamente definidas. Cada diretório abaixo é responsável por formar um caminho a partir da raiz (/), sendo que a página inicial em / é representada pelo arquivo page.js localizado na pasta app.

Dentro de cada diretório, que representa um novo caminho na aplicação, há um arquivo page.js. Este arquivo é responsável por definir o código que determina a aparência e o comportamento da interface do usuário associada a esse caminho específico. Em resumo, essa estrutura de diretórios e arquivos permite a organização lógica e modular do *front-end* da aplicação, tornando mais claro onde cada parte da interface do usuário está definida e como ela se relaciona com as entidades e funcionalidades da aplicação (Figura 12).

Figura 12 – Estrutura de pastas da aplicação.

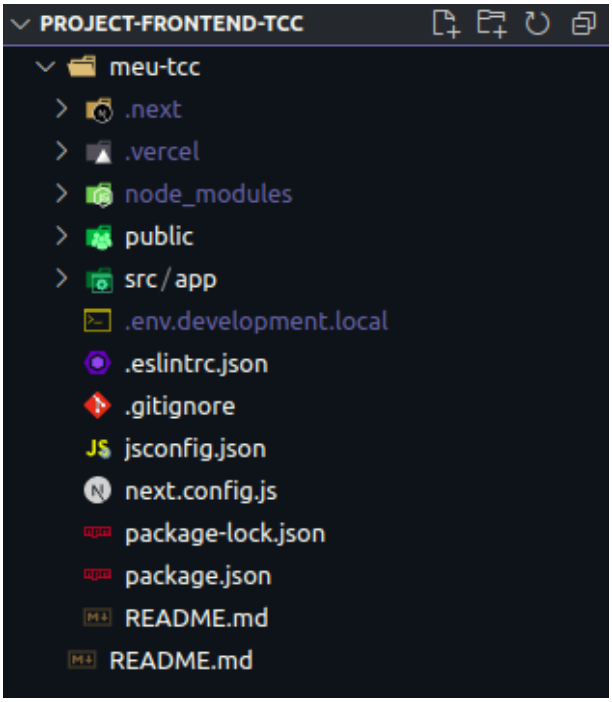


Fonte: O autor (2023).

Dessa forma, desenvolveu-se um componente que serve como um modelo para todas as páginas da aplicação, permitindo que ele seja facilmente incorporado em todos os códigos, funcionando como um *template* reutilizável. Este componente, denominado pageLayout.js (conforme mostrado na Figura 12), está localizado na raiz do projeto e foi importado para outros diretórios com o propósito de ser reutilizado. Ele inclui elementos como a barra de ferramentas (*toolbar*) e outros botões cruciais para a funcionalidade da aplicação. Além disso, outros componentes foram desenvolvidos com a finalidade de torná-los reutilizáveis, como o 'menu.js'. Esse componente tem a responsabilidade de constituir o menu lateral da aplicação, proporcionando uma navegação intuitiva entre os projetos, fases e atividades.

Além disso, a pasta *public*, vista na Figura 13, é utilizada para armazenar recursos públicos e estáticos que serão servidos diretamente para o navegador dos usuários. Isso inclui imagens, arquivos de estilo CSS, JavaScript estático e qualquer outro recurso é necessário ser acessível publicamente pela *web*.

Figura 13 – Estrutura geral das pastas no projeto.



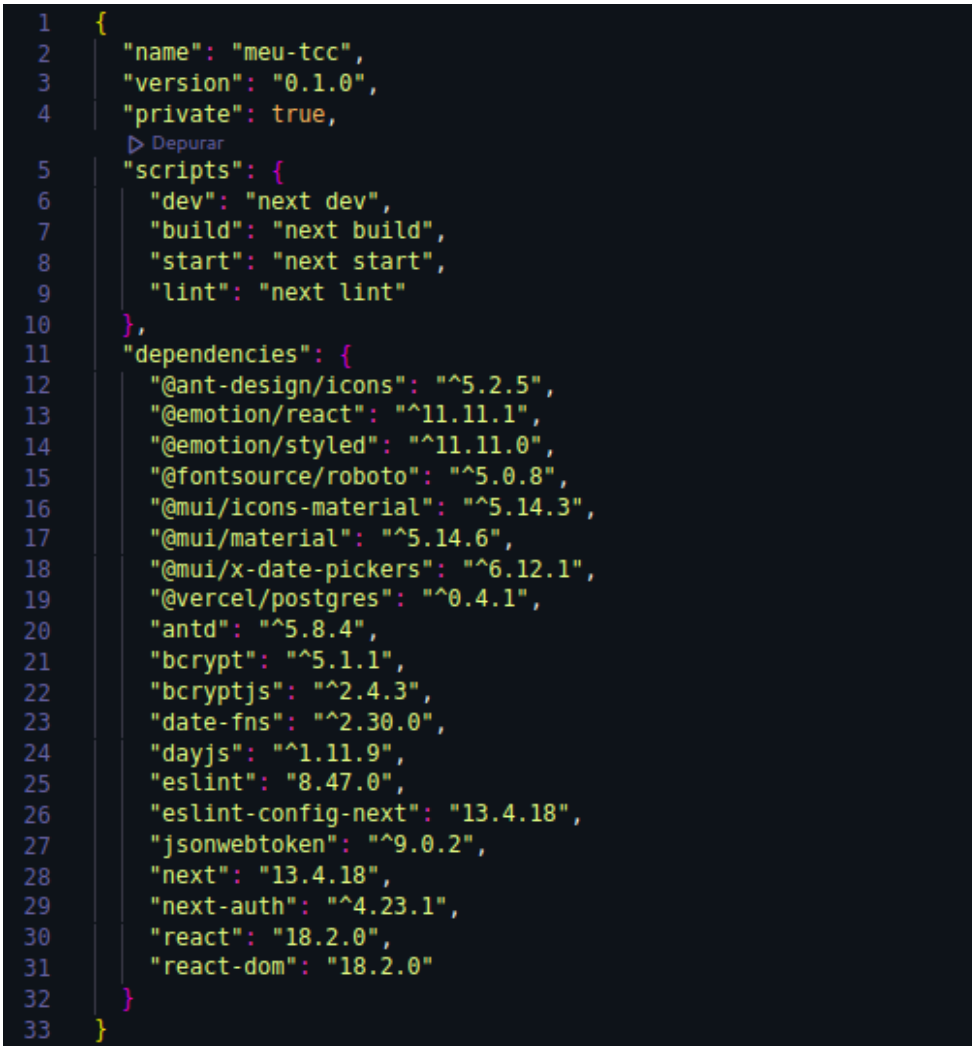
Fonte: O autor (2023).

Para a criação das interfaces de usuário no *front-end*, foram utilizadas duas bibliotecas de design de interface (UI) e componentes para o desenvolvimento de aplicativos *web*: o MUI (*Material-UI*) e o AntD (*Ant Design*). Essas bibliotecas oferecem conjuntos completos de componentes pré-projetados e estilizados que podem ser facilmente integrados em nossos projetos de desenvolvimento *web*, resultando em interfaces de usuário coesas e atraentes. Essa escolha simplificou significativamente o processo de desenvolvimento e agilizou a criação das interfaces, contribuindo para uma experiência mais eficiente.

O arquivo 'package.json', na Figura 14, é um arquivo de manifesto que contém informações e especificações de dependências do projeto. Ele desempenha um papel crucial em qualquer projeto Node.js, sendo responsável por gerenciar as bibliotecas e as configurações essenciais do projeto. No contexto de um projeto Next.js, o arquivo 'package.json' adquire uma importância ainda maior, pois gerencia as dependências específicas do Next.js e outras configurações relacionadas ao projeto.

Para alcançar isso, ele trabalha em conjunto com o sistema de gerenciamento de pacotes do Node.js, normalmente utilizando os comandos 'npm' (*Node Package Manager*) ou 'yarn'. Essas ferramentas permitem a instalação, atualização e gerenciamento das bibliotecas e dependências necessárias para o desenvolvimento do projeto Next.js. Como resultado, estão listadas as dependências específicas presentes neste projeto na Figura 14.

Figura 14 – Dependências definidas no arquivo package.json.



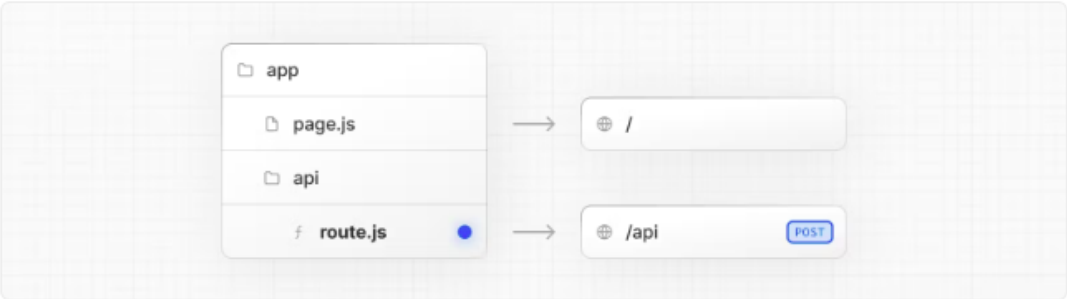
```
{
  "name": "meu-tcc",
  "version": "0.1.0",
  "private": true,
  ▷ Depurar
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "@ant-design/icons": "^5.2.5",
    "@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@fontsource/roboto": "^5.0.8",
    "@mui/icons-material": "^5.14.3",
    "@mui/material": "^5.14.6",
    "@mui/x-date-pickers": "^6.12.1",
    "@vercel/postgres": "^0.4.1",
    "antd": "^5.8.4",
    "bcrypt": "^5.1.1",
    "bcryptjs": "^2.4.3",
    "date-fns": "^2.30.0",
    "dayjs": "^1.11.9",
    "eslint": "8.47.0",
    "eslint-config-next": "13.4.18",
    "jsonwebtoken": "^9.0.2",
    "next": "13.4.18",
    "next-auth": "^4.23.1",
    "react": "18.2.0",
    "react-dom": "18.2.0"
  }
}
```

Fonte: O autor (2023).

4.3 DESENVOLVIMENTO DO BACK-END

O processo de desenvolvimento do *back-end* da aplicação Next.js desempenha um papel fundamental na interação com o banco de dados, criação e gerenciamento de tabelas, bem como na implementação das operações de *CRUD (Create, Read, Update e Delete)*. Para isso, foi criada uma estrutura organizada na pasta /api, que representa o componente de *back-end* do projeto (Figura 17).

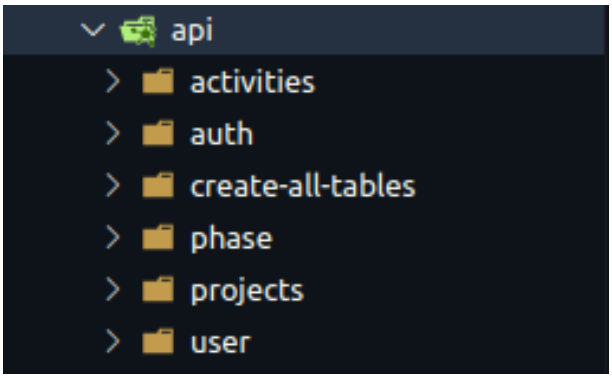
Figura 15 – Manipuladores de Rota.



Fonte: Vercel (2023).

Os Manipuladores de Rota, também conhecidos como *Route Handlers*, viabilizam a criação de manipuladores de solicitações personalizados para rotas específicas por meio do uso das APIs de Solicitação e Resposta da *web*. Consequentemente, nesta pasta, foram desenvolvidos códigos destinados a realizar operações de solicitação, como inserção (*POST*), atualização (*PUT*), entre outras, nas tabelas do banco de dados. Portanto, a estrutura do projeto nesse trecho ficou dessa forma:

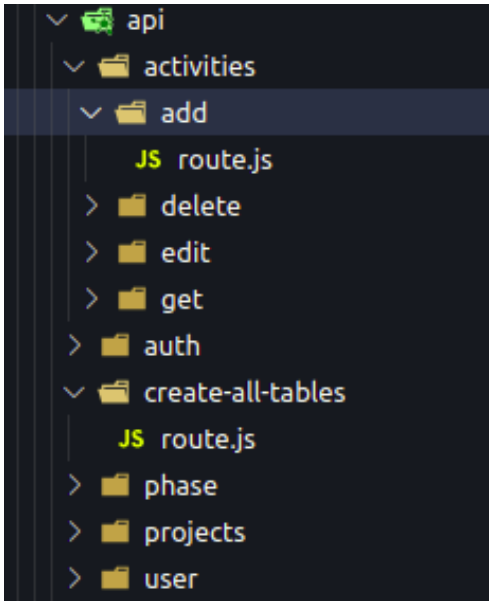
Figura 16 – Estrutura da pasta /api no código fonte do projeto.



Fonte: O autor (2023).

Em cada diretório, foram implementadas operações do CRUD com uma abordagem organizacional clara. Dentro de cada pasta, existe um arquivo chamado route.js, que encapsula a lógica específica para aquele caminho em particular. Um arquivo de rota ou route.js permite criar manipuladores de solicitação personalizados para uma rota específica. Os seguintes métodos HTTP são suportados: *GET, POST, PUT, PATCH, DELETE, HEAD e OPTIONS*. Por exemplo, se houver uma estrutura de diretórios como /api/activities/get, o caminho correspondente será /api/activities/get. Ao acessá-lo com o método apropriado, neste caso o método *GET*, os dados do banco de dados são recuperados e processados conforme definido no arquivo route.js. Isso permite uma estrutura organizada e modular para gerenciar as diferentes operações da API.

Figura 17 – Estrutura da pasta /api no código fonte do projeto.



Fonte: O autor (2023).

A mesma lógica foi aplicada a todas as outras tabelas presentes no banco de dados. Além disso, foi criada uma rota específica, "/api/create-all-tables", com o objetivo de gerar todas as tabelas e seus relacionamentos através de uma solicitação *GET*. O código correspondente está disponível na Figura 18.

Figura 18 – Código para criação de tabelas na API.

```
import { sql } from '@vercel/postgres';
import { NextResponse } from 'next/server';

export async function GET(request) {
  try {
    // Criar a tabela de fases
    await sql`
      CREATE TABLE IF NOT EXISTS Phases (
        id serial PRIMARY KEY,
        name varchar(255),
        description text
        project_id int REFERENCES Projects(id)
      );
    `;

    // Criar a tabela de atividades
    await sql`
      CREATE TABLE IF NOT EXISTS Activities (
        id serial PRIMARY KEY,
        name varchar(255),
        description text,
        status varchar(255),
        phase_id int REFERENCES Phases(id)
      );
    `;

    // Criar a tabela de projetos
    await sql`
      CREATE TABLE IF NOT EXISTS Projects (
        id serial PRIMARY KEY,
        name varchar(255),
        description text,
        current_phase_id int REFERENCES Phases(id)
      );
    `;

    return NextResponse.json({ message: 'Tables created successfully' }, { status: 200 });
  } catch (error) {
    return NextResponse.json({ error }, { status: 500 });
  }
}
```

Fonte: O autor (2023).

Códigos semelhantes aos exemplificados acima foram elaborados para cada entidade e para cada operação do CRUD, fazendo uso da dependência do PostgreSQL do Vercel e da biblioteca com funcionalidades exclusivas para o servidor do Next.js.

4.4 DESENVOLVIMENTO DO MVP DA APLICAÇÃO

O desenvolvimento da aplicação teve início com a criação do projeto a partir de um modelo predefinido fornecido pelo site da Vercel. O código-fonte foi hospedado no GitHub para controle de versão. A estrutura foi estabelecida com sucesso utilizando o App Router, permitindo a criação das telas necessárias para atender aos objetivos do projeto.

Dessa forma, com relação aos requisitos funcionais discutidos anteriormente, foram implementados os seguintes recursos para testar o MVP:

- Simplificação da Autenticação de Usuários.
- Acesso às Configurações das Fases do PDP.
- Gerenciamento Versátil de Múltiplos Projetos.

- Administração das Fases do PDP por Projeto.
- Possibilidade de Criação Personalizada de Atividades.
- Visualização do Progresso.

Quanto aos requisitos não funcionais, a usabilidade e a interface intuitiva receberam atenção, sendo priorizadas na medida do possível. Em relação ao desempenho e escalabilidade, a escolha do local de hospedagem, do banco de dados e do *framework* foi cuidadosamente considerada, com foco na presença de boa documentação e na capacidade de testar a ideia com um MVP. Questões relacionadas ao desempenho e escalabilidade estão sujeitas a discussões futuras para aprimorar o projeto.

No que diz respeito à segurança e autenticação, o Next.js fornece suporte para autenticação, com a distinção entre autenticação (verificando quem é o usuário) e autorização (controlando o que um usuário pode acessar). Embora a autenticação tenha sido simplificada no projeto para atender à ideia inicial de gerenciamento de projetos dentro do PDP, melhorias de segurança e autenticação podem ser consideradas em projetos futuros em produção.

Quanto à compatibilidade com navegadores, é importante notar que a compatibilidade pode variar dependendo da configuração do projeto e dos recursos da *web* utilizados. O Next.js oferece uma base sólida para o desenvolvimento *web* moderno, mas a compatibilidade com navegadores mais antigos pode requerer a implementação de *polyfills* e testes em diferentes navegadores.

Finalmente, em relação ao backup e recuperação de dados, a escolha de modelagem de banco de dados no MVP foi feita visando a simplificação dos testes, mas a implementação de um sistema sólido de backup e recuperação de dados é essencial ao considerar a aplicação em produção.

Com base nisso, idealizou-se as seguintes interfaces para o desenvolvimento do projeto, utilizando a ferramenta Figma, alinhadas com os objetivos estabelecidos anteriormente para a aplicação *web*:

- Tela Inicial (/): Esta tela principal contém botões de navegação para as páginas essenciais da aplicação, incluindo a página inicial (/), a página "Sobre"da aplicação (/about), informações sobre as fases do PDP (/pdp), página de contato (/contato), página de login (/login) e a página de cadastro (/register).
- Tela de Login (/login): A tela de login apresenta dois campos de entrada, um para o *e-mail* e outro para a senha, além de um botão para efetuar o login através da conta do Google (embora essa funcionalidade não esteja disponível no MVP da aplicação).
- Tela "Sobre"(/about): Nesta tela, é fornecida uma descrição detalhada da aplicação e seu objetivo.

- Tela Informativa das Fases do PDP (/pdp): Esta tela exibe as quatro fases principais do PDP, que podem ser adicionadas aos projetos gerenciados na aplicação. Além disso, cada fase é acompanhada por uma descrição resumida.
- Tela de Visão Geral de Projetos (/overview): A tela de visão geral de projetos foi configurada para incluir um menu lateral, que redireciona para os seguintes endereços: /projects/[id], /activities e /stages/[fase]. Isso permite a navegação fácil entre os principais pontos de gerenciamento de projetos. Além disso, esta tela exibe informações cruciais sobre os projetos em andamento.

Com base nisso, deu-se início ao processo de criação das telas, seguindo a concepção previamente estabelecida e as funcionalidades planejadas. O desenvolvimento teve seu ponto de partida na tela inicial (Figura 19), localizada na raiz do projeto, dentro da pasta "app". Para a construção dessa tela e das subsequentes, optou-se por utilizar a fonte "Roboto"do Google, bem como as ferramentas fornecidas pelo React e Next.js. Além disso, foram incorporados componentes predefinidos pelas bibliotecas mencionadas anteriormente, agilizando assim o processo de desenvolvimento.

Figura 19 – Tela Inicial desenvolvida.

Fonte: O autor (2023).

A partir da tela inicial, na Barra de Ferramentas (*Toolbar*), disponibilizou-se botões que direcionam os usuários para as páginas "Sobre", "Fases do PDP"e "Contato", as quais foram desenvolvidas da seguinte maneira (Figura 20, 21 e 22):

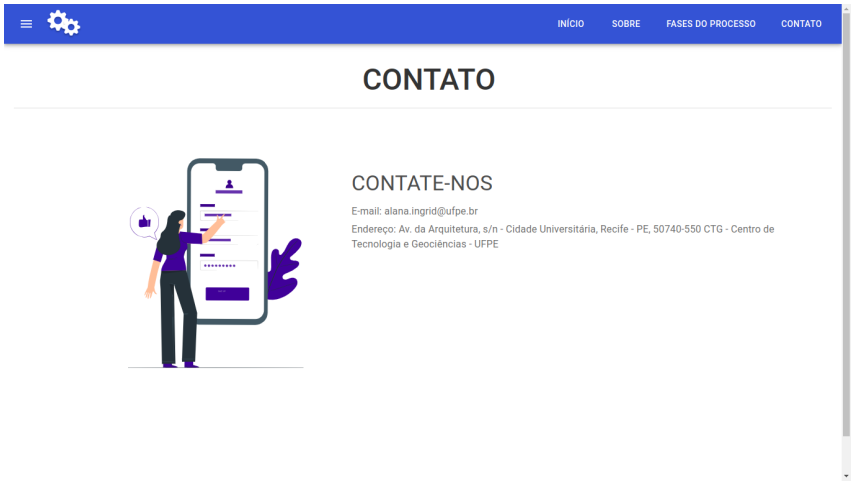
Figura 20 – Tela de Sobre projetada.

Fonte: O autor (2023).

Figura 21 – Tela de Fases do PDP projetada.

Fonte: O autor (2023).

Figura 22 – Tela de Contato projetada.



Fonte: O autor (2023).

A partir da tela inicial, pode ser acessada duas outras telas: "Login"e "Register"(Figuras 23 e 24). Ambas as telas oferecem campos para inserção das informações do usuário.

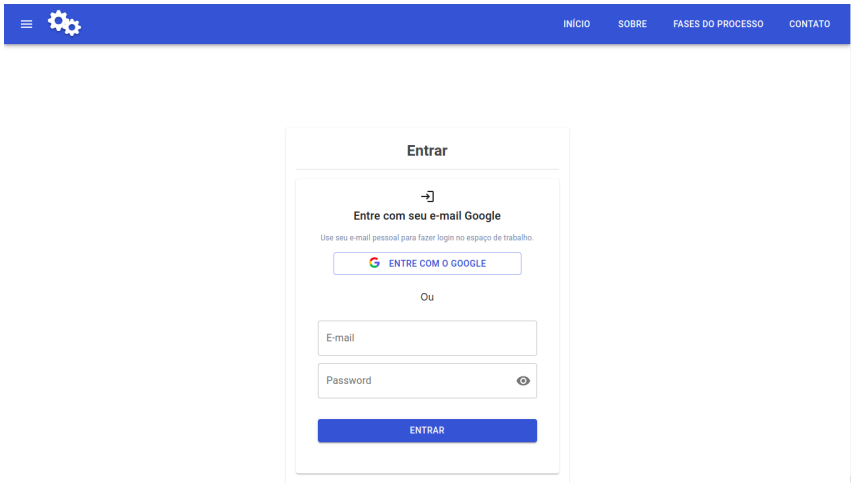
Na tela de registro, é necessário fornecer o *e-mail*, senha e nome do usuário. Esses valores são salvos no banco de dados, e a senha é armazenada de forma criptografada utilizando uma biblioteca chamada "bcrypt", que está presente na API e é acionada pelo *endpoint* responsável por adicionar um novo usuário dessa API (/api/user/add-user).

O *bcrypt* desempenha um papel fundamental na segurança do aplicativo, pois garante que as senhas dos usuários sejam armazenadas de forma segura e não possam ser facilmente decifradas em caso de violação de dados.

Ao acessar a tela de login e inserir o *e-mail* e senha, ao pressionar "enter", será redirecionado para a tela de visão geral (/overview). Nesse momento, as informações do usuário são recuperadas do banco de dados para fins de validação.

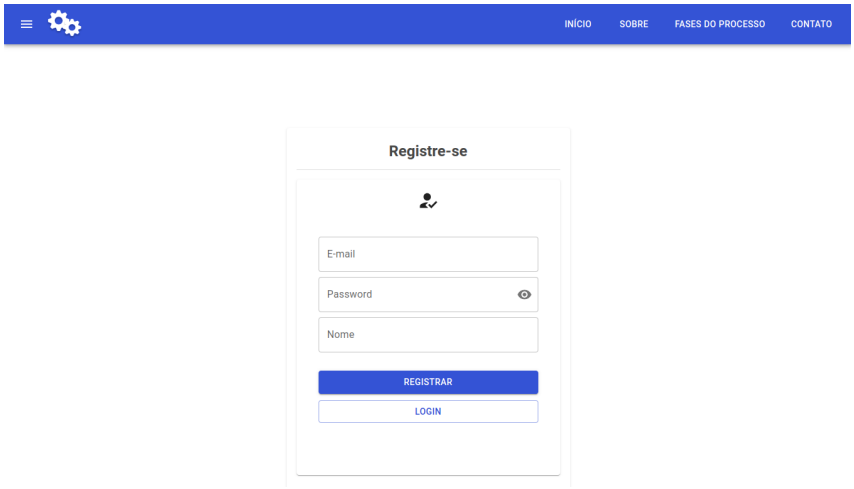
Importante notar que, na versão atual da aplicação, o botão de login com o Google não está disponível.

Figura 23 – Tela de Login projetada.



Fonte: O autor (2023).

Figura 24 – Tela de Cadastro projetada.

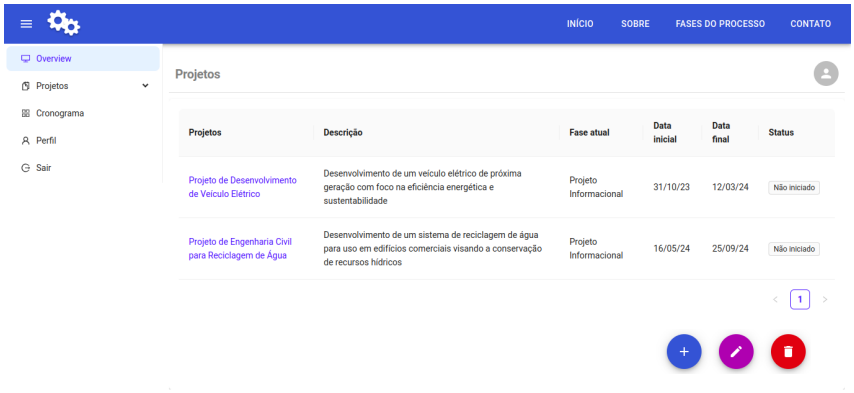


Fonte: O autor (2023).

Na tela de Visão Geral (/overview), você pode visualizar informações abrangentes sobre o projeto (Figura 25). Isso inclui detalhes como o nome do projeto, sua descrição, a fase atual em que se encontra, os prazos de início e término do projeto e o status atual. Abaixo dessa tabela de informações, é encontrado uma seção de ações que contém botões flutuantes. Ao acioná-los, são exibidos diálogos que permitem adicionar, editar ou excluir um projeto (Figuras 26, 27 e 28).

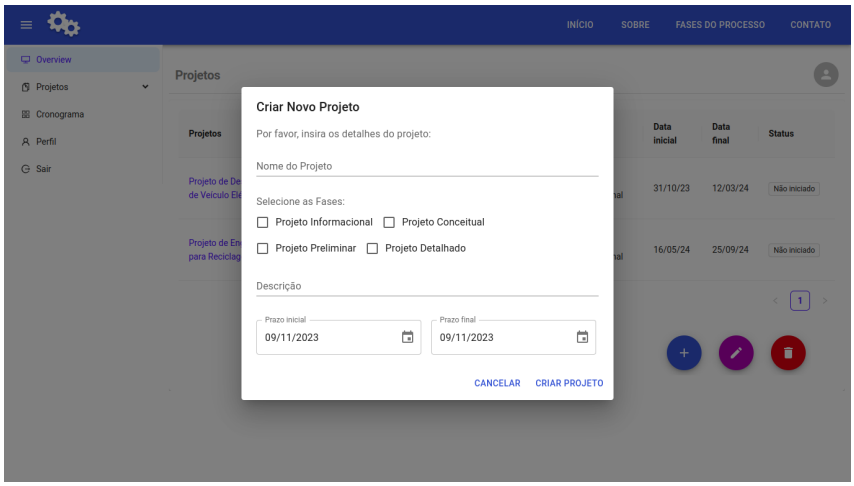
Além disso, há um menu lateral que oferece opções de redirecionamento para outras telas, incluindo "Atividades"(/activities), "Cronograma"(/cronograma) e "Perfil"(/perfil). Também existe um botão de saída que redireciona para a tela inicial da aplicação, permitindo ao usuário encerrar sua sessão quando desejado. Importante destacar que os dados dos projetos exibidos são apenas ilustrativos.

Figura 25 – Tela de Overview projetada.



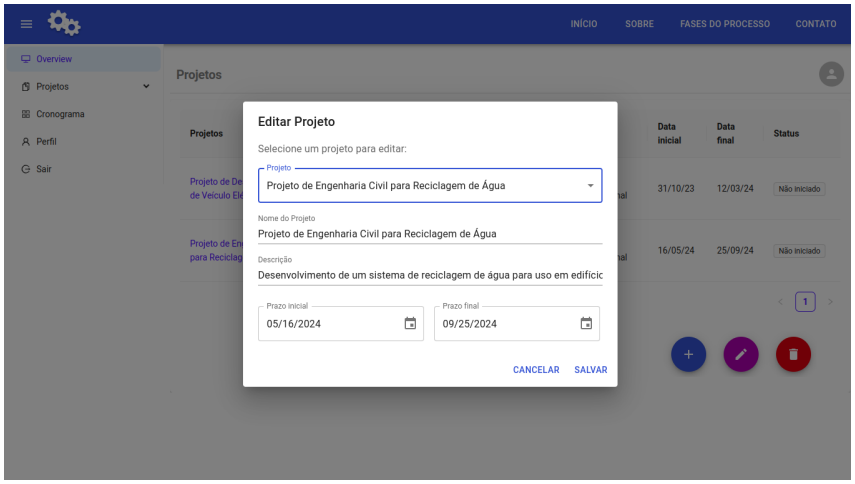
Fonte: O autor (2023).

Figura 26 – Diálogo para adicionar projeto.



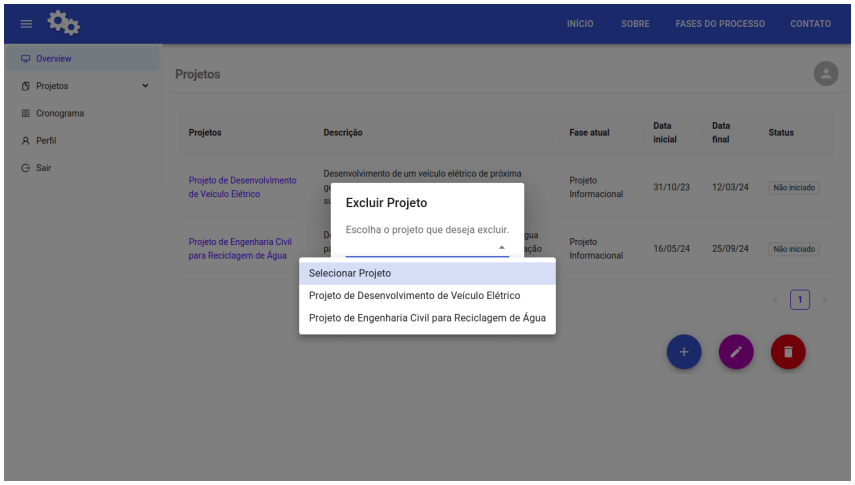
Fonte: O autor (2023).

Figura 27 – Diálogo para editar projeto.



Fonte: O autor (2023).

Figura 28 – Diálogo para excluir projeto.

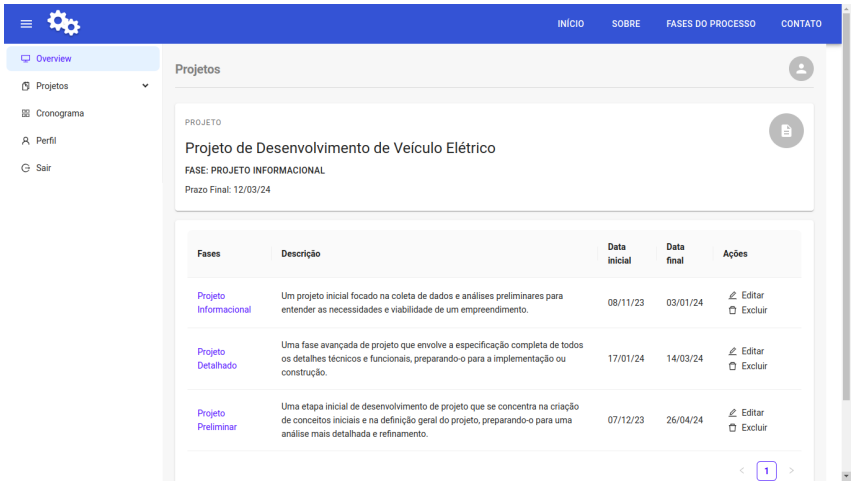


Fonte: O autor (2023).

Como observado na tela de *Overview*, na tabela de informações, há um link para cada projeto. Ao clicar em qualquer um dos projetos, será redirecionado para o caminho /projects/[id], e o "id"será substituído pelo ID do projeto específico que você clicou. Esse ID é recuperado a partir do banco de dados.

Dessa forma, exemplificando com um projeto específico, a tela mostrada na Figura 29 será encontrada.

Figura 29 – Tela de Informações do Projeto.



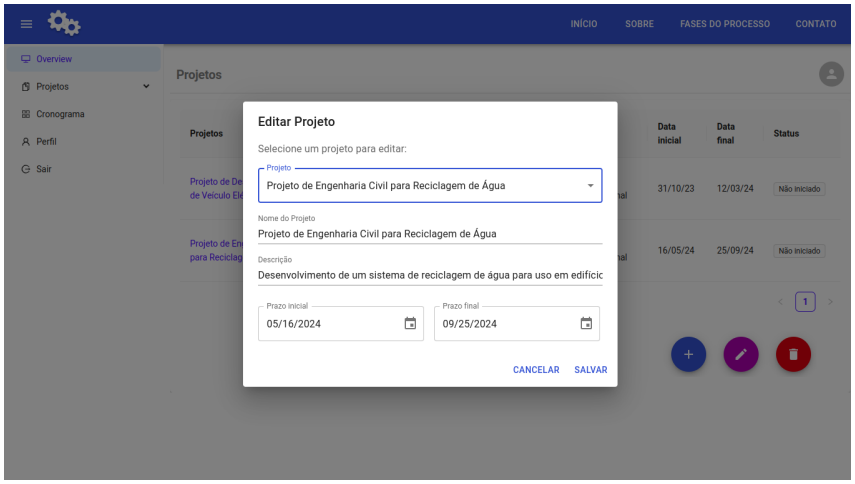
Fonte: O autor (2023).

Esta tela oferece informações específicas de um projeto em particular, incluindo o nome do projeto, sua fase atual e a data de conclusão prevista. Além disso, apresenta uma tabela contendo detalhes das fases que compõem o projeto, como seus nomes, descrições, prazos e opções de ação.

Na tabela, a opção "Editar"em uma linha abre um diálogo de edição específico para aquela fase em particular (Figura 30). O botão "Excluir"permite a remoção da fase correspondente na tabela. Adicionalmente, há um botão para adicionar uma nova fase, caso seja necessário. Ao clicar nesse botão, é aberto um diálogo onde é possível inserir informações sobre a nova fase e adicioná-la ao projeto (Figura 31).

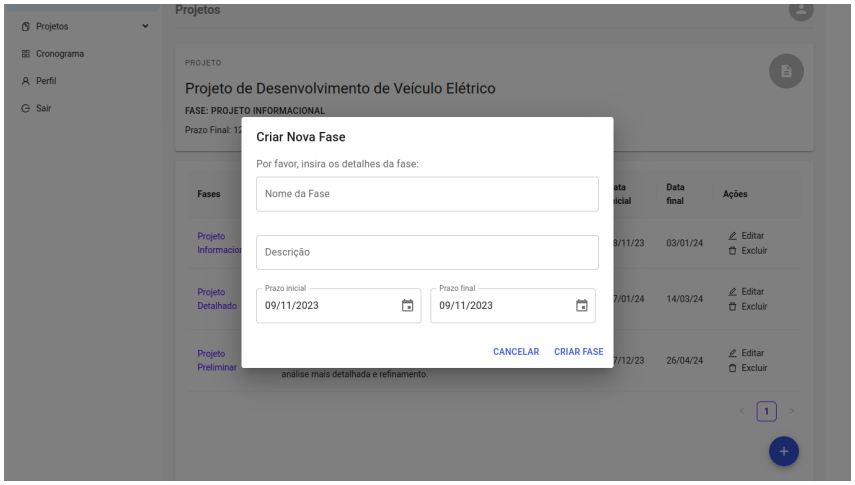
É importante destacar que todas as ações executadas nos botões da interface (adicionar, editar e excluir) resultam na adição, edição ou exclusão das informações no banco de dados. Essas operações são executadas pelo *back-end*, que foi configurado para realizar as operações de *CRUD (Create, Read, Update, Delete)* através do *endpoint* /api correspondente.

Figura 30 – Diálogo de edição de uma fase.



Fonte: O autor (2023).

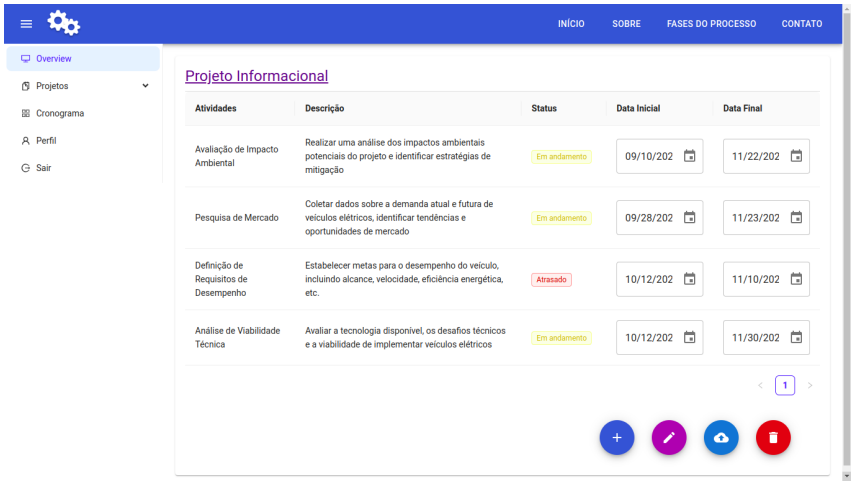
Figura 31 – Diálogo de criação de nova fase.



Fonte: O autor (2023).

A partir da tabela de informações das fases, também é possível clicar em uma das fases. Ao fazer isso, será redirecionado para o caminho /stages/[fase], onde "fase"representa o ID da fase armazenado no banco de dados. Ao selecionar uma fase específica, você terá acesso à tela de informações detalhadas sobre a fase escolhida (Figura 32).

Figura 32 – Tela informacional sobre a fase.



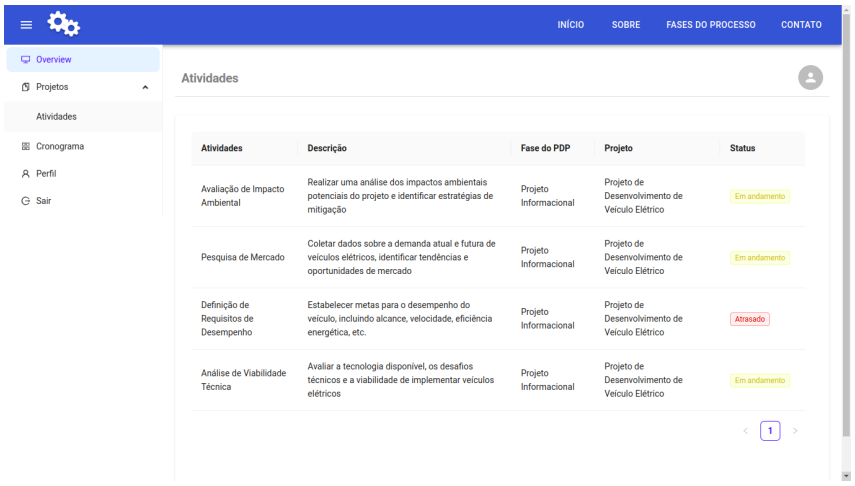
Fonte: O autor (2023).

O mesmo princípio se aplica às fases, onde é possível visualizar as atividades relacionadas a uma fase específica. Isso inclui detalhes como o nome da atividade, descrição, status e prazos correspondentes. Além disso, abaixo desses detalhes, existe uma seção de ações que permite adicionar uma nova atividade a essa fase, editar uma atividade existente, excluí-la ou até mesmo realizar *uploads* de arquivos relacionados a uma fase específica.

É importante mencionar que a ação de *upload* de arquivos está desativada devido à falta de disponibilidade de bancos de dados gratuitos no Vercel para armazenamento de arquivos, conforme discutido anteriormente.

Através do menu lateral, ao clicar no botão "Atividades", é possível ser direcionado para o caminho /activities. Nessa página, são exibidas informações de todas as atividades, abrangendo todos os projetos e suas respectivas fases (Figura 33).

Figura 33 – Tela de Atividades.

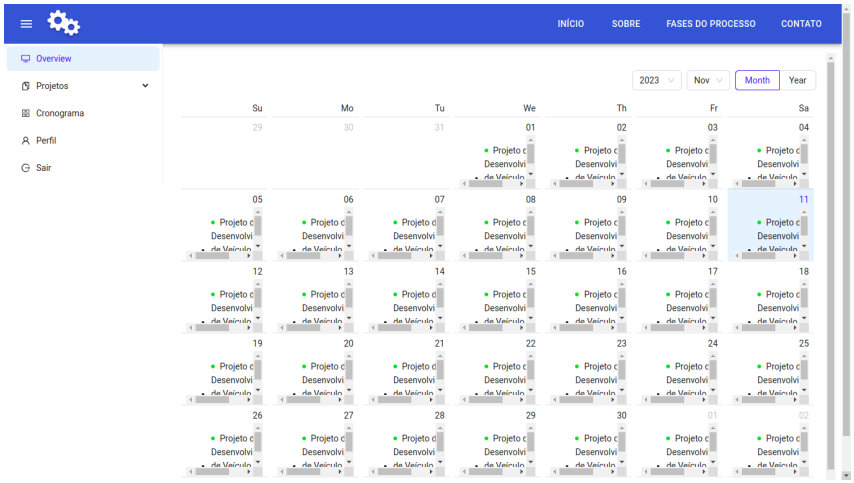


Fonte: O autor (2023).

A partir do menu lateral, também está disponível o "Cronograma". Esta seção exibe

um calendário com os projetos agendados de acordo com suas datas de início e término correspondentes (Figura 34.

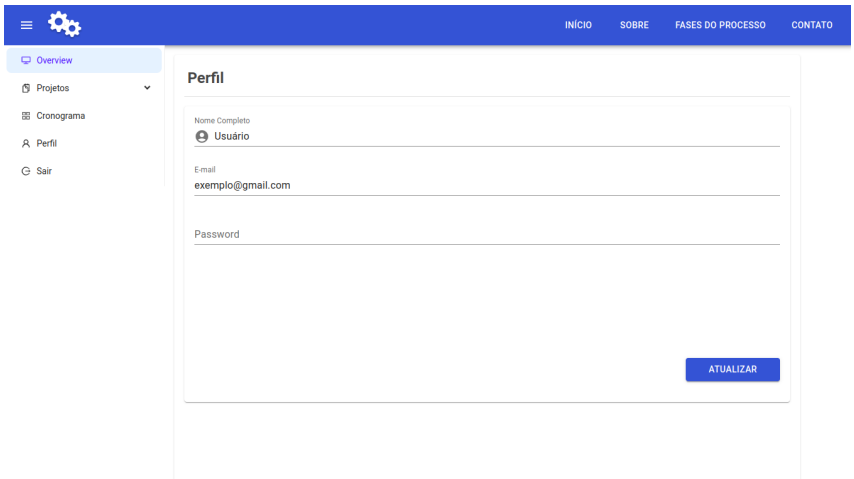
Figura 34 – Tela de Cronograma.



Fonte: O autor (2023).

No botão "Perfil", você acessa uma tela que exibe informações do usuário. Nessa tela, é possível editar e enviar novamente as informações para o banco de dados (Figura 35).

Figura 35 – Tela de Perfil.



Fonte: O autor (2023).

Dessa forma, foi apresentada uma visão abrangente da aplicação, incluindo suas telas e funcionalidades. Demonstrou-se os requisitos funcionais e não funcionais da aplicação, destacando sua capacidade de gerenciar projetos, suas fases e as atividades relacionadas a eles.

4.4.1 Tecnologias e ferramentas utilizadas

As tecnologias empregadas no projeto são essenciais para o desenvolvimento e sua gestão. No *front-end*, utilizou-se a linguagem de programação JavaScript em conjunto com o *framework* Next.js. Essa combinação permite criar uma interface de usuário responsiva e eficaz. Para o *back-end*, foi explorado as ferramentas oferecidas pelo Next.js para criar uma API incorporada ao próprio projeto, facilitando as operações de criação, leitura, atualização e exclusão (CRUD) no banco de dados.

No que diz respeito às ferramentas de desenvolvimento, foi utilizado o Git, uma ferramenta de controle de versão que auxilia a rastrear e gerenciar as diferentes iterações do código-fonte. Para compartilhar e colaborar nos repositórios Git, foi utilizado o GitHub, uma plataforma que simplifica a colaboração entre desenvolvedores.

Além disso, a IDE escolhida foi o Visual Studio Code, uma ferramenta versátil e altamente personalizável que desempenha um papel central no processo de desenvolvimento. As extensões disponíveis ampliam suas funcionalidades, permitindo a adaptabilidade da IDE às necessidades específicas e às tecnologias utilizadas.

Para gerenciar as atividades relacionadas ao projeto, foi definida a utilização do Trello, uma plataforma de gestão de tarefas que auxilia na organização, priorização e acompanhamento das atividades em andamento. Além disso, foi utilizado o Figma como ferramenta para criar e visualizar as interfaces das telas do projeto, permitindo uma idealização eficaz e visualização prévia antes da implementação.

5 CONCLUSÃO

Com base nos resultados apresentados, é possível concluir que este trabalho alcançou com sucesso seu objetivo principal de desenvolver uma aplicação web que permite a gestão eficiente de projetos de produtos, seguindo as fases estabelecidas no Processo de Desenvolvimento de Produto (PDP).

A relevância deste trabalho para um estudante de Engenharia Mecânica é notável, pois fornece um exemplo prático da aplicação de habilidades técnicas e gerenciais em um contexto real. Ele demonstra a importância da gestão eficiente de projetos de produtos, uma habilidade essencial para um engenheiro mecânico, e destaca a interseção entre a engenharia e o gerenciamento de projetos.

Em relação aos Objetivos de Desenvolvimento Sustentável (ODS), este trabalho está alinhado com o ODS 9 (Indústria, Inovação e Infraestrutura). Ao melhorar a eficiência na gestão de projetos de produtos, promove a inovação na indústria, o que é fundamental para o desenvolvimento sustentável.

Em termos de implicações gerenciais, a aplicação desenvolvida pode ter um impacto significativo nas empresas ao simplificar o processo de gerenciamento de projetos, promovendo a colaboração entre equipes e impulsionando a inovação. Isso pode resultar em maior competitividade no mercado e em uma gestão mais eficaz de recursos, o que é essencial para o sucesso das empresas no campo da engenharia mecânica.

Em síntese, este trabalho não se limita a cumprir seus objetivos com sucesso, mas também serve como uma peça fundamental na Engenharia Mecânica, na promoção do desenvolvimento sustentável por meio do ODS 9 e na melhoria das práticas gerenciais das empresas. Ele contribui de forma inequívoca para o progresso do conhecimento prático e teórico, iluminando o caminho para uma indústria mais eficiente, inovadora e sustentável, preparando estudantes e empresas para os desafios e oportunidades que o futuro reserva.

5.1 TRABALHOS FUTUROS

Considerando o futuro, é essencial otimizar o desempenho e a escalabilidade, fortalecer a segurança, aprofundar a modelagem de dados, melhorar o design de UX e manter a estabilidade da aplicação. Testes abrangentes, integração com APIs externas, segurança de dados, validação de entradas e restrição de acesso são necessários para proteger informações sensíveis. Além disso, registros de auditoria e monitoramento constante são cruciais para identificar e responder a ameaças de segurança. Essas melhorias garantirão a relevância contínua da aplicação diante das crescentes demandas.

REFERÊNCIAS

AGILE MANIFESTO. **Agile manifesto**. [S.l.: s.n.], 2001. Disponível em: http://agilemanifesto.org/. Acesso em: 10 abr. 2023.

BANKS, Alex; PORCELLO, Eve. **Learning React**. [S.l.]: O'Reilly Media, 2017.

BROWN, Tim. **Change by design: How design thinking transforms organizations and inspires innovation**. [S.l.]: HarperBusiness, 2009.

BROWN, Tim. **Design thinking**. Harvard Business Review, v. 86, n. 6, p. 84–92, 2008.

BUDD, Andy et al. **CSS Mastery: Advanced Web Standards Solutions**. [S.l.]: Apress, 2016.

CASTRO, E.; HYSLOP, B. **Visual QuickStart Guide: HTML and CSS.** Eighth Edition. Peachpit Press, 2014.

CDL Bom Despacho. **Desenhando o sucesso: design thinking como inovação no varejo.** Disponível em: https://www.cdlacibom.com.br/artigo/desenhando-o-sucesso-design-thinking-co Acesso em: 26 de setembro de 2023.

CERVO, Amado Luiz; BERVIAN, Pedro Alcino; SILVA, Roberto. **Metodologia científica**. 6. ed. [S.l.]: Pearson Universidades, 2006.

COOPER, R. G.; KLEINSCHMIDT, E. J. **New products: What separates winners from losers?** Journal of Product Innovation Management, Wiley Online Library, v. 20, n. 2, p. 169–184, 2003.

COOPER, Robert G. **Winning at New Products: Accelerating the Process from Idea to Launch**. 3. ed. [S.l.]: Basic Books, 2001.

IBM. **What is Java Spring Boot?**. Disponível em: https://www.ibm.com/topics/java-spring-boot. Acesso em: 07 de ago. 2023.

KELLEY, Tom; KELLEY, David. **Creative confidence: Unleashing the creative potential within us all**. Crown Business, 2013.

JOHANSSON, Johny. **Global Marketing: Foreign Entry, Local Marketing, and**

Global Management. McGraw-Hill Education, 2009.

KOTLER, Philip; KELLER, Kevin Lane. **Administração de marketing**. 15. ed. Pearson Universidades, 2019.

LARMAN, Craig. **Agile and iterative development: a manager's guide**. Pearson Education (US), 2003.

LIKER, Jeffrey K. **The Toyota way: 14 management principles from the world's greatest manufacturer**. McGraw-Hill, 2004.

MARTINS, Joaquim R. R. A.; LAMBE, Andrew B. **Multidisciplinary Design Optimization: A Survey of Architectures.**. *Aerospace Research Central (ARC)*. Publicado online em 20 Ago. 2013.

MARTIN, R. **Design of business: Why design thinking is the next competitive advantage**. Harvard Business Press, 2009.

MARTIN, Robert C. **Clean architecture: a craftsman's guide to software structure and design**. Prentice Hall Press, 2017.

METODOLOGIA Agil e Scrum. **Ciclo de desenvolvimento agile.** [S.l.]: s.n., 2016. Disponível em: http://metodologiagil.com/wp-content/uploads/2016/09/ciclo_desenvolvimento_agile_2.png. Acesso em: 3 abr. 2023.

MITAL, A.; DESAI, A.; SUBRAMANIAN, A. **Product Development: A Structured Approach to Consumer Product Development, Design, and Manufacture**. 2. ed. Elsevier, 2014.

MONTGOMERY, Cynthia A; PORTER, Michael E. **Strategy: Seeking and Securing Competitive Advantage**. Boston: Harvard Business Press, 1991.

NADLER, David A; TUSHMAN, Michael L. A model for diagnosing organizational behavior. *Organization Dynamics*, Elsevier, v. 13, n. 2, p. 35–51, 1995.

NODE.JS. **About Node.js®**. Disponível em: https://nodejs.org/. Acesso em: 07 ago. 2023.

PLATTNER, H.; MEINEL, C.; LEIFER, L. **Design thinking: Understand-improve-**

apply. [S.l.]: Springer, 2013.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de Software: Uma Abordagem Profissional.** 8. ed. Porto Alegre: AMGH Editora, 2016.

QUARKUS. **What is Quarkus?**. Disponível em: https://quarkus.io/about/. Acesso em: 07 ago. 2023.

RODRIGUES, L. S.; PAULA, I. C. de; ECHEVESTE, M. A. S. **Modelo teórico de integração de ferramentas de projeto no Processo de Desenvolvimento de Produto.** In: Simpósio de Engenharia de Produção, 15., 2008, Bauru. Anais... Bauru: 2008. Disponível em: <www.simpep.feb.unesp.br/anais.php>. Acesso em: 14/19/2023.

ROZENFELD, Henrique. Workshop Gestão de Projetos em Desenvolvimento de Produto. USP – São Carlos, 31/03/08. Disponível em: http://www.producao.ufrgs.br/arquivos/arquivos/080331_Workshop_Rozenfeld.pdf., Acesso em: 31 de maio de 2023.

SCHWABER, K.; SUTHERLAND, J. **The Scrum Guide**. [S.l.]: Scrum.org, 2017.

SHAW, M.; GARLAN, D. **Software architecture: perspectives on an emerging discipline**. [S.l.]: Prentice Hall Professional Technical Reference, 1996.

TECHTARGET. **front end and back end**. Disponível em: https://www.techtarget.com/whatis/definition/front-end#:~:text=The%20back%20end%20refers%20to,one%20or%20more%20programming%20languages. Acesso em: 07 ago. 2023.

ULRICH, K. T.; EPPINGER, S. D. **Product design and development**. 6th. New York: McGraw-Hill, 2015.

VERCEL, Inc. Next.js. **What is Next.js?**. Disponível em: https://nextjs.org/docs#what-is-nextjs. Acesso em: 07 ago. 2023.

VERCEL, Inc. Next.js. **Documentation**. Disponível em: https://nextjs.org/docs/app/building-your-application/routing#terminology. Acesso em: 17 de agosto de 2023.

WARD, A. C.; SOBEK II, D. K. **Lean product and process development**. [S.l.]: Lean Enterprise Institute, 2016.

WHEELWRIGHT, S. C.; CLARK, K. B. **Revolutionizing product development:**

Quantum leaps in speed, efficiency, and quality. New York: Free Press, 2011.

WOMACK, J. P.; JONES, D. T. **Lean thinking: banish waste and create wealth in your corporation**. Free Press, 2003.

49EDUCAÇÃO. **Ciclo lean detalhado.** Disponível em: https://49educacao.com.br/startup/lean-startup/. Acesso em: 10 abr. 2023.