UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

Bacharelado de Sistemas de Informação

Leonardo Ferreira de Souza

Navinix: Desenvolvimento de um Sistema de Gerenciamento para Micro e Pequenos Negócios

Diamantina/MG

2023

Leonardo Ferreira de Souza

Navinix: Desenvolvimento de um Sistema de Gerenciamento para Micro e Pequenos Negócios

Monografia apresentada ao Curso de Sistemas de Informação do Departamento de Computação da Universidade Federal dos Vales do Jequitinhonha e Mucuri como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Marcelo Ferreira Rego

Diamantina/MG

2023

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

FOLHA DE APROVAÇÃO

Leonardo Ferreira de Souza

NAVINIX: DESENVOLVIMENTO DE UM SISTEMA DE GERENCIAMENTO PARA MICRO E PEQUENOS NEGÓCIOS

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, como requisitos parcial para conclusão do curso.

Orientador: Prof. Dr. Marcelo Ferreira Rego

Aprovada em 13 de dezembro de 2023

BANCA EXAMINADORA

Prof. Dr. Marcelo Ferreira Rego
Faculdade de Ciências Exatas - DECOM - UFVJM

Profª. Drª. Cinthya Rocha Tameirão
Faculdade de Ciências Exatas - DECOM - UFVJM

Prof. Dr. Áthila Rocha Trindade
Faculdade de Ciências Exatas - DECOM - UFVJM

Documento assinado eletronicamente por **Marcelo Ferreira Rego**, **Servidor (a)**, em 20/12/2023, às 12:46, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Áthila Rocha Trindade**, **Servidor (a)**, em 20/12/2023, às 13:17, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Cinthya Rocha Tameirão**, **Servidor (a)**, em 20/12/2023, às 13:32, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufvjm.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1284941** e o código CRC **1871D4A7**.

Referência: Processo nº 23086.000729/2023-83 SEI nº 1284941

Dedico esse trabalho a pessoa que me inspirou a sempre ir mais longe. Obrigado por tudo pai.

Agradecimentos

Desejo expressar minha eterna gratidão...

Aos meus pais, Jean Carlos e Mirian pelo amor incondicional, apoio incansável e pelos sacrifícios que fizeram para que eu pudesse alcançar meus objetivos acadêmicos.

À minha querida namorada Natália Peres, por ser minha fonte constante de apoio, compreensão e motivação. Seu amor e paciência foram fundamentais para que eu superasse os desafios deste percurso acadêmico.

Ao meu orientador, Marcelo Ferreira Rego, pela orientação sábia, paciência e inspiração ao longo deste caminho desafiador.

Aos meus amigos e colegas, que compartilharam risadas, desafios e momentos de estudo. Juntos, enfrentamos os altos e baixos deste percurso acadêmico.

Este trabalho é dedicado a todos aqueles que, de alguma forma, contribuíram para minha jornada acadêmica. Obrigado por fazerem parte desta conquista.

"Eu posso não ter ido onde gostaria, mas acho que cheguei onde deveria. - Douglas Adams"

Resumo

O aumento do número de microempreendedores individuais no Brasil reflete a resposta à crise econômica e à pandemia, indicando uma necessidade impulsionada por condições econômicas desafiadoras. Ferramentas que auxiliem no gerenciamento do negócios podem ser um importante aliado para as micro e pequenas empresas aumentarem sua eficiência e melhorar sua competitividade. Desse modo, este trabalho tem como objetivo implementar um sistema para aprimorar a gestão de negócios, fornecendo uma solução tecnológica que atenda às demandas dos microempreendedores individuais diante dos desafios econômicos. Para o desenvolvimento deste trabalho, foram utilizadas abordagens dos métodos ágeis, como Scrum e histórias de usuário, para orientar o desenvolvimento. Foram desenvolvidos dois sistemas para este trabalho. Um sistema web foi construído com as linguagens PHP e o framework Laravel, responsável pelo processamento de dados e interações com o banco de dados. O segundo sistema foi desenvolvido em React Native para plataformas móveis, no qual os usuários irão interagir diretamente.

Palavras-chaves: Ferramenta de gestão, estoque, aplicativo mobile, força de vendas

Abstract

The increase in the number of individual microentrepreneurs in Brazil reflects a response to the economic crisis and the pandemic, indicating a need driven by challenging economic conditions. Tools that assist in business management can be an important ally for micro and small enterprises to increase their efficiency and enhance competitiveness. Thus, this work aims to implement a system to improve business management by providing a technological solution that meets the demands of individual microentrepreneurs in the face of economic challenges. For the development of this work, approaches from agile methods such as Scrum and user stories were used to guide the development process. Two systems were developed for this work. A web system was built using PHP and the Laravel framework, responsible for data processing and interactions with the database. The second system was developed in React Native for mobile platforms, where users will interact directly.

Key-words: Management tool, inventory, mobile app, sales force

Lista de Ilustrações

Lista de Tabelas

Lista de Abreviaturas e Siglas

MEIs	*Microempreendedor individual*
IBGE	*Instituto Brasileiro de Geografia e Estatística*
PHP	*Hypertext Preprocessor (Pré-Processador de Hipertexto)*
XP	*Extreme Programming (Programação extrema)*
OpenUP	*Open Unified Process (Processo Unificado Aberto)*
SGBD	*Sistema de gerenciamento de banco de dados*
MER	*Modelo entidade relacionamento*
DER	*Diagrama Entidade-Relacionamento*
ORM	*Object Relational Mapper (Mapeamento objeto-relacional)*
HTTP	*Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto)*
API	*Application Programming Interface (Interface de Programação de Aplicação)*

Sumário

1 Introdução

O atual cenário econômico brasileiro apresenta desafios expressivos, especialmente em decorrência da crise econômica global e da pandemia de COVID-19. O empreendedorismo tem surgido como uma resposta a esses desafios, como apontado por Soares et al. (2021) e Governo Federal (2020a). Recentemente, o Brasil testemunhou um aumento considerável no número de microempreendedores individuais (MEIs), ultrapassando a marca dos 10 milhões de registros, conforme destacado em Governo Federal (2020b).

Segundo Soares et al. (2021), a decisão de empreender no Brasil muitas vezes não é motivada por uma vocação intrínseca, mas sim pela necessidade impulsionada por condições econômicas desafiadoras. Durante os primeiros nove meses de 2020, um período crítico devido à pandemia, o número de MEIs aumentou consideravelmente, indicando a busca por alternativas em meio à crise.

Dentro de uma empresa a gestão de estoque e vendas podem ser consideradas áreas críticas, visto que elas tem impacto direto nos resultados financeiros e na satisfação dos clientes. Uma eficiente gestão de estoque e vendas pode aumentar a eficiência dos processos internos, reduzir custos, otimizar recursos e, consequentemente, melhorar a competitividade da empresa no mercado. Por outro lado, a falta de uma gestão adequada pode levar a problemas como excesso ou falta de estoque, perda de vendas, insatisfação dos clientes e prejuízos financeiros.

Um aspecto relevante a ser considerado é o impacto significativo da rápida evolução tecnológica na configuração dos padrões de acesso à informação e comunicação no Brasil. Conforme indicado por dados recentes do Instituto Brasileiro de Geografia e Estatística (IBGE), a posse de dispositivos móveis, especialmente celulares, registrou um crescimento substancial em todas as faixas etárias entre 2019 e 2021 (do Povo, 2022). Em contrapartida, o uso de microcomputadores teve um declínio durante o mesmo período, sinalizando uma clara preferência dos brasileiros pela conveniência e mobilidade oferecidas pelos dispositivos móveis, conforme relatório da Agência de Notícias IBGE (2022).

A crescente necessidade de mobilidade e flexibilidade no ambiente empresarial torna o desenvolvimento de softwares compatíveis para celulares uma questão importante a ser considerada na gestão de estoque e vendas. A falta de compatibilidade pode prejudicar a adoção dessas ferramentas especialmente por micro e pequenas empresas, limitando sua capacidade de gerenciamento de estoque e vendas em tempo real e reduzindo sua competitividade no mercado.

Nesse contexto, surge a motivação deste trabalho. Ao observar um notável aumento de 33% no número de microempreendedores individuais no terceiro trimestre de 2021, em comparação com o mesmo período de 2019, antes da pandemia, conforme dados do Valor Investe (2021), e considerando as dificuldades econômicas enfrentadas, percebeu-se a necessidade

de desenvolver uma solução que pudesse auxiliar e potencializar a gestão desses negócios emergentes.

A partir do contato com microempreendedores, identificou-se uma necessidade premente do controle do estoque e das vendas, em uma ferramenta acessível por dispositivos móveis e criada para atender demandas específicas das micro e pequenas empresas. Esse sistema deve ser acessível e de fácil utilização, possibilitando aos usuários gerenciarem seus estoques e suas vendas de maneira eficiente e eficaz.

Este estudo propõe uma solução tecnológica voltada para pequenos negócios, com o objetivo de aprimorar suas práticas comerciais.

1.1 Objetivos

Neste capítulo, será delineado o conjunto de metas estabelecidas para orientar o desenvolvimento do projeto. Esses objetivos estão relacionados à proposta de oferecer uma solução para os desafios frequentemente encontrados na gestão de pequenos negócios.

1.1.1 Objetivo Geral

Desenvolver e implementar um aplicativo de gerenciamento para pequenos negócios, visando oferecer uma solução para as necessidades de gestão empresarial. A partir desse propósito central, desdobram-se objetivos específicos que orientarão as etapas de desenvolvimento e avaliação do sistema.

1.1.2 Objetivos Específicos

- **Desenvolver o Aplicativo**
  - Criar uma interface funcional para o cadastro e autenticação de usuários.
  - Implementar a lógica necessária para garantir a segurança das informações do sistema.
- **Desenvolver funcionalidades relacionadas ao gerenciamento do produto**
  - Desenvolver funcionalidades para inserção, modificação e exploração de produtos.
  - Selecionar produtos comercializados em uma venda, incorporando funcionalidades como a aplicação de descontos, acréscimos e a especificação de quantidades.
- **Desenvolver funcionalidades relacionadas ao gerenciamento de clientes**
  - Criar uma interface dedicada à administração de clientes.

  - Coletar informações para personalizar o atendimento e fortalecer o relacionamento com o cliente.

- **Desenvolver funcionalidades relacionadas ao processo de vendas**
  - Implementar um processo de vendas intuitivo, desde a adição de produtos ao carrinho até a escolha de clientes e formas de pagamento.
  - Oferecer opções flexíveis ao usuário, permitindo a escolha entre salvar vendas como orçamentos ou finalizar transações.

- **Desenvolver funcionalidades para a análise de dados**
  - Desenvolver telas de relatórios para análise de dados de vendas.
  - Integrar diferentes tipos de gráficos para uma compreensão mais profunda das tendências e padrões presentes nos dados.

- **Desenvolver funcionalidades relacionadas a configurações e segurança**
  - Criar uma seção de configurações para gerenciar informações pessoais.
  - Comprometer-se com práticas de segurança, como segregação de informações de acesso e implementação de um banco de dados central criptografado.

2 Trabalhos Relacionados

Com o intuito de situar o leitor no contexto da pesquisa e fornecer diretrizes para o desenvolvimento desta monografia, este capítulo tem por objetivo apresentar trabalhos prévios que abordam propostas semelhantes, auxiliando na elaboração de uma abordagem apropriada.

O trabalho de Oliveira (2014) teve como objetivo aprimorar a experiência de compra de clientes em lojas do segmento de vestuários e sapatos, buscando agilizar o processo de busca por informações dos produtos em estoque. Para isso, foram realizadas pesquisas quantitativas e qualitativas para identificar as necessidades dos clientes e verificar a satisfação deles com relação ao processo de busca de informações nas lojas. A partir dos resultados obtidos, foi identificada a necessidade de desenvolver um sistema para tornar esse processo mais rápido e eficiente.

Foram propostos dois sistemas: um para desktop e outro para celular com sistema Android. O primeiro foi desenvolvido em Java e feito para administrar as informações de estoque e realizar vendas de produtos. O segundo sistema também foi desenvolvido em Java com a utilização da plataforma Android Studio, e contém as funções de listar os produtos e a sua quantidade em estoque. Ambas aplicações compartilham a mesma base de dados.

Motivados pela necessidade de criar soluções tecnológicas que permitissem a realização de vendas de ingressos e o controle das vendas de bebidas em eventos de forma mais eficiente e automatizada. Rodrigues and Ribeiro (2020) propuseram a prototipagem e modelagem de dois aplicativos distintos. O primeiro deles visa facilitar a venda de ingressos para eventos, enquanto o segundo se concentra no controle das vendas de bebidas durante os eventos. É importante ressaltar que o desenvolvimento completo dos softwares não foi abordado neste trabalho, que se concentrou na criação de protótipos e na modelagem dos sistemas propostos.

Slominski (2016) analisou os benefícios que a utilização de ferramentas de gestão de estoque pode trazer para empresas de pequeno e médio porte. Foram utilizadas somente ferramentas gratuitas, pois o trabalho buscava trazer melhorias para a gestão de estoque sem realizar grandes investimentos em softwares.

Foram utilizados alguns indicadores para poder fazer a medição da eficiência do estoque, e através desses indicadores foi possível observar uma melhoria no desempenho com a utilização das ferramentas apresentadas. Um ponto importante apresentado é o da importância de se conhecer as necessidades da empresa, para que seja escolhida uma ferramenta eficaz na resolução do problema.

O trabalho de Oliveira (2019) teve como objetivo criar um sistema web responsivo para fazer o gerenciamento de estoque de uma empresa de telecomunicações. Para o desenvolvimento desse sistema foi utilizado o framework Laravel baseado na linguagem PHP. A metodologia utilizada se baseia nas técnicas de XP, adaptando a sua literatura e realizando apenas algumas

de suas práticas.

Foi desenvolvido um único sistema que consegue interagir com plataformas web e mobile por possuir elementos de design responsivo. Tal solução trouxe maior dinamismo ao sistema sem ter a necessidade de ser desenvolvido um outro software separadamente, e dessa forma foi possível atingir uma maior quantidade de usuários.

Prates and Patino (2004) em seu artigo buscaram analisar as características de pequenas empresas em relação à utilização da Tecnologia da Informação como ferramenta estratégica para o processo de planejamento, direção e controle. O objetivo dos autores foi investigar as dificuldades e benefícios encontrados pelas organizações na implantação das tecnologias nos processos da empresa, além de avaliar o impacto da resistência por parte dos funcionários.

A resistência dos funcionários foi identificada como a principal dificuldade, evidenciando a falta de treinamento prévio sobre os benefícios da tecnologia. Os benefícios mais importantes incluíram o aumento da satisfação do usuário e melhorias nos controles, como a redução de redundância de operações e maior velocidade de resposta.

Ramsdorf (2014) em seu artigo destaca a influência do mercado de aplicativos para celular na economia e eficiência dos processos, e em seu artigo, o autor analisa o uso de aplicativos gratuitos para automação de processos em micro e pequenas empresas. O estudo de caso utilizado foi uma empresa familiar de médio porte no setor de aparas de papel, onde o autor buscou verificar como os aplicativos podem otimizar o tempo, reduzir custos e melhorar a qualidade dos processos em empresas de menor porte.

Para implementar novas tecnologias com sucesso, os gestores precisam identificar quais processos automatizar e quais usuários terão acesso aos novos recursos. A falta de cultura digital e preocupações com segurança de dados podem afetar a adoção dessas ferramentas. No entanto, o autor destaca que a digitalização traz vantagens como redução de custos, otimização de tempo e organização de dados para as empresas.

3 Referencial Teórico

Este capítulo tem como objetivo apresentar as principais metodologias de desenvolvimento de software que serviram como base para a elaboração deste trabalho, bem como as principais tecnologias utilizadas. Serão apresentadas metodologias ágeis e tradicionais, ressaltando as características e vantagens de cada uma delas. Além disso, serão descritas as linguagens de programação e ferramentas utilizadas com o objetivo de destacar as escolhas técnicas que foram feitas para garantir a qualidade do sistema proposto.

3.1 Metodologias de Desenvolvimento

O desenvolvimento de software, como evidenciado por de Azevedo Terceiro (2013), é uma tarefa complexa que abrange diversos fatores que podem influenciar sua execução. Diante dessa complexidade, surgiram métodos estruturados e planejamentos adequados para auxiliar as organizações a planejar e executar projetos de forma eficiente. Na presente seção será apresentado uma breve introdução sobre as metodologias de desenvolvimento de software, enfatizando a metodologia escolhida para ser utilizada neste trabalho.

3.1.1 Modelo Cascata

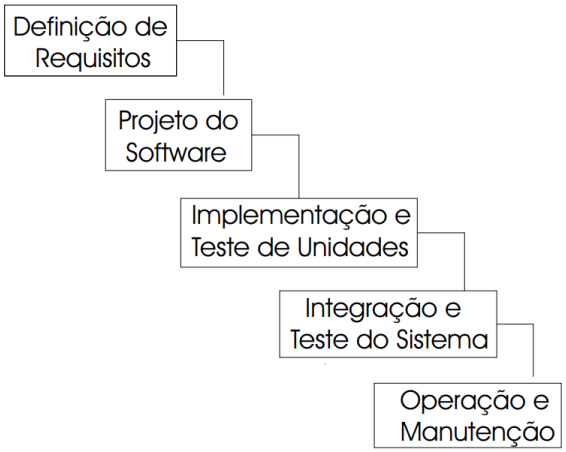
Pressman (2001) ressalta que o modelo cascata é uma metodologia tradicional de desenvolvimento de software caracterizada por uma sequência rigorosa de etapas que devem ser seguidas uma após a outra. Cada etapa tem associada a ela uma documentação padrão, que deve ser aprovada antes de se prosseguir para a próxima etapa. As etapas geralmente incluem definição de requisitos, projeto do software, implementação, teste unitário, integração, teste do sistema, operação e manutenção.

Ele também afirma que embora o modelo cascata ofereça uma estrutura clara e definida para o desenvolvimento de software, seu principal problema é a rigidez que o torna pouco adaptável às mudanças que frequentemente ocorrem durante o desenvolvimento de um projeto. Esse modelo deve ser utilizado somente quando os requisitos do projeto são completamente compreendidos e não há riscos significativos de mudanças. Caso contrário, a adoção de uma abordagem mais flexível e ágil pode ser mais adequada.

Conforme destacado por Franco (2007) e exemplificado pela Figura 1, o modelo cascata divide o desenvolvimento de software em etapas distintas, iniciando-se pela Definição de Requisitos, onde as necessidades do cliente são identificadas e definidas. Em seguida, a fase de Projeto de Software aprofunda esses requisitos, transformando-os em uma especificação estruturada. A etapa subsequente, denominada Implementação e Testes de Unidade, concentra-se na

a estrutura de dados, na arquitetura do software, nos procedimentos e na interface do programa de maneira abrangente. Após a codificação, o estágio de Integração e Teste do Sistema assegura a conformidade do software com os requisitos e critérios estabelecidos. Finalmente, a fase de Operação e Manutenção engloba a entrega do software ao cliente e sua implantação.

Figura 1 – Modelo Cascata



Fonte: Pressman (2001)

Contudo, é importante destacar que, como apontado por Franco (2007), o modelo cascata é raramente adotado na prática devido à sua inflexibilidade e à complexidade inerente de prever todos os requisitos do projeto antecipadamente, especialmente em projetos complexos e em constante evolução. Esse modelo tende a ser mais apropriado para situações em que os requisitos do software são conhecidos e estáveis.

Soares (2004a) em seu artigo também afirma que o uso de metodologias tradicionais apresentam algumas desvantagens, dentre as quais destacam-se a dificuldade e o alto custo de antecipar a análise dos fatores que influenciarão o desenvolvimento de software. Além disso, a inflexibilidade no planejamento pode se tornar um problema caso haja necessidade de realizar modificações.

As metodologias tradicionais, também conhecidas como sequenciais ou cascata, são adequadas para projetos onde os requisitos do software são claramente definidos e previsíveis. No entanto, os requisitos do software estão em constante mudança e são influenciados por diversos fatores, o que pode tornar difícil manter um escopo previsível, mesmo com a utilização de metodologias estruturadas e sequenciais (Soares, 2004a).

3.1.2 Metodologias Ágeis

O conceito de agilidade na entrega de serviços e produtos fez parte da história da indústria manufatureira dos Estados Unidos. Nos anos 1980, em resposta à eficácia das práticas de produção enxuta das fábricas japonesas e preocupações no Congresso sobre a lucratividade

da indústria dos EUA, um programa federal foi iniciado, e este programa tinha como objetivo desenvolver o que mais tarde se tornaria a base do conceito de manufatura ágil. Franco (2007) enfatiza que, a partir desse programa, foram realizadas pesquisas estratégicas que resultaram em um amplo conjunto de estratégias de negócios, modelos e estudos de caso que trouxeram valiosas contribuições para a compreensão da agilidade nos contextos de negócios e indústria.

A Engenharia de Software inicialmente se inspirou na indústria de manufatura, mas nos anos 90, devido à rigidez dos métodos tradicionais, surgiram abordagens mais ágeis. Porém, o termo "ágil"para descrever essas abordagens foi estabelecido oficialmente somente em 2001, quando um grupo de especialistas criou o Manifesto Ágil. Esse manifesto trouxe transformações significativas para a indústria de desenvolvimento de software, destacando a importância das pessoas e permitindo respostas ágeis às mudanças que ocorrem no cenário empresarial (Prikladnicki et al., 2014).

A escolha entre metodologias tradicionais e ágeis no desenvolvimento de software depende das necessidades e características do projeto. As tradicionais são mais apropriadas quando os requisitos do projeto são previsíveis e bem definidos desde o início, enquanto as ágeis são ideais para projetos em que os requisitos são mais voláteis e sujeitos a mudanças ao longo do tempo.

Dentro do contexto das metodologias ágeis, é importante destacar que cada uma delas apresenta características que as tornam mais adequadas a diferentes situações. Por exemplo, as metodologias *Scrum*, *Extreme Programming* (XP) e *Open Unified Process* (OpenUP) são amplamente utilizadas e compartilham princípios essenciais de flexibilidade e colaboração, mas divergem em suas ênfases e abordagens.

Prikladnicki et al. (2014) destacam as principais características dessas três metodologias ágeis. O Scrum se concentra na gestão de projetos, com papéis definidos e cerimônias estruturadas. Por outro lado, o XP concentra-se nas práticas de engenharia de software, enfatizando aspectos como testes e programação em pares. Já o OpenUP oferece uma abordagem mais abrangente, abordando todo o ciclo de vida do projeto. As diferenças entre essas metodologias incluem a definição de papéis, a utilização de artefatos, a escalabilidade entre outros aspectos.

3.1.2.1 Scrum

A escolha de adotar o Scrum como metodologia de desenvolvimento neste trabalho levou em consideração as características e desafios específicos do projeto em questão, visto que o ambiente em que o projeto está inserido requer flexibilidade e capacidade de adaptação devido à natureza volátil e incerta dos requisitos. Nesse sentido, o Scrum se mostra uma escolha adequada, pois permite lidar de forma eficiente com essas demandas, proporcionando agilidade e respostas rápidas às mudanças que possam surgir.

O Scrum se adequa às necessidades do desenvolvimento de um aplicativo mobile, pois

possibilita a entrega de valor em curtos intervalos de tempo, garantindo a qualidade do produto e a satisfação dos usuários. Além disso, a transparência e a inspeção contínua permitem um acompanhamento eficaz do processo de desenvolvimento, assegurando que o aplicativo atenda às expectativas e necessidades dos usuários de forma contínua e iterativa.

O Scrum compreende que o desenvolvimento de sistemas envolve diversas variáveis que estão sujeitas a alterações ao longo do processo. Franco (2007) aborda esses aspectos e apresenta uma perspectiva realista do desenvolvimento de software, enfatizando a relevância de um processo flexível e adaptável:

> A idéia central do Scrum é que o desenvolvimento de sistemas envolve diversas variáveis (ambientais e técnicas) e elas possuem grande probabilidade de mudar durante a execução do projeto (por exemplo: requisitos, prazos, recursos, tecnologias etc.). Estas características tornam o desenvolvimento do sistema de software uma tarefa complexa e imprevisível, necessitando de um processo flexível e capaz de responder às mudanças. (Franco, 2007).

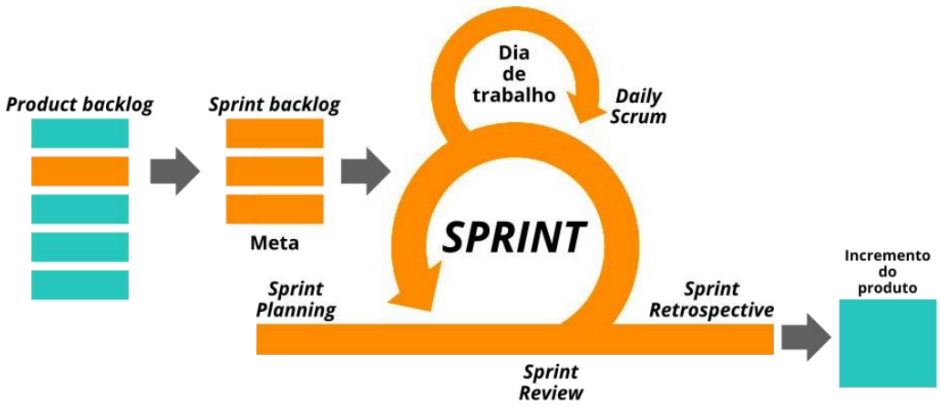
A primeira referência ao termo ”Scrum” no contexto do desenvolvimento de produtos foi feita por Takeuchi e Nonaka em 1986, onde eles consolidaram as melhores práticas utilizadas por empresas japonesas inovadoras, descrevendo um processo adaptativo, rápido e auto gerenciado (Franco, 2007). De acordo com Franco (2007), posteriormente Takeuchi e Nonaka aprimoraram e refinaram esses processos, transformando-os em uma abordagem empírica que incorpora ideias da teoria de controle de processo industrial para o desenvolvimento de software, introduzindo conceitos de flexibilidade, adaptabilidade e produtividade.

O Scrum adota a abordagem de dividir o desenvolvimento em iterações denominadas sprints, com um curto tempo de duração, onde equipes pequenas são responsáveis pelo desenvolvimento das funcionalidades definidas no início de cada sprint. Soares (2004b) destaca que o Scrum valoriza a realização de reuniões diárias de acompanhamento, de curta duração, como uma prática essencial para discutir o progresso do projeto, identificar dificuldades e solucionar possíveis impedimentos encontrados ao longo do processo de desenvolvimento.

Schwaber (1995) agrupou as práticas e ferramentas utilizadas pela equipe durante a sprint em 4 pontos principais: Reunião de planejamento do sprint; Backlog do sprint; Reunião diária do Scrum; Reunião de revisão do sprint. Ramos and Junior (2017) elaboraram o diagrama da Figura 2, o qual apresenta uma visão geral dos processos, com pontos semelhantes aos propostos por Schwaber (1995), porém adaptados às especificidades do seu trabalho. Esse diagrama proporciona uma representação visual mais elucidativa dos processos que ocorrem no desenvolvimento utilizando a metodologia SCRUM.

No início do projeto, o *product backlog* é criado, consistindo em uma lista que contém todos os requisitos que se tem conhecimento. Com base nesse artefato, é dado início à *sprint*. A

Figura 2 – Ciclo de desenvolvimento proposto pelo SCRUM



Fonte: Ramos and Junior (2017)

primeira etapa é conhecida como *Sprint Planning*. Nessa fase, os membros do projeto negociam o que será desenvolvido, priorizando os pontos de maior importância do *product backlog*, e a partir dessa seleção, é criada a *Sprint Backlog*, que detalha as tarefas específicas que a equipe deve realizar para atender aos requisitos selecionados.

Conforme mencionado por Ramos and Junior (2017), uma vez concluído o *Sprint Planning*, dá-se início ao desenvolvimento dos *Sprints*, seguindo a priorização dos itens do *Product Backlog*. Diariamente, são realizadas reuniões conhecidas como *Daily Scrum*, que têm como propósito acompanhar o progresso das atividades individuais da equipe e planejar as tarefas para o próximo dia de forma dinâmica e iterativa. No último dia do *Sprint*, ocorre o *Sprint Review*, onde o produto é revisado e adaptado conforme as necessidades identificadas, resultando em um incremento do produto ao final da *sprint*. (Ramos and Junior, 2017)

Segundo Schwaber (1995), o Scrum envolve a formação de duas equipes distintas: a equipe de gerenciamento e a equipe de desenvolvimento. Cada equipe possui características específicas, sendo elas:

- Sob a liderança do Gerente de Produto, a equipe de gerenciamento assume a responsabilidade de estabelecer o conteúdo inicial e o cronograma do lançamento, além de monitorar sua evolução à medida que o projeto avança e as circunstâncias mudam. Além dessas atribuições, a equipe de gerenciamento tem a responsabilidade de gerenciar o *backlog*, avaliar e mitigar os riscos envolvidos, e determinar o conteúdo que será incluído no lançamento.
- As equipes de desenvolvimento devem ter um tamanho reduzido, com a presença de desenvolvedores e profissionais responsáveis pelo controle de qualidade. Essas equipes são responsáveis por identificar as mudanças necessárias para implementar os itens do *backlog* e gerenciar todos os problemas associados a essas alterações. A composição das equipes de desenvolvimento é baseada no conhecimento e na experiência dos membros em relação às funcionalidades específicas que serão desenvolvidas por cada equipe.

De acordo com Franco (2007), o Scrum fundamenta-se principalmente em pessoas, não em processos, atribuindo, por isso, grande importância aos papéis desempenhados por cada indivíduo no desenvolvimento do projeto. A seguir, serão apresentados os principais stakeholders, representando indivíduos ou grupos com interesse direto ou indireto no projeto.

Cliente: é um participante ativo nas tarefas relacionadas à definição da lista de recursos do software que está sendo desenvolvido ou melhorado, fornecendo detalhes sobre os requisitos e restrições do produto final desejado;

Gerente: é responsável por tomar as decisões finais e fornecer informações visuais de maneira gráfica, seguindo as normas e convenções estabelecidas para o projeto. Ele também é responsável por negociar as metas e requisitos do projeto com os clientes;

Equipe Scrum: é o grupo responsável pelo projeto, tendo autoridade para tomar as decisões necessárias e se organizar para alcançar os objetivos estabelecidos. Eles são responsáveis por estimar o esforço necessário, elaborar e revisar a lista de recursos do produto e identificar obstáculos que precisam ser removidos do projeto;

Scrum Master: é responsável por assegurar que o projeto siga as práticas, valores e regras estabelecidas pelo Scrum, e que o andamento do projeto atenda às expectativas do cliente. Ele mantém contato com a equipe Scrum, o cliente e o gerente durante o projeto. Além disso, é responsável por identificar e remover qualquer impedimento que possa surgir ao longo do projeto, com o objetivo de garantir que a equipe esteja trabalhando da forma mais eficiente possível;

Product Owner: é o responsável oficial por conceber, gerenciar, supervisionar e tornar visível a lista de recursos do produto. Ele é selecionado pelo Scrum Master, pelo cliente e pelo gerente de projeto. Ele é responsável por tomar as decisões finais sobre as tarefas necessárias para transformar a lista de recursos em um produto final, contribuir na estimativa do trabalho de desenvolvimento necessário e detalhar as informações sobre a lista de recursos utilizada pela equipe Scrum.

3.1.2.2 Kanban

A decisão de utilizar o Kanban neste trabalho baseia-se em sua habilidade de oferecer uma visão clara do progresso por meio do uso de cartões Kanban, permitindo a priorização de tarefas, controle do fluxo de trabalho e adaptação ágil a mudanças. Dessa forma, o Kanban emerge como uma escolha adequada, proporcionando benefícios significativos para o desenvolvimento e acompanhamento do projeto.

De acordo com Silva and Anastácio (2019), após a devastação causada pela Segunda Guerra Mundial, o Japão enfrentou uma crise econômica que demandou soluções estratégicas

para redução de custos e aumento da produtividade. Nesse contexto, a empresa Toyota desenvolveu a técnica Kanban na década de 1960, utilizando cartões coloridos como forma de controlar estoques e produção. Silva and Anastácio (2019) ressalta que o Kanban vai além do controle de estoque e produção, sendo uma filosofia que valoriza a excelência, simplicidade e o respeito pelas pessoas.

Desde a sua criação pela Toyota, o Kanban passou por um contínuo desenvolvimento e evolução, tornando-se a técnica de gerenciamento de demandas que conhecemos hoje. Fundamentalmente, o Kanban é constituído por um quadro visual com colunas e cartões que representam as demandas. Essa abordagem proporciona uma visão clara do estado das tarefas, aprimorando a capacidade de priorização, identificação de obstáculos e fomentando a melhoria contínua do fluxo de trabalho.

Essa técnica, como destaca Silva and Anastácio (2019), contribui para a otimização dos processos, o aumento da produtividade e a entrega consistente de resultados. Ao utilizar o Kanban, as equipes têm maior controle sobre suas demandas, podendo acompanhar visualmente o progresso, adaptar-se a mudanças e realizar ajustes necessários para alcançar um desempenho eficaz. Em suma, a abordagem visual e centrada no fluxo de trabalho do Kanban pode trazer vantagens substanciais para o gerenciamento eficiente deste projeto, promovendo melhorias contínuas e resultados satisfatórios.

3.2 Linguagens de programação

Nesta seção será feita uma rápida apresentação das linguagens de programação que foram utilizadas para a realização do trabalho, além de explicar brevemente as vantagens que a linguagem de programação escolhida possui em relação às suas semelhantes, levando em conta os objetivos que se deseja alcançar neste trabalho.

3.2.1 PHP

A linguagem de programação PHP, criada em 1994 por Rasmus Lerdorf, foi originalmente concebida como um conjunto de scripts para criar páginas web dinâmicas em servidores. No ano seguinte, em 1995, foi adicionado o suporte para interpretar dados de formulários HTML e também para acesso a banco de dados SQL, o que ampliou sua capacidade e popularidade (Zago, 2011).

Segundo Zago (2011), uma das vantagens do PHP é que o código fonte da aplicação não fica exposto para os clientes, e a interação com o banco de dados e a aplicação é feita somente por meio do HTML puro, que é retornado pelo servidor. Zago (2011) aponta também uma outra vantagem do PHP, que é o suporte a um grande número de banco de dados, incluindo o PostgreSQL, que também será usado neste trabalho.

Uma das vantagens de escolher uma linguagem de programação como o PHP, além da experiência e conhecimento prévio, é a sua capacidade de lidar com as tarefas requeridas pelo projeto. No caso em questão, a linguagem oferece as ferramentas necessárias para implementar as ações descritas no levantamento de requisitos, além de poder se comunicar com as demais ferramentas selecionadas, como o banco de dados escolhido, garantindo um bom desempenho e integração entre os componentes do sistema.

3.2.2 JavaScript

O *JavaScript* teve sua origem na *Netscape Communications Corporation*, passando pelos nomes Mocha e *LiveScript*, até ser oficialmente lançado como *JavaScript* em 1995, quando foi integrado ao navegador Netscape. Conforme apontado por Grillo and de Mattos Fortes (2008), essa linguagem foi concebida para implementar tecnologias de processamento no lado do cliente. Apesar das semelhanças sintáticas com o Java, o termo "*JavaScript*"foi escolhido, mesmo não compartilhando outras relações substanciais com a linguagem Java.

O *JavaScript* destaca-se por seu apoio a diversos estilos de programação, como orientação a objetos e funcional, além de contar com uma comunidade ativa, vasta documentação e bibliotecas notáveis como React, Angular e Vue.js, contribuindo assim para a eficiência no desenvolvimento (Grillo and de Mattos Fortes, 2008). As atualizações contínuas introduzem novos recursos e melhorias, consolidando o *JavaScript* como uma escolha popular e robusta para o desenvolvimento de softwares

O JavaScript se apresenta como uma solução eficaz para mitigar os desafios enfrentados pelos programadores ao aprender e desenvolver em diversas linguagens voltadas para diferentes plataformas e sistemas. Essa abordagem, conforme destacado por da Silva Cruz et al. (2018), não apenas simplifica o processo de aprendizagem para os desenvolvedores, mas também beneficia as empresas ao eliminar a necessidade de equipes específicas para cada plataforma. Tendo em consideração esses aspectos, a decisão de empregar o JavaScript como base para o aplicativo mobile apresenta-se como uma escolha vantajosa.

3.3 Banco de Dados

No desenvolvimento do aplicativo proposto, a forma como os dados são organizados e gerenciados desempenha um papel significativo. Este capítulo aborda as ferramentas utilizadas para efetuar operações nos bancos de dados.

3.3.1 PostgreSQL

O PostgreSQL é um sistema gerenciador de banco de dados objeto relacional, ou SGBD, que se destaca por ser um software de código aberto. O PostgreSQL foi inicialmente desenvolvido

na Universidade de Berkeley, Califórnia, e liderado pelo professor Michael Stonebraker (Souza et al., 2011).

Conforme Souza et al. (2011) apresentaram em seu trabalho, a utilização do PostgreSQL possibilita a realização de consultas complexas, com junções de tabelas e com inúmeras condições, além de permitir a utilização de disparadores, que são funções executadas mediante a algum evento do banco de dados.

O Postgres se destaca como uma opção confiável e escalável para projetos de desenvolvimento de software devido à sua capacidade de utilizar recursos como os disparadores (triggers). Esses recursos permitem uma maior flexibilidade e personalização no gerenciamento de banco de dados, trazendo maior facilidade para projetos com necessidades específicas.

3.3.2 pgAdmin

A interface gráfica disponibilizada pelo pgAdmin é um recurso relevante no desenvolvimento do sistema proposto, uma vez que permite o acesso aos dados de forma mais visual e simplificada, tornando a criação de tabelas, consultas e outras operações no banco de dados mais intuitiva e diminuindo a quantidade de erros que poderiam ocorrer.

3.4 Frameworks

No caso deste trabalho, foram selecionados o Laravel como framework *backend* e o React Native para a criação do aplicativo móvel. Embora possa parecer mais fácil e eficiente utilizar um único framework para toda a aplicação, existem razões válidas para a escolha de dois frameworks diferentes.

3.4.1 Laravel

Dentre as opções de frameworks da linguagem PHP disponíveis, foi escolhido o Laravel para ser utilizado no presente trabalho. O Laravel é um framework de código aberto que foi criado por Taylor Otwell em 2011 e tem como objetivo poupar o tempo dos desenvolvedores de software, já que o framework traz consigo diversas funcionalidades e recursos que já foram previamente testados e validados (Silva, 2019).

O Laravel segue a arquitetura MVC (*Model View Controller*) que, como o próprio nome já diz, divide o sistema em 3 camadas: 1. A camada de Modelo, que é a responsável por fazer toda a comunicação com o banco de dados do sistema; 2. A camada de Visão, que é responsável por exibir os dados da aplicação através das páginas html; 3. A camada de Controle, que fica responsável por intermediar as requisições dos usuários feitas através da camada da View, com as respostas fornecidas pela camada do Model (Silva, 2019).

O Laravel será utilizado neste projeto como uma ferramenta para desenvolver uma API RESTful, que será acessada pelo aplicativo. Essa escolha se deve à sua capacidade de construir um backend seguro e escalável, proporcionando uma estrutura sólida para a comunicação entre o servidor e o aplicativo móvel.

3.4.2 React Native

Na implementação deste projeto, o React Native será utilizado para construir a interface do usuário do aplicativo, enquanto a comunicação de dados será gerenciada por meio da API RESTful fornecida pelo Laravel.

O React Native foi desenvolvido pelo Facebook em 2015, e o seu maior diferencial é a possibilidade de desenvolver aplicativos móveis multiplataforma (Android e iOS), e diferentemente dos seus concorrentes, o React Native converte o código do aplicativo para a linguagem nativa do sistema operacional, tornando os aplicativos mais ágeis e acessíveis.

A escolha de utilizar dois frameworks diferentes, Laravel e React Native, foi feita pensando na experiência de uso para os usuários finais, além de permitir uma arquitetura mais flexível e escalável para o sistema como um todo.

No decorrer deste trabalho, foram empregadas ferramentas de suporte para a organização do código e a integração do software com o banco de dados. Na próxima seção, serão descritas essas ferramentas, bem como suas vantagens em relação ao projeto em questão.

3.4.3 Visual Studio Code

O Visual Studio Code é um editor de código gratuito e de código aberto, desenvolvido pela Microsoft (2023). Ele possui uma grande capacidade de integração com diversas linguagens de programação, incluindo o PHP, e frameworks como o React Native. Uma das vantagens do Visual Studio Code é a sua ampla variedade de extensões disponíveis, que possibilita a personalização do ambiente de desenvolvimento de acordo com as necessidades de cada projeto.

3.4.4 Notion/Kanban

O sucesso na condução de um projeto está intrinsecamente ligado à transição eficiente dos requisitos para tarefas passíveis de gerenciamento. No âmbito deste trabalho, esse processo é efetuado mediante a conversão dos requisitos identificados em tarefas tangíveis. Cada requisito é analisado e subdividido em unidades executáveis, as quais são então designadas como tarefas específicas. Essas tarefas, por sua vez, são encapsuladas em tickets que, de maneira estruturada, incorporam informações cruciais, como prazos, responsáveis e status de conclusão.

Para garantir a eficiência na gestão dos tickets do projeto em questão, optou-se por utilizar o aplicativo web Notion, que oferece a funcionalidade de ”quadro Kanban”. Ele permite

a criação de listas personalizadas, com cartões que representam as tarefas individuais, e permite que esses cartões sejam movidos entre as listas, indicando assim o andamento das tarefas. Além disso, o Notion possui recursos adicionais como comentários, anexos de arquivos, etiquetas e lembretes para auxiliar na gestão do projeto.

O Notion é uma ferramenta versátil e eficiente, amplamente utilizada para a gestão de projetos por meio de quadros Kanban. Com essa funcionalidade, é possível visualizar o progresso das tarefas, identificar gargalos e equilibrar o trabalho. O Notion oferece flexibilidade na personalização das colunas, além de recursos colaborativos que permitem a atualização em tempo real. Sua integração com outras ferramentas e a possibilidade de adicionar metadados tornam o Notion uma solução abrangente e eficaz para a implementação do Kanban, permitindo uma gestão mais eficiente e organizada dos projetos.

A implementação da metodologia Kanban em conjunto com o SCRUM é uma abordagem estratégica para otimizar o fluxo de trabalho e gerenciar projetos de forma ágil. A combinação dessas metodologias proporciona uma visão abrangente do progresso das tarefas e do andamento do projeto, permitindo uma priorização eficiente das demandas. Essa integração promove uma gestão mais eficiente, aumentando a produtividade e proporcionando entregas consistentes ao longo do projeto.

3.4.5 Nuvemshop

Para a elaboração do logotipo neste projeto, optou-se pela utilização da plataforma *Nuvemshop*, uma ferramenta reconhecida na América Latina por sua capacidade de proporcionar uma experiência robusta para o desenvolvimento e gerenciamento de lojas online (Nuvemshop, 2023). A opção por essa plataforma específica não apenas simplifica o processo de design, mas também contribui para a construção da identidade visual do projeto.

4 Metodologia Adotada para o Desenvolvimento do Software

Para o presente trabalho, foi adotado um modelo de desenvolvimento ágil baseado na metodologia SCRUM, complementado pela técnica de controle de fluxo de trabalho Kanban. Além disso, uma característica fundamental desse modelo é a entrega incremental e iterativa do software. Esse enfoque permite que o desenvolvimento seja dividido em ciclos curtos e frequentes, nos quais as funcionalidades são desenvolvidas, testadas e entregues de forma incremental ao longo do tempo.

A combinação do SCRUM com o Kanban e a entrega incremental e iterativa tornam o modelo de desenvolvimento adotado altamente adaptável, flexível e eficiente, capaz de atender às demandas em constante evolução do projeto. Através da combinação do SCRUM, que oferece uma estrutura para a gestão de projetos, com o Kanban, que proporciona um controle visual do fluxo de trabalho, é possível obter uma visão clara e organizada das tarefas em andamento.

A metodologia Scrum é amplamente utilizada para desenvolvimento de projetos ágeis e é baseada em um ciclo de desenvolvimento iterativo e incremental. No entanto, para se adequar às necessidades específicas do projeto, é possível fazer adaptações no ciclo de desenvolvimento. A fim de garantir a consistência e eficácia do processo, foi utilizado como base o diagrama desenvolvido por Almeida (2017), que apresenta um ciclo de desenvolvimento com etapas similares às adotadas neste projeto.

Com base no diagrama proposto por Almeida (2017), foi elaborado o fluxograma apresentado na figura 3 que descreve o modelo de desenvolvimento e gerenciamento de sistemas adotado, englobando os principais processos. Esse fluxograma proporciona uma visão clara das etapas, desde a definição dos requisitos até a entrega do sistema, abrangendo o planejamento, implementação e testes. Além disso, o fluxograma aborda decisões presentes no decorrer do projeto, como a aceitação dos testes realizados e a conclusão das *sprints*.

4.1 Início

Para iniciar o projeto, é crucial estabelecer claramente os principais *stakeholders* envolvidos, como o *Product Owner*, o *Scrum Master* e a equipe de desenvolvimento. O *Product Owner* será responsável por descrever as necessidades do software para o *Scrum Master* e a equipe de desenvolvimento, a fim de garantir que todos os requisitos/funcionalidades requeridos sejam abrangidos no projeto.

Figura 3 – Diagrama de fluxo para o desenvolvimento e gestão de sistemas

Fonte: O próprio autor

4.2 Levantamento de Requisitos

De acordo com Carvalho et al. (2014) apud Cohn (2004), a História de Usuário é um método de levantamento de requisitos que descreve funcionalmente as necessidades do cliente. Conforme Carvalho et al. (2014) apud Cohn (2004), as conversas desempenham um papel fundamental no processo de levantamento de requisitos, permitindo detalhar as funcionalidades, enquanto os cartões e critérios de aceitação são utilizados para registrar de forma sucinta os requisitos e validar as histórias de usuário.

Para o presente trabalho, as histórias de usuário serão utilizadas como uma abordagem central para a obtenção dos requisitos. Através de entrevistas e interações diretas com o usuário final, ele terá a oportunidade de compartilhar suas experiências, fornecer *feedbacks* e contribuir ativamente na definição das histórias de usuários. Essas histórias servirão como base para definir os requisitos funcionais e não funcionais do projeto, permitindo uma compreensão mais aprofundada das necessidades dos usuários e garantindo que a solução final esteja alinhada com suas expectativas.

4.3 Backlog

Após a coleta inicial dos requisitos, o *Scrum Master* tem a responsabilidade de catalogar e classificar as histórias de usuário, de acordo com a ordem de prioridade estabelecida pelo *Product Owner*, para criar o *Product Backlog*. Esse artefato é essencial durante o planejamento de cada *sprint*, garantindo a coerência dos requisitos coletados e a sua priorização de acordo com a relevância para o negócio. Assim, o *Product Backlog* serve como base para definir as funcionalidades a serem desenvolvidas e implementadas ao longo do projeto, assegurando um alinhamento consistente com os objetivos estabelecidos.

4.4 Planejamento da Sprint

Durante a etapa de planejamento da *Sprint*, o *Scrum Master* e a equipe realizam a seleção de itens do *Product Backlog* e os desdobram em tarefas específicas a serem incluídas no *Sprint Backlog*, levando em consideração a prioridade previamente estabelecida. Esse processo de fragmentação permite uma melhor compreensão e detalhamento das atividades necessárias para o cumprimento das histórias de usuário selecionadas. A criação do *Sprint Backlog* é um passo fundamental para o planejamento e execução da *Sprint*, garantindo a definição clara das tarefas a serem realizadas durante o período determinado.

4.5 Execução e Implementação

Durante a fase de execução e implementação as funcionalidades são desenvolvidas de maneira incremental, seguindo a ordem de prioridades estabelecida no *Sprint Backlog*. Essa abordagem permite que a equipe de desenvolvimento teste a integração da funcionalidade implementada com as funcionalidades das *sprints* anteriores durante a etapa de validação. A adoção desse método incremental garante uma evolução controlada do projeto, possibilitando ajustes e aprimoramentos contínuos ao longo do processo de desenvolvimento.

4.6 Validação e Testes

Ao final de cada *sprint*, é realizado um processo de avaliação para verificar se os objetivos estabelecidos foram alcançados. Se houver falhas ou problemas que impeçam a aceitação plena da implementação, é possível adaptar o *Product Backlog* para incluir correções e melhorias nas próximas *sprints*. Dessa forma, é garantido o aprimoramento contínuo e a entrega de um produto de qualidade.

4.7 Entrega da Sprint

No encerramento de uma *sprint*, é conduzida uma revisão a fim de avaliar o cumprimento dos objetivos estabelecidos e a aceitação das funcionalidades implementadas. Caso surjam problemas, são realizadas as devidas correções para garantir a entrega satisfatória do produto final. Após a aprovação de todos os elementos, são entregues os incrementos do sistema, isto é, as funcionalidades desenvolvidas e implementadas ao longo da *sprint*.

5 Etapas do Desenvolvimento

Neste capítulo, exploraremos as diversas etapas do desenvolvimento do projeto, fornecendo uma visão detalhada do uso da metodologia adotada para o desenvolvimento do sistema proposto. Ao desdobrar cada etapa, destacaremos os processos, estratégias e ferramentas utilizadas para alcançar os objetivos propostos. Esta seção proporcionará informações sobre o fluxo de trabalho, permitindo uma compreensão abrangente do desenvolvimento do projeto.

5.1 Levantamento de requisitos

Para obter os requisitos do sistema, adotou-se uma abordagem colaborativa entre o desenvolvedor e um usuário final. A comunicação para obtenção dos requisitos do sistema ocorreu de forma informal, por meio de discussões e interações diretas com o usuário. Com base nessas conversas, foram identificadas as necessidades e demandas do usuário, as quais foram convertidas em histórias de usuário.

Através dessa interação direta, foi possível obter *insights* valiosos, identificar os principais requisitos funcionais e não funcionais, bem como entender as preferências e prioridades do usuário. Essa abordagem focada em um único *stakeholder* permitiu um processo de levantamento de requisitos mais ágil e eficiente, garantindo que o sistema atendesse às expectativas e necessidades específicas.

5.1.1 Histórias de usuário

Com base nas informações levantadas durante a análise dos requisitos, foram elaboradas histórias de usuário que descrevem de maneira clara e concisa as necessidades e desejos dos usuários. O perfil do usuário considerado para a elaboração das histórias de usuário é o de uma microempreendedora especializada na comercialização de cosméticos, com ênfase na categoria de perfumaria.

Esse processo permite compreender o escopo do projeto e estabelecer a ordem de importância para o desenvolvimento dos itens. Essas histórias de usuário servirão como guia ao longo do desenvolvimento do projeto, delineando os requisitos do sistema. Ao final, essas histórias serão utilizadas no processo de validação, estabelecendo os critérios de aceitação necessários para alcançar uma conclusão e entrega bem-sucedidas.

A seguir, serão detalhadas as histórias de usuário que foram elaboradas a partir das necessidades e desejos dos usuários. Cada história será apresentada com suas respectivas estimativas de tempo necessário para o desenvolvimento.

História: H-01
Como administrador, eu desejo registrar informações relacionadas aos detalhes do meu negócio dentro do aplicativo, incluindo o nome, um código único, a data de criação, se a organização está ativa ou não e os dados de acesso para a minha organização, de forma que essas informações estejam acessíveis no futuro.
Estimativa: 24 horas

História: H-02
Como administrador, eu desejo criar credenciais de acesso únicas para cada usuário do aplicativo da minha organização, incluindo um nome de usuário e uma senha, a fim de garantir que somente eu ou outras pessoas autorizadas por mim possam ter acesso aos dados e funcionalidades do aplicativo.
Estimativa: 8 horas

História: H-03
Como administrador eu gostaria de poder acessar os dados de uma organização, anteriormente, cadastrada, mediante informações de credenciais, como nome de usuário e senha
Estimativa: 4 horas

História: H-04
Como administrador, eu desejo que todos os usuários tenham que registrar endereço de e-mail e uma senha para poder acessar minha organização dentro do aplicativo de maneira segura e exclusiva.
Estimativa: 14 horas

História: H-05
Como gerente, eu desejo visualizar uma representação gráfica dos meus produtos mais vendidos e sua respectiva quantidade, para um determinado período de tempo, de forma a ter uma melhor compreensão das tendências de vendas e tomar decisões estratégicas.
Estimativa: 20 horas

História: H-06
Como gerente eu desejo ter acesso às informações detalhadas sobre as vendas dos meus produtos em um período específico, incluindo dados como nome do produto, quantidade vendida, e valor total das vendas.
Estimativa: 10 horas

História: H-07
Como gerente, eu desejo registrar detalhes sobre os produtos dentro do aplicativo, incluindo informações como o nome, categoria, preço de aquisição, preço de venda e quantidade disponível em estoque, para que esses dados possam ser utilizados como base para gerenciar e realizar as vendas futuras.
Estimativa: 16 horas

História: H-08
Como gerente, eu desejo realizar a atualização do estoque de produtos previamente cadastrados no aplicativo, incluindo a adição de quantidades adicionais para aumentar a disponibilidade dos itens em questão.
Estimativa: 6 horas

História: H-09
Como gerente, eu desejo editar as informações de um produto já cadastrado no aplicativo, incluindo o nome, o grupo, o preço de compra, o preço de venda e a quantidade em estoque, para que esses dados atualizados possam ser utilizados para futuras operações de venda.
Estimativa: 12 horas

História: H-10
Como gerente, eu desejo uma opção para desativar um produto já cadastrado no aplicativo, para que ele não possa mais ser selecionado para venda ou para visualização em relatórios.
Estimativa: 3 horas

História: H-11
Como vendedor, eu desejo ter acesso a uma lista completa de todos os produtos cadastrados no aplicativo, incluindo suas informações, como nome, grupo, preço de compra, preço de venda e quantidade em estoque, para que eu possa ter uma visão geral dos itens disponíveis para venda.
Estimativa: 14 horas

História: H-12
Como vendedor, desejo adicionar produtos a um carrinho de vendas no aplicativo, para que eu possa selecionar os itens desejados e realizar a venda de forma organizada e facilitada.
Estimativa: 16 horas

História: H-13
Como vendedor, eu desejo escolher um produto já cadastrado no aplicativo e adicioná-lo ao meu carrinho de vendas para realizar a compra posteriormente.
Estimativa: 2 horas

História: H-14
Como vendedor, eu desejo excluir um item selecionado do meu carrinho de vendas antes de finalizar a minha compra.
Estimativa: 2 horas

História: H-15
Como vendedor, eu gostaria de ter a capacidade de finalizar uma venda utilizando os produtos selecionados no meu carrinho de vendas, incluindo todos os detalhes da transação, como valor total, itens vendidos e informações de pagamento.
Estimativa: 18 horas

História: H-16
Como gerente, quero poder registrar um novo cliente no sistema, para que eu possa armazenar informações detalhadas sobre o cliente, como nome, endereço, número de telefone, e-mail, e quaisquer detalhes adicionais.
Estimativa: 18 horas

História: H-17
Como gerente, quero visualizar os detalhes completos de um cliente, para que eu possa entender melhor o histórico do cliente, seu comportamento de compra e quaisquer notas ou registros relevantes.
Estimativa: 4 horas

História: H-18
Como vendedor, quero poder pesquisar clientes pelo nome, para que eu possa localizar clientes rapidamente e acessar suas informações sem problemas.
Estimativa: 8 horas

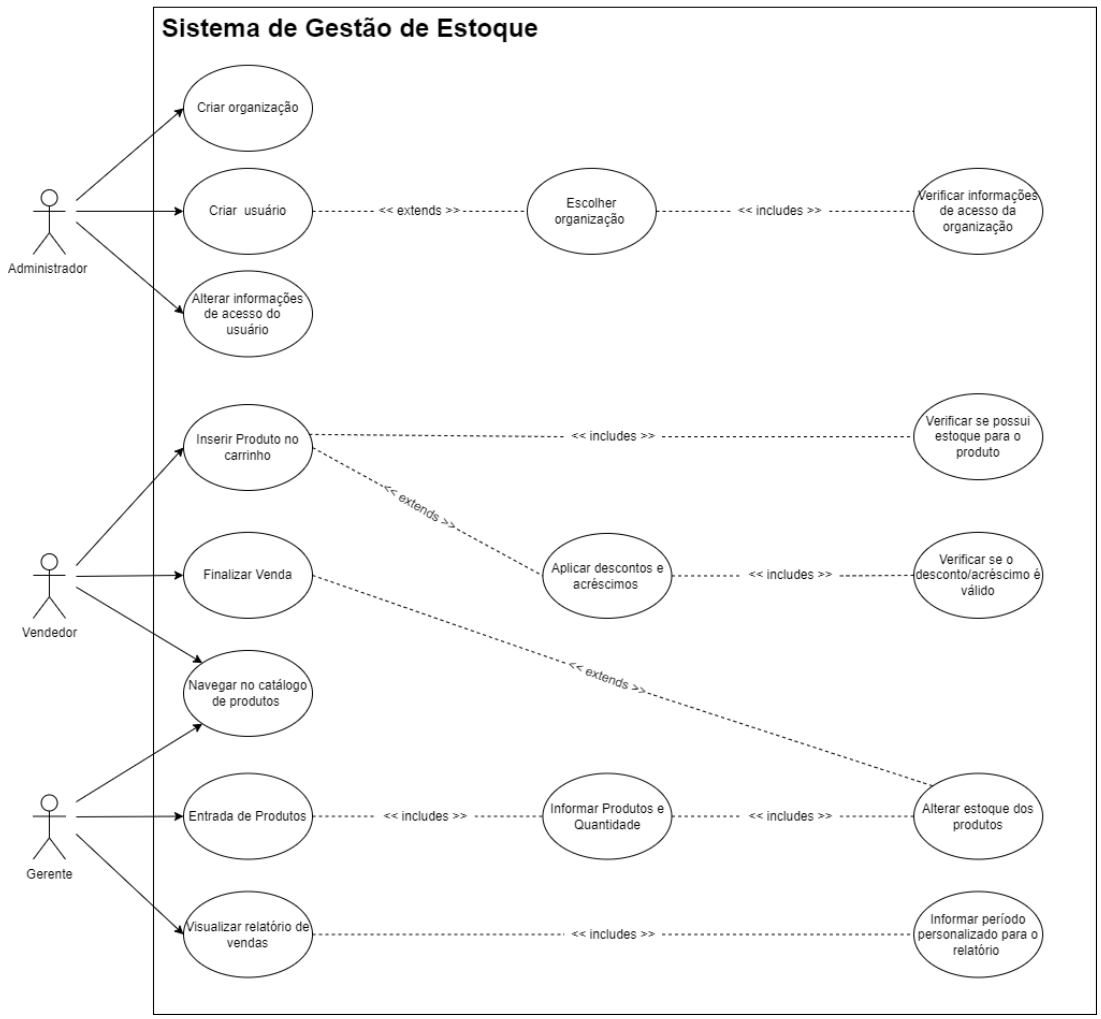
História: H-19
Como gerente, desejo poder atualizar os detalhes de um cliente, para que eu possa manter os registros de clientes sempre atualizados com as informações mais recentes.
Estimativa: 8 horas

História: H-20
Como gerente, desejo poder remover um cliente da base de dados, para que eu possa manter o sistema livre de registros obsoletos ou duplicados.
Estimativa: 4 horas

História: H-21
Como gerente, desejo poder cadastrar produtos que serão utilizados para a produção de outros produtos, para que eu possa gerenciar o meu estoque controlando os insumos necessários para a produção e garantindo a disponibilidade de materiais conforme a demanda.
Estimativa: 6 horas

Com base nas histórias de usuário, foi elaborado um diagrama UML comportamental apresentado na figura 4, representando os atores envolvidos no sistema, as funcionalidades requisitadas por cada ator e as interações entre elas. Esse diagrama de casos de uso auxilia na compreensão do funcionamento do aplicativo e na identificação das necessidades e requisitos do sistema, fornecendo uma visão clara das partes envolvidas e das ações possíveis.

Figura 4 – Diagrama de Casos de Uso



Fonte: O próprio autor

5.2 Backlog

Com base nas histórias de usuário, elaborou-se um backlog que compreende as tarefas a serem executadas ao longo do projeto, utilizando o método MoSCoW. Este método facilita a priorização das tarefas, considerando sua relevância e impacto no projeto, assegurando a categorização dos requisitos de acordo com sua importância.

O método MoSCoW, conforme descrito por Gasparetto and Canal (2023), organiza os requisitos em quatro categorias distintas. Na categoria *"Must have"* (deve ter), estão os requisitos de alta prioridade, ou seja, as funcionalidades essenciais que precisam estar presentes para que o produto seja entregue ao cliente. A categoria *"Should have"* (deveria ter) engloba funcionalidades importantes, porém não vitais, que podem ser negociadas e implementadas após as prioridades *"Must have"*. Em "*"Could have"* (poderia ter), são colocadas as funcionalidades desejáveis, mas não essenciais. Por fim, *"Won't have"* (não vai ter) abrange requisitos inviáveis no momento, seja por questões financeiras ou estratégicas.

Por meio da classificação MoSCoW, estabelecemos diretrizes definidas para o desenvolvimento do projeto. Abaixo, apresentamos uma lista de tarefas organizadas com base nessa classificação, para guiar a implementação:

- **Must-Have** (Deve Ter): H-01, H-02, H-03, H-04, H-05, H-13, H-15, H-16, H-19, H-20
- **Should-Have** (Deveria Ter): H-06, H-08, H-12, H-14, H-18, H-21
- **Could-Have** (Poderia Ter): H-07, H-09, H-10, H-11, H-17
- **Won't-Have** (Não Deve Ter): Nenhuma história foi identificada como neste grupo.

Na Figura 5, é apresentado um quadro Kanban que abrange as histórias de usuário H-01 até H-05 e detalha o processo de desenvolvimento das tarefas relacionadas ao cadastro e login de um cliente. No contexto deste projeto, foram aplicadas cinco colunas distintas no quadro: 'Backlog', 'A fazer', 'Em andamento', 'Aguardando Feedback' e 'Concluído', que representam os diferentes estágios do processo.

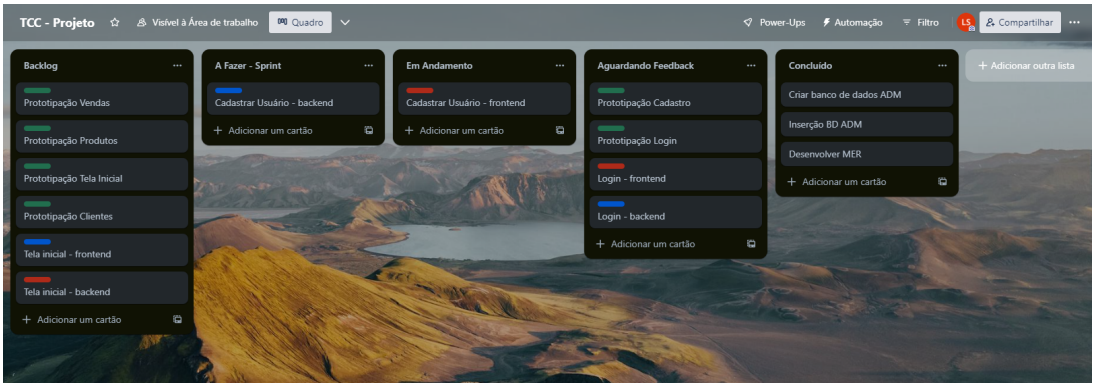


Figura 5 – Quadro *Kanban* com o progresso das atividades

Vale destacar que a configuração do quadro Kanban, conforme apresentado na Figura 5, será mantida ao longo de todas as próximas *sprints* deste projeto. Isso garante a consistência e uniformidade na abordagem durante todo o processo de desenvolvimento, o qual será detalhado nos planejamentos, execuções e validações das sprints nas seções subsequentes.

5.3 Sprints

Nessa seção, serão analisadas em detalhes as fases do desenvolvimento do software, abrangendo o planejamento, a execução e a validação de cada *sprint*. Este enfoque visa oferecer uma compreensão aprofundada do ciclo de desenvolvimento, destacando as estratégias específicas adotadas em cada etapa. A Tabela 1 apresenta a relação entre as *sprints* e as histórias de usuário, proporcionando uma visão clara da implementação progressiva do sistema em resposta aos requisitos levantados.

Tabela 1 – Histórias de Usuário por Sprint

Sprint	Histórias de Usuário (H)
01	Não teve foco em nenhuma história específica
02	Não teve foco em nenhuma história específica
03	H-01, H-04
04	H-02, H-03
05	H-05
06	H-06, H-07, H-08, H-09, H-10, H-11, H-21
07	H-16, H-19, H-20
08	H-12
09	H-13, H-14, H-15, H-18
10	H-17
11	Não teve foco em nenhuma história específica

Fonte: O próprio autor

Iniciamos o desenvolvimento do projeto após apresentar o diagrama de caso de uso, que ilustra as interações dos usuários com o sistema. A partir deste ponto, adotamos um enfoque abrangente, incorporando histórias de usuário em conjunto com o diagrama de casos de uso para conduzir todas as *Sprints*. Este procedimento abrange as etapas de planejamento, execução, implementação e a fase de validação e testes. Essa abordagem é aplicada de maneira integral durante todo o ciclo de desenvolvimento do projeto.

5.3.1 Sprint 01 - Banco de Dados

Está *Sprint* tem como foco principal estabelecer a base de dados do projeto, criando o Modelo de Entidade-Relacionamento (MER) e implementando-o no sistema de gerenciamento de banco de dados PostgreSQL. Essa etapa desempenha um papel fundamental na estruturação eficaz do sistema e na organização dos dados a serem manipulados, ela teve duração de duas semanas.

Durante essa *Sprint*, uma série de tarefas e atividades específicas foram conduzidas para atingir os objetivos estabelecidos. Primeiramente, houve uma análise detalhada dos requisitos do sistema, com o propósito de identificar todas as entidades, relacionamentos e atributos necessários para a criação do MER. Em seguida, foi utilizada a ferramenta de modelagem de banco de dados *Lucidchart* para desenvolver um MER completo que represente com precisão todas as informações e interações do sistema.

Com base no MER criado, a próxima etapa envolveu a criação do banco de dados. Neste processo, tabelas, relacionamentos, chaves primárias e estrangeiras foram definidos de acordo com o modelo estabelecido. Por fim, a implementação do banco de dados PostgreSQL foi integrada ao sistema em desenvolvimento. Foram utilizadas as funcionalidades de *Object-Relational Mapping* (ORM) fornecidas pelo Laravel para realizar o acesso ao banco de dados, permitindo uma interação fluida entre o sistema e a base de dados.

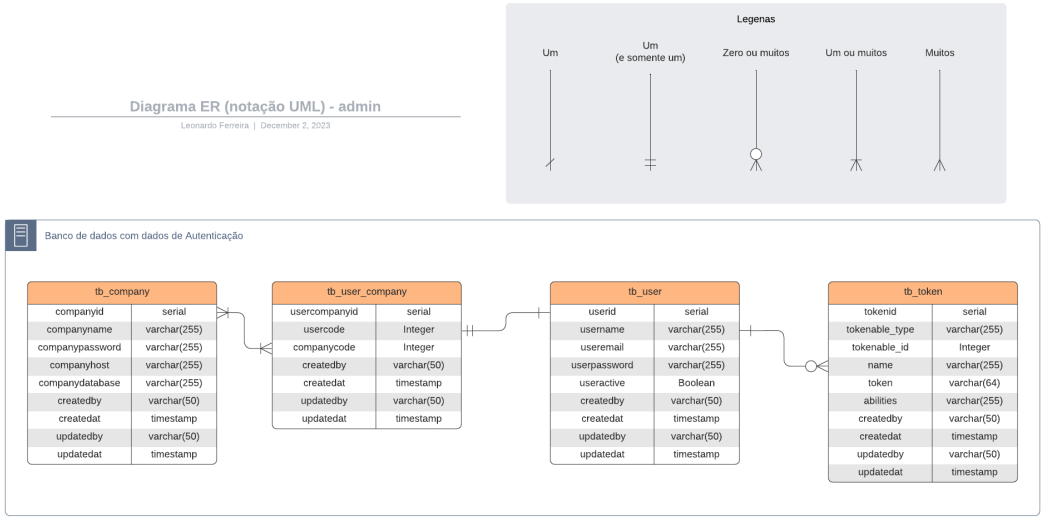
Os principais resultados entregues nesta *Sprint* compreendem o Modelo de Entidade-Relacionamento (MER) e o Banco de Dados PostgreSQL, que foi organizado conforme as diretrizes do MER. Adicionalmente, os módulos responsáveis pelo acesso ao banco de dados foram desenvolvidos e integrados ao sistema, assegurando que nas próximas etapas do desenvolvimento o sistema esteja adequado para executar as operações de inserção, leitura, edição e exclusão de registros no banco de dados.

Nos parâmetros desta Sprint, é imprescindível atingir a criação bem-sucedida do banco de dados PostgreSQL, juntamente com a funcionalidade correta dos módulos de acesso ao banco de dados integrados ao *framework* Laravel. Esta etapa desempenhou um papel crítico no progresso contínuo do projeto, uma vez que estabelece a base de dados fundamental para o desenvolvimento das funcionalidades subsequentes.

Dentro do contexto deste projeto, foi elaborado um Diagrama de Entidade-Relacionamento (DER) para modelar os dados do sistema. A notação adotada para o DER segue a convenção de 'notação pé-de-galinha', conforme definido em Lucidchart (2023). Esse diagrama ajuda a entender como os dados gerenciados pelo sistema se relacionam, como ilustrado nas Figuras 6 e 7.

Neste sistema, usamos uma abordagem de gerenciamento com duas bases de dados diferentes. A primeira base de dados, como mostrado na Figura 6, contém informações administrativas e de configuração vitais para o funcionamento do sistema, como o nome da organização e detalhes de acesso ao banco de dados, que podem ser armazenados em servidores diferentes. Isolando as informações administrativas, garantimos que apenas pessoas autorizadas possam acessar esses dados confidenciais.

Figura 6 – Diagrama entidade relacionamento para dados administrativos



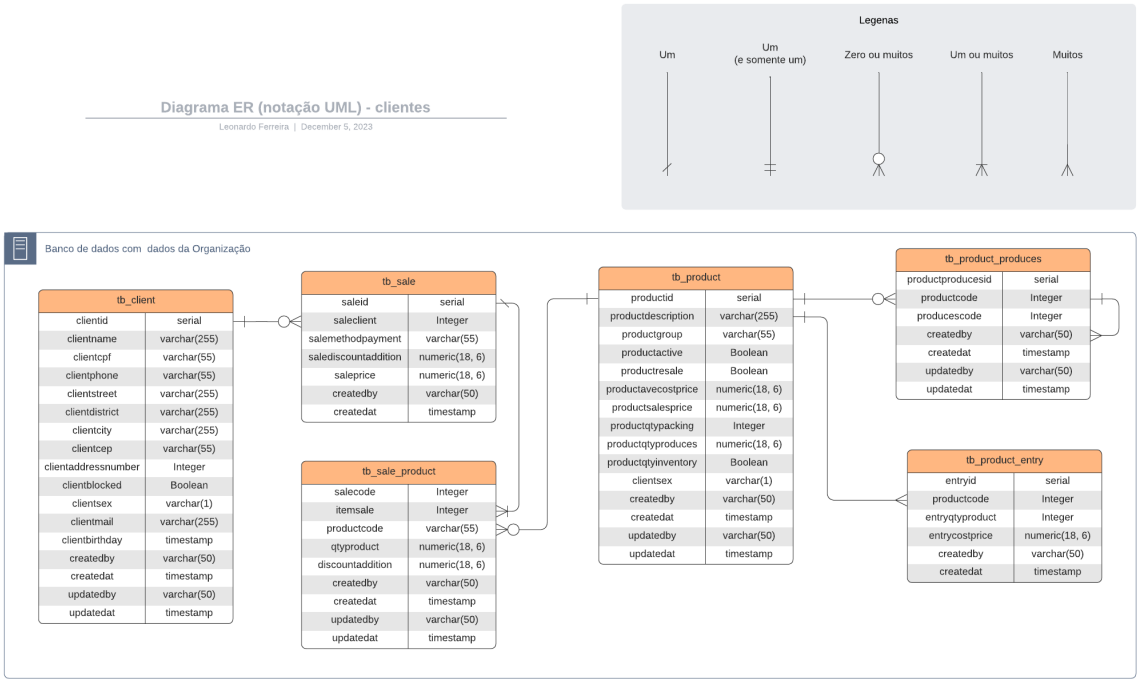
Fonte: O próprio autor

Neste banco de dados, a tabela "tb_company"armazena informações da organização,

incluindo dados de acesso ao banco de dados, como servidor e nome do banco. Por sua vez, a tabela "tb_user"registra dados de acesso dos usuários, como login e senha. Para estabelecer a relação entre usuários e organizações, utiliza-se a tabela "tb_user_company". Por fim, a tabela "tb_token"registra dados da sessão do usuário. Ao realizar o login, é gerado um registro de token nessa tabela, permitindo a verificação de usuários válidos.

A segunda base de dados, representada na Figura 7, é designada para armazenar os dados operacionais da organização, incluindo registros de vendas, modalidades de pagamento, catálogo de produtos, níveis de estoque e outros elementos pertinentes. Esse arranjo possibilita alocar o banco de dados de cada novo cliente em servidores independentes. Consequentemente, em situações de sobrecarga ou interrupção no tráfego de dados de um servidor, novos clientes podem ser direcionados para servidores adicionais, otimizando o balanceamento de carga.

Figura 7 – Diagrama entidade relacionamento para dados de clientes



Fonte: O próprio autor

Este banco de dados é composto por três tabelas principais e suas tabelas auxiliares. As principais incluem "tb_cliente,"destinada a armazenar informações sobre clientes, "tb_product,"para dados de produtos, e "tb_sale,"voltada para informações de vendas. A tabela "tb_sale_product"estabelece a relação entre as tabelas de vendas e produtos, detalhando quais produtos foram utilizados em cada transação. Os registros de entrada de mercadorias são mantidos na tabela "tb_product_entry", enquanto produtos derivados de outros produtos são registrados na tabela "tb_product_produces".

5.3.2 Sprint 02 - Prototipação

Nesta *Sprint*, que se estendeu ao longo de três semanas, foi dada ênfase na etapa inicial do desenvolvimento do aplicativo, que se concentrou na criação de um protótipo visual do sistema.

A primeira etapa desta *Sprint* foi dedicada à concepção das telas do aplicativo. Nesse processo, foram desenvolvidos *wireframes* e *mockups* que representavam as principais interfaces do aplicativo. Essas representações visuais abrangeram desde a página inicial até as telas de login, o painel principal, formulários de cadastro, páginas de produtos entre outras. Esse esforço permitiu uma compreensão clara de como o aplicativo seria estruturado e como os usuários interagiriam com ele.

Nesta Sprint, ocorreu a criação de um protótipo visual completo do aplicativo, abrangendo todas as telas principais e suas interações. Esse protótipo serviu como referência nas etapas seguintes do desenvolvimento, proporcionando uma organização dos dados e uma base sólida para a implementação do sistema. Além de antecipar a visualização do produto, essa fase possibilitou a validação das decisões de design, assegurando que o aplicativo atendesse plenamente aos requisitos de usabilidade e experiência do usuário, direcionando o desenvolvimento subsequente em direção a um aplicativo eficiente e intuitivo.

De modo geral, o aplicativo começa com uma tela de login para garantir a segurança e permite que novos usuários se cadastrem. Após o login, os usuários acessam uma tela inicial com gráficos que mostram informações importantes sobre as vendas, ajudando a entender o desempenho do negócio. O aplicativo também oferece telas para gerenciar produtos e clientes, onde você pode adicionar, editar ou remover informações. Na tela de vendas, é possível escolher produtos, colocá-los em um carrinho de vendas, selecionar o cliente e finalizar as transações.

A tela de login e cadastro do aplicativo é a primeira a ser exibida, apresentando dois botões distintos: "Entrar"e "Cadastrar", conduzindo os usuários de forma clara e direta às suas respectivas ações, conforme apresentado na Figura 8. Ao selecionar o botão "Entrar", uma caixa de diálogo é exibida na parte inferior da tela, possibilitando que os usuários insiram com facilidade seu endereço de email e senha, conforme apresentado na Figura 8b.

O botão "Cadastrar"direciona os usuários para a criação de uma nova conta. Novamente, conforme apresentado na figura 8c, uma caixa de diálogo se abre, solicitando informações vitais, como nome, endereço de email e senha. Além disso, os usuários têm a opção de especificar se estão criando uma nova organização ou se estão se associando a uma já existente, e para isso é necessário fornecer um nome de usuário e senha para acessar a organização.

A tela inicial do aplicativo, conforme apresentado na Figura 9, oferece uma visão abrangente e informativa do sistema para os usuários. No cabeçalho, estão disponíveis opções para acessar as configurações, sair do aplicativo e selecionar um período de tempo para filtrar as informações exibidas. No corpo da tela, são apresentadas caixas com dados de vendas em

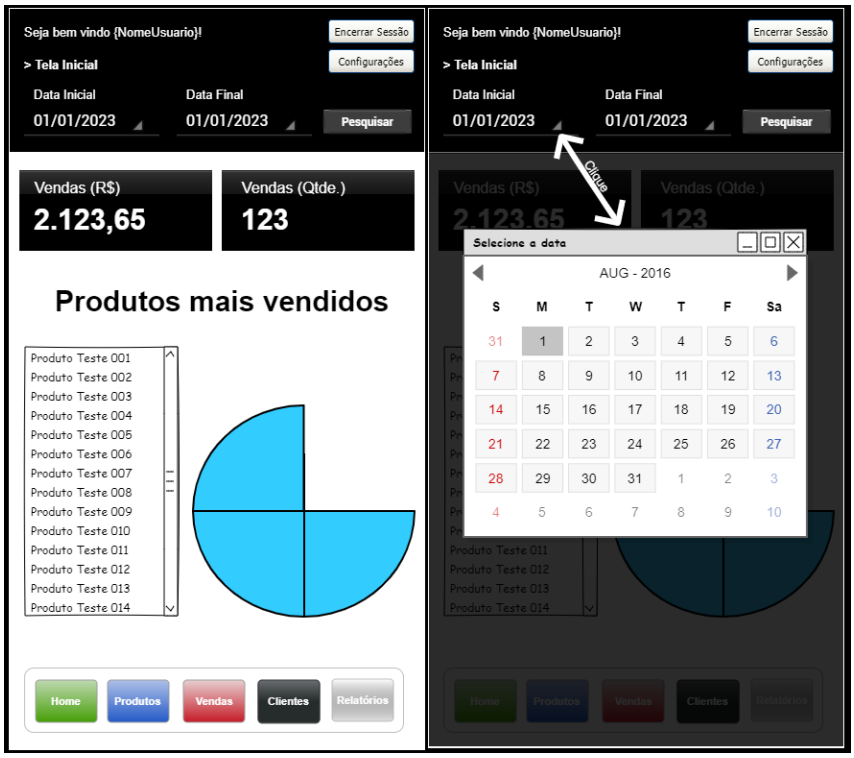
Figura 8 – Protótipo criado contendo as telas de login e cadastro

(a) Protótipo login (b) Protótipo entrar (c) Protótipo cadastrar (d) Carregando dados

Fonte: O próprio autor

dinheiro e em quantidade, bem como informações sobre o lucro e o *ticket* médio. Destaca-se a presença de um gráfico circular que ilustra os produtos mais vendidos, proporcionando uma visualização rápida e intuitiva.

Figura 9 – Protótipo criado contendo a tela inicial do aplicativo



Fonte: O próprio autor

No rodapé da tela, estão localizadas as opções de navegação, que estão presentes em

todas as telas de cadastro do sistema. Essa abordagem facilita a navegação e a interação dos usuários em todo o aplicativo, tornando a experiência mais fluida e intuitiva. A tela inicial oferece, assim, um painel central para os usuários acessarem informações cruciais sobre as vendas e operações da organização.

5.3.3 Sprint 03 - Cadastro e autenticação de usuário

Esta *sprint* teve duração de 2 semanas, e o foco principal foi o desenvolvimento das funcionalidades relacionadas ao processo de cadastro de novos usuários e a implementação da autenticação no sistema, contemplando as histórias de usuário H-01 e H-04. Cada organização terá acesso a um banco de dados exclusivo para o seu negócio, no qual poderá armazenar suas informações com segurança. Além disso, haverá a opção de acesso a bancos de dados de outras organizações, mediante a inserção correta do nome de usuário e senha da mesma. Isso assegura tanto o isolamento quanto a segurança dos dados.

Para desenvolver essa *sprint*, foi usada a ferramenta *Eloquent*, uma ORM (*Object Relational Mapping* - Mapeamento Objeto-Relacional) nativa do *framework* Laravel. Essa escolha se fundamentou na capacidade dessa ferramenta de simplificar as operações de interação com o banco de dados, ao aplicar conceitos do paradigma de orientação a objetos ao ambiente de banco de dados relacional.

A configuração da ferramenta *Eloquent* envolveu a definição das informações de acesso ao banco de dados no arquivo de configuração responsável por gerenciar as variáveis de ambiente. No contexto específico deste trabalho, as informações acessadas pelo sistema foram separadas em dois blocos distintos. Um deles diz respeito às informações administrativas de acesso ao sistema, enquanto o outro bloco foi destinado ao armazenamento das informações específicas de cada cliente.

A arquitetura do sistema desenvolvido segue um modelo no qual cada cliente dispõe de um banco de dados exclusivo para o armazenamento de seus registros de vendas e controle de estoque. Essa abordagem proporciona a flexibilidade necessária para distribuir bancos de dados em diferentes servidores, o que resulta em uma alocação eficiente de recursos e possibilita a escalabilidade do sistema.

Com o intuito de garantir um eficaz controle de acessos, foi implementado um banco de dados administrativo contendo informações de todas as organizações e clientes registrados. Nesse banco de dados central, são mantidos dados de usuários, como endereços de email e senhas, que são armazenadas após passarem por um processo de criptografia utilizando o algoritmo bcrypt. Essa abordagem assegura que mesmo em situações de exposição de dados, as senhas dos usuários permanecem protegidas, pois não podem ser revertidas para sua forma original, oferecendo um alto nível de segurança e privacidade aos usuários do sistema.

Para facilitar o acesso ao banco de dados, foi desenvolvida uma função incorporada

a cada um dos *Models* do sistema, com a finalidade de receber o código do usuário e através dele buscar os dados de acesso para estabelecer a conexão com o banco de dados específico. Essa abordagem assegura que, em cada requisição do sistema, a conexão seja estabelecida com o banco de dados específico de cada organização. A Figura 10 apresenta o código em JavaScript com a função desenvolvida para estabelecer uma conexão com o banco de dados específico de cada organização.

Figura 10 – Função responsável pela conexão ao banco de dados

```
36  public static function connect($UserID){
37    $DBdata = User_CompanyTB::select('companyhost', 'companydatabase')
38      ->join('companytb', function($join){$join->on('companycode', '=', 'companyid');})
39      ->where('usercode', $UserID)
40    ->first();
41
42    \Config::set('database.connections.pgsqlclient.host', $DBdata['companyhost']);
43    \Config::set('database.connections.pgsqlclient.database', $DBdata['companydatabase']);
44
45    \DB::purge('pgsqlclient');
46    return (new static)->setConnection('pgsqlclient');
47  }
```

Fonte: O próprio autor

Toda vez que o usuário faz alguma requisição ao sistema, o código do cliente é encaminhado ao *Model*, dessa forma, na resposta sempre constará apenas dados relacionados a empresa do cliente em questão. A Figura 11 apresenta o código de uma função para atender uma requisição feita pelo usuário para acessar os dados dos clientes, que exemplifica a situação descrita anteriormente.

Figura 11 – Exemplo de requisição para a busca de dados no banco

```
11  public function getClients(Request $request){
12    date_default_timezone_set('America/Sao_Paulo');
13    $userCode = (\Request::header('usercode') ?? -1);
14    $clientBlocked = (\Request::header('blocked') ?? false);
15    $clientName = (\Request::header('name') ?? '');
16
17    return ClientsTB::GetClients($userCode, $clientBlocked, $clientName);
18  }
```

Fonte: O próprio autor

No decorrer deste projeto, o padrão estabelecido para as requisições será mantido consistentemente ao longo de todo o desenvolvimento. Essa abordagem padronizada tem por objetivo garantir uma operação uniforme nas interações do sistema com o banco de dados, independentemente da complexidade das operações ou da natureza das informações acessadas. Essa padronização proporciona maior clareza e eficiência ao processo, permitindo uma compreensão contínua das operações e promovendo a integração consistente entre os componentes do sistema. Portanto, a escolha de manter essa uniformidade em todas as etapas do projeto é uma estratégia que visa garantir a qualidade e o sucesso da implementação do sistema.

5.3.4 Sprint 04 - Criação da tela de login

Essa *sprint* foca no desenvolvimento da interface destinada às funcionalidades de login e cadastro de usuários e teve duração de duas semanas. Inicialmente a prioridade foi a implementação da interface para essas funcionalidades, o que incluiu a criação das telas de login e cadastro do usuário, bem como a definição das interações do usuário com o sistema. Posteriormente o foco da *sprint* se voltou para a implementação das ações do usuário, como o preenchimento dos campos da tela de login e cadastro de usuário, bem como a validação da interface de entrada de dados.

No contexto deste trabalho, foi empregado um conceito essencial do desenvolvimento em *React Native*, conhecido como "helper". As *helpers* consistem em funções ou módulos auxiliares criados com o propósito de simplificar e encapsular lógicas específicas, a fim de promover a organização, reutilização e manutenção do código. É importante ressaltar que as *helpers*, por sua natureza, são independentes dos componentes de interface do usuário, permitindo que sejam importadas e utilizadas em diversos pontos do aplicativo.

Neste trabalho, foi adotado um conjunto de funções *helpers* destinadas a efetuar requisições ao servidor. Essa abordagem se justifica pelo fato de que a realização de requisições ao servidor se faz presente em praticamente todas as telas do aplicativo. Ao adotar essas funções *helpers*, nosso objetivo principal é evitar a redundância de código por meio da reutilização, promovendo a eficiência e a manutenibilidade do projeto. Na Figura 12, encontra-se o código de uma dessas funções, a qual utiliza a funcionalidade "*fetch*"para executar as requisições.

Logo a seguir, demonstraremos a aplicação prática de uma das funções discutidas anteriormente. Na Figura 13, apresentamos um exemplo concreto da utilização dessa função. Esse exemplo ilustra como a função auxilia na realização de requisições ao servidor, simplificando a implementação e contribuindo para a funcionalidade do aplicativo.

Ao final dessa sprint, a interface de login e cadastro de usuários estavam funcionais e prontas para uso. A partir da conclusão dessa etapa, os usuários podem interagir com o sistema, abrindo caminho para o desenvolvimento das próximas etapas do projeto.

5.3.5 Sprint 05 - Criação da tela inicial

A tela inicial é a primeira tela que o usuário acessa após a realização do login. Durante essa *sprint*, houve o desenvolvimento simultâneo do *frontend* e do *backend* com o objetivo de fornecer aos usuários uma visão ampla e rica em informações. Essa tela é composta por vários elementos, entre eles um gráfico de pizza que representa o volume de venda dos produtos. Para implementar essa funcionalidade, foi adotado o uso da biblioteca *Victory Native*, que é especializada em gráficos. Essa abordagem não apenas agilizou o processo de desenvolvimento, economizando tempo significativo, como também proporcionou uma solução altamente informativa e interativa aos usuários.

Figura 12 – Função utilizada para realizar as requisições ao servidor

app > components > Helper.js > requestAPI

```
function requestAPI(URLreq, Params = {}){
  var headerParams = {"Authorization": global.userData.tokenAPI};
  Object.keys(Params).forEach(key => {
    if (Params[key] !== undefined){
      headerParams[key] = Params[key];
    }
  });

  if (global.userData.id > 0){
    headerParams['usercode'] = global.userData.id;
  }

  return fetch(URLreq, {method: 'GET', headers: headerParams}).then(response => {
    if ((response.status !== 200) && (response.status !== 201)){
      return {cod: 501, data: {message: 'Erro ao validar token. Por favor, faça o login novamente.'}};
    }
    return response.json();
  }).then(responseData => { return responseData; }).then(data => {
    if (!data){
      return {cod: 501, data: {message: 'Erro ao validar token. Por favor, faça o login novamente.'}};
    }
    return data;
  }).catch(err => {
    console.log(err);
  });
}

function requestWithBodyAPI(URLreq, bodyJSON, method = 'POST', Params = {}){ ···
}

export { requestAPI, requestWithBodyAPI }
```

Fonte: O próprio autor

Figura 13 – Exemplo da chamada da função de requisição ao servidor

```
componentDidMount(){
  requestAPI(global.URLreq+'listdbs').then(data => {
    this.setState({listOrg: data});
  });
}
```

Fonte: O próprio autor

Na tela inicial do aplicativo, bem como nas demais telas, garantimos a autenticação dos usuários por meio do *middleware* "*auth:sanctum*" do *Laravel*. Esse *middleware* atua como um intermediário nas solicitações HTTP, intervindo no processo de requisição antes de atingir os controladores do aplicativo. Ele é responsável por verificar a autenticidade do *token* de autenticação fornecido pelo cliente, gerado durante o processo de autenticação do usuário. Quando uma solicitação é direcionada ao servidor, o *middleware* verifica a validade do *token* de autenticação apresentado, certificando-se de que corresponda a um usuário que tenha realizado com sucesso o processo de autenticação.

Esse mecanismo garante a segurança das operações, pois permite a execução de ações específicas apenas por usuários autenticados, o que protege os dados e recursos contra acessos não autorizados.

5.3.6 Sprint 06 - Criação da tela de Produtos

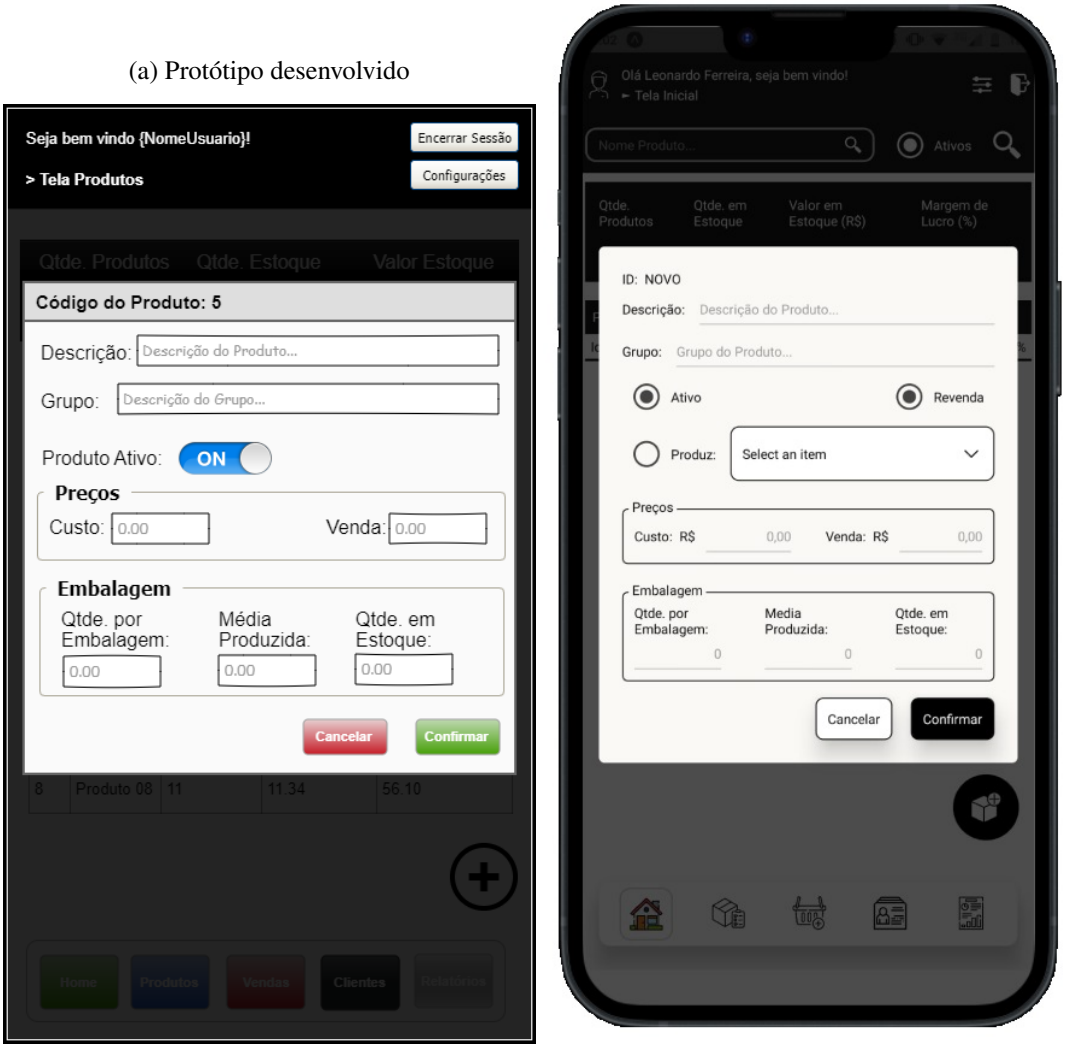
A *sprint* 6, teve duração de duas semanas, com foco na implementação das seguintes histórias de usuário: H-06 até H-11 e H-21. Ela resultou em uma ampla gama de recursos e aprimoramentos relacionados à gestão de produtos no aplicativo. Este processo foi caracterizado pela integração completa entre a interface do usuário e o sistema de gerenciamento de banco de dados. A tela de produtos foi integrada ao fluxo do aplicativo, buscando trazer aos usuários uma experiência de uso natural e consistente em relação a outras funcionalidades do sistema.

Como é comum em projetos de desenvolvimento de software, a tela de produtos passou por uma evolução significativa desde seu estágio inicial até sua versão final. A Figura 14 proporciona uma visão real de como essa tela evoluiu e se adaptou ao longo do projeto.

Figura 14 – Tela de inclusão e alteração de produtos

(a) Protótipo desenvolvido

(b) Tela do aplicativo



Fonte: O próprio autor

No protótipo inicial, mostrado na Figura 14a, podemos observar que o formulário de inclusão e edição de dados do produto contém um conjunto menor de campos conforme planejado

originalmente. Entretanto, na Figura 14b, que corresponde à versão final da tela, notamos a adição de novos campos e algumas pequenas modificações de interface. Essas alterações visam atender a requisitos que surgiram durante o processo de desenvolvimento, o que é uma situação comum na construção de aplicativos.

Essas divergências entre o protótipo e a versão final não apenas demonstram a capacidade de adaptação do modelo de desenvolvimento, mas também representam características de uma abordagem ágil, ressaltando a flexibilidade e adaptabilidade que a metodologia de desenvolvimento empregada fornece para lidar com as necessidades em evolução do projeto. Essas imagens fornecem um exemplo prático de como o processo de desenvolvimento de software pode ser dinâmico e orientado pelas demandas do usuário.

É importante ressaltar que as funcionalidades criadas para a tela de produtos desempenharam um papel de destaque como modelo para a implementação das outras partes do sistema. Isso ocorre porque as demais telas compartilham funcionalidades similares às que foram desenvolvidas nesta *sprint*. Essa abordagem traduz-se em eficiência e consistência no desenvolvimento do aplicativo, resultando em uma progressão mais harmoniosa e organizada.

5.3.7 Sprint 07 - Criação da Tela de Clientes

Nessa sprint, que teve uma duração de duas semanas, foi desenvolvida a tela de gerenciamento de clientes. A interface manteve o mesmo padrão de *layout* adotado na tela de produtos, com adaptações nos campos de acordo com as necessidades descritas nas histórias H-16, H-19 e H-20.

Vale destacar que o uso do *middleware* "*auth:sanctum*", conforme previamente descrito nas *sprints* anteriores, auxilia na preservação da integridade e confidencialidade dos dados sensíveis dos clientes, permitindo que apenas usuários autorizados acessem essas informações.

5.3.8 Sprint 08 - Criação da tela de Vendas

Durante esta *sprint*, que se estendeu por duas semanas, o foco principal recaiu sobre o desenvolvimento da tela de vendas, uma demanda originada da história de usuário H-12, que almejava a disponibilização de dados de vendas e a capacidade de filtrar esses dados com base em datas. Para tal, incorporamos elementos da tela inicial com o propósito de disponibilizar tais informações e permitir a filtragem desses dados, de modo a atender os requisitos do sistema. A tela de vendas foi projetada de maneira a exibir uma lista de vendas não finalizadas, as quais são salvas no banco de dados como orçamentos. Seguindo a abordagem padrão adotada nas telas de produtos e clientes, essa tela apresenta um botão projetado para a criação de novas vendas.

É importante ressaltar que a implementação da tela para criação de novas vendas somente será implementada em *sprints* futuras. Essa situação gerou um desafio para esta *sprint*, pois como a funcionalidade para efetuar vendas ainda não estava plenamente operacional, e era

necessário testar a funcionalidade de conversão de orçamentos em vendas, então foram utilizados dados estáticos para povoar a tabela de vendas. Tal abordagem foi adotada temporariamente, permitindo a continuidade do desenvolvimento.

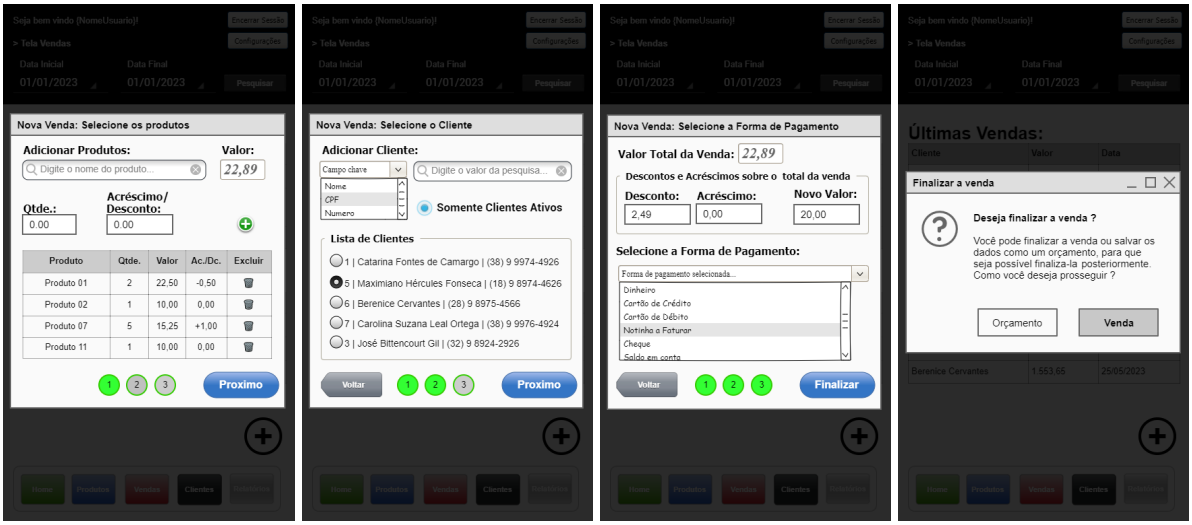
5.3.9 Sprint 09 - Criação do Carrinho de Vendas

A *sprint* 9, que abrangeu um período de três semanas, foi responsável pelo desenvolvimento das funcionalidades relacionadas ao processo de vendas, implementando as funcionalidades solicitadas nas histórias de usuário H-13 até H-15 e H-18, com o foco na funcionalidade de registrar uma nova venda no sistema.

A Figura 15 exibe uma sequência de quatro imagens ilustrativas do procedimento de realizar uma venda no sistema, evidenciando as etapas de adicionar produtos, escolher o cliente e selecionar o método de pagamento. Através do uso desse protótipo, conseguimos verificar se a interface escolhida atende às necessidades dos usuários sem precisar investir tempo no desenvolvimento completo. Dessa forma é possível garantir que as funcionalidades solicitadas pelos usuários fossem integradas corretamente no sistema.

Figura 15 – Sequência com os protótipos das telas para a criação de nova venda

(a) Adicionar produtos ao carrinho (b) Selecionar cliente da venda (c) Selecionar forma de pagamento (d) Tela de finalização de venda



Fonte: O próprio autor

No carrinho de vendas, os usuários têm a possibilidade de escolher o cliente para a venda, podendo selecioná-lo a partir de uma lista que apresenta todos os clientes disponíveis. Da mesma maneira, em relação aos produtos, os usuários podem escolher os itens desejados, definir as quantidades e aplicar descontos ou acréscimos específicos por produto. A lista dos produtos adicionados à venda é visível na tela, permitindo a exclusão de itens conforme necessário.

A forma de pagamento utilizada na venda é uma escolha que os usuários podem

fazer a partir de uma lista predefinida, garantindo uma padronização e facilitando a operação. Adicionalmente, o sistema possibilita a aplicação de descontos ou acréscimos no valor total da venda, oferecendo flexibilidade na gestão de preços.

Um recurso adotado nessa etapa de desenvolvimento foi a implementação de transações no banco de dados. A adoção desse mecanismo estabelece que todas as operações dentro de uma transação devem ser concluídas com êxito para que as alterações sejam efetivamente registradas. Se ocorrer um erro em qualquer parte da transação, todas as alterações serão desfeitas, garantindo a consistência e a integridade dos dados no banco de dados. Dessa forma, mitigamos a possibilidade de vendas com dados inconsistentes no banco de dados, o que afetaria a confiabilidade dos relatórios e dados apresentados pelo sistema.

5.3.10 Sprint 10 - Criação da tela de Relatórios

Ao longo dessa *sprint*, que teve duração de duas semanas, a prioridade foi a criação da tela de relatórios, abrangendo a história de usuário H-10. Para dar mais eficiência ao projeto e acelerar o desenvolvimento, optamos por incorporar o componente *Victory Native* desenvolvido por terceiros. Essa estratégia permitiu uma implementação ágil dos gráficos e tabelas necessários para apresentar os dados. Os componentes pré-desenvolvidos já dispunham de funcionalidades avançadas para exibir informações, economizando tempo que teria sido gasto no desenvolvimento personalizado desses recursos.

A utilização de componentes de terceiros proporcionou uma abordagem pragmática, permitindo que o esforço fosse destinado a outras áreas do projeto. Isso englobou a integração eficaz desses componentes na estrutura do aplicativo, bem como a personalização de sua estética e funcionalidades de acordo com as necessidades específicas do projeto, assegurando uma experiência do usuário consistente com as demais telas do sistema.

Ao desenvolver a tela de relatórios, tomamos como referência o protótipo mostrado na Figura 16. Este protótipo não incorporou dados reais ou simulações, sendo focalizado principalmente em ilustrar a disposição dos elementos na tela e esboçar as informações planejadas para os relatórios finais. Essa representação simplificada do protótipo oferece uma primeira visão da aparência planejada para a tela de relatórios, permitindo uma compreensão visual da disposição dos componentes e do *layout* geral destinado à funcionalidade.

5.3.11 Sprint 11 - Criação da tela de Configurações

Durante a última *sprint* de desenvolvimento, que teve uma duração de uma semana, o foco concentrou-se na criação das telas de configurações do aplicativo. Vale destacar que o tempo desta *sprint* difere do padrão usual de duas semanas. Essa decisão foi tomada porque as funcionalidades abordadas nesta *sprint* não eram tão complexas e não exigiam o tempo estendido de desenvolvimento que é típico das *sprints*. Embora as telas de configurações não tenham sido

Figura 16 – Protótipo da tela de relatórios

Fonte: O próprio autor

diretamente solicitadas pelos usuários entendeu-se que essa funcionalidade seria importante para melhorar o gerenciamento do sistema.

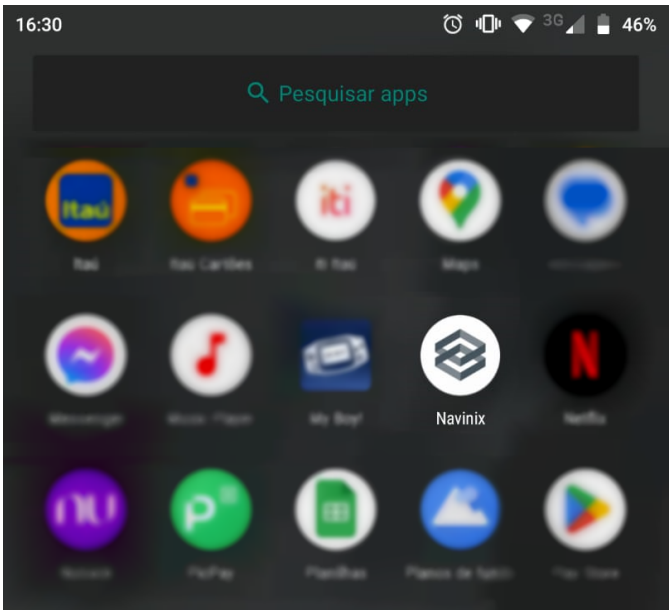
6 Resultados Obtidos

Neste capítulo iremos relatar e analisar os resultados alcançados nesse trabalho. Será apresentada uma demonstração abrangente das funcionalidades, utilizando simulações de produtos específicos do cliente consultado durante o levantamento de requisitos. Isso proporcionará uma visão completa do projeto, contribuindo para consolidar o entendimento acerca do impacto positivo e das contribuições que o trabalho ofereceu ao atender às necessidades e expectativas previamente definidas pelo usuário.

6.1 Apresentação do Aplicativo

O primeiro contato do usuário com o aplicativo desenvolvido é feito a partir do acesso à lista de softwares instalados no celular, conforme pode ser visto na Figura 17, que mostra o ícone para acessar o aplicativo desenvolvido no dispositivo móvel.

Figura 17 – Ícone do aplicativo no dispositivo móvel



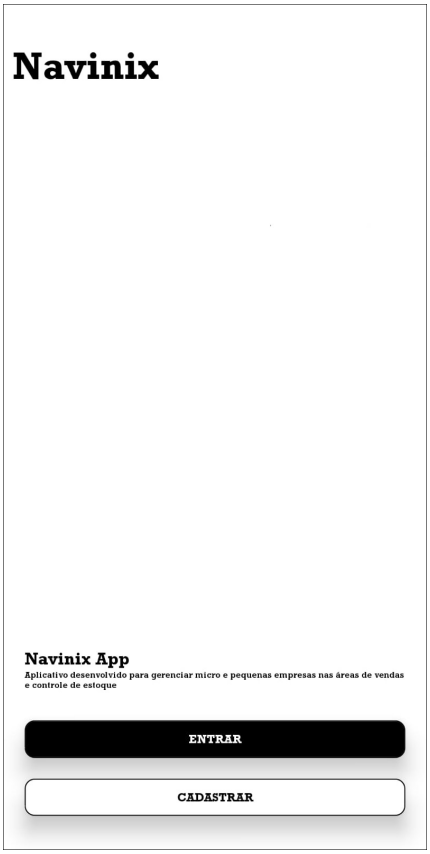
Fonte: O próprio autor

É importante destacar que o ícone exibido foi concebido através da plataforma online Nuvemshop (2023), dedicada à criação de logos para empresas. A utilização dessa ferramenta especializada conferiu um caráter profissional ao ícone do aplicativo, contribuindo na construção de uma identidade visual do sistema.

6.2 Tela inicial

A estética minimalista permeia todo o aplicativo, refletindo-se de maneira evidente na tela de login, conforme ilustrado na Figura 18. No topo desta tela, destaca-se o nome do aplicativo, proporcionando uma identificação visual clara e direta. Ao centro, um breve texto de apresentação oferece aos usuários uma visão concisa e informativa acerca dos propósitos e funcionalidades do aplicativo. Na parte inferior da tela, encontram-se dois botões: o botão de login, destinado aos usuários registrados que buscam acesso às suas contas, e o botão de cadastro, que viabiliza a criação de novas contas no sistema.

Figura 18 – Tela inicial do sistema



Fonte: O próprio autor

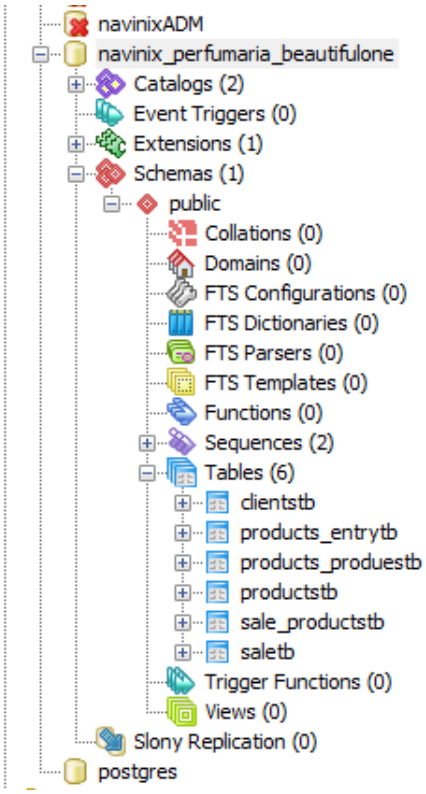
Ao selecionar a opção de cadastro, uma caixa de diálogo será aberta na parte inferior da tela do aplicativo, como ilustrado na Figura 19a, proporcionando a possibilidade de cadastrar um novo usuário e uma nova organização mediante a inserção das informações pertinentes e a seleção da opção ”Cadastrar Empresa”. Por outro lado, caso a intenção seja apenas cadastrar um novo usuário vinculado à uma empresa previamente cadastrada, há a opção de escolher a empresa desejada a partir de uma lista disponível e informar a senha definida no momento do cadastro da organização.

Em seguida, o sistema criará um novo banco de dados, caso se trate de uma nova empresa. Após a conclusão desse processo, uma mensagem de confirmação será exibida, indicando

Figura 19 – Cadastro de novo usuário

(a) Tela de cadastrado no aplicativo

(b) Banco de dados criado



Fonte: O próprio autor

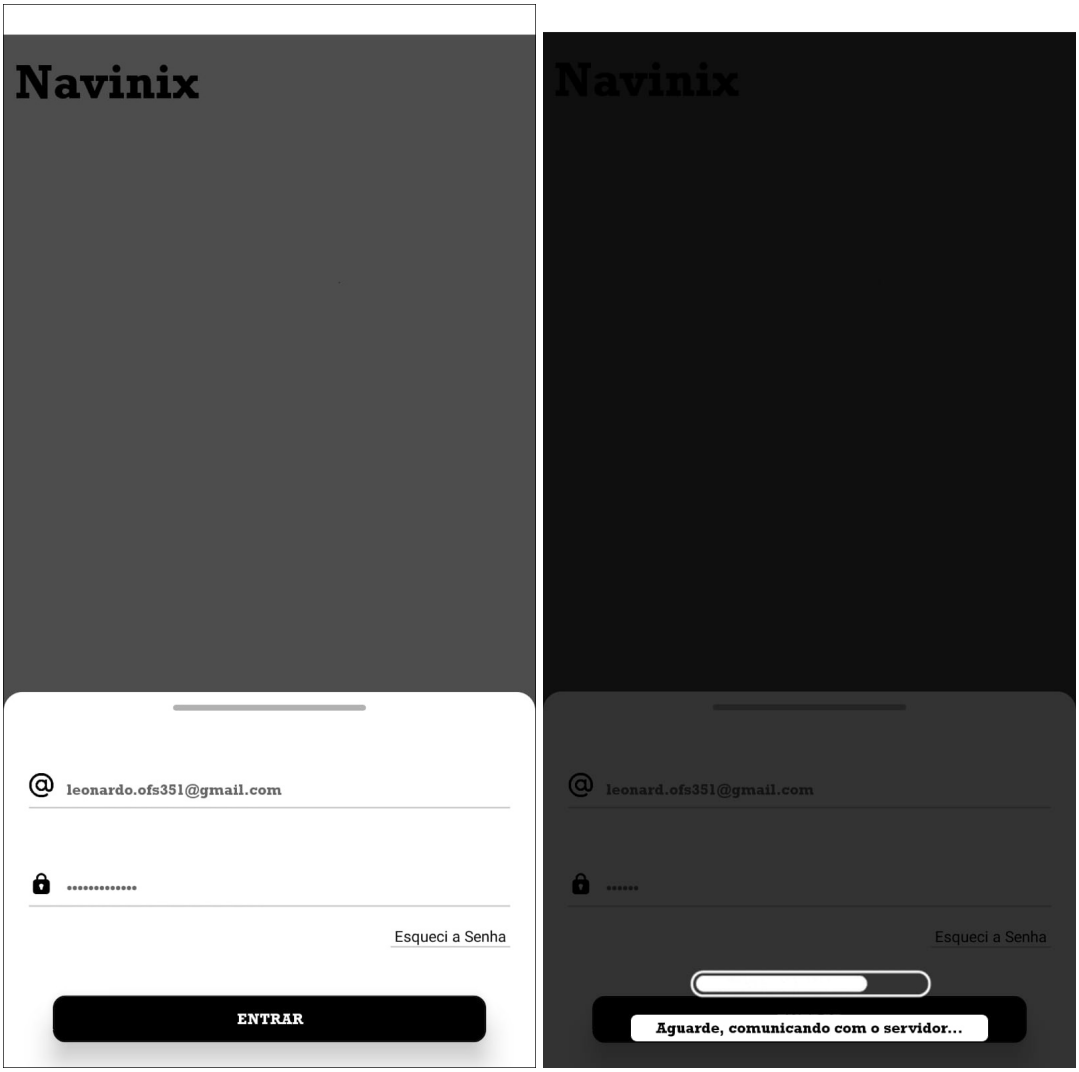
o sucesso da operação. O banco de dados recém-criado abrigará todas as tabelas designadas para armazenar os dados específicos da empresa, como representado na Figura 19b.

Após a conclusão do processo de cadastro e criação do banco de dados pelo sistema, o usuário estará habilitado para realizar o login, como mostrado na Figura 20a. Durante o procedimento de login, o sistema estabelece a comunicação com o servidor para autenticação. Vale ressaltar que o tempo exigido para essa comunicação pode variar conforme a velocidade da conexão à internet. Com o intuito de proporcionar uma experiência transparente ao usuário e mantê-lo informado sobre o progresso dessa operação, o sistema exibe uma tela de carregamento, representada na Figura 20b Esta tela é um indicador visual para informar ao usuário que o processo pode demandar um certo tempo, especialmente em conexões mais lentas.

Após efetuar o login, o usuário é direcionado para a tela *Inicial* do aplicativo, con-

Figura 20 – Processo de autenticação de usuário

(a) Tela de login do aplicativo (b) Tela de carregamento



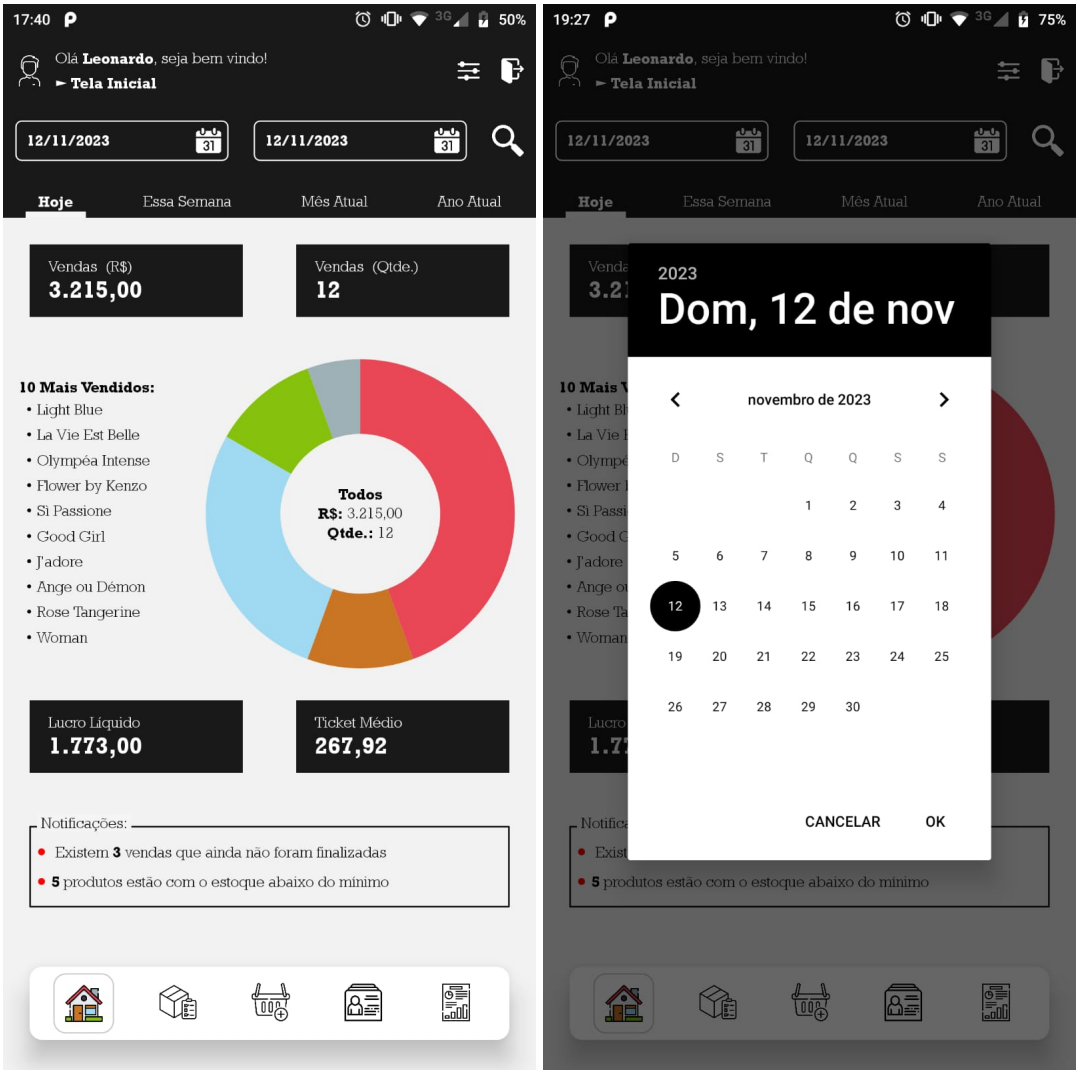
Fonte: O próprio autor

forme apresentado na Figura 21a. Essa interface oferece botões de navegação centralizados para acessar as principais seções, como "Produtos", "Clientes", "Vendas"e "Relatórios". Além disso, notificações e alertas relevantes são exibidos na parte inferior da tela, mantendo o usuário informado.

Além disso, o sistema apresenta, na tela Inicial, relatórios de vendas em períodos pré-definidos que permitem ao usuário uma rápida análise do volume de vendas. Os períodos considerados pelos relatórios são: o dia atual, semana, mês ou ano corrente. Os dados exibidos na tela estão relacionados às vendas realizadas pelo aplicativo, incluindo informações sobre o número total de vendas e o respectivo valor. Destaque para a exibição dos produtos mais vendidos por meio de um gráfico central na tela. Esse gráfico é interativo, permitindo ao usuário obter detalhes específicos ao clicar em uma seção, onde são apresentados os dados referentes ao produto selecionado no centro do gráfico. Adicionalmente, o sistema permite selecionar um

Figura 21 – Imagens da tela *Home* do aplicativo

(a) Tela *Home* do aplicativo (b) Calendário para a seleção de data



Fonte: O próprio autor

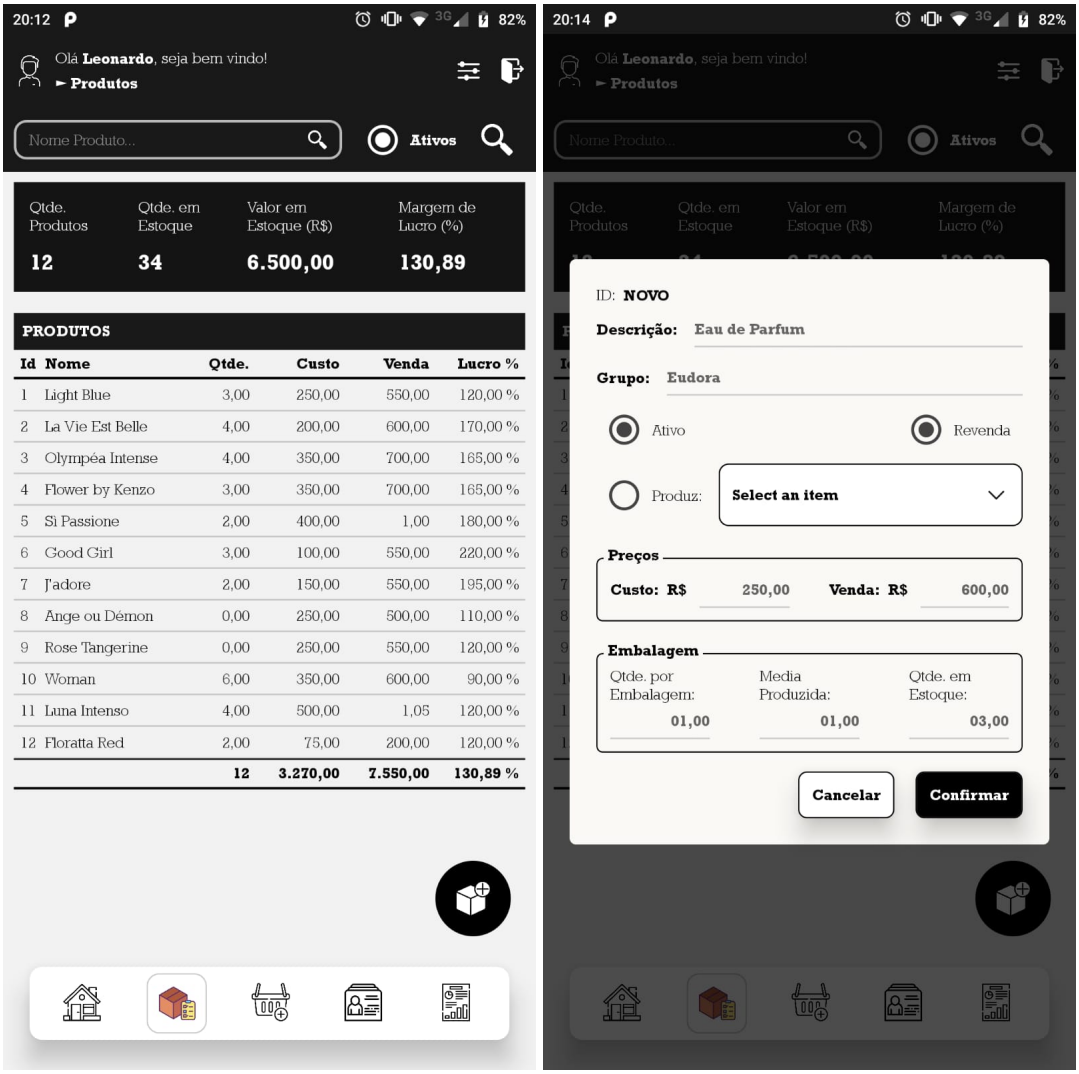
período específico para gerar o relatório de vendas, como pode ser visto na Figura 21b. Para tal, foi utilizado um componente de calendário para facilitar a entrada de dados necessários para a geração do relatório.

6.3 Produtos

A tela inicial do cadastro de produtos, apresentada na Figura 22a, oferece uma perspectiva abrangente para o gerenciamento dos produtos cadastrados no aplicativo. Nessa tela, os usuários têm a possibilidade de inserir novos produtos, modificar informações de produtos já existentes e explorar o catálogo de produtos de maneira organizada. Cada produto é exibido com detalhes como nome, preço e quantidade em estoque, oferecendo uma visualização clara do conjunto de produtos registrados.

Figura 22 – Telas referentes aos produtos

(a) Listagem de produtos cadastrados (b) Adicionando um novo produto



Fonte: O próprio autor

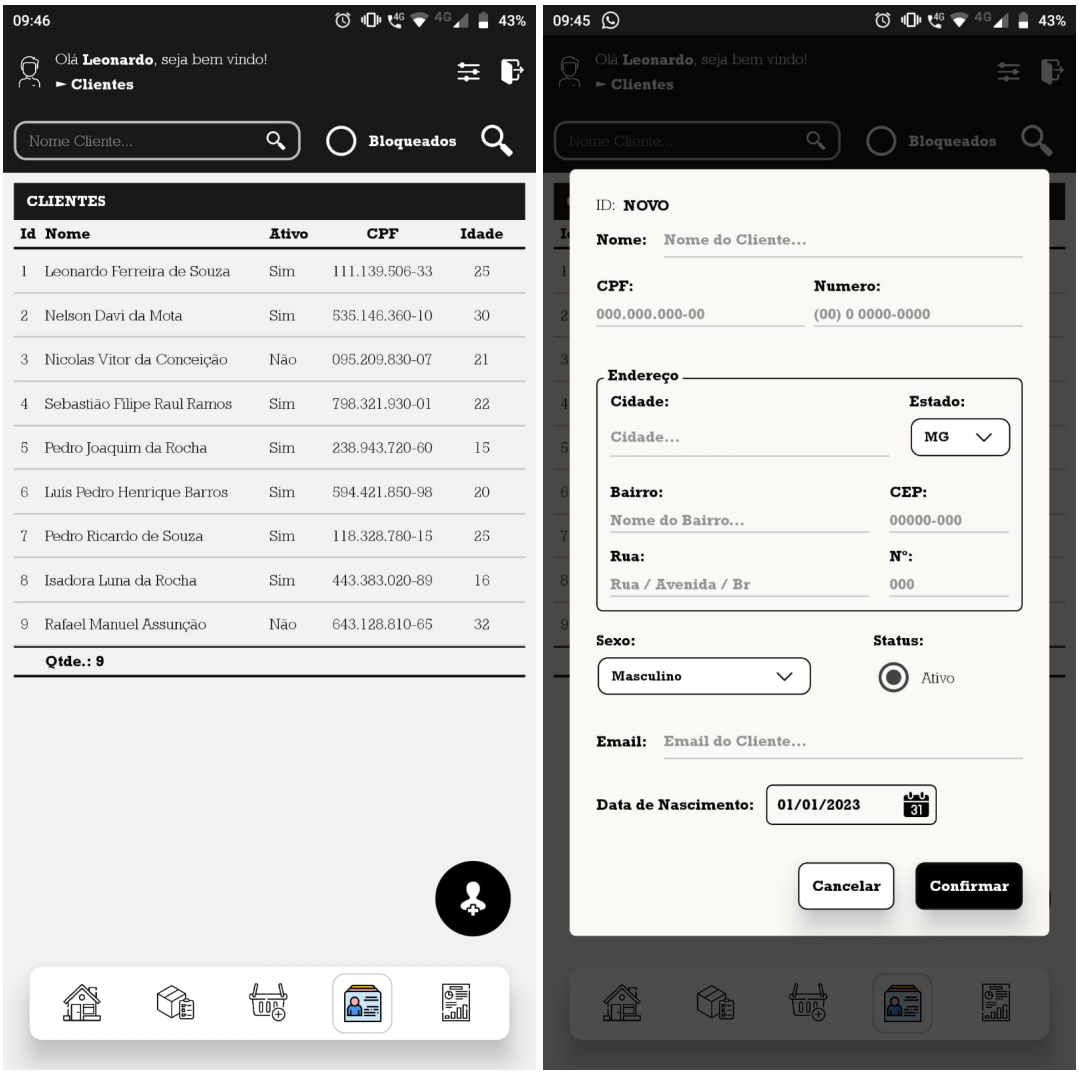
Na tela para adicionar novos produtos, apresentada na Figura 22b, os usuários podem inserir os dados de um novo produto. Além disso, é possível registrar informações como o nome do produto, o grupo à qual pertence, se o produto está ativo, e se é destinado para revenda ou consumo próprio. Para produtos utilizados na produção de outros itens, os usuários podem especificar qual produto será fabricado. Além disso, é possível informar os preços de custo e venda, proporcionando uma abordagem completa à precificação do produto. A tela também oferece a opção de incluir detalhes sobre a embalagem, se aplicável, permitindo também a inserção de informações específicas sobre a quantidade de itens já disponíveis em estoque.

6.4 Clientes

A tela inicial do cadastro de clientes, apresentada na Figura 23a, oferece uma visão abrangente para a gestão de clientes dentro do aplicativo. A partir dessa tela, os usuários podem adicionar novos clientes, realizar modificações nas informações de clientes já existentes e explorar a lista de clientes de maneira organizada. Cada cadastro é composto por informações básicas, como nome, telefone, idade além do indicativo de bloqueio ou não do cliente.

Figura 23 – Telas referentes aos clientes

(a) Listagem de clientes cadastrados (b) Adicionando um novo cliente



Fonte: O próprio autor

Em relação a opção de adicionar novos clientes, como pode ser visto na Figura 23, os usuários têm a possibilidade de inserir os dados de um novo cliente. Nessa tela específica, é possível fornecer uma variedade de informações, incluindo nome, CPF, telefone, endereço, sexo, email, data de nascimento e a determinação do status de bloqueio do cliente, um elemento que impactará diretamente no processo de venda. A obtenção dessas informações dos clientes

é crucial no contexto empresarial, pois contribuem para uma gestão eficaz do relacionamento com o cliente.

A coleta e uso de informações podem viabilizar estratégias de retenção de cliente, como por exemplo, a personalização do atendimento. Esses dados são considerados valiosos, pois proporcionam um conhecimento aprofundado do perfil de cada cliente. Dessa forma, a organização pode oferecer uma experiência adaptada às necessidades específicas e fomentar a lealdade do cliente. Além disso, a empresa pode antecipar necessidades, direcionar promoções e criar um ambiente propício para que os clientes retornem e realizem compras adicionais.

6.5 Vendas

A tela inicial do cadastro de vendas, ilustrada na Figura 24, mantém a consistência visual observada nas demais telas do aplicativo. Essa tela é composta por diversos elementos visuais, entre eles duas tabelas: uma apresenta os detalhes das vendas mais recentes, enquanto a outra enumera os orçamentos cadastrados. Na tabela com os orçamentos cadastrados, o sistema oferece uma opção para finalizar uma venda previamente salva.

Figura 24 – Tela de vendas do aplicativo

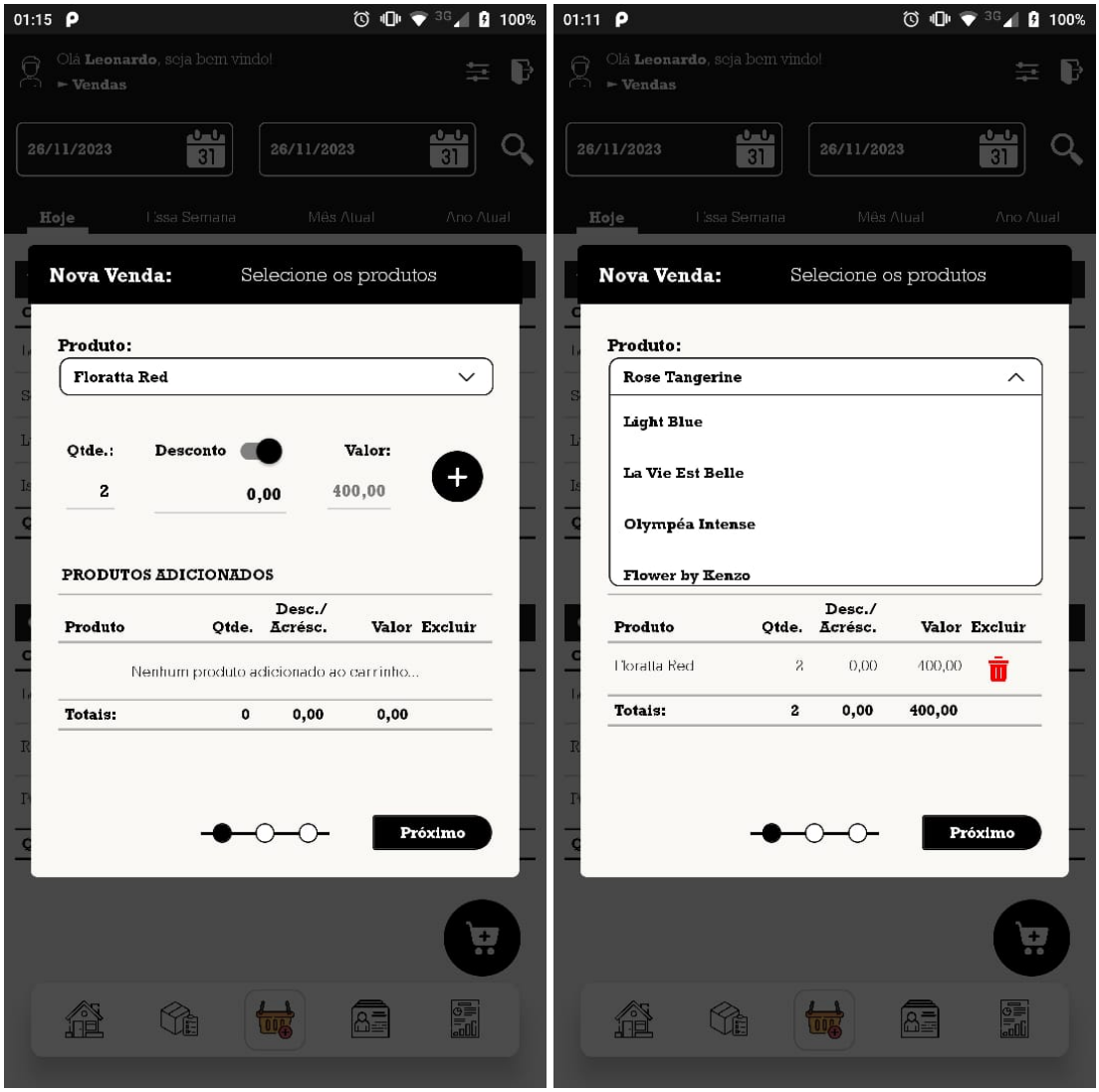
Fonte: O próprio autor

Ao iniciar uma nova venda, o sistema exibe a tela de inserção de produtos ao carrinho de vendas que permite aos usuários adicionar novos produtos ao carrinho, como pode ser visto

na Figura 25a. Nessa etapa, os usuários têm a opção de buscar produtos pelo nome. De modo que o resultado é exibido em uma lista com opções para seleção, conforme apresentado na Figura 25b. Para cada produto escolhido, é possível especificar a quantidade desejada, além de aplicar descontos ou acréscimos individuais. O campo de pesquisa facilita a localização do produto desejado, enquanto o valor do produto selecionado é exibido ao lado. Uma vez que todas as informações são preenchidas, os usuários podem adicionar o produto ao carrinho.

Figura 25 – Telas de inserção de produtos em uma venda

(a) Inserção de produtos ao carrinho de vendas (b) Busca de produtos na lista



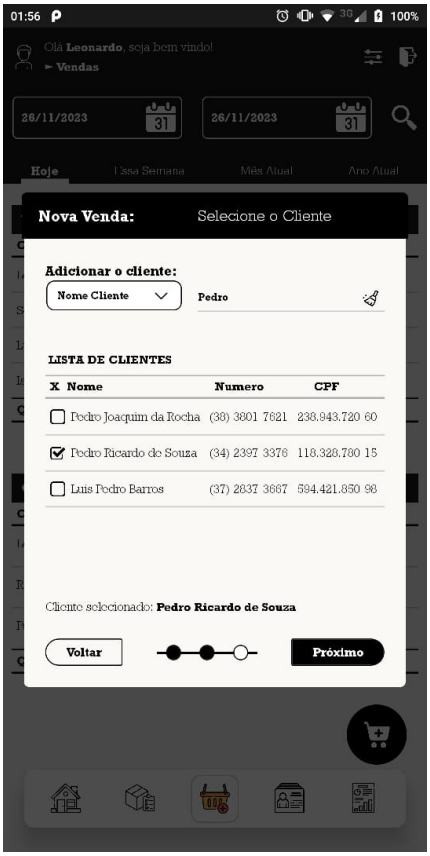
Fonte: O próprio autor

A parte inferior da tela de inserção de produtos ao carrinho de vendas, contém uma grade que exibe de forma clara e organizada os produtos já adicionados, como pode ser visto na Figura 25a, proporcionando uma visão abrangente dos itens selecionados até o momento.

Após selecionar os itens e clicar em próximo na tela de inserção de produtos ao carrinho de venda, os usuários podem escolher o cliente para a venda, como indicado na Figura

26. Nessa etapa, é possível listar os clientes a fim de escolher o cliente da venda. Aqui, destaca-se uma funcionalidade que permite aos usuários selecionarem o campo de pesquisa, seja por nome, número de telefone ou CPF. Adicionalmente, existe a opção de aplicar filtros para exibir exclusivamente clientes ativos, simplificando o processo de busca e seleção do cliente desejado.

Figura 26 – Tela seleção de cliente para venda

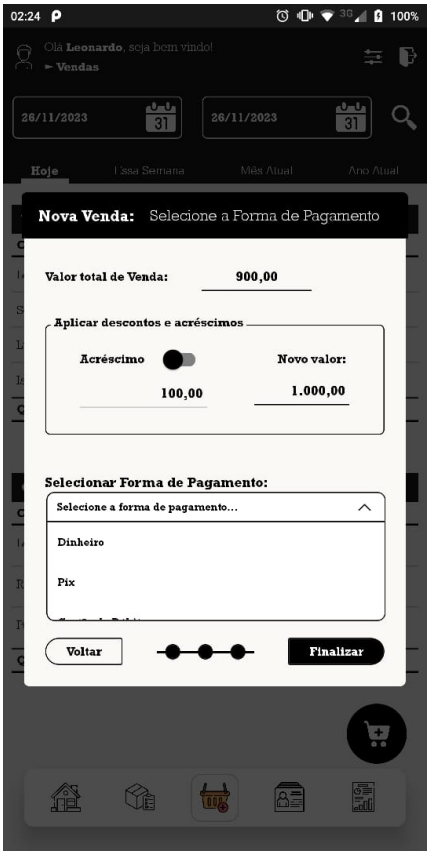


Fonte: O próprio autor

Na etapa seguinte, conforme ilustrado na Figura 27, os usuários podem selecionar a forma de pagamento para a venda. Inicialmente, o valor total da venda é apresentado, proporcionando uma compreensão clara do montante a ser pago. Adicionalmente, há a opção de aplicar descontos ou acréscimos, conforme necessário. Logo abaixo dessas opções, encontra-se a lista de formas de pagamento disponíveis, permitindo que o cliente escolha a opção que melhor atenda às suas preferências ou condições.

Após a conclusão dos passos anteriormente mencionados, o sistema oferece ao usuário a opção de salvar a venda como orçamento ou finalizar a operação. Essa funcionalidade visa oferecer flexibilidade ao usuário, permitindo a escolha entre manter a venda como uma estimativa ou concluir a operação. A Figura 28 ilustra essa etapa, proporcionando uma representação visual do momento em que o cliente decide o destino da operação.

Figura 27 – Tela seleção de forma de pagamento para venda



Fonte: O próprio autor

6.6 Relatórios

A tela de relatórios disponibiliza opções para listagem de dados que podem ser usados para análise das vendas, como pode ser visto na Figura 29. Nessa tela, os usuários têm a flexibilidade de escolher entre três tipos distintos de relatórios, sendo um relatório de vendas por clientes, outro de vendas por produtos e o último de vendas por forma de pagamento. Cada uma dessas categorias foi elaborada para disponibilizar informações específicas relacionadas a diversos aspectos do negócio. Isso confere aos usuários a flexibilidade de concentrar-se nas áreas que consideram mais relevantes e estratégicas para a administração das vendas.

Dentro de cada tipo de relatório, os usuários podem ainda personalizar a representação visual dos dados, optando entre quatro tipos de gráficos: *donut*, linhas ou histograma. Essa funcionalidade permite uma análise mais detalhada e adaptada às preferências individuais, proporcionando uma compreensão mais profunda das tendências e padrões presentes nos dados de vendas.

Ao selecionar um tipo específico de gráfico, ele é exibido na parte superior da tela, fornecendo uma representação clara e intuitiva dos dados analisados. Logo abaixo do gráfico, os dados são apresentados de forma tabular, oferecendo uma visão detalhada e numérica dos mesmos.

Figura 28 – Confirmação da venda

Fonte: O próprio autor

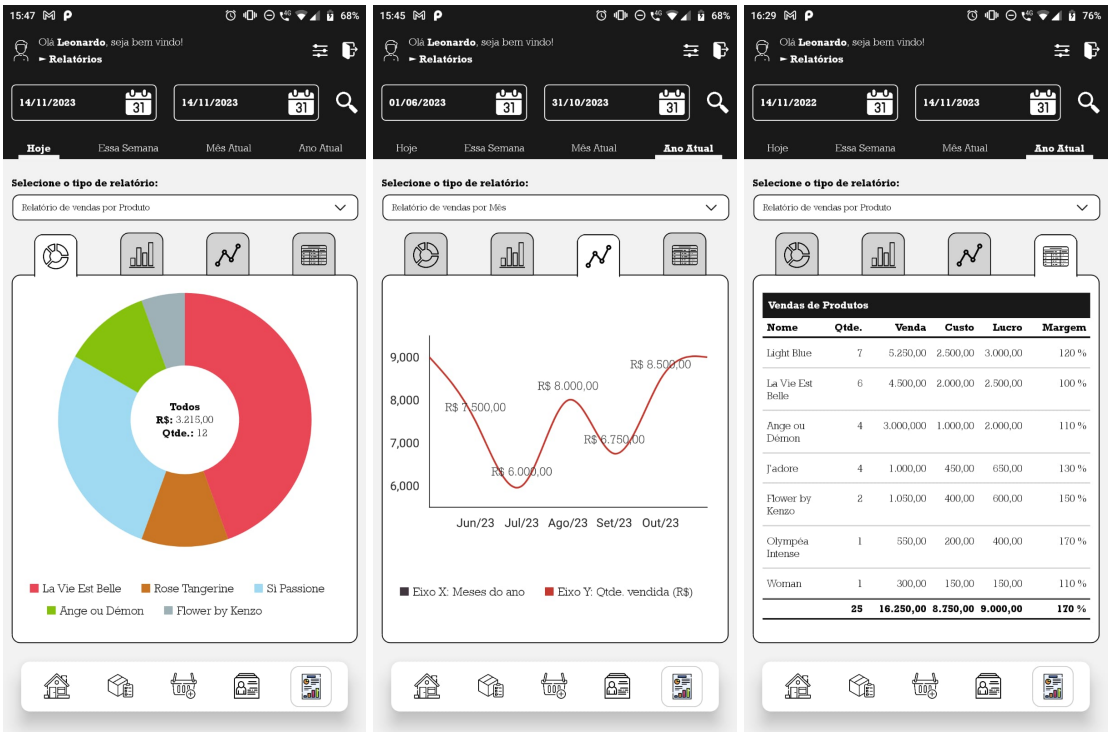
6.7 Configurações

A tela de configurações, ilustrada na Figura 30a, apresenta quatro opções para o usuário. A primeira opção, "Voltar à Tela Inicial", permite ao usuário retornar à página principal do aplicativo. Em seguida, a opção "Acessar Meus Dados"possibilita a visualização e edição das informações pessoais do usuário. A terceira opção, "Alterar Senha", oferece a funcionalidade de modificar a senha de acesso à conta. Por fim, a opção "Encerrar Sessão"permite ao usuário efetuar o *logout*, encerrando sua sessão no aplicativo.

Ao optar por encerrar a sessão, o aplicativo redireciona o usuário para a tela inicial do sistema, conforme exemplificado na figura 30b, completando, dessa forma, o ciclo operacional do sistema.

Figura 29 – Telas de relatórios do aplicativo

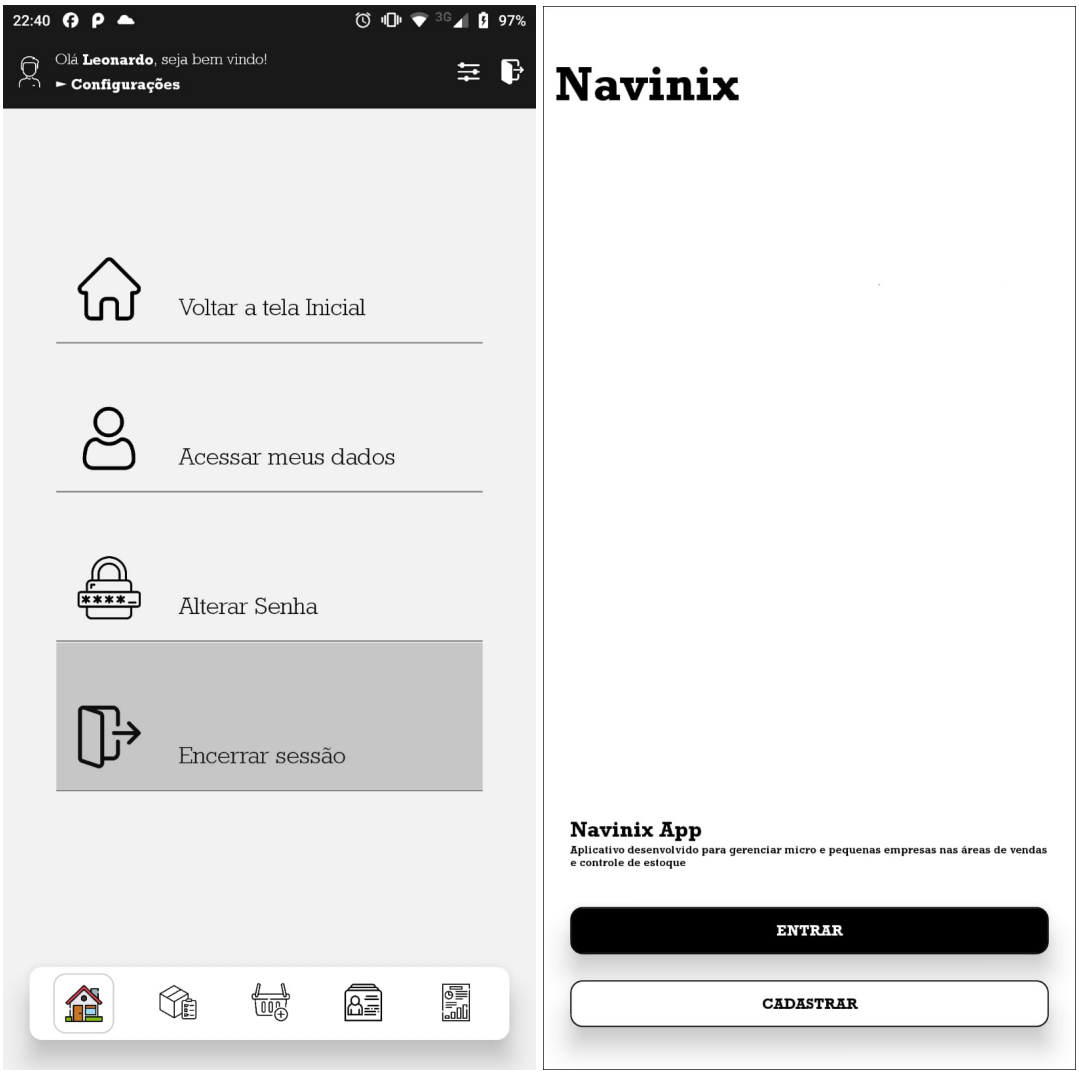
(a) Gráfico de *donnut* (b) Gráfico de linhas (c) Vendas com dados tabelados



Fonte: O próprio autor

Figura 30 – Configurações do sistema

(a) Tela de configurações do sistema (b) Tela inicial do aplicativo



Fonte: O próprio autor

7 Considerações Finais

O desenvolvimento deste projeto representou um desafio empolgante e significativo na busca por proporcionar aos usuários uma solução abrangente e eficaz para o gerenciamento de negócios. Ao longo de várias *sprints* de desenvolvimento, foram abordadas diversas funcionalidades, desde o cadastro de usuários e autenticação até a criação de telas de gerenciamento de produtos, clientes, vendas e relatórios.

O uso do *framework Laravel* e sua ferramenta *Eloquent ORM* demonstrou ser uma escolha sólida para simplificar as interações com o banco de dados, enquanto o paradigma de orientação a objetos foi aplicado ao ambiente de banco de dados relacional. O isolamento de informações de acesso para dados administrativos e específicos do cliente, além do uso de um banco de dados central criptografado, ilustra o compromisso com a segurança e a privacidade dos usuários.

A abordagem ágil adotada neste projeto permitiu uma adaptação contínua às necessidades em evolução, refletindo-se na evolução das interfaces das telas, como ilustrado pela tela de produtos. Isso demonstra a dinâmica do desenvolvimento de software e a capacidade de responder a demandas do projeto em constante evolução. As telas de relatórios, eficientemente implementadas com o uso de componentes de terceiros, economizaram tempo e ofereceram funcionalidades avançadas, proporcionando uma experiência mais completa e flexível para os usuários com a adição de telas de configurações.

A validação dos objetivos do trabalho foi realizada por meio da coleta contínua de *feedback* pelo usuário que participou ativamente no levantamento de requisitos, incorporada às iterações subsequentes do aplicativo. Este processo assegurou a consideração das preferências e necessidades dos usuários em cada fase do desenvolvimento, proporcionando uma validação prática e direta do sucesso na entrega dos resultados almejados.

Este projeto ilustra a importância da integração de boas práticas de segurança, usabilidade e agilidade no desenvolvimento de sistemas de gerenciamento de negócios. O sistema resultante representa uma solução robusta que visa atender às necessidades de negócios em constante mudança.

À medida que o desenvolvimento continua e se expande, o objetivo é aprimorar ainda mais as funcionalidades, atender a requisitos específicos dos clientes e garantir que o sistema permaneça seguro e eficiente. No entanto, é importante mencionar que, no escopo deste trabalho, não foi possível abordar a usabilidade do sistema em detalhes devido a restrições de tempo. Assim, em trabalhos futuros, um foco adicional poderá ser dedicado à usabilidade, com o intuito de melhorar a experiência do usuário.

Identificamos uma lacuna na área de entrada de produtos, visto que essa funcionalidade

não foi desenvolvida. Consequentemente, não pudemos implementar a dinâmica de preços com base em margens de lucro. A capacidade de ajustar dinamicamente os preços dos produtos, considerando custos, margens de lucro desejadas e estratégias de mercado, auxiliam para uma gestão de preços eficaz. Este aspecto será um ponto central em nossos esforços futuros de desenvolvimento.

Adicionalmente, como parte do trabalho futuro, pretende-se explorar e estabelecer um protocolo de testes para avaliar o comportamento do sistema em cenários envolvendo números variados de empresas. Isso servirá para verificar como o sistema se adapta e mantém seu desempenho diante da criação e manipulação de diferentes bases de dados de estoque.

Essas considerações delineiam as direções estratégicas para futuros desenvolvimentos, buscando aprimorar a utilidade, segurança e eficiência do sistema. Este estudo é uma jornada contínua, comprometida em enfrentar desafios futuros com uma abordagem proativa e centrada no usuário.

Referências

Agência de Notícias IBGE. Internet já é acessível em 90,0% dos domicílios do país em 2021. https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/34954-internet-ja-e-acessivel-em-90-0-dos-domicilios-do-pais-em-2021, 2022. Citado na página 14.

Hudson Andrade Almeida. Desenvolvimento e avaliação de um sistema web para controle e gestão das amostras geológicas do setor cegeo-ict da ufvjm de diamantina/mg. Universidade Federal dos Vales do Jequitinhonha e Mucuri - UFVJM, 2017. Citado na página 30.

Lorena Adrian Cardoso Carvalho, Marcelo Werneck Barbosa, and Vinícius Bernardo Silva. Proposta e avaliação de uma abordagem lúdica para o ensino de histórias de usuário e scrum. *Gestão e Projetos: GeP*, 5(3):44–58, 2014. Citado na página 31.

Mike Cohn. *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional, 2004. Citado na página 31.

Vitor da Silva Cruz, Erick Eduardo Petrucelli, and Eder Carlos Salazar Sotto. A linguagem javascript como alternativa para o desenvolvimento de aplicaÇÕes multiplataforma. *Revista Interface Tecnológica*, 15(2):39–49, 2018. Citado na página 26.

Antonio Soares de Azevedo Terceiro. Caracterização da complexidade estrutural em sistemas de software livre. Programa Multiinstitucional de Pós-Graduação em Ciência da Computação, 2013. Citado na página 19.

Correio do Povo. Mais de 155 milhões de brasileiros possuem celular para uso pessoal, aponta ibge. https://www.correiodopovo.com.br/jornalcomtecnologia/mais-de-155-milh~oes-de-brasileiros-possuem-celular-para-uso-pessoal-aponta-ibge-1.891007, 2022. Citado na página 14.

Eduardo Ferreira Franco. Um modelo de gerenciamento de projetos baseado nas metodologias ágeis de desenvolvimento de software e nos princípios da produção enxuta. Escola Politécnica da Universidade de São Paulo, 2007. Citado 4 vezes nas páginas 19, 20, 21 e 22.

Iago Rossi Gasparetto and Ana Paula Canal. *Ferramenta Para A Priorização De Requisitos De Software Em Projetos Ágeis.* Trabalho de Conclusão de Curso Ciência Da Computação - Universidade Franciscana (UFN), Santa Maria, RS, Brasil. Disponível em https://tfgonline.lapinf.ufn.edu.br, 2023. Citado na página 39.

Governo Federal. Brasil registra saldo de quase 700 mil empresas abertas nos primeiros quatro meses do ano. https://www.gov.br/pt-br/noticias/financas-impostos-e-gest

ao-publica/2020/06/brasil-registra-saldo-de-quase-700-mil-empresas-abertas-nos-primeiros-quatro-meses-do-ano, 2020a. Citado na página 14.

Governo Federal. Brasil ultrapassa a marca de 10 milhões de microempreendedores individuais (meis). https://www.gov.br/economia/pt-br/assuntos/noticias/2020/abril/brasil-ultrapassa-a-marca-de-10-milhoes-de-microempreendedores-individuais-meis, 2020b. Citado na página 14.

Filipe Del Nero Grillo and Renata Pontin de Mattos Fortes. Aprendendo javascript. https://repositorio.usp.br/directbitstream/4cd7f9b7-7144-40f4-bfd0-7a1d9a6bd748/nd_72.pdf, 2008. Citado na página 26.

Lucidchart. Símbolos e notação de diagramas entidade-relacionamento. https://www.lucidchart.com/pages/pt/simbolos-de-diagramas-entidade-relacionamento, 2023. Citado na página 42.

Microsoft. Visual studio code. https://code.visualstudio.com, 2023. Citado na página 28.

Nuvemshop. Nuvemshop. https://www.nuvemshop.com.br/midia/companhia, 2023. Citado 2 vezes nas páginas 29 e 55.

Fábio Oliveira. Protótipo de aplicativo de força de vendas para dispositivos móveis baseados na plataforma android. Universidade Regional de Blumenau, 2014. Citado na página 17.

Renan Oliveira. Desenvolvimento de sistema de gestão de estoque personalizado com o framework laravel. Universidade Estadual do Centro-Oeste, 2019. Citado na página 17.

Glaúcia Aparecida Prates and Marco Túlio Ospina Patino. Tecnologia da informação em pequenas empresas: Fatores de Êxito, restrições e benefícios. *Revista de administração contemporânea*, 8(1):9–26, 2004. Citado na página 18.

Roger S. Pressman. Engenharia de software - uma abordagem profissional. AMGH Editora Ltda., 2001. Citado 2 vezes nas páginas 19 e 20.

Rafael Prikladnicki, Renato Willi, and Fabiano Milani. *Métodos ágeis para desenvolvimento de software*. Bookman Editora, 2014. Citado na página 21.

Aline Bentes Ramos and Dalton Chaves Vilela Junior. A influÊncia do papel do scrum master no desenvolvimento de projetos scrum. *Gestão e Projetos - GeP*, 8(3):80–99, 2017. Citado 2 vezes nas páginas 22 e 23.

Rennata Algarte Ramsdorf. Estudar o uso de aplicativos livres da tecnologia da informação na gestão de micro e pequenas empresas. UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANA, 2014. Citado na página 18.

Felipe Pasianot Rodrigues and Kaue Ribeiro. Sistema web e mobile para eventos. Faculdade de Tecnologia - Fatec, 2020. Citado na página 17.

Ken Schwaber. Scrum development process. Business Object Design and Implementation, 1995. Citado 2 vezes nas páginas 22 e 23.

Jessica Belém Silva and Francisca Alexandra Macedo Anastácio. Método kanban como ferramenta de controle de gestão. *Revista multidisciplinar e de psicologia*, 13(43):1018–1027, 2019. Citado 2 vezes nas páginas 24 e 25.

Wesley Willians Ramos Silva. *Laravel 5.7: Essencial*. Leanpub, 2019. Citado na página 27.

Juliana Coelho Slominski. A importância da realização da gestão de estoque em pequenas empresas. Universidade Federal do Paraná, 2016. Citado na página 17.

Amanda Fagundes Soares, Gabriela Fonseca de Moura, and Rayla dos Santos Oliveira Dias. O impacto da pandemia do covid-19 no empreendedorismo: um estudo acerca da percepção dos empreendedores do município de resende - rj. Simpósio de Excelência em Gestão e Tecnologia, 2021. Citado na página 14.

Michel S. Soares. Comparação entre metodologias Ágeis e tradicionais para o desenvolvimento de software. Universidade Presidente Antônio Carlos, 2004a. Citado na página 20.

Michel Santos Soares. Metodologias Ágeis extreme programming e scrum para o desenvolvimento de software. Universidade Presidente Antônio Carlos, 2004b. Citado na página 22.

Arthur Câmara Souza, Hugo Richard Amaral, and Luis Eduardo O. Lizardo. Postgresql: uma alternativa para sistemas gerenciadores de banco de dados de código aberto. *Anais do Congresso Nacional Universidade, EAD e Software Livre*, 2(2):449–467, 2011. Citado na página 27.

Valor Investe. Com pandemia, criação de pequenas empresas aumenta 19trimestre. https://valorinveste.globo.com/objetivo/empreenda-se/noticia/2021/12/07/com-pandemia-criacao-de-pequenas-empresas-aumenta-19percent-no-terceiro-trimestre.ghtml, 2021. Citado na página 14.

Carlos Alberto Zago. Desenvolvimento de um site de comércio eletrônico utilizando php e mysql. *Anais do Congresso Nacional Universidade, EAD e Software Livre*, 2(2):449–467, 2011. Citado na página 25.