Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Desenvolvimento de um *e-commerce* para uma marca autoral de roupas

Ian Langkammer Batista

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:
Janniele Aparecida Soares Araujo

COORIENTAÇÃO:
George Henrique Godim Da Fonseca

Fevereiro, 2024
João Monlevade–MG

Ian Langkammer Batista

Desenvolvimento de um *e-commerce* para uma marca autoral de roupas

Orientador: Janniele Aparecida Soares Araujo

Coorientador: George Henrique Godim Da Fonseca

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Fevereiro de 2024

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

B333d Batista, Ian Langkammer.
Desenvolvimento de um e-commerce para uma marca autoral de roupas. [manuscrito] / Ian Langkammer Batista. - 2024.
66 f.: il.: color., tab..

Orientadora: Profa. Dra. Janniele Aparecida Soares Araujo.
Coorientador: Prof. Dr. George Henrique Godim da Fonseca.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Graduação em Sistemas de Informação .

1. Comércio eletrônico. 2. Comportamento do consumidor. 3. Disseminação seletiva da informação. 4. Satisfação do consumidor. 5. Software - Desenvolvimento. I. Araujo, Janniele Aparecida Soares. II. Fonseca, George Henrique Godim da. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 004.41:658.84

Bibliotecário(a) Responsável: Flavia Reis - CRB6-2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Ian Langkammer Batista

Desenvolvimento de um e-commerce para uma marca autoral de roupas

Monografia apresentada ao Curso de de Sistemas de Informação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação

Aprovada em 23 de Fevereiro de 2024

Membros da banca

Dra Janniele Aparecida Soares Araujo - Orientadora (Universidade Federal de Ouro Preto)
Dr George Henrique Godim da Fonseca - Coorientador (Universidade Federal de Ouro Preto)
Dr Fernando Bernardes de Oliveira - (Universidade Federal de Ouro Preto)
Dr Samuel Souza Brito - (Universidade Federal de Ouro Preto)

Janniele Aparecida Soares Araujo, orientadora do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 16/10/2024

Documento assinado eletronicamente por **Janniele Aparecida Soares Araujo**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 16/10/2024, às 09:29, 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 http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0795435** e o código CRC **A678B7DF**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3808-0819 - www.ufop.br

Com todo o meu coração, dedico este trabalho aos meus pais, fontes inesgotáveis de amor, apoio e inspiração. A cada desafio enfrentado, vocês estiveram ao meu lado, proporcionando força e motivação para seguir em frente. À minha família e amigos, cujo encorajamento constante e amizade leal iluminaram os dias mais difíceis. Cada palavra de ânimo, cada gesto de carinho, construiu a base sólida sobre a qual este trabalho se ergue. Aos meus orientadores, expresso minha profunda gratidão. Suas orientações sábias e paciência infinita foram fundamentais para o desenvolvimento deste trabalho. Seu comprometimento e expertise moldaram não apenas o meu projeto, mas também o meu crescimento acadêmico e pessoal. A todos vocês, que de diversas maneiras contribuíram para minha jornada acadêmica, meu mais sincero agradecimento. Este trabalho é não apenas meu, mas nosso, reflexo do apoio e cooperação que tornaram possível superar desafios e alcançar este marco. Vocês foram a luz que me guiou quando a jornada parecia obscura, e por isso, dedico a vocês este trabalho com imensa gratidão e carinho.

Agradecimentos

Agradeço de coração aos meus pais, Gisele Langkammer e Francislei De Souza Batista, peças fundamentais em minha jornada, por me proporcionarem uma base sólida na vida e por incutirem em mim os valores que me guiaram até aqui. Sua dedicação, amor e ensinamentos moldaram não apenas a pessoa que sou, mas também a perseverança que me impulsionou a buscar meus sonhos. Assim também, não posso deixar de agradecer a toda minha família por também sempre me apoiarem e oferecerem todo suporte para continuar trilhando minha jornada árdua, agradeço a ti meus tios Mércia e Wesley, Geane e Amarildo, minha tia Juliana, as minhas amadas madrinhas Normélia e Alessandra, aos meus avós que são tão especiais Dario, Emília e Maria das Graças.

Aos meus orientadores, que desempenharam um papel essencial nesta jornada, dedico sinceros agradecimentos. Sua paciência, orientação e *insights* valiosos foram o alicerce sobre o qual construí este trabalho. Em cada dúvida, cada obstáculo, vocês estiveram presentes, acendendo luzes e guiando-me com maestria.

A todos que, de alguma forma, contribuíram para este projeto e para a minha formação, meu mais profundo agradecimento. Este trabalho é resultado não apenas do meu esforço, mas da colaboração, apoio e confiança de pessoas extraordinárias que tornaram possível transformar ideias em realidade.

Não posso deixar de expressar minha profunda gratidão aos amigos que conquistei ao longo da jornada acadêmica. Companheiros incansáveis nas madrugadas de estudo, nas intermináveis sessões de trabalho em equipe e nos desafios de aprendizado, essas amizades se tornaram alicerces essenciais na minha formação. Juntos, enfrentamos as dificuldades das matérias, compartilhamos a pressão dos prazos e celebramos as pequenas vitórias. Sem a colaboração e o suporte mútuo desses amigos incríveis, nada disso teria sido possível. A amizade e camaradagem que cultivamos foram fundamentais para que todos nós conseguíssemos superar obstáculos e alcançar este momento tão significativo de formatura. Agradeço a cada um por fazer parte dessa jornada inesquecível e por tornar a caminhada acadêmica não apenas educativa, mas também repleta de memórias valiosas.

"A inovação distingue um líder de um seguidor."

— Steve Jobs (1955 – 2011)

Resumo

Este trabalho descreve o desenvolvimento de um *e-commerce* para uma marca autoral de roupas, com o intuito de expandir o negócio e aprimorar a experiência de compra dos clientes. Com o crescimento do comércio eletrônico e a mudança de comportamento dos consumidores, a presença online tornou-se fundamental para alcançar um público amplo e oferecer uma experiência de compra conveniente e personalizada. A partir da implementação de uma plataforma que oferece uma experiência de compra enriquecida e a maximização do conhecimento das tecnologias utilizadas. Este estudo sugere o potencial de criar um ambiente digital mais integrado e dinâmico, promovendo a satisfação do usuário e impulsionando a disseminação de informações relevantes para o consumo consciente e a tomada de decisões informadas.

Palavras-chaves: *e-commerce*, clientes, experiência, comércio eletrônico

Abstract

This work describes the development of an e-commerce for an authorial clothing brand, in order to expand the business and improve the customers' shopping experience. With the growth of e-commerce and changing consumer behavior, an online presence has become critical to reaching a broad audience and providing a convenient and personalized shopping experience. Through the implementation of a platform that offers an enriched shopping experience and the maximization of knowledge of the technologies used. This study demonstrates the potential to create a more integrated and dynamic digital environment, promoting user satisfaction and driving the dissemination of relevant information to conscious consumption and informed decision-making.

Key-words*: e-commerce, customers, experience*

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
B2B	*Business-to-Business*
B2C	*Business-to-Consumer*
B2G	*Business-to-Government*
BMC	*Business Model Canvas*
CDN	*Content Delivery Network*
C2C	*Consumer-to-Consumer*
C2G	*Consumer-to-Government*
CEP	Código de Endereçamento Postal
COSI	Colegiado do Curso de Sistemas de Informação
DECSI	Departamento de Computação e Sistemas
DOM	*Document Object Model*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
ICEA	Instituto de Ciências Exatas e Aplicadas
I/O	*Input/Output*
JM	João Monlevade
npm	*Node Package Manager*
REPL	*Read-Eval-Print-Loop*
SEO	*Search Engine Optimization*
SI	Sistemas de Informação
SPA	*Single-page application*
SSR	*Server-side rendering*

SSL	*Secure Sockets Layer*
TLS	*Transport Layer Security*
UI	*User Interface*
UFOP	Universidade Federal de Ouro Preto
UX	*User Experience*

Sumário

1 Introdução

O comércio eletrônico tem desempenhado um papel cada vez mais relevante no contexto atual, impulsionado pela crescente adoção da tecnologia e pela mudança de comportamento dos consumidores. No setor da moda, em particular, a presença *online* é essencial para alcançar um público amplo e oferecer uma experiência de compra conveniente e personalizada. Nesse contexto, o presente trabalho tem como objetivo principal o desenvolvimento de um *e-commerce* para uma marca autoral de roupas, visando a expansão do negócio e aprimorando a experiência de compra dos clientes.

A marca autoral de roupas em questão busca estabelecer sua presença no mercado digital, reconhecendo a importância de uma plataforma online como meio de alcançar novos consumidores e atender às demandas de um público cada vez mais conectado. O *e-commerce* surge como uma estratégia essencial para permitir que a marca amplie sua visibilidade, expanda sua base de clientes e ofereça uma experiência de compra diferenciada, alinhada à identidade e aos valores da marca.

Ao desenvolver um *e-commerce* para uma marca autoral de roupas, é necessário considerar diversos aspectos, como a arquitetura da plataforma, o *design* de interface, a integração de sistemas de pagamento seguro, a acessibilidade para dispositivos móveis, entre outros. Além disso, a personalização e a exclusividade são elementos fundamentais para transmitir a essência da marca e atrair consumidores em um mercado altamente competitivo.

Neste trabalho, serão utilizadas técnicas e ferramentas de desenvolvimento *web*, bem como princípios de *design* centrado no usuário, a fim de criar uma plataforma de *e-commerce* que atenda às necessidades da marca autoral de roupas e ofereça uma experiência de compra envolvente e intuitiva para os clientes. Serão consideradas também as boas práticas de segurança e usabilidade, visando garantir a confiabilidade do sistema e a satisfação dos usuários.

Com base em uma sólida fundamentação teórica e por meio da aplicação prática dos conceitos e técnicas de desenvolvimento de *e-commerce*, este trabalho contribuirá para a compreensão dos desafios e oportunidades associados à criação de uma plataforma online para uma marca autoral de roupas. Espera-se que os resultados obtidos possam auxiliar não apenas a marca em questão, mas também outras empresas do setor que desejam ingressar no comércio eletrônico, aproveitando as vantagens oferecidas pelo ambiente digital.

Em suma, este trabalho se propõe a apresentar o desenvolvimento de um *e-commerce* para uma marca autoral de roupas, destacando sua importância estratégica, os aspectos técnicos envolvidos e as contribuições esperadas para a empresa e para o campo do comércio

eletrônico.

1.1 Objetivos

Em linhas gerais, o objetivo desse projeto se concentra no desenvolvimento e realização do *deploy* de um sistema *web* que se especializa no comércio de roupas de uma marca do próprio autor. Além disso, a maximização dos conhecimentos em desenvolvimento de sistemas *web* se torna um objetivo essencial, especializando-se em tecnologia em alta no mercado como: *Typescript*, *React*, *Node* e *Firebase*. Vale ressaltar que o termo *deploy* se refere ao fato de disponibilizar para a internet o seu serviço, seja ele uma *SPA* (*Single Page Application*), *API* (*Application Programming Interface*), etc...

1.2 Objetivos específicos

A seguir serão listados os objetivos específicos desse trabalho:

- Descrever e apresentar o Quadro de Modelo de Negócios
- Identificar e construir personas relacionadas aos usuários do sistema
- Desenvolver protótipos de alta fidelidade
- Oferecer um sistema interativo e de fácil acesso
- Maximização do conhecimento acerca das tecnologias utilizadas

1.3 Organização

Esta monografia está organizada em seis capítulos. A Introdução apresenta os objetivos do trabalho e uma visão geral do negócio, tanto como a motivação para o projeto. A Revisão Bibliográfica cobre os conceitos teóricos e as tecnologias utilizadas, enquanto a Metodologia descreve a modelagem do negócio, desenvolvimento de protótipos e criação de personas. O capítulo de Desenvolvimento detalha o *backend*, *frontend* e o processo de *deploy*. Por fim, em Resultados, são analisados as características da plataforma resultante, e a Conclusão discute as considerações finais e ideias para os trabalhos futuros.

1.4 *NICEBOYS*

De forma sucinta, a *NICEBOYS* se resume em um coletivo de artistas que se movem em prol da arte. Focados no estilo de *street wear*, a *NICEBOYS* tem conquistado espaço em algumas regiões do Brasil, sobretudo em Belo Horizonte. No entanto, existem

grandes barreiras para compradores de outros estados por falta de canais de comunicação e logística de transporte. Além disso, uma marca exige visibilidade para que o negócio tenha sucesso, assim, torna-se mister possuir uma plataforma *online* para promover ainda mais a sua imagem em meio ao mercado.

Durante a pandemia, se tornou nítido que todas as empresas que não tinham boas soluções tecnológicas tiveram grandes prejuízos, resultando na insatisfação dos seus clientes, o que implica em queda nas vendas e consequentemente o lucro da empresa. Mediante ao exposto, acredita-se que uma boa estratégia para o momento da marca seria investir no desenvolvimento de um *e-commerce* para que seja possível resolver ou ao menos mitigar algumas barreiras que os impedem de crescer.

2 Revisão bibliográfica

A fim de contextualizar o leitor acerca do desenvolvimento desse projeto, este capítulo de revisão tem como objetivo a exposição de alguns conceitos utilizados para metodologia e implementações, tecnologias utilizadas e trabalhos relacionados que foram utilizados ao longo de todo processo de desenvolvimento da plataforma.

2.1 Referencial teórico

No referencial teórico deste trabalho, serão fundamentados e explorados diversos conceitos essenciais para o entendimento e desenvolvimento do projeto. O *e-business* e *e-commerce*, que compreende a realização de transações comerciais online, será abordado como elemento central para a compreensão das práticas de negócios digitais. Além disso, o *Business Model Canvas* (BMC) será empregado como uma ferramenta estratégica para a visualização e análise do modelo de negócios proposto. A criação de personas, representações fictícias de usuários, será utilizada para direcionar as estratégias de design e desenvolvimento, assegurando uma abordagem centrada no usuário. A prototipação, por sua vez, surgirá como uma metodologia essencial para a materialização e avaliação iterativa das soluções propostas, garantindo um processo eficiente e alinhado às necessidades identificadas. A integração destes conceitos visa proporcionar uma base teórica robusta e abrangente para o desenvolvimento do projeto, contribuindo para a eficácia e sucesso das iniciativas propostas no âmbito do *e-business*.

2.1.1 *E-business e e-commerce*

Segundo Laudon e Laudon (2016) o *E-business* (ou negócio eletrônico) é um termo amplo que abrange todas as atividades comerciais que ocorrem pela internet ou por meio de sistemas eletrônicos. Envolve o uso da tecnologia da informação e comunicação para realizar transações comerciais, gerenciar processos de negócios e interagir com clientes, fornecedores e parceiros. O *e-business* inclui uma ampla gama de atividades, como *marketing online*, atendimento ao cliente via internet, colaboração eletrônica com fornecedores e parceiros, automação de processos internos, entre outros.

Por outro lado, o *e-commerce* (ou comércio eletrônico) é definido por Laudon e Laudon (2016) como uma parte específica do *e-business* que se refere à compra e venda de produtos ou serviços pela internet. O *e-commerce* envolve a transação eletrônica de bens e serviços entre empresas (B2B), empresas e consumidores (B2C), consumidores e consumidores (C2C) e até mesmo governos e empresas ou consumidores (B2G ou C2G).

No *e-commerce*, os produtos são exibidos em plataformas *online*, onde os consumidores podem pesquisar, selecionar, adicionar ao carrinho, fazer o pagamento e concluir a compra. Isso geralmente envolve o uso de sistemas de pagamento eletrônico e logística para entrega dos produtos aos clientes.

O *e-commerce* tem se tornado cada vez mais popular devido à conveniência, variedade de produtos, facilidade de comparação de preços e acesso global que oferece aos consumidores. Ele também proporciona vantagens para as empresas, permitindo alcance de novos mercados, redução de custos operacionais e melhorias na eficiência.

Em resumo, o *e-business* é um conceito mais amplo que abrange todas as atividades comerciais eletrônicas, enquanto o *e-commerce* é uma parte específica do *e-business* que se concentra na compra e venda de produtos ou serviços pela internet. Ambos desempenham um papel fundamental na economia digital atual.

2.1.2 *Business Model Canvas*

O *Business Model Canvas*, ou Quadro de Modelo de Negócios, é uma ferramenta estratégica utilizada para descrever e analisar o modelo de negócios de uma empresa de forma visual e concisa. Foi desenvolvido por Alexander Osterwalder e Yves Pigneur (OSTERWALDER; PIGNEUR, 2010), e tem sido amplamente adotado como uma metodologia eficaz para desenhar, comunicar e inovar modelos de negócios.

O BMC é representado por um quadro dividido em nove blocos principais, que capturam os principais aspectos do modelo de negócios. Esses blocos são os seguintes:

1. Segmentos de Clientes: descreve os diferentes grupos de clientes ou segmentos de mercado que a empresa pretende atender.
2. Proposta de Valor: define os produtos ou serviços oferecidos pela empresa e como eles agregam valor aos clientes.
3. Canais de Distribuição: descreve os canais pelos quais a empresa se comunica e entrega sua proposta de valor aos clientes.
4. Relacionamento com o Cliente: aborda como a empresa interage e estabelece relacionamentos com os clientes ao longo do ciclo de vida do cliente.
5. Fontes de Receita: identifica as principais fontes de receita da empresa, ou seja, como ela gera receita a partir de seus produtos ou serviços.
6. Recursos Chave: engloba os recursos necessários para operar o modelo de negócios, incluindo recursos físicos, financeiros, intelectuais e humanos.

7. Atividades Chave: representa as principais atividades que a empresa realiza para entregar sua proposta de valor, como produção, marketing, logística, entre outras.

8. Parcerias Chave: descreve as parcerias estratégicas ou alianças que a empresa estabelece para fortalecer seu modelo de negócios.

9. Estrutura de Custos: engloba todos os custos associados à operação do modelo de negócios, incluindo custos fixos e variáveis.

Ao preencher cada bloco do BMC, a empresa pode ter uma visão holística e compreensão clara de seu modelo de negócios, identificando pontos fortes, fraquezas, oportunidades e ameaças. Além disso, o BMC facilita a análise e a geração de ideias para a inovação e a melhoria do modelo de negócios.

O uso do BMC é especialmente valioso para empreendedores, *startups* e empresas que desejam desenvolver, adaptar ou comunicar seu modelo de negócios de maneira eficaz. Ele permite uma representação visual e fácil de entender do modelo de negócios, facilitando a colaboração entre as partes interessadas e o alinhamento estratégico da organização.

2.1.2.1 Personas

De acordo com Cooper, Reimann e Cronin (2007) as personas são representações fictícias de pessoas reais, criadas com base em informações demográficas, comportamentais, necessidades, desejos e características específicas. Elas são utilizadas como ferramenta no design centrado no usuário e na estratégia de marketing para compreender e se conectar melhor com o público-alvo de um produto, serviço ou iniciativa. A ideia é desenvolver perfis detalhados e realistas que representem diferentes tipos de usuários que interagem com um produto ou serviço.

As personas geralmente incluem informações como nome, idade, ocupação, interesses, comportamentos, objetivos, necessidades e desafios. Além disso, elas podem ter características demográficas, como sexo, localização geográfica, nível de educação e estado civil. Essas informações são agregadas para criar uma visão geral dos usuários-alvo e ajudar a equipe a entender suas motivações, expectativas e preferências.

O uso de personas tem várias vantagens. Elas ajudam a equipe a ter empatia com os usuários, a compreender suas necessidades e a tomar decisões de design mais informadas. As personas também auxiliam na identificação de lacunas nas ofertas atuais e na criação de soluções mais alinhadas com as necessidades do público-alvo. Além disso, elas facilitam a comunicação efetiva entre os membros da equipe e evitam a tomada de decisões baseadas em suposições ou opiniões pessoais.

As personas são usadas em diversas áreas, como design de produtos, desenvolvimento de *software*, marketing, publicidade e estratégia de negócios. Elas fornecem uma

representação concreta dos usuários, permitindo que as empresas projetem produtos e serviços mais personalizados e relevantes, direcionem suas campanhas de marketing de forma mais precisa e melhorem a experiência geral do cliente.

No entanto, é importante ressaltar que as personas são construções fictícias baseadas em informações reais. Elas não representam indivíduos reais, mas sim um conjunto de características e necessidades comuns encontradas em determinado grupo de usuários. Portanto, é necessário realizar pesquisas contínuas e atualizar as personas conforme novas informações e *insights* são obtidos.

2.1.3 Prototipação

Como Martins (2007) afirmou, a interface gráfica é necessária em todo produto de *software*, visto que ela é o ponto de contato com os usuários. Portanto, a prototipação se revela como uma técnica amplamente utilizada no processo de design e desenvolvimento de produtos, serviços e sistemas. Consiste na criação de modelos ou representações tangíveis de um conceito ou ideia para testar, iterar e validar o seu funcionamento, usabilidade e aceitação pelo usuário.

O objetivo da prototipação é permitir que os designers, engenheiros e profissionais envolvidos no processo de desenvolvimento obtenham *feedback* dos usuários de forma rápida e econômica, antes de investir recursos significativos na produção final. Essa abordagem iterativa e orientada pelo usuário ajuda a identificar e corrigir problemas, melhorar a experiência do usuário e otimizar a solução final.

Martins (2007) ainda discute sobre os diferentes tipos de protótipos, que variam em complexidade e fidelidade. Protótipos de baixa fidelidade são geralmente rápidos e fáceis de construir, usando materiais simples como papel, cartolina ou prototipagem digital. Eles são usados nas fases iniciais do processo de design para explorar conceitos, comunicar ideias e obter *feedback* sobre a direção geral do projeto.

Por outro lado, os protótipos de alta fidelidade são mais detalhados e representam com maior precisão a solução final. Eles podem ser construídos com materiais mais avançados, como plástico, madeira ou metal, e podem incluir funcionalidades interativas ou simular a aparência e o comportamento do produto final. Esses protótipos são usados em fases mais avançadas do processo de design para testar especificações técnicas, validar a usabilidade e realizar testes de aceitação com os usuários.

A prototipação pode ocorrer em diferentes estágios do desenvolvimento de um projeto, desde o conceito inicial até a fase de refinamento final. Ao longo desse processo, os protótipos são refinados e atualizados com base no *feedback* dos usuários e nas necessidades identificadas. Isso permite a iteração contínua, onde cada versão do protótipo é testada e refinada, resultando em um produto final mais alinhado às necessidades e expectativas do

usuário.

A prototipação também promove a colaboração e a comunicação entre as partes interessadas e envolvidas no projeto, pois oferece uma representação concreta e tangível das ideias. Além disso, a prototipação ajuda a mitigar riscos e incertezas, reduzindo a possibilidade de retrabalho e custos desnecessários durante o desenvolvimento.

Em resumo, a prototipação é uma técnica essencial no processo de design e desenvolvimento, permitindo a criação de modelos tangíveis para testar, iterar e validar soluções antes de sua implementação final. Ela ajuda a obter *feedback* do usuário, aperfeiçoar a usabilidade e melhorar a experiência do usuário, resultando em produtos e serviços mais eficazes e satisfatórios.

2.2 Tecnologias

Nessa seção serão descritas sobre as principais tecnologias utilizadas nesse projeto.

2.2.1 *Firebase*

Firebase [1] é uma plataforma de desenvolvimento de aplicativos móveis e *web* baseada em nuvem, fornecida pela *Google*. Ela oferece um conjunto abrangente de serviços e ferramentas para ajudar os desenvolvedores a criar, melhorar e escalar aplicativos de maneira eficiente.

A principal proposta de valor do *Firebase* é permitir que os desenvolvedores se concentrem na lógica do aplicativo, enquanto a plataforma cuida da infraestrutura necessária para seu funcionamento. Com o *Firebase*, os desenvolvedores podem aproveitar uma série de recursos prontos para uso, como autenticação de usuários, armazenamento em nuvem, banco de dados em tempo real, mensagens e notificações, hospedagem de aplicativos, análise de dados, testes A/B e muito mais.

A utilização do *Firebase* simplifica o processo de desenvolvimento, permitindo que os desenvolvedores utilizem uma infraestrutura confiável e escalável, sem a necessidade de gerenciar e configurar servidores ou se preocupar com tarefas de manutenção. Isso acelera o tempo de desenvolvimento, reduz custos e permite que os desenvolvedores se concentrem na experiência do usuário e nas funcionalidades do aplicativo. Com seus recursos prontos para uso e fácil integração, o *Firebase* ajuda os desenvolvedores a criar aplicativos de alta qualidade, escaláveis e com recursos avançados, sem a necessidade de gerenciar uma infraestrutura complexa.

[1] https://firebase.google.com/

2.2.2 *Typescript*

"*JavaScript* é uma linguagem de programação de alto nível, mais conhecida como a linguagem de *script* para páginas *web*, mas é usada também em vários outros ambientes sem *browser*, tais como *Node.js*, *Apache CouchDB* e *Adobe Acrobat*" (JAVASCRIPT, 2020).

Mediante a citação acima, surge o *TypeScript* [2] que é uma linguagem de programação de código aberto desenvolvida pela *Microsoft*. Ela é uma extensão do *JavaScript*, adicionando recursos de tipagem estática e outros recursos avançados ao *JavaScript* tradicional. O *TypeScript* foi projetado para tornar o desenvolvimento de aplicativos *JavaScript* mais robusto, escalável e eficiente, especialmente em projetos de grande porte.

Uma das principais características do *TypeScript* é a adição de tipos estáticos ao *JavaScript*. Isso significa que os desenvolvedores podem declarar o tipo de cada variável, parâmetro de função, propriedade de objeto e retorno de função. Essas declarações de tipo são verificadas em tempo de compilação, o que ajuda a evitar erros comuns e facilita a detecção de problemas antes mesmo de executar o código. Além disso, o *TypeScript* oferece recursos avançados, como inferência de tipo, união de tipos, tipos genéricos e muito mais.

Outro benefício do *TypeScript* é o suporte para recursos modernos do *ECMAScript*, como classes, módulos, *arrow functions*, desestruturação, entre outros. Isso permite que os desenvolvedores aproveitem as últimas funcionalidades do *JavaScript*, mantendo a compatibilidade com versões anteriores.

O *TypeScript* também inclui um compilador que traduz o código *TypeScript* em *JavaScript* puro, que pode ser executado em qualquer ambiente *JavaScript*. Isso significa que os aplicativos desenvolvidos em *TypeScript* podem ser executados em navegadores, servidores e em qualquer ambiente onde o *JavaScript* seja suportado.

Além da verificação de tipo estático, o *TypeScript* oferece ferramentas de desenvolvimento poderosas, como *IntelliSense*, refatoração de código, suporte a depuração e uma ampla gama de bibliotecas e definições de tipo disponíveis na comunidade.

O uso do *TypeScript* traz benefícios significativos para projetos de desenvolvimento de *software*. Ele ajuda a reduzir erros e aprimorar a qualidade do código, facilita a manutenção e o refatoramento, melhora a produtividade dos desenvolvedores e facilita a colaboração em equipes. Além disso, o *TypeScript* é amplamente adotado pela comunidade de desenvolvedores e possui suporte robusto em várias ferramentas e *frameworks* populares, como *Angular*, *React*, *Vue.js*, *Node.js* e outros.

[2] https://www.typescriptlang.org/

2.2.3 *Node*

Node.js [3], comumente referido como *Node*, é um ambiente de tempo de execução de código aberto baseado no motor *JavaScript* V8 da *Google Chrome*. Ele permite que os desenvolvedores executem código *JavaScript* no lado do servidor, em vez de apenas no navegador, ampliando assim a capacidade do *JavaScript* para além do cliente da *web*.

O *Node.js* possui uma arquitetura orientada a eventos e baseada em assincronia, o que significa que ele é capaz de lidar com um grande número de conexões simultâneas sem bloquear a execução do código. Isso é especialmente vantajoso em cenários onde a escalabilidade e o desempenho são críticos, como em aplicativos em tempo real, servidores da *web*, *APIs* e microsserviços.

Uma das principais características do *Node* é a capacidade de realizar operações de entrada e saída (I/O) de forma assíncrona, sem bloquear a execução do programa. Isso é alcançado por meio do uso de chamadas de retorno (*callbacks*), promessas ou recursos mais recentes, como *async/await*. Essa abordagem assíncrona permite que o *Node* seja altamente eficiente e escalável, lidando com um grande número de solicitações simultâneas de forma eficaz.

"*Node.js* usa um modelo de entrada ou saída não bloqueante e orientado por eventos, isso o torna leve e eficiente, perfeito para tempo real com muitos dados aplicativos que são executados em dispositivos. *Node.js*, inicialmente desenvolvido por Ryan Dahl, também fornece um *Read-Eval-Print-Loop* (REPL) ambiente para teste interativo" (JS; JS, 2020).

O *Node.js* também possui um sistema de gerenciamento de pacotes integrado chamado npm (*Node Package Manager*). O npm é um repositório abrangente de bibliotecas e módulos de código-fonte abertos que podem ser facilmente instalados e usados em projetos *Node*. Isso promove a reutilização de código, a colaboração entre desenvolvedores e a criação mais rápida de aplicativos por meio da integração de pacotes de terceiros.

Além disso, o *Node.js* possui uma vasta comunidade de desenvolvedores e é suportado por uma grande variedade de *frameworks* e bibliotecas populares, como *Express.js*, *Socket.io*, *Sequelize*, entre outros. Essas ferramentas facilitam o desenvolvimento de aplicativos *web*, *APIs RESTful*, aplicações em tempo real, bancos de dados e integração com outros serviços.

O uso do *Node.js* traz várias vantagens para o desenvolvimento de aplicativos, como a capacidade de compartilhar código entre o cliente e o servidor usando *JavaScript*, a escalabilidade horizontal e vertical, a fácil integração com bancos de dados não relacionais e relacionais, além da vasta disponibilidade de bibliotecas e módulos para acelerar o desenvolvimento.

[3] https://nodejs.org/

2.2.4 *React*

React [4], também conhecido como *React.js* ou *ReactJS*, é uma biblioteca *JavaScript* de código aberto desenvolvida pelo *Facebook*. Ela é amplamente utilizada para criar interfaces de usuário (UI) interativas e responsivas em aplicativos *web* de uma página (SPAs) e sites de várias páginas.

"*React* utiliza características de *JavasSript* moderno para muitos de seus padrões. O maior desvio do *React* para o *JavaScript* dá-se pela utilização da sintaxe *JSX*. O *JSX* estende a sintaxe padrão do *Javascript* hablitando a utilizar código similar a *HyperText Markup Language* (HTML) que pode viver lado a lado ao *JSX*" (MDN, 2020).

A principal proposta de valor do *React* é a construção de UIs componentizadas. Ele introduz o conceito de componentes reutilizáveis, que são blocos independentes de código que encapsulam o estado (dados) e o comportamento de uma parte específica da interface do usuário. Os componentes podem ser compostos e combinados para criar interfaces complexas, permitindo que os desenvolvedores criem aplicativos escaláveis e de fácil manutenção.

O *React* utiliza uma abordagem declarativa para a construção de interfaces, o que significa que os desenvolvedores descrevem como a interface deve ser exibida com base no estado atual e não precisam se preocupar com manipulações diretas no DOM (*Document Object Model*). Isso melhora a eficiência e a performance do aplicativo, pois o *React* é capaz de atualizar apenas as partes necessárias da interface quando o estado é alterado.

Outro conceito fundamental no *React* é o Virtual DOM. Ele é uma representação virtual da estrutura do DOM atualizado pelo *React*. Quando o estado de um componente muda, o *React* compara o Virtual DOM com o DOM real, identifica as diferenças e aplica apenas as alterações necessárias no DOM real. Isso minimiza as manipulações diretas no DOM, o que pode ser um processo computacionalmente intensivo, resultando em uma melhor performance do aplicativo.

O *React* também suporta a reatividade bidirecional de dados através do uso de *props* (propriedades) e *state* (estado). As *props* são utilizadas para passar dados de um componente pai para um componente filho, enquanto o *state* é usado para gerenciar o estado interno de um componente. Quando o *state* é atualizado, o *React* re-renderiza o componente e suas partes dependentes, mantendo a interface sempre sincronizada com os dados.

Além disso, o *React* possui uma comunidade ativa e um ecossistema rico de ferramentas, bibliotecas e *frameworks* complementares, como o *React Router* para roteamento, o *Redux* para gerenciamento de estado global, o *Next.js* para renderização do lado do servidor (SSR) e muitos outros.

[4] https://react.dev/

2.2.5 *Figma*

Figma [5] é uma ferramenta de design de interface do usuário (UI) baseada na *web*, que permite aos designers criar e colaborar em projetos de design de forma eficiente. Ela foi lançada em 2016 pela *Figma Inc.* e se tornou uma escolha popular entre os profissionais de design.

A principal característica do *Figma* é a sua abordagem baseada na nuvem. Isso significa que os arquivos de design são armazenados na nuvem e podem ser acessados e editados por várias pessoas simultaneamente, facilitando a colaboração entre designers, desenvolvedores e outras partes interessadas. O *Figma* permite que várias pessoas trabalhem no mesmo projeto em tempo real, tornando o processo de design mais ágil e eficiente.

O *Figma* oferece uma interface de usuário intuitiva e recursos poderosos para a criação de designs de alta qualidade. Ele suporta a criação de interfaces para aplicativos móveis, *web*, protótipos interativos e outros tipos de projetos de design. A ferramenta possui recursos avançados de desenho vetorial, como criação de formas, edição de camadas, importação de imagens, criação de estilos de texto e muito mais.

Além disso, o *Figma* possui recursos de prototipagem integrados, permitindo que os designers criem protótipos interativos para testar a funcionalidade e a experiência do usuário. Os protótipos podem ser compartilhados com a equipe de desenvolvimento ou com os clientes para obter *feedback* e validar as ideias de design antes da implementação.

Outra vantagem do *Figma* é a sua capacidade de integração com outras ferramentas e serviços. Ele permite a importação de recursos, como ícones e imagens, diretamente de bancos de dados de terceiros, como o *Iconfinder* e o *Unsplash*. Além disso, o *Figma* pode ser integrado a plataformas de gerenciamento de projetos, como o *Jira* e o *Trello*, facilitando a sincronização e a organização do trabalho.

O *Figma* é amplamente adotado pela comunidade de design devido à sua facilidade de uso, recursos avançados e capacidade de colaboração em tempo real. Ele é usado por designers de diferentes áreas, desde pequenas equipes de design até grandes empresas. Além disso, o *Figma* oferece planos gratuitos e pagos, tornando-o acessível para uma ampla gama de profissionais.

2.3 Trabalhos relacionados

Na busca por fundamentar e enriquecer o presente trabalho, foram utilizados dois artigos que desempenharam papéis cruciais no embasamento teórico deste estudo. O primeiro, intitulado "Desenvolvimento e Manutenção de um Marketplace para a Indústria da Moda" (OLIVEIRA, 2022), forneceu uma rica gama de informações relacionadas

[5] https://www.figma.com/

ao desenvolvimento do sistema com ferramentas e estratégias semelhantes às que serão abordadas neste trabalho. As análises detalhadas e as conclusões sólidas deste artigo foram de valor inestimável ao aprofundar minha compreensão sobre o desenvolvimento e implementação de um sistema de vendas *online*.

O segundo artigo, intitulado "E-commerce no Brasil: revisão sistemática de literatura de 2011 a 2021" (COSTA et al., 2021), destacou-se por seu estudo abrangente sobre as atividades de comércio e serviços na internet. Este trabalho ofereceu uma visão aprofundada das dinâmicas do mercado *online*, destacando a crescente importância do comércio eletrônico para marcas estabelecidas. As conclusões apresentadas neste artigo fortaleceram substancialmente a argumentação para a implementação de um e-commerce para a marca em questão, embasando a relevância estratégica desta decisão.

Ao integrar as descobertas destes dois artigos, foi possível construir uma base sólida e abrangente para o desenvolvimento do presente trabalho, contribuindo significativamente para a robustez das análises e argumentos apresentados ao longo desta monografia.

3 Metodologia

Nesta seção o texto entrará em uma jornada estratégica e prática para a construção integral do projeto. A metodologia adotada é intrinsecamente ligada ao propósito de desenvolver e aprimorar o entendimento do negócio em questão. Inicialmente, será empregado a ferramenta do *Business Model Canvas* para modelar o negócio de maneira holística, delineando de forma clara os elementos chave que compõem a estrutura da organização. Este modelo oferecerá uma visão abrangente das principais áreas.

Em seguida, o texto explorará o universo das personas, criando representações fictícias, mas fundamentadas, dos potenciais usuários ou clientes. Este exercício contribuirá para a compreensão profunda das necessidades, desejos e características do público-alvo, informando decisões estratégicas em consonância com as demandas reais do mercado.

A etapa seguinte envolverá a materialização das ideias através da criação de protótipos. Estes modelos iniciais serão desenvolvidos com o intuito de visualizar e testar as funcionalidades e experiências propostas, proporcionando *insights* valiosos para refinamento e aprimoramento. O processo de criação garantirá flexibilidade e adaptabilidade ao longo do desenvolvimento.

Assim, esta seção de metodologia delineia não apenas os passos práticos iniciais para a execução do projeto, mas também ressalta a abordagem estratégica que fundamenta cada escolha metodológica, visando à construção de um trabalho robusto, inovador e alinhado com as necessidades reais do mercado e dos usuários.

3.1 Modelagem do negócio

No que concerne a modelagem do negócio, a abordagem estratégica se baseou na utilização do *Business Model Canvas* (Figura 1), uma ferramenta eficaz que permitiu estruturar e visualizar de maneira abrangente os elementos fundamentais do empreendimento. Por meio do BMC, foi possível explorar e delinear de forma sistêmica aspectos cruciais, como proposta de valor, segmentos de clientes, canais de distribuição, relacionamento com clientes, fontes de receita, recursos chave, atividades principais, parcerias chave e estrutura de custos.

A utilização do *Canvas* não apenas proporcionou uma visão consolidada do modelo de negócio, mas também facilitou a identificação de oportunidades de inovação e a compreensão das inter-relações entre os diferentes componentes. O processo de modelagem, orientado pelo BMC, serviu como alicerce para a definição estratégica da empresa, proporcionando clareza na concepção e permitindo uma abordagem mais informada e

eficiente na busca pelos objetivos propostos. Este método de modelagem capacitou o projeto a antecipar desafios, otimizar recursos e posicionar a marca de forma alinhada com as expectativas e necessidades do público-alvo.

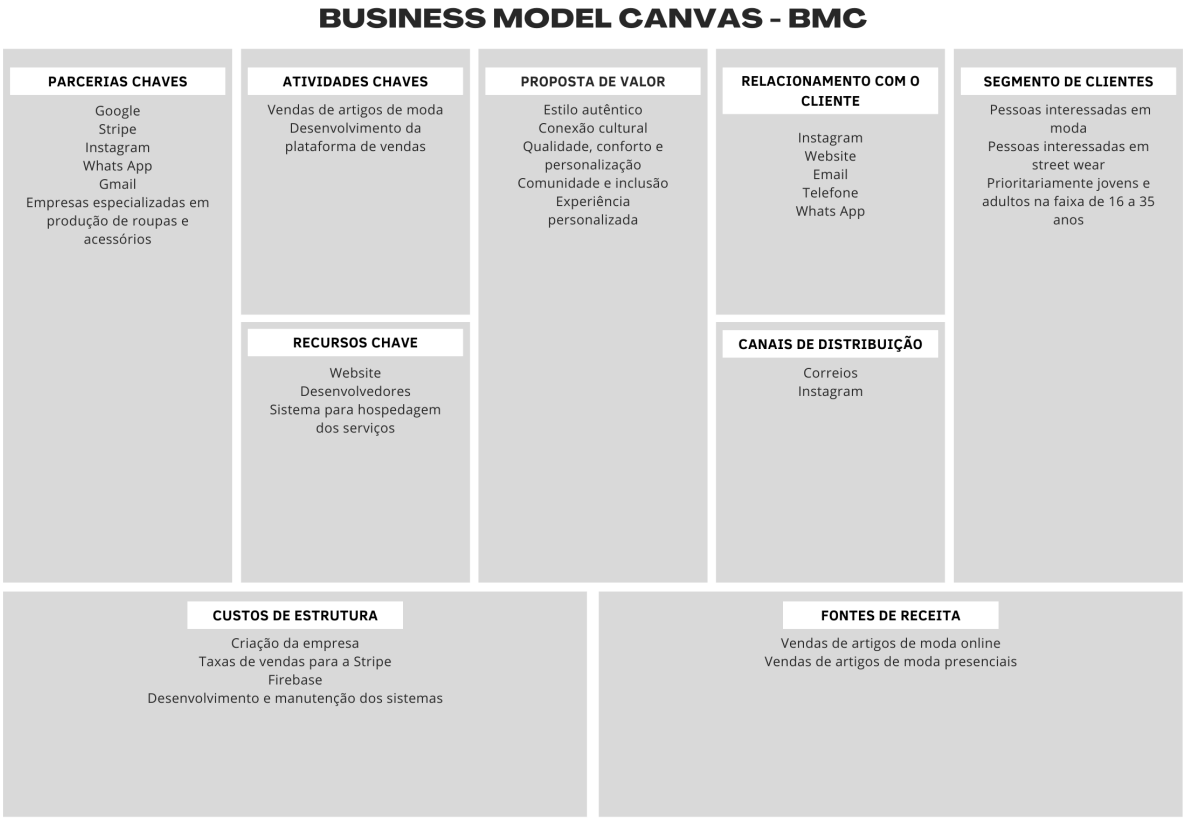

Figura 1 – *Bussiness Model Canvas*

Fonte: Elaborado pelo autor

3.2 Personas

A construção de personas não apenas desempenha um papel crucial, mas se revela como a base essencial para uma compreensão aprofundada e empática do público-alvo. Nesse sentido, a criação de personas transcende uma simples estratégia, é uma ferramenta poderosa que permite mergulhar nas diferentes perspectivas e desafios enfrentados pelos potenciais clientes da marca.

Ao elaborar personas como Gabriel, Virginia, Lucas e Juliana, o trabalho visa criar personificações tangíveis daqueles que a empresa almeja servir. Essa abordagem, intrinsecamente focada no usuário, não apenas informará as estratégias de marketing, mas também guiará o processo de prototipagem e desenvolvimento. Compreender as nuanças e motivações por trás das personas permite antecipar desafios, adaptar soluções e, acima de tudo, criar uma experiência que ressoe autenticidade e valor para cada indivíduo representado.

- Persona 1 - Gabriel de Oliveira

  Descrição: Skatista apaixonado por moda, Gabriel, 21 anos, reside em Alcobaça, Bahia. Sua afinidade com a marca originou-se por meio da música, despertando seu interesse em adquirir produtos exclusivos. Entretanto, enfrenta desafios logísticos, já que os canais de transporte e venda atuais não atendem plenamente às suas necessidades. A persona de Gabriel destaca a importância de aprimorar a acessibilidade e conveniência para consumidores de regiões distantes.

- Persona 2 - Virginia Fernandes Bastos

  Descrição: Tatuadora talentosa, Virginia, 24 anos, reside em Belo Horizonte. Seu forte interesse na marca é acompanhado por uma barreira: a dificuldade de se manter atualizada sobre novos produtos e promoções. A persona de Virginia destaca a importância de estratégias de comunicação eficazes, garantindo que os clientes, mesmo distantes, possam acompanhar facilmente as atualizações da marca.

- Persona 3 - Lucas Souza Mendes

  Descrição: Estudante de design gráfico, Lucas, 22 anos, de Porto Alegre, Rio Grande do Sul, é um entusiasta da arte urbana e música alternativa. Sua busca por peças únicas que expressem sua individualidade é constante. A persona de Lucas destaca a oportunidade de aprimorar a diversidade de produtos, incorporando elementos de design únicos e inspirados nas tendências culturais emergentes.

- Persona 4 - Juliana Costa Lima

  Descrição: Jovem empresária, Juliana, 27 anos, de São Paulo, está envolvida na cena do rap e hip-hop. Sua busca por moda que represente seu estilo vibrante é constante. A persona de Juliana destaca a importância de estratégias de marketing dinâmicas, focadas nas redes sociais e eventos culturais, para alcançar consumidores engajados na cena urbana.

Ao delinear essas personas, foi possível obter uma compreensão abrangente das diversas facetas do público-alvo, permitindo moldar estratégias que atendam às suas necessidades e fortaleçam a conexão com a marca.

3.3 Protótipos

A escolha do *Figma* como ferramenta para a criação de protótipos revelou-se estrategicamente vantajosa durante a etapa de desenvolvimento. A natureza colaborativa do *Figma* possibilitou uma eficiente interação entre as pessoas envolvidas no desenvolvimento dos protótipos, permitindo que diferentes pessoas contribuíssem simultaneamente para o design e avaliação do protótipo. A capacidade de acesso em tempo real, além da opção

de inserir comentários diretamente na plataforma, facilitou a comunicação e a troca de *feedback*, acelerando o ciclo de desenvolvimento e garantindo uma convergência mais rápida para a versão final do protótipo.

Além disso, a versatilidade do *Figma* proporciona uma experiência de usuário excepcional ao criar *wireframes* e simular a interatividade da aplicação. A funcionalidade de prototipagem permitiu não apenas visualizar a aparência final do projeto, mas também testar a navegabilidade e a usabilidade, garantindo uma representação fiel das funcionalidades planejadas. Essa abordagem refinada e detalhada foi fundamental para identificar aprimoramentos necessários, resultando em um protótipo robusto que reflete as expectativas do usuário de maneira precisa.

Se torna mister ressaltar que os protótipos foram desenvolvidos pensando tanto para os usuários de computadores e notebooks, como também para celulares e tablets, uma vez que grande parte do acesso a internet é feito por meio de aparelhos celulares. Essa preocupação com o projeto é de importância fundamental para que seja possível oferecer a melhor experiência para os usuários em qualquer plataforma que ele utilize para acessar o sistema.

A priori, as telas de carregamento (Figura 2) desenvolvidas para situações em que o sistema está realizando o carregamento de algumas funcionalidades, conteúdos de imagem, entre outros. A posteriori, as telas usualmente chamadas de *'Home'* (Figura 3), ou seja, a tela principal do projeto, essa será a primeira página que o usuário irá encontrar quando acessar a plataforma.

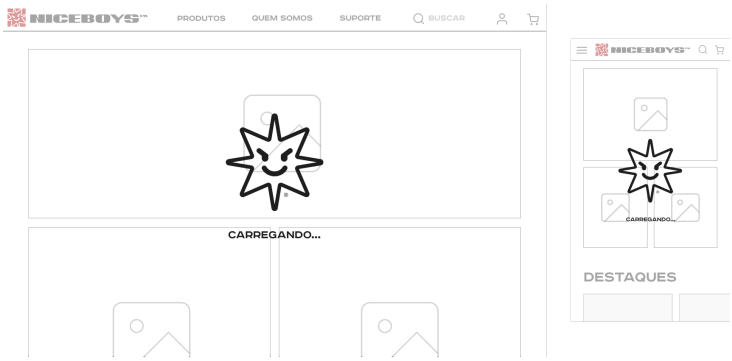

Figura 2 – Protótipo da tela de carregamento

Fonte: Elaborado pelo autor

Seguindo a sequência, também foram desenvolvidos protótipos das telas de suporte (Figura 4) para que os usuários possam relatar *bugs* ou fazer uma solicitação de ajuda com algum problema que eles possam ter vindo a enfrentar. Duas telas fundamentais também foram prototipadas, sendo elas a de produtos (Figura 5) e a tela de um único produto (Figura 6).

Por fim, foram criados não só um protótipo simples para a tela de *login* (Figura 7)

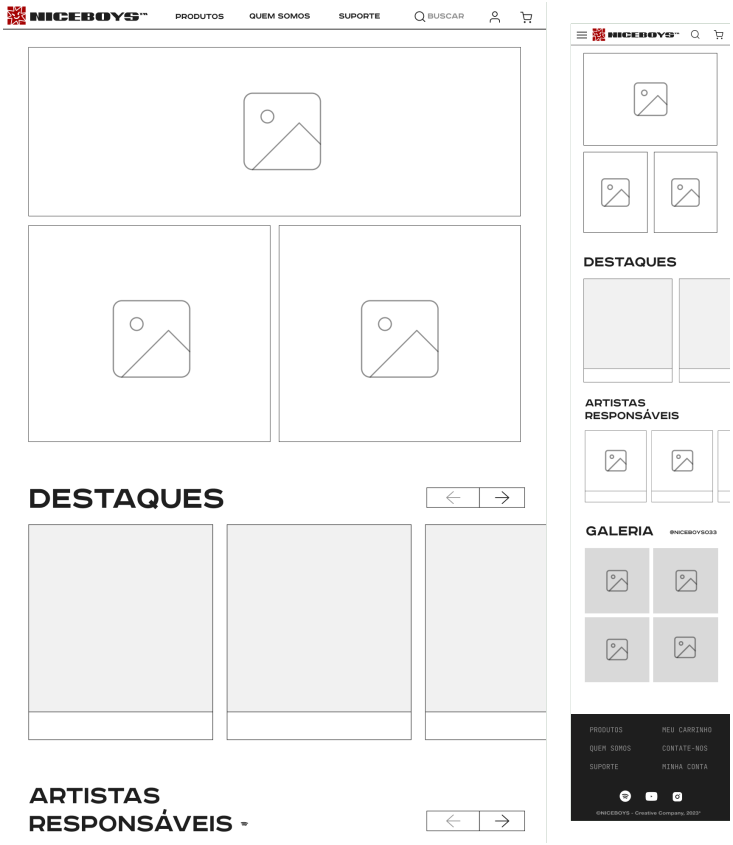

Figura 3 – Protótipo da tela *'Home'*

Fonte: Elaborado pelo autor

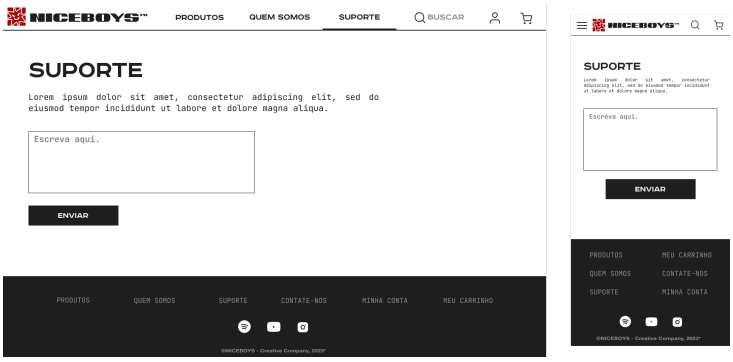

Figura 4 – Protótipo da tela de suporte

Fonte: Elaborado pelo autor

como também protótipos das telas (Figuras 8, 9) que acompanharão o usuário durante o fluxo de *checkout*.

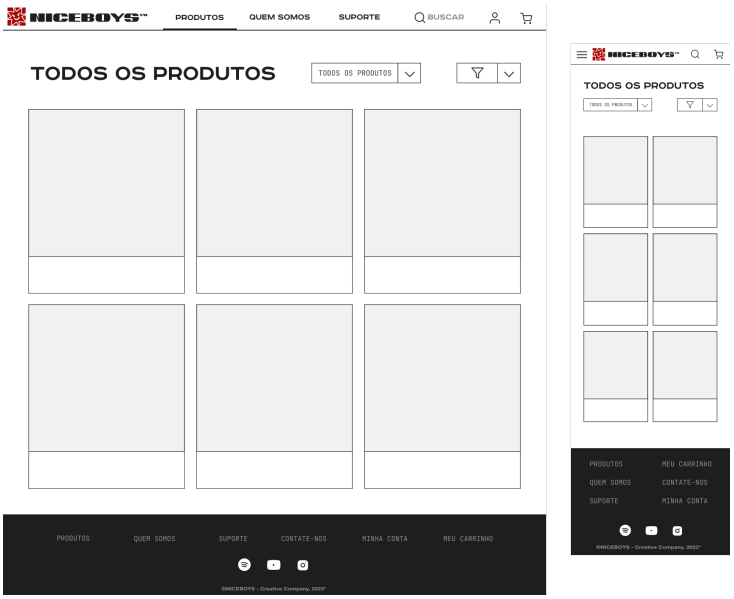

Figura 5 – Protótipo da tela de produtos

Fonte: Elaborado pelo autor

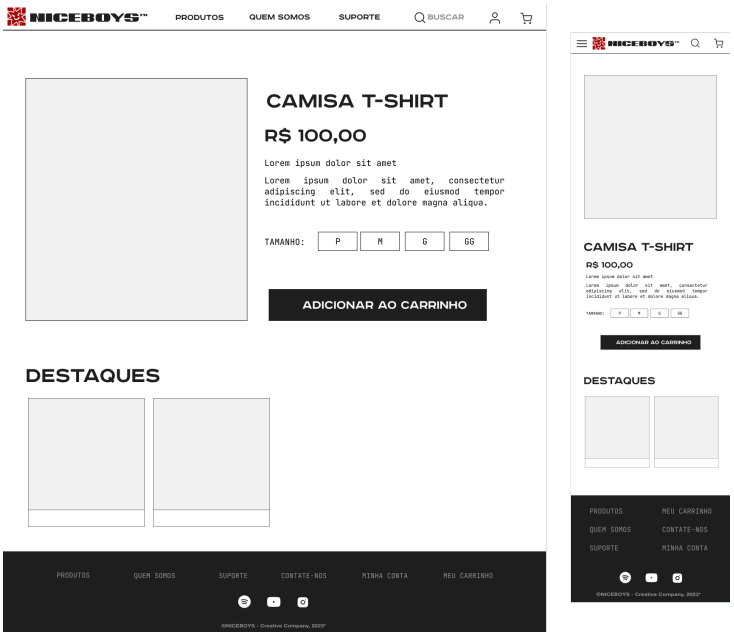

Figura 6 – Protótipo da tela de um único produto

Fonte: Elaborado pelo autor

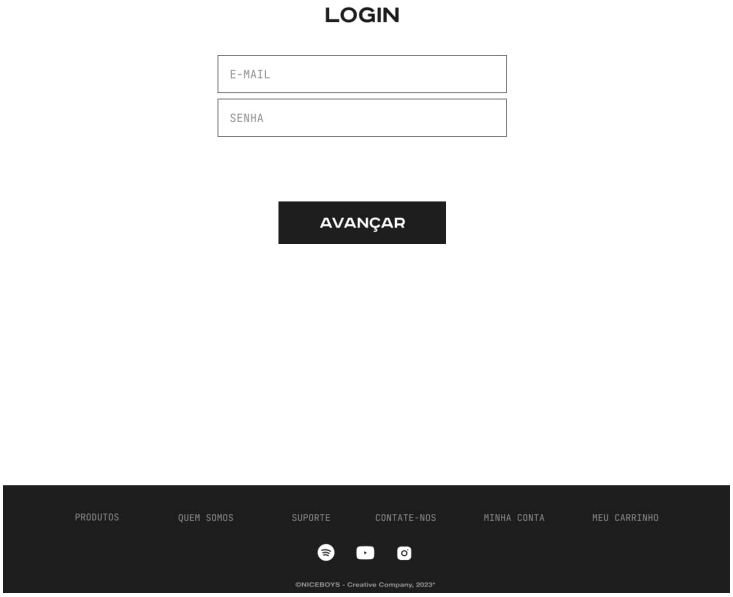

Figura 7 – Protótipo da tela de *login*

Fonte: Elaborado pelo autor

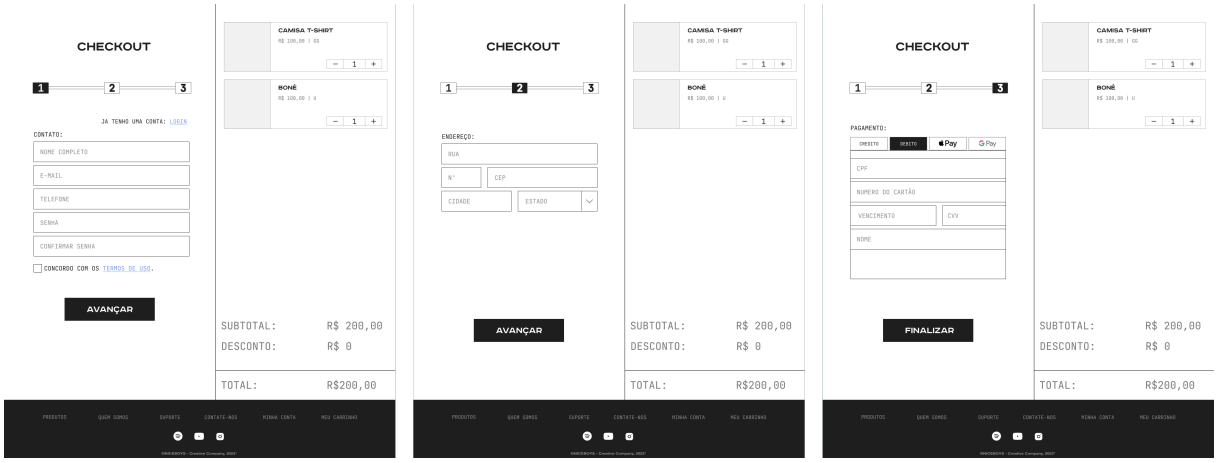

Figura 8 – Protótipo das telas no fluxo de *checkout*

Fonte: Elaborado pelo autor

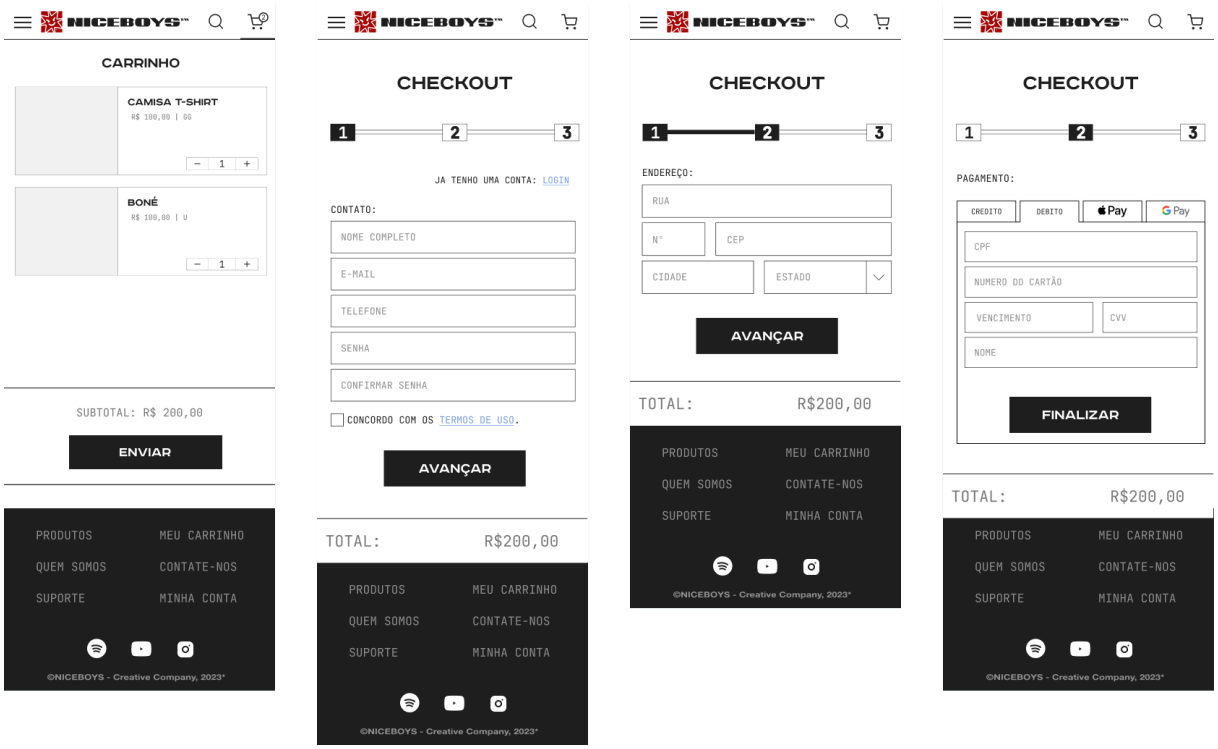

Figura 9 – Protótipo das telas de *checkout mobile*

Fonte: Elaborado pelo autor

4 Desenvolvimento

De acordo com Marcoratti (2014), o processo de *software* é crucial para o desenvolvimento de sistemas. Isso implica que desenvolver um sistema de *e-commerce* é uma jornada complexa e planejada com estratégia, onde cada escolha tecnológica e decisão arquitetural influencia a experiência do usuário e a eficiência operacional. Nesta parte, a seção irá examinar os elementos essenciais do progresso do projeto, com foco no *backend*, *frontend* e *deploy*.

A concretização de um projeto visionário depende crucialmente do desenvolvimento do *e-commerce*. A escolha de bibliotecas, configuração de ambiente e cada linha de código são passos cruciais para desenvolver uma plataforma eficaz, escalável e correspondente às exigências do mercado.

O objetivo deste capítulo é oferecer uma visão minuciosa e clara do processo de desenvolvimento utilizado. Examinando as ferramentas e tecnologias selecionadas, explicando cada decisão com base em critérios como eficácia, capacidade de escalonamento e compatibilidade com os objetivos do projeto. A procura pela maximização do conhecimento em tecnologias emergentes e a padronização de código são elementos cruciais discutidos ao longo deste capítulo.

4.1 Levantamento de requisitos

Conforme observado por Wazlawick (2016), embora a visão geral do sistema não seja rigidamente prescrita por um formato específico, é crucial que contenha elementos essenciais, os quais são discernidos por meio de entrevistas com os principais envolvidos no projeto. Portanto, se torna fulcral realizar um levantamento de requisitos para ter essa visão geral do projeto.

A análise de requisitos desempenha um papel crucial na elaboração de qualquer projeto, especialmente em monografias acadêmicas e grandes projetos de *software*. É através desse processo que se define claramente o escopo, os objetivos e as funcionalidades necessárias para atender às necessidades do usuário final. Ao compreender e documentar de forma precisa os requisitos do projeto, é possível evitar retrabalhos, garantir a satisfação do cliente e direcionar eficientemente o desenvolvimento, tornando-se, assim, uma etapa fundamental para o sucesso do trabalho.

De acordo com Machado (2018), uma análise de requisitos eficaz envolve a identificação e a especificação, representando a fase primordial no ciclo de desenvolvimento de sistemas. Embora nem todos os requisitos possam ser completamente definidos, o processo

de identificação já permite antecipar possíveis equívocos e aperfeiçoar o procedimento.

4.1.1 Requisitos funcionais

Os requisitos funcionais (Tabela 1) referem-se às especificações das funções ou serviços que o sistema deve fornecer. Eles descrevem o comportamento do sistema em resposta a entradas específicas do usuário e condições operacionais. Filho (2003) afirma que os comportamentos apresentados por um programa ou sistema diante da ação do usuário é definido como requisitos funcionais, esses requisitos são essenciais para determinar como o sistema deve funcionar para atender às necessidades dos usuários. No contexto de um *e-commerce*, os requisitos funcionais podem incluir funcionalidades como cadastro de usuários, pesquisa de produtos, carrinho de compras, processamento de pedidos, entre outros. Eles formam a base para o desenvolvimento e teste do sistema, garantindo que ele atenda aos objetivos do negócio e às expectativas dos clientes.

Tabela 1 – Tabela de requisitos funcionais

Fonte: Elaborado pelo autor

Requisito	**Descrição**
Cadastro de Usuários	Os clientes devem poder criar contas com informações pessoais, como nome, endereço, e-mail e senha.
Catálogo de Produtos	Deve haver uma interface para exibir os produtos disponíveis, incluindo imagens, descrições, preços e disponibilidade em estoque.
Pesquisa e Ordenação	Os usuários devem poder pesquisar produtos por nome, além de ordenar os resultados por preço.
Carrinho de Compras	Os clientes devem poder adicionar itens ao carrinho, visualizar o total da compra e fazer alterações antes de partir para o *checkout*.
Processo de *Checkout*	Deve haver um fluxo claro e intuitivo para os clientes inserirem informações de entrega, escolherem métodos de pagamento e finalizarem suas compras.
Gerenciamento de Pedidos	Os usuários devem poder visualizar o histórico de pedidos e acompanhar o status dos pedidos.
Sistema de Pagamento	Deve integrar com um *gateway* seguro de pagamento, sendo *Stripe* o escolhido.
Gerenciamento de Estoque	O administrador deve ter o poder de cadastrar, excluir, editar ou desativar os produtos que serão exibidos no catálogo de produtos.
Funcionalidades de Cupons	Deve ser possível aplicar descontos com cupons promocionais.

4.1.2 Requisitos não-funcionais

Os requisitos não-funcionais (Tabela 2) descrevem as características do sistema que não se referem diretamente às suas funcionalidades, mas sim a atributos de qualidade como desempenho, segurança, usabilidade e escalabilidade. Esses requisitos são igualmente importantes, pois impactam diretamente na experiência do usuário e na eficácia do sistema como um todo. Em um *e-commerce*, os requisitos não funcionais podem abranger aspectos como tempo de carregamento das páginas, segurança dos dados do cliente, compatibilidade com diferentes navegadores e dispositivos, entre outros.

"Os requisitos ainda podem ser classificados em obrigatórios e desejados, ou seja, aqueles que devem ser obtidos de qualquer maneira e aqueles que podem ser obtidos caso isso não cause maiores transtornos no processo de desenvolvimento" (WAZLAWICK, 2011).

Tabela 2 – Tabela de requisitos não-funcionais

Fonte: Elaborado pelo autor

Requisito	**Descrição**
Desempenho	O sistema deve ser capaz de lidar com um grande número de acessos simultâneos sem degradação significativa do tempo de resposta.
Segurança	Todas as transações e informações dos clientes devem ser protegidas por protocolos de segurança robustos, como *SSL/TLS*, para garantir a confidencialidade e integridade dos dados.
Usabilidade	A interface do usuário deve ser intuitiva e fácil de usar, proporcionando uma experiência de compra agradável e sem complicações.
Compatibilidade	O sistema deve ser compatível com uma variedade de navegadores *web* e dispositivos móveis, garantindo uma experiência consistente para todos os usuários.
Manutenibilidade	O código-fonte e a infraestrutura do sistema devem ser bem modularizados para facilitar a manutenção e futuras atualizações.
Escalabilidade	O sistema deve ser projetado para escalar facilmente, permitindo aumentar a capacidade e os recursos conforme a demanda cresce ao longo do tempo.
Disponibilidade	O sistema deve estar disponível 24 horas por dia, 7 dias por semana, com tempo de inatividade mínimo para manutenção planejada.

4.1.3 Diagramas de caso de uso

Os diagramas (Figura 10) são utilizados para representar visualmente as interações entre os diferentes atores e os sistemas, identificando os principais cenários de uso. Esses diagramas fornecem uma visão clara e concisa das funcionalidades oferecidas pelo sistema, destacando as ações que os usuários podem realizar e como o sistema responde a essas ações. Ao mapear os casos de uso, é possível entender de forma abrangente os requisitos do sistema e garantir que todas as funcionalidades essenciais sejam adequadamente consideradas e implementadas.

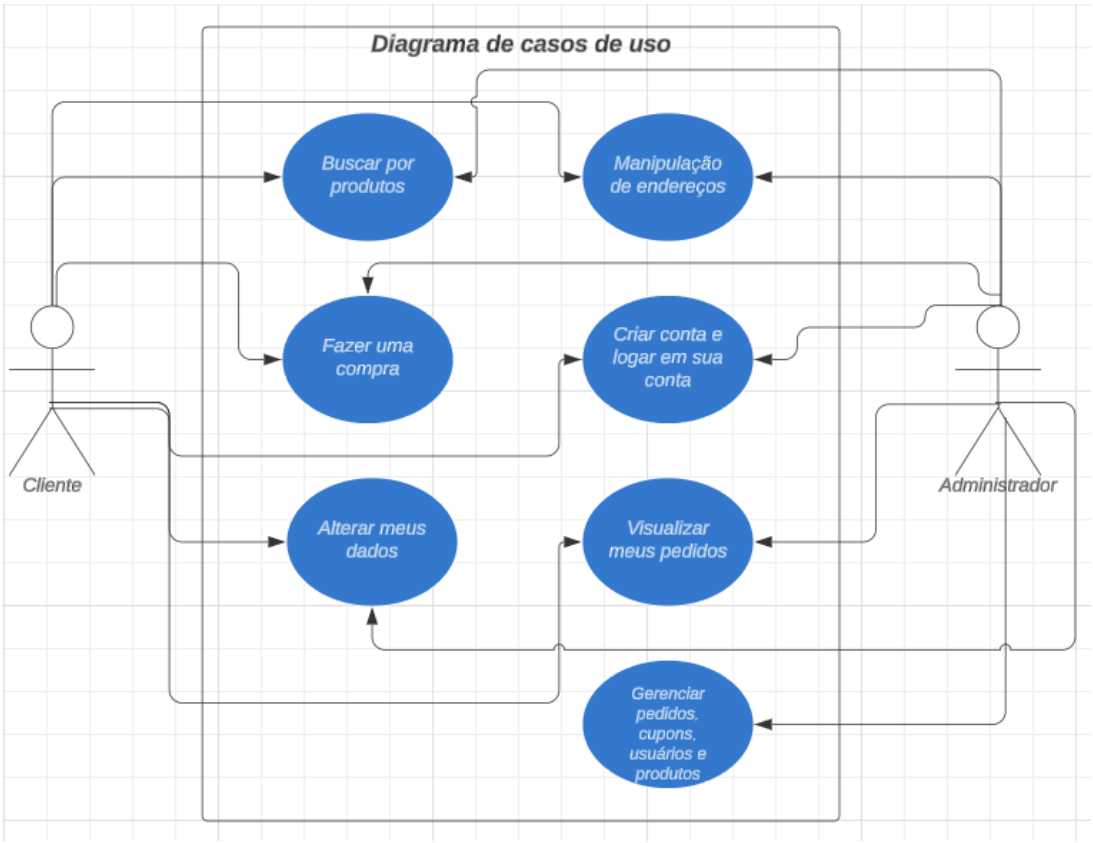

Figura 10 – Diagrama de casos de uso do sistema

Fonte: Elaborado pelo autor

4.2 *Backend*

A seguir o texto discorre sobre os aspectos que envolvem o desenvolvimento do *backend* desse sistema.

4.2.1 Ambiente de Desenvolvimento

Segundo Artia (2019), as ferramentas utilizadas em um projeto são fundamentais para seu sucesso. Em virtude do exposto, o ambiente de desenvolvimento no *backend*

foi configurado para promover uma padronização eficiente, garantindo a consistência e qualidade do código. Esse processo se iniciou estabelecendo uma base sólida com a configuração do *Eslint*, incorporando alguns *plugins* essenciais para a detecção e correção automática de potenciais erros e manutenção de uma sintaxe uniforme. Juntamente com o *Eslint*, o arquivo de configuração do *Prettier* foi ajustado para refletir os padrões visuais desejados, proporcionando uma formatação consistente ao código-fonte.

Além disso, o arquivo de configuração do *TypeScript* (*tsconfig*) foi ajustado para aperfeiçoar a compilação e as verificações estáticas de tipos durante o desenvolvimento. Essas escolhas visam não apenas assegurar a coesão do código, mas também acelerar o processo de desenvolvimento ao oferecer *feedbacks* instantâneos sobre eventuais problemas.

Com o ambiente de desenvolvimento configurado, o projeto avançou para a etapa de instalação das bibliotecas necessárias. A geração do arquivo *yarn.lock* foi realizada, garantindo consistência nas versões das dependências e facilitando a replicação do ambiente em diferentes instâncias.

4.2.2 Bibliotecas

Durante o desenvolvimento do *backend*, a escolha cuidadosa de bibliotecas desempenhou um papel fundamental na eficiência, funcionalidade e integração de serviços essenciais. As bibliotecas selecionadas foram estrategicamente incorporadas para garantir a construção robusta e eficaz do sistema de *e-commerce*.

- ***Express*:** o *framework Express* foi utilizado para a construção da *API*. Sua simplicidade e flexibilidade oferecem uma base sólida para o desenvolvimento de rotas, *middleware* e gestão de requisições, contribuindo para a criação de uma *API* escalável e eficiente. "O *Express* por si só provavelmente não faz tudo o que você precisa, e você provavelmente se encontrará com um grande número de outras bibliotecas que integrará em seus aplicativos". (HAHN, 2016)

- ***Firebase Admin*:** essencial para a conexão entre o *backend* e os serviços do *Firebase*, o *Firebase Admin* foi empregado para habilitar a comunicação fluida com o *Firestore*, o banco de dados principal. Isso permitiu operações eficientes de leitura e escrita, garantindo uma interação segura e consistente com os dados armazenados.

- ***Firebase Functions*:** a integração do *Firebase Functions* foi crucial para a implementação de funções *serverless* acionadas por eventos específicos. Isso proporcionou uma abordagem escalável para tarefas automatizadas e processamento assíncrono, aprimorando a eficiência operacional.

- ***Stripe*:** para facilitar a integração do sistema com o processador de pagamento *Stripe*, foi escolhido a biblioteca oficial para *Node.js*. Configurada com o *token* de

acesso gerado pela plataforma. Essa biblioteca permitiu a criação de uma experiência de pagamento segura e eficaz durante o *checkout*, enriquecendo a funcionalidade do *e-commerce* e diminuindo as barreiras de implementação.

Essas escolhas estratégicas de bibliotecas refletem o compromisso em utilizar ferramentas confiáveis e eficientes, contribuindo para o desenvolvimento de um *backend* sólido e altamente funcional.

4.2.3 Banco de dados

Quando se discute sobre modelagem de dados para um sistema, mesmo quando se opta por um banco não relacional, é crucial compreender como certas informações são armazenadas. Embora os bancos de dados não relacionais ofereçam flexibilidade e escalabilidade para lidar com diferentes tipos de dados, entender a estrutura de armazenamento ajuda a garantir uma organização eficiente dos dados e a otimização do desempenho do sistema. Além disso, compreender como os dados são armazenados possibilita uma melhor gestão da integridade e da consistência dos dados, garantindo que as informações sejam recuperadas de maneira precisa e oportuna quando necessário. Portanto, é possível verificar na Figura 11 o resultado da modelagem dos dados, que demonstram como as informações serão salvas no *Firestore*.

A escolha pelo *Firestore* como banco de dados no projeto foi motivada por diversos fatores técnicos e estratégicos. Inicialmente, o autor já possuía familiaridade com o ecossistema *Firebase*, incluindo o *Firestore*, o que proporcionou um ponto de partida sólido e acelerou o processo de desenvolvimento. Essa experiência prévia permitiu aproveitar ao máximo as funcionalidades nativas da plataforma, garantindo uma integração eficiente com outros serviços do *Firebase*, como o de autenticação e de armazenamento de arquivos.

4.2.4 Estruturação de pastas

Após os passos citados nas seções acima, o projeto se encaminha para a estruturação das pastas (Figura 12). Quando os arquivos de um projeto são bem organizados, isso reflete diretamente na qualidade do código e do desempenho do sistema, tornando mais fácil a localização das funções, classes, tipagens do sistema, entre outros.

- **Pasta principal:** no diretório principal dos projetos são armazenados arquivos básicos, sendo o *README.md* e *.gitignore* arquivos referentes ao versionamento de código com o auxílio do *GitHub*. Além disso, existe a pasta *'functions'* onde fica armazenado todo código que será executado nas *functions* do *Firebase*. Os arquivos referentes ao *Firebase* serão explicados na seção de *deploy* e hospedagem.

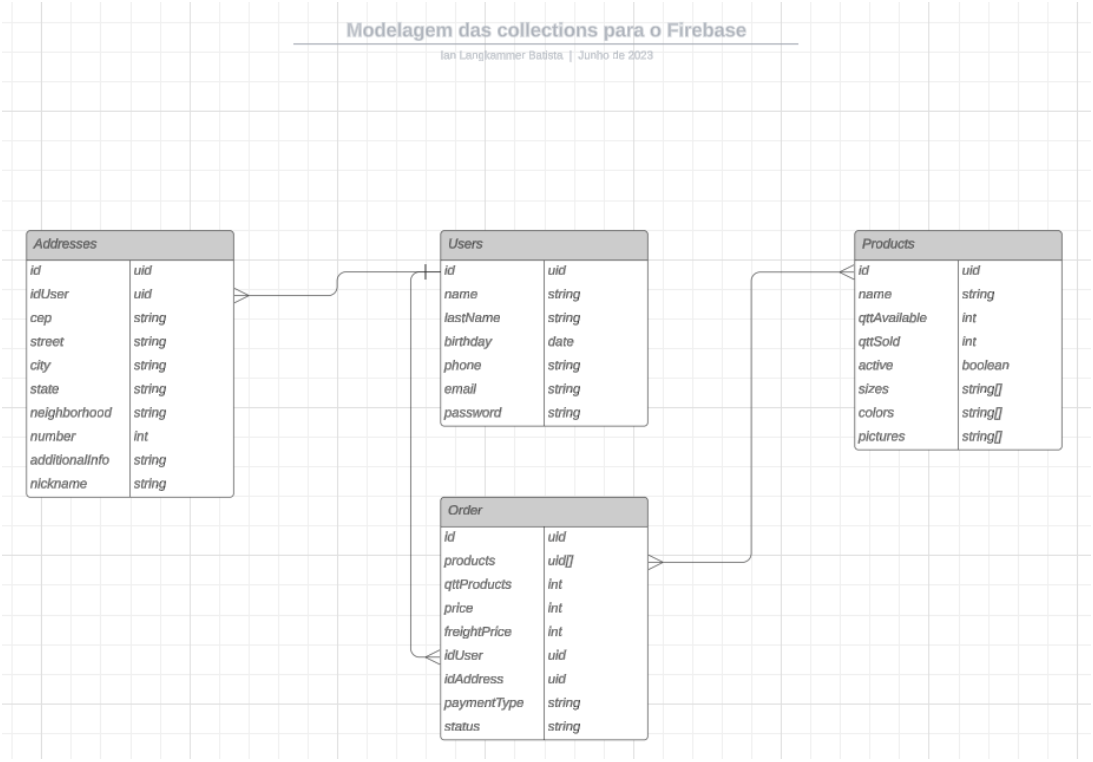

Figura 11 – *Collections* do *Firebase*

Fonte: Elaborado pelo autor

- ***functions*:** dentro dessa pasta, estão amarzenados os arquivos supracitados na seção de ambiente de desenvolvimento, além disso é possível visualizar o diretório *'src'* que contém todo o código, enquanto isso o *package.json* e *yarn.lock* garantem o versionamento das bibliotecas e versões das tecnologias utilizadas no projeto.

- ***api*:** se encontra os arquivos referentes ao instanciamento e configurações do *Express*, além da adição das rotas e controladores responsáveis por processar e responder os *requests* feitos para a *API*.

- ***services*:** nesse diretório existem outras duas pastas, sendo a *'firebase'* e a *'stripe'*, no qual se utiliza as bibliotecas *Firebase Admin* e *Stripe* para inicializar a conexão com os serviços do *Firebase* e da *Stripe* e consumir suas funcionalidades, que serão utilizados posteriormente pela *API* e as outras futuras implementações do projeto.

- ***utils*:** uma pasta comum de se encontrar em todo tipo de projeto, basicamente ela armazena métodos, tipos e classes que irão auxiliar processos de conversão de *strings*, validação de parâmetros, mapeamento de entidades, entre outros.

- ***functions*:** esse arquivo fica responsável exclusivamente por consumir a biblioteca *Firebase Functions* para instanciar os gatilhos disparados pelo *Firestore* ao criar, excluir, editar ou acessar os documentos armazenados.

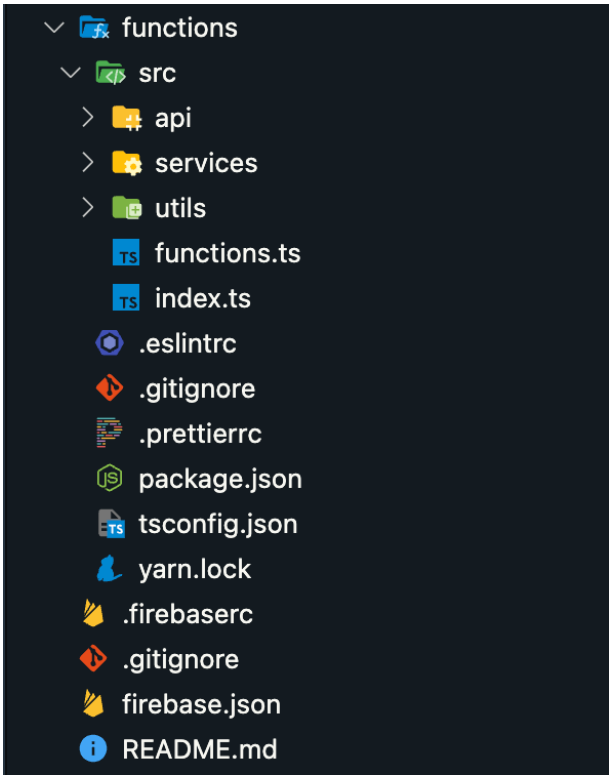

Figura 12 – Estrutura de pastas do *backend*

Fonte: Elaborado pelo autor

- ***index*:** por fim, esse é o arquivo onde todo o projeto é iniciado, dentro dele são exportados os gatilhos instanciados no arquivo *'functions'* e também configurado a hospedagem da *API* em *Express* com o *Firebase*.

4.2.5 *API*

A seção da *API* representa a camada que o usuário não enxerga do sistema de *e-commerce*, onde a comunicação entre o *frontend* e o *backend* se torna tangível. Para proporcionar uma visão clara da estrutura e funcionalidades da *API*, é essencial compreender a organização dos arquivos que a compõem.

O arquivo *index.ts* atua como a instância central da *API*, configurando o *Express* e definindo sua única rota, a rota em questão é a *'/stripe'*, que é mapeada para o arquivo *stripe.ts*. Este arquivo contém as definições de rotas específicas e seus respectivos *handlers*, permitindo a comunicação eficaz entre o *frontend* e o *backend*.

O arquivo *stripe.ts* desempenha um papel crucial na facilitação dos pagamentos e na criação de pedidos dentro do sistema. No centro desse arquivo, encontra-se um método dedicado a lidar com solicitações de pagamento. Este método realiza o mapeamento detalhado de produtos, custos de frete e cupons de desconto, comunicando-se de maneira eficiente com a plataforma de pagamento *Stripe*.

Além de integrar-se à *Stripe* para a geração de uma *URL* segura para o cliente finalizar o pagamento, este método também assume a responsabilidade de criar registros de pedidos no *Firestore*, garantindo uma trilha precisa e organizada de todas as transações. A interação harmoniosa entre esse método e a *Stripe* reforça a segurança e eficiência durante o processo de *checkout*, contribuindo para uma experiência de compra fluida e confiável.

A organização e funcionalidades desses arquivos na *API* ilustram a coesão entre as diversas partes do sistema, oferecendo um ambiente robusto, eficaz e escalável para o funcionamento do *e-commerce*.

4.2.6 Gatilhos do *Firestore* (*Triggers*)

A implementação de gatilhos no *Firestore* desempenha um papel crucial no gerenciamento dinâmico de cupons, estabelecendo uma comunicação eficiente entre o sistema e a plataforma de pagamento *Stripe*. Através do arquivo *functions.ts* do *Firebase*, dois métodos foram desenvolvidos para monitorar eventos específicos no *Firestore*, resultando em interações automáticas e em tempo real com a *Stripe*.

O método *onCreateCoupon* é acionado sempre que um novo documento é criado na coleção *Coupons* do *Firestore*. A função correspondente estabelece uma comunicação imediata com a *Stripe*, utilizando os dados do cupom recém-criado para gerar um cupom correspondente na plataforma de pagamento. O percentual de desconto, duração e nome do cupom são cuidadosamente mapeados para criar uma representação equivalente na *Stripe*. Após a criação bem-sucedida, o *ID* do cupom gerado é registrado no documento correspondente no *Firestore*, garantindo uma correlação direta entre os sistemas.

Quando um cupom é removido no *Firestore*, o método *onDeleteCoupon* é acionado. Este gatilho desencadeia a remoção imediata do cupom equivalente na plataforma *Stripe*, assegurando que não haja cupons desativados ou não utilizáveis. A comunicação direta com a *Stripe* para a exclusão do cupom baseia-se no *ID* associado a esse cupom no momento da criação. Essa abordagem garante a sincronização consistente entre os dois sistemas, mantendo a integridade dos cupons e a precisão nas operações de exclusão.

Esses gatilhos, operando de maneira síncrona e automatizada, contribuem significativamente para a eficiência operacional do sistema de *e-commerce*, mantendo a consistência entre o *Firestore* e a plataforma de pagamento *Stripe*.

4.3 *Frontend*

Durante essa seção, será discutido sobre o desenvolvimento do *frontend*.

4.3.1 Ambiente de desenvolvimento

Para iniciar o desenvolvimento do sistema, era necessário dar início em toda estrutura do projeto, inclusive o ambiente de desenvolvimento. Pensando nisso, foi estabelecido um ambiente de desenvolvimento eficiente e padronizado. O *Vite* é utilizado como um *bundler* extremamente rápido que simplifica o processo de desenvolvimento, enquanto o *ESLint* é configurado para garantir a consistência do código e a detecção de possíveis erros ou más práticas durante o desenvolvimento. O *TypeScript* é integrado para adicionar tipagem estática ao *JavaScript*, aumentando a robustez e a manutenibilidade do código. Além disso, o *Prettier* é configurado para garantir a formatação consistente do código, mantendo-o limpo e legível. Essas ferramentas combinadas proporcionam uma base sólida para o desenvolvimento de aplicações *React*, promovendo boas práticas de codificação e prevenindo erros comuns.

4.3.2 Bibliotecas

- ***React*:** biblioteca *JavaScript* de código aberto mantida pelo *Facebook* para a criação de interfaces de usuário, especialmente para aplicações de página única. *React* utiliza uma abordagem baseada em componentes para construir *UIs* declarativas e reativas.
- ***Vite*:** *bundler* de desenvolvimento extremamente rápido que oferece uma experiência de desenvolvimento moderna e eficiente para projetos *frontend*, facilitando a configuração de um ambiente de desenvolvimento de alto desempenho.
- ***Firebase*:** plataforma de desenvolvimento de aplicativos móveis e da *web* fornecida pelo *Google*, oferecendo uma variedade de serviços para autenticação de usuários, armazenamento de dados em tempo real, análises e muito mais.
- ***Axios*:** biblioteca *JavaScript* para fazer requisições *HTTP* a partir do navegador ou de um servidor *Node.js*. É amplamente utilizada devido à sua simplicidade e suporte a *promises*.
- ***Styled Components*:** biblioteca para *React* e *React Native* que permite estilizar componentes utilizando *JavaScript* e *CSS-in-JS*. Facilita a criação de interfaces de usuário consistentes e escaláveis.
- ***React Hook Form*:** biblioteca para gerenciamento de formulários em *React*, utilizando hooks. Oferece uma abordagem simples e intuitiva para lidar com validação, envio e manipulação de dados de formulários.
- ***Yup*:** biblioteca de validação de esquemas para *JavaScript*. É amplamente utilizada para validar dados de formulários no lado do cliente, oferecendo uma sintaxe expressiva e flexível para definir esquemas de validação.

4.3.3 Estruturação de pastas

Nessa subseção será apresentado a estruturação de pastas (Figura 13), no entanto, agora para o *frontend*.

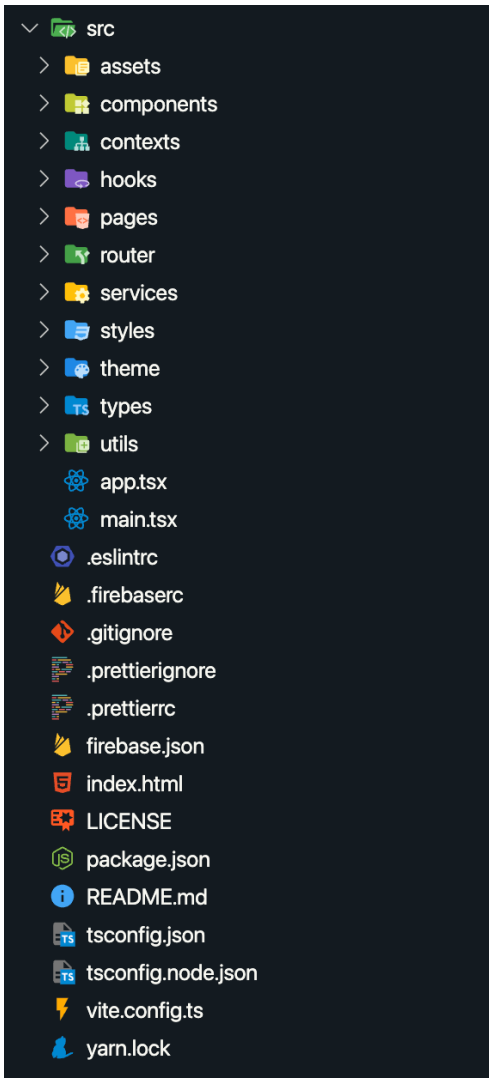

Figura 13 – Estruturação de pastas do *frontend*

Fonte: Elaborado pelo autor

- ***assets*:** armazena arquivos estáticos, como imagens, fontes e outros recursos visuais utilizados na interface do aplicativo.
- ***components*:** contém os componentes reutilizáveis do aplicativo, como botões, *cards* e formulários, facilitando a organização e manutenção do código.
- ***contexts*:** responsável por armazenar os contextos utilizados na aplicação, permitindo o compartilhamento de estados entre componentes, como exemplo informações do usuário logado, carrinho, entre outros.

- ***hooks*:** neste diretório ficam os *hooks* dos contextos criados, para que seja possível a utilização de todas as informações globais em qualquer componente do sistema.
- ***pages*:** aqui são armazenadas as páginas da aplicação, cada uma representando uma rota específica do aplicativo.
- ***router*:** contém as configurações e definições do roteador utilizado na aplicação, com o auxílio do *React Router*, nessa pasta existe toda configuração responsável pelo controle de navegação entre as páginas.
- ***services*:** armazena os serviços de integração com o *Firebase*, *APIs* externas, como requisições *HTTP/HTTPS* para obter ou enviar dados para o *backend*.
- ***styles*:** contém os estilos globais da aplicação, como cores, tipografia e outros estilos compartilhados entre os componentes.
- ***theme*:** neste diretório são definidos os temas da aplicação dentro da *Styled Components*, permitindo a customização dos estilos de acordo com as preferências ou necessidades do usuário.
- ***utils*:** armazena funções utilitárias e *helpers* que podem ser reutilizadas em diferentes partes do projeto, contribuindo para a organização e manutenção do código.
- ***types*:** esse diretório contém somente um arquivo com todas as tipagens globais da aplicação para o *TypeScript*.

4.3.4 Principais funcionalidades

Ao longo dessa etapa do projeto, o texto explora detalhadamente as principais funcionalidades desenvolvidas no *frontend* desse *e-commerce*, um ambiente virtual onde os usuários realizam suas compras. Esta análise permitirá uma compreensão aprofundada das características e da usabilidade da interface, bem como das tecnologias empregadas para sua implementação. Ao destacar as funcionalidades essenciais do *frontend*, será possível evidenciar não apenas a qualidade do projeto mas também o cumprimento dos objetivos propostos para desenvolvimento. Vale ressaltar que nessa seção serão destacadas somente as principais funcionalidades que não foram abordadas na seção seguinte de resultados.

4.3.4.1 Carrinho

Para aperfeiçoar a experiência de compra dos clientes, foi implementado um sistema robusto de gerenciamento de carrinho. Todas as informações pertinentes ao carrinho são armazenadas em um contexto global do *React*, garantindo acessibilidade de acesso em vários dispositivos e praticidade em todos os componentes da aplicação. Além disso, para assegurar a continuidade das informações mesmo após o fechamento do navegador, foi utilizado o

Local Storage nativo do *JavaScript*. Dessa forma, quando o cliente acessa novamente o *site* após sair ou navegar entre páginas, as últimas informações do carrinho são automaticamente resgatadas do *cache* do navegador. Essa abordagem simplifica significativamente o processo de compra, proporcionando uma experiência fluida e intuitiva aos usuários.

4.3.4.2 *Checkout*

Ao prosseguir para o processo de *checkout*, o cliente é conduzido por um processo simples, projetado para simplificar ao máximo a conclusão da compra. Ao clicar no botão "Comprar" no carrinho, o cliente é redirecionado para a primeira etapa do processo. Se o cliente não possuir uma conta, ele será encaminhado para uma tela de pré-cadastro (Figura 14), onde poderá inserir somente as informações essenciais, sendo elas o nome, endereço de e-mail e senha, para criar uma conta. Essa etapa garante que o cliente tenha uma conta para gerenciar seus pedidos e informações de forma eficiente no futuro.

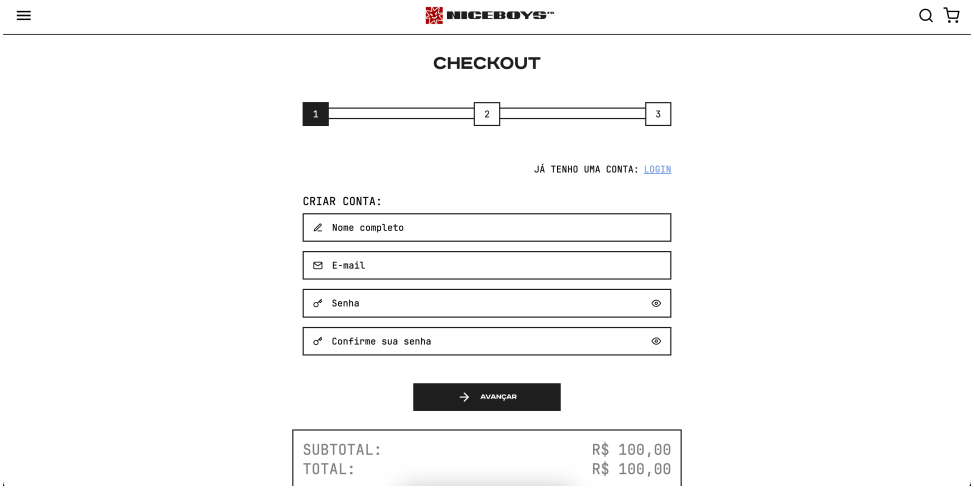

Figura 14 – Primeira etapa do fluxo de *checkout*

Fonte: Elaborado pelo autor

Caso o cliente já possua uma conta, ele será direcionado diretamente para a segunda etapa do *checkout* (Figura 15). Nesta etapa, o cliente pode cadastrar ou selecionar um endereço de entrega já registrado em sua conta. Isso proporciona conveniência e agilidade, permitindo que o cliente selecione rapidamente o endereço para entrega de seus produtos. Vale ressaltar que durante essa tela, existe uma integração com uma *API* chamada 'ViaCEP', onde foi utilizado um *endpoint* para enviar o CEP digitado pelo usuário e recuperar automaticamente todas as informações como o nome da rua, bairro, etc...

Na última etapa do processo de *checkout* (Figura 16), o cliente é apresentado com opções para selecionar a modalidade de entrega desejada, como PAC ou SEDEX. Além disso, o cliente tem a oportunidade de inserir um cupom de desconto, caso possua algum, para aplicar descontos especiais à sua compra. Após a finalização de todas as etapas, o cliente é encaminhado para uma página gerada pela *Stripe*, onde pode finalizar o pagamento

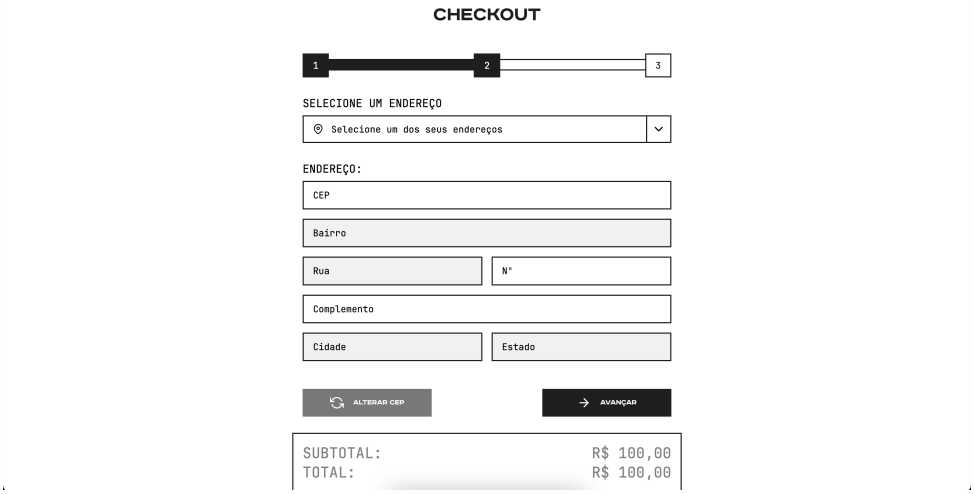

Figura 15 – Segunda etapa do fluxo de *checkout*

Fonte: Elaborado pelo autor

de forma segura e criptografada. Nessa página, o cliente tem acesso a todas as informações relevantes do pedido, incluindo itens selecionados, modalidade de entrega escolhida e total a ser pago. Este processo garante uma experiência de compra segura, transparente e conveniente para os clientes, promovendo a confiança e satisfação do consumidor.

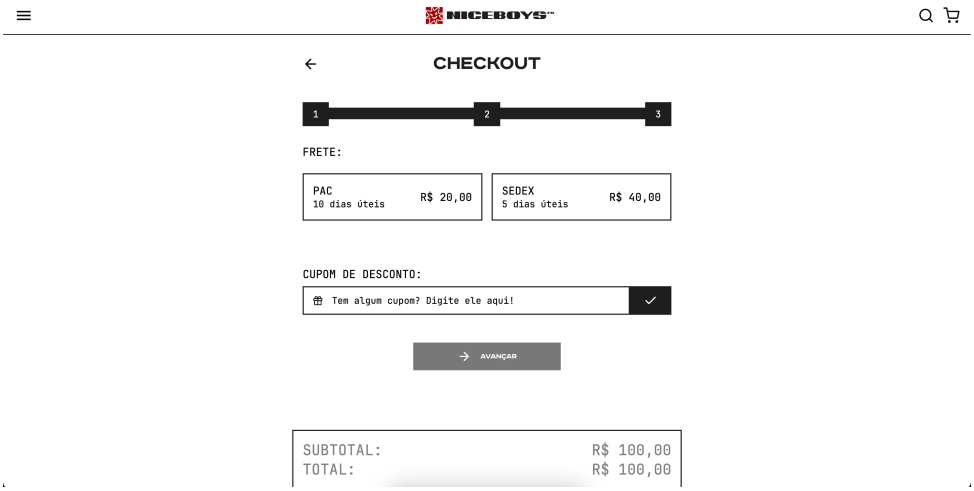

Figura 16 – Terceira etapa do fluxo de *checkout*

Fonte: Elaborado pelo autor

4.3.4.3 Painel de administração

Por fim, no painel de administração, acessível apenas para usuários designados como administradores do sistema, são apresentadas quatro opções principais: Pedidos, Estoque, Cupons e Usuários. Ao selecionar uma dessas opções, o administrador é redirecionado para uma tabela que lista todos os itens relacionados à categoria escolhida.

Por exemplo, ao acessar a opção "Estoque", o administrador visualiza uma lista de todos os produtos cadastrados (Figura 17), juntamente com seu status (ativo ou desativado) e a possibilidade de buscar por um produto específico. Além disso, o administrador pode adicionar um novo item ao estoque, sendo então direcionado para um formulário específico para preenchimento das informações relevantes (Figura 18). Nesse formulário, o administrador tem a capacidade de manipular todos os detalhes do produto, como nome, quantidade disponível em estoque, além de poder desativar, ativar ou excluir produtos conforme necessário.

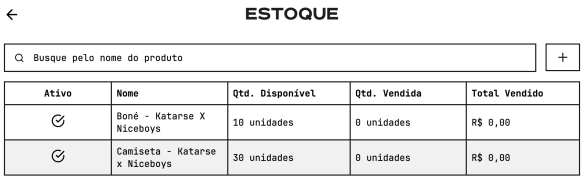

Figura 17 – Painel de manutenção dos produtos

Fonte: Elaborado pelo autor

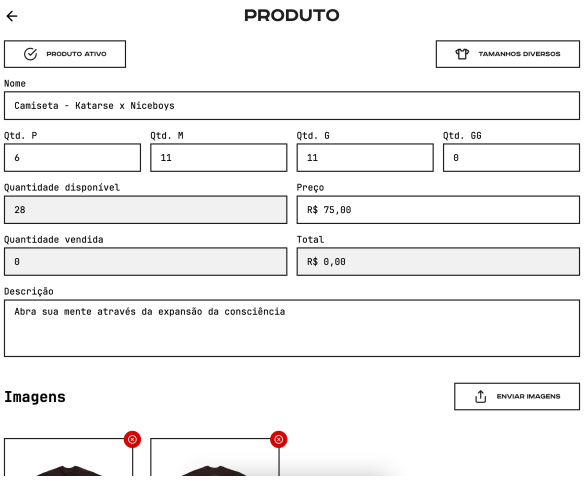

Figura 18 – Formulário de um produto

Fonte: Elaborado pelo autor

Essa mesma lógica se aplica às outras opções do painel de administração. Por exemplo, na seção "Pedidos", o administrador pode visualizar todos os pedidos registrados, bem como gerenciar informações relacionadas a eles, como status de pagamento e entrega. Da mesma forma, nas seções "Cupons"e "Usuários", o administrador tem acesso a uma lista

dos cupons disponíveis e dos usuários cadastrados, respectivamente, com a capacidade de adicionar, editar ou excluir itens conforme necessário, com ressalva dos usuários, que só podem ser visualizados.

Em resumo, o painel de administração oferece uma interface intuitiva e poderosa para que os administradores possam gerenciar eficientemente todas as áreas essenciais do sistema, garantindo assim o bom funcionamento e a qualidade dos serviços prestados.

4.4 *Deploy* e Hospedagem

Ao longo dessa seção será descrito o processo de implantação *(deploy)* das *cloud functions* utilizando a ferramenta *firebase-tools*, bem como a hospedagem da *build* de um projeto *react* utilizando o *Firebase Hosting* e também o *firebase-tools*.

4.4.1 *Deploy* das *Cloud Functions*

O deploy das *cloud functions* é essencial para a implementação de lógicas de servidor e serviços do lado do servidor em um ambiente hospedado na nuvem, garantindo escalabilidade e confiabilidade. Para tal, é necessário instalar o *firebase-tools* com o *yarn* e fazer login na *CLI* do *Firebase*. Uma vez autenticado, é possível associar o projeto de *backend* com o *Firebase*, permitindo hospedar a *API* que foi desenvolvida com *Express* e também os gatilhos do *Firestore*.

No entanto, antes disso é necessário configurar as informações do *Firebase* no projeto. Então, foi executado o comando *'firebase init'* com o *firebase-tools*, e realizado todo o passo a passo para configurar as *Functions* do *Firebase* nesse *backend* desenvolvido. Como consequência dessa configuração, são gerados os arquivos *.firebaserc* e *firebase.json* que aparecem na Figura 12.

Figura 19 – Executando o comando *'firebase init'*

Fonte: Elaborado pelo autor

O arquivo *.firebaserc* armazena as informações dos projetos do *Firebase* que foram associados com a aplicação. Enquanto isso, o *firebase.json* armazena configurações em formato *JSON* dos serviços que você vai usar, como o *Hosting*, *Functions* ou *Emulators*.

Finalmente, basta executar no terminal o comando *'firebase deploy'* e a *CLI* irá cuidar de todo o resto do processo, basta aguardar alguns instantes e se você verificar no seu projeto do *Firebase* na aba *Functions*, irá encontrar suas *functions* hospedadas e disponíveis na internet.

Figura 20 – *Functions* hospedadas no *Firebase*

Fonte: Elaborado pelo autor

4.4.2 *Deploy* e hospedagem da *SPA*

Para realizar a implantação do *frontend* foi feito os mesmos passos descritos para as *Functions*, a principal diferença é que o projeto do *frontend* utiliza somente o serviço de *Hosting*, gerando também os mesmos arquivos já explicados acima.

Todavia, é importante ressaltar que as configurações no arquivo *firebase.json* para o serviço de *Hosting* se difere em algumas desse mesmo arquivo do *backend*, pois se tratam de serviços diferentes do *Firebase*.

Além disso, no caso do serviço de *Hosting* é obrigatório compilar manualmente o projeto antes de fazer o *deploy* para o *Firebase*. Então, é necessário executar o comando *'yarn build'*, que utilizou o *Vite* como aliado para realizar essa *build*. Na Figura 21 é possível verificar o resultado do comando de *build* e visualizar o diretório da *build* gerado em *'dist'*.

Por fim, bastou executar o mesmo comando utilizado para as *Functions*, sendo ele o *'firebase deploy'*, assim, em alguns instantes a *SPA* também já estava disponível na internet.

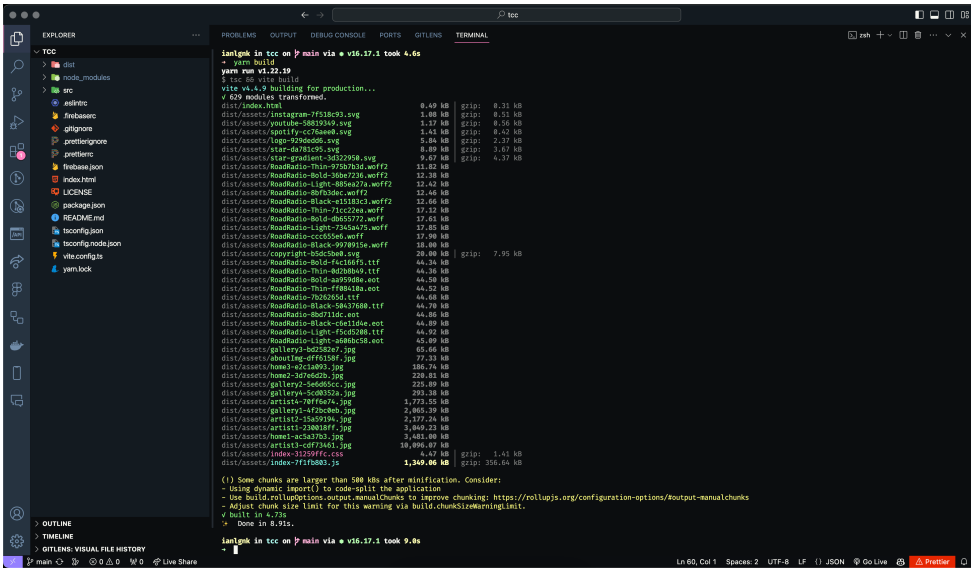

Figura 21 – Execução do comando *'yarn build'*

Fonte: Elaborado pelo autor

5 Resultados

A fase de implementação desse projeto de *e-commerce* culminou em resultados palpáveis e uma plataforma de vendas, que serão apresentados abaixo.

- Página Principal (Figura 22): a página inicial é a vitrine virtual da plataforma, projetada para cativar os usuários desde o primeiro acesso. Os elementos visuais e a disposição estratégica dos produtos de destaque visam proporcionar uma experiência visualmente atrativa e intuitiva. Além disso, essa página oferece um *insight* inicial sobre a imagem que a marca quer passar, contando com galeria e os artistas envolvidos no projeto, fortalecendo o laço de conexão com o usuário.

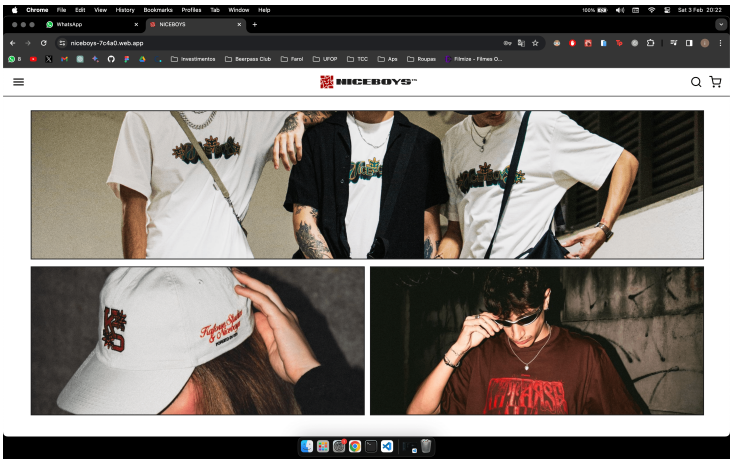

Figura 22 – Tela *'Home'*

Fonte: Elaborado pelo autor

- Página Sobre Nós (Figuras 23): a página 'Sobre Nós' foi projetada para estabelecer uma conexão emocional com os usuários, apresentando a história e valores da marca de maneira envolvente e informativa.

- Página de Suporte (Figura 24): a página de suporte foi concebida para facilitar a comunicação entre os usuários e a equipe de suporte. A inclusão de formulários e informações relevantes permite que os usuários relatem problemas, solicitem assistência ou forneçam *feedback* de maneira eficiente.

- Página de Produtos (Figuras 25, 26, 32): a página de produtos exibe a diversidade do catálogo, proporcionando aos usuários uma navegação fácil e atraente. Recursos como filtros e busca por produtos foram implementados para facilitar a descoberta e a seleção de itens desejados.

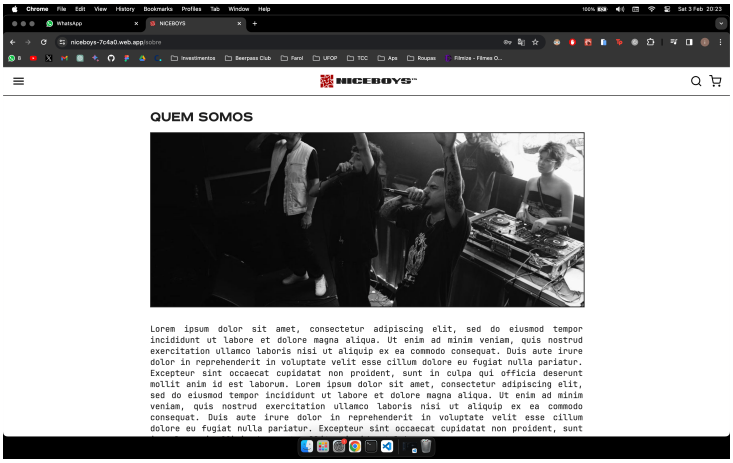

Figura 23 – Tela Sobre Nós

Fonte: Elaborado pelo autor

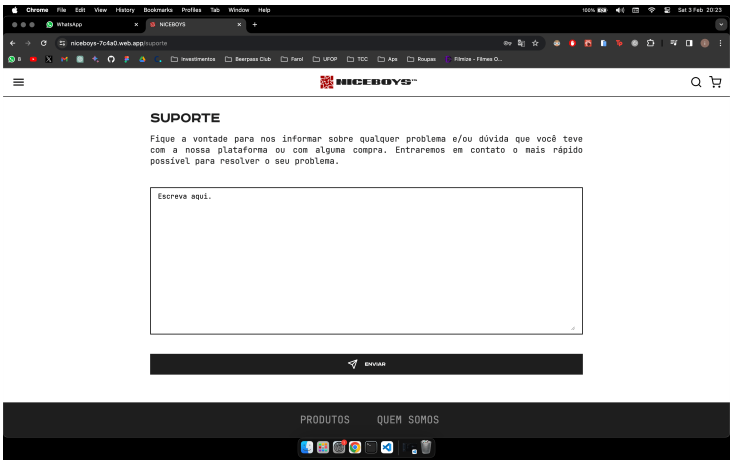

Figura 24 – Tela de Suporte

Fonte: Elaborado pelo autor

- Páginas de *Login* e Cadastro (Figuras 27, 28): as páginas referentes a autenticação e criação de acesso são simples e com poucos elementos, visando praticidade para o usuário. O cadastro exige somente os dados essenciais para que o usuário não fique confuso ou tenha aversão a preencher o formulário e criar a sua conta.

- Páginas de Esqueci Minha Senha (Figuras 29): a página para resgate de senha é a mais simples, visto que ela tem somente uma função básica, portanto conta com poucos elementos para o usuário não se perder em como fazer a operação de recuperar senha.

- Página de *Checkout* (Figuras 30, 31, 32, 33): a página de *checkout* é o último estágio crucial da jornada do usuário. Focada na simplicidade e eficiência, o design responsivo garante uma experiência consistente em diferentes dispositivos, contribuindo para a facilidade de uso durante todo o processo de *checkout*. A implementação do *gateway*

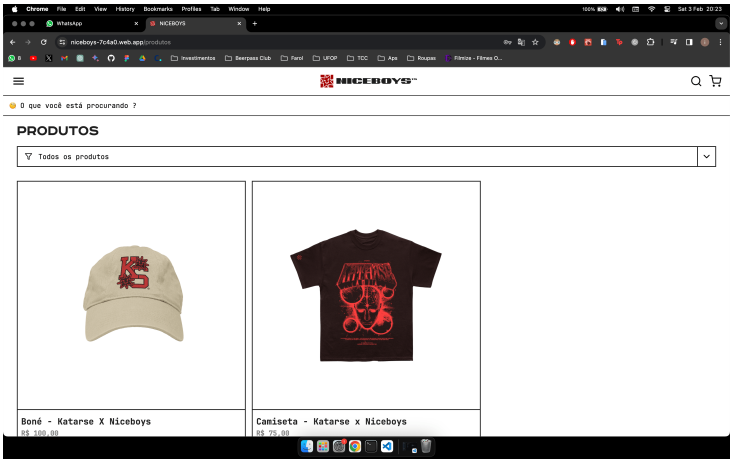

Figura 25 – Tela de Produtos

Fonte: Elaborado pelo autor

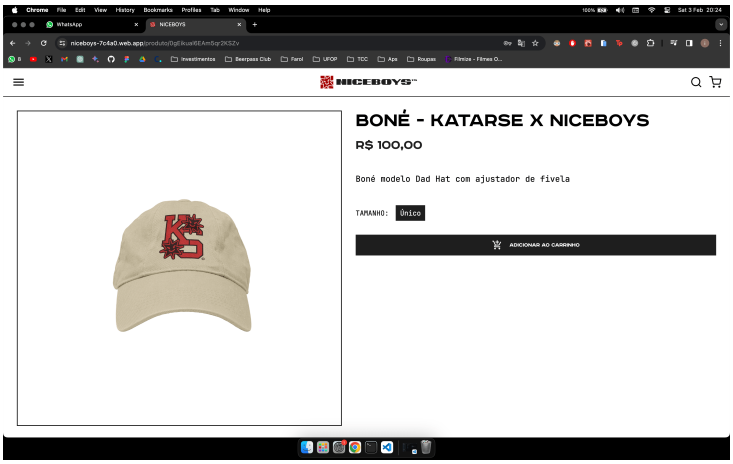

Figura 26 – Tela de um Produto

Fonte: Elaborado pelo autor

de pagamento *Stripe* reforça a segurança e confiabilidade nas transações, promovendo a conclusão bem-sucedida do processo de compra.

- Páginas de Usuário e Administrador (Figuras 34, 35): as páginas destinadas a usuários e administradores foram cuidadosamente projetadas para oferecer funcionalidades específicas, simples e diretas. Os usuários podem visualizar os seus pedidos, endereços, acessar e alterar informações da conta, enquanto os administradores tem todos os acessos supracitados, além de acesso as ferramentas de gerenciamento de produtos, cupons, pedidos e usuários.

- Meus Pedidos, Meus Endereços e Meus Dados: as interfaces dentro do painel do usuário conta com 'Meus Pedidos', 'Meus Endereços' e 'Meus Dados' permitem que os usuários revisem suas compras passadas, gerenciem informações de entrega e alterem informações da sua própria conta, proporcionando uma experiência personalizada e conveniente.

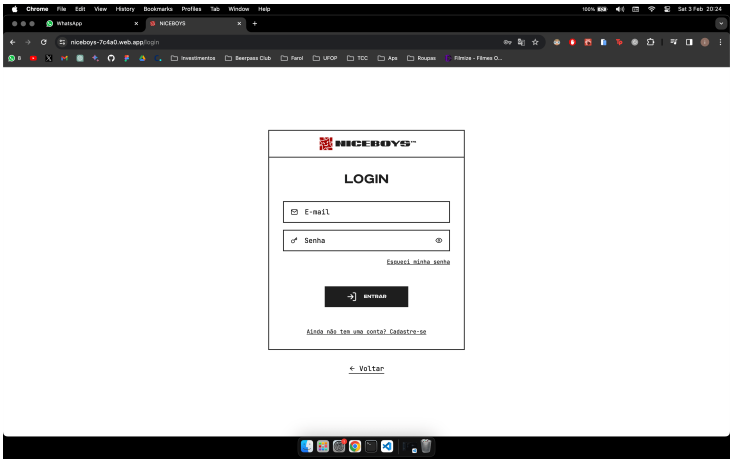

Figura 27 – Tela de *Login*

Fonte: Elaborado pelo autor

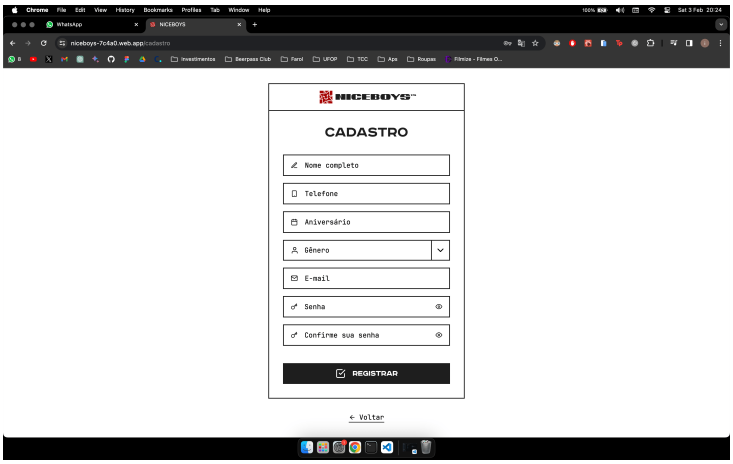

Figura 28 – Tela de Cadastro

Fonte: Elaborado pelo autor

- Manipulação de Pedidos, Produtos, Cupons e Usuários: a inclusão de interfaces dentro do painel de administrador dedicadas a gerência de cupons, pedidos, usuários e produtos oferece aos administradores controle total sobre o catálogo e promoções, além de visualização de detalhes de pedidos, alteração de status e também visualização dos clientes, permitindo assim uma gestão eficiente e ágil desses aspectos fundamentais.

- Carrinho (Figuras 36): o carrinho foi desenvolvido com foco na simplicidade e usabilidade. O usuário pode visualizar e editar facilmente os itens selecionados, além de receber informações claras sobre preços e opções de *checkout*.

- Pesquisa e Menu: os recursos de pesquisa e menu foram implementados para aprimorar a navegação do usuário. A barra de pesquisa permite busca rápida e automatizada por produtos, enquanto o menu proporciona acesso fácil às diversas funcionalidades da plataforma.

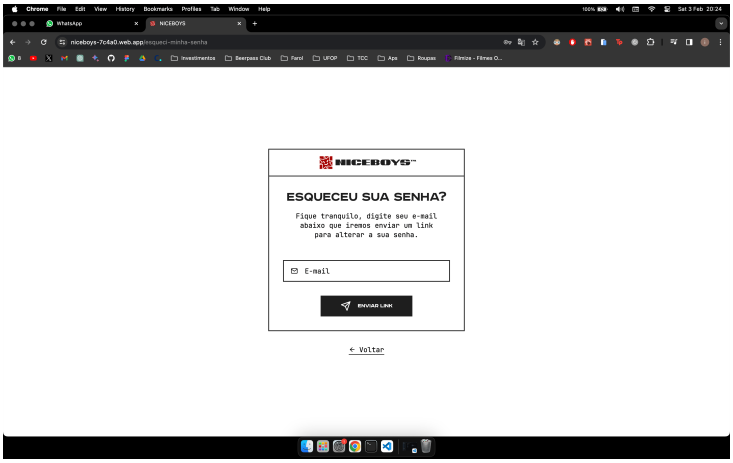

Figura 29 – Tela de Esqueci Minha Senha

Fonte: Elaborado pelo autor

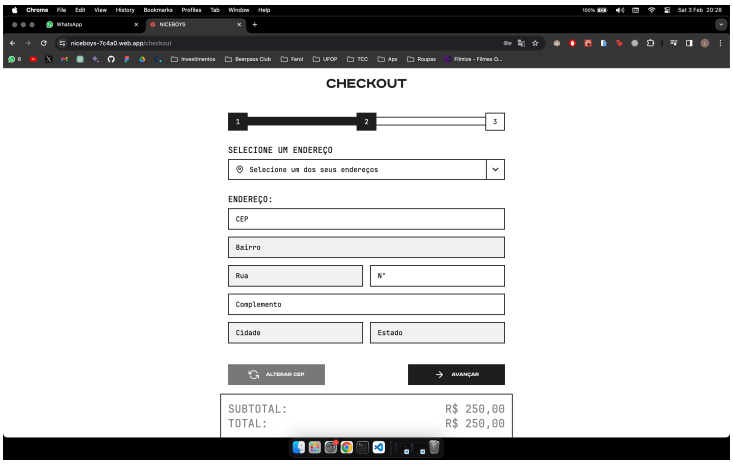

Figura 30 – Tela de *Checkout* (Endereço)

Fonte: Elaborado pelo autor

- Alerta de sistema (Figura 33): esse componente foi simples e bem elaborado, utilizado para oferecer feedbacks para as operações de usuário.

A materialização dessas páginas ilustra não apenas a implementação bem-sucedida dos requisitos propostos, mas também a capacidade de criar uma plataforma de *e-commerce* coesa, funcional e orientada ao usuário. Esses resultados evidenciam o comprometimento em transformar conceitos teóricos em soluções práticas e tangíveis.

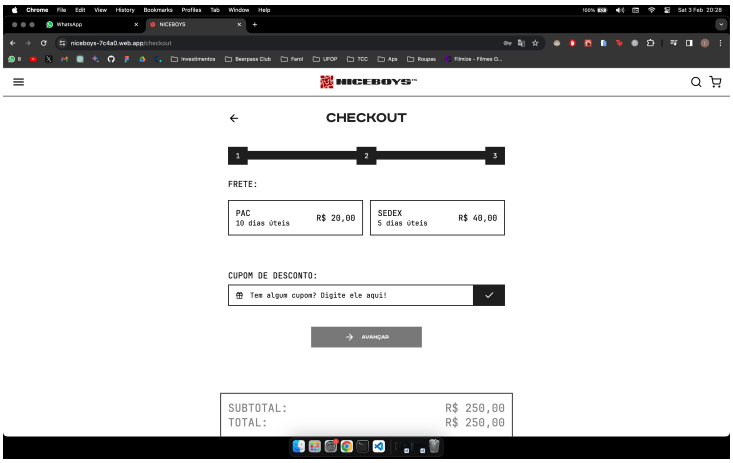

Figura 31 – Tela de *Checkout* (Frete e Cupom)

Fonte: Elaborado pelo autor

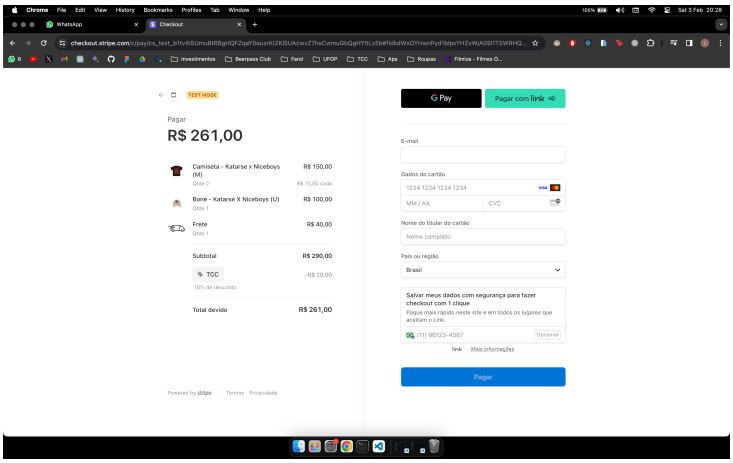

Figura 32 – Pagamento via *Stripe*

Fonte: Elaborado pelo autor

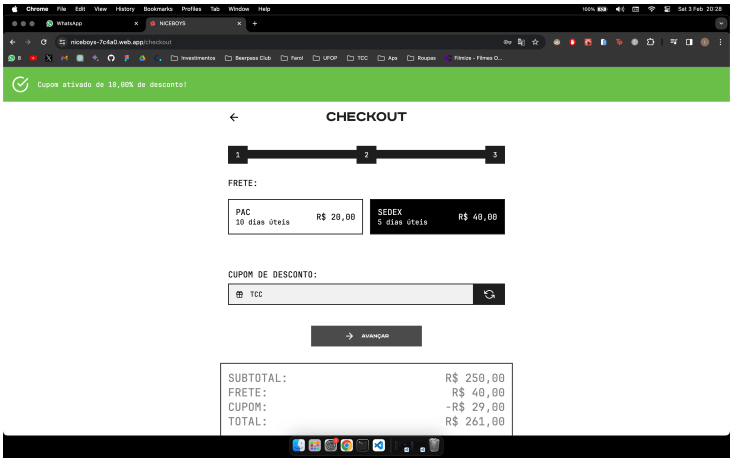

Figura 33 – Alerta de Sistema na tela de *Checkout*

Fonte: Elaborado pelo autor

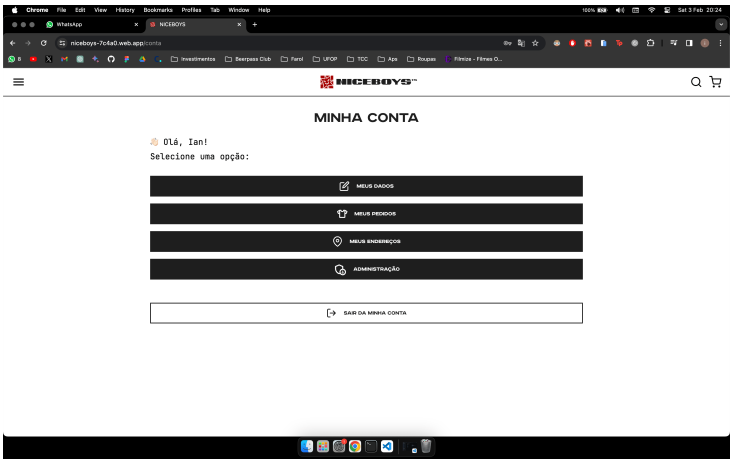

Figura 34 – Painel do usuário

Fonte: Elaborado pelo autor

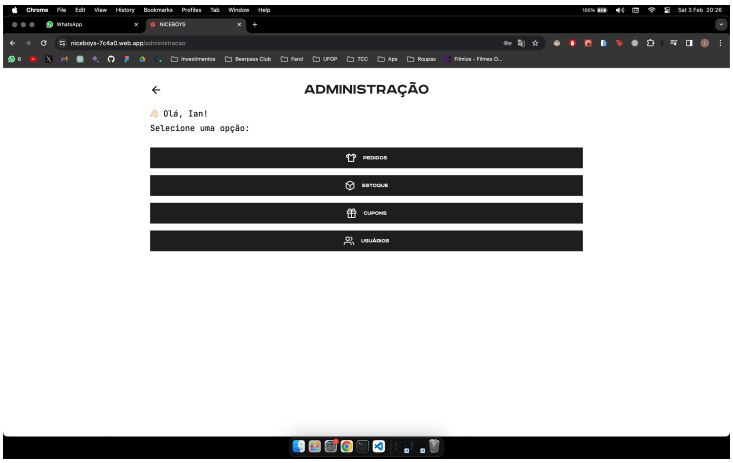

Figura 35 – Painel do administrador

Fonte: Elaborado pelo autor

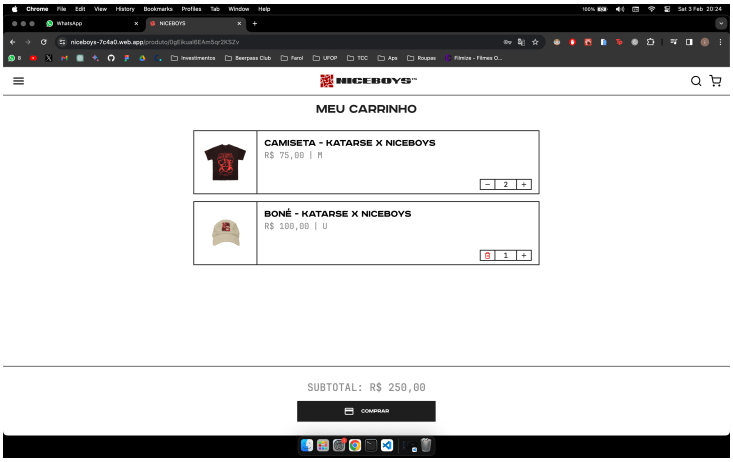

Figura 36 – Componente do Carrinho

Fonte: Elaborado pelo autor

6 Conclusão

Ao finalizar este projeto de conclusão de curso, é possível observar como cada etapa convergiu para o alcance bem-sucedido dos objetivos inicialmente traçados. Em linhas gerais, o foco esteve na concepção, desenvolvimento e implementação de um sistema *web* especializado no comércio de roupas, uma marca que ecoa a visão e autenticidade do próprio autor.

A busca pela maximização dos conhecimentos em tecnologias emergentes, como *TypeScript*, *React*, *Node* e *Firebase*, não apenas enriqueceu a bagagem técnica, mas também impulsionou a eficiência e inovação no desenvolvimento do *e-commerce*. A abordagem centrada no cliente, evidenciada pelo uso do *Business Model Canvas* e pela criação de personas, ampliou a compreensão das necessidades do público-alvo, culminando em uma experiência de usuário mais envolvente e personalizada.

Ao cumprir objetivos específicos, como a implementação de funcionalidades-chave de autenticação e processo de venda automático, não apenas fortalece a robustez do sistema, mas também a eficácia operacional do negócio do autor. O sucesso do *deploy* do sistema não só viabilizou o funcionamento eficiente da plataforma, mas também contribuiu diretamente para aprofundar os conhecimentos com o *Firebase*, uma vez que os *deploys* do *frontend* e *backend* foram feitos utilizando essa ferramenta.

Embora seja possível reconhecer que sempre há espaço para aprimoramentos contínuos, é com satisfação que ao fim da monografia se afirma: todos os objetivos propostos foram alcançados. Este projeto representa não apenas a conclusão de uma etapa acadêmica, mas também o início de uma nova fase no universo da arquitetura e desenvolvimento de sistemas, onde a tecnologia se entrelaça com a visão empreendedora para criar experiências únicas e impactantes.

6.1 Trabalhos futuros

Na próxima seção, será apresentado as ideias que se categorizam como trabalhos futuros para essa monografia.

6.1.1 Tela de cupom

A implementação de uma descrição sobre o que o campo *stripeId* se trata no cadastro de cupons pode fortalecer a integridade do sistema, evitando potenciais erros, inconsistências ou incoerência na administração de descontos. Essa melhoria contribuirá para uma gestão mais precisa e segura das promoções, assegurando que cada cupom seja

devidamente vinculado ao *Stripe* e os administradores consigam entender como encontrá-lo na plataforma da *Stripe*, mantendo a transparência e eficácia nas transações.

6.1.2 Configuração dinâmica de cores

Introduzir uma configuração dinâmica para cores no catálogo de produtos, pois oferecerá uma experiência de compra mais flexível e personalizada. Permitir que os administradores ajustem dinamicamente as opções de cores facilitará a adaptação às demandas do mercado e às preferências dos clientes. Essa personalização não apenas enriquecerá a variedade de produtos, mas também simplificará a gestão do catálogo.

6.1.3 Validação de produtos sem estoque

Da forma como o sistema foi projetado, a verificação se o tamanho selecionado está disponível em estoque acontece no momento em que o botão do tamanho é selecionado, todavia a desativação dinâmica do botão de compra quando determinado tamanho não estiver em estoque é uma medida preventiva crucial para evitar pedidos inválidos ou frustrações dos clientes. Evoluir essa funcionalidade proporcionará uma experiência de compra mais transparente, evitando potenciais conflitos pós-compra devido à falta de estoque. Essa melhoria garantirá que os clientes possam fazer escolhas informadas e efetuar compras apenas quando os produtos desejados estiverem disponíveis.

6.1.4 Travas de segurança para evitar frustrações

Como já foi exposto acima, hoje a validação de tamanho em estoque só ocorre no momento em que o item é selecionado, porém o cliente pode estar com as informações do carrinho em *cache*, abrir o *site* e seguir direto pro fluxo de *checkout*, ele finalizará a compra no entanto pode não haver aquele tamanho em estoque, resultando em frustração para o cliente. Portanto, incluir validações adicionais para os tamanhos escolhidos durante todo o processo de compra é essencial para evitar problemas futuros. A validação no carrinho e durante o *checkout* garantirá que os clientes não enfrentem contratempos devido a seleções incorretas de tamanhos, contribuindo para uma experiência de compra aprimorada. Essa medida preventiva também reduzirá a incidência de devoluções ou trocas, aperfeiçoando a eficiência operacional do negócio.

6.1.5 *Feedback* com *badge* no carrinho

Por fim, surgiu a ideia de adicionar uma *badge* visível no ícone do carrinho, exibindo o número atual de itens no carrinho, visto que isso é uma melhoria significativa na experiência do usuário. Essa funcionalidade fornece aos clientes uma visão rápida e intuitiva do conteúdo do carrinho enquanto navegam pelo site. A implementação de um contador dinâmico no

ícone do carrinho aumenta a transparência sobre os itens adicionados, incentivando os usuários a explorarem e finalizarem suas compras de maneira mais eficaz. Essa melhoria contribuirá não apenas para a usabilidade do site, mas também para a satisfação do cliente ao proporcionar uma experiência de compra mais informada e visualmente intuitiva.

6.1.6 Testes de estresse para performance e escalabilidade

Como mais uma sugestão para trabalhos futuros, surge a realização de testes de estresse, que é fundamental para garantir que o sistema de *e-commerce* possa lidar adequadamente com picos de acesso e grandes volumes de transações simultâneas. A implementação desses testes permitiria identificar eventuais gargalos e melhorar a escalabilidade do sistema, garantindo que ele se mantenha estável em cenários de alta demanda. Ademais, seria interessante explorar a capacidade de escalabilidade horizontal do *Firestore*, aproveitando sua infraestrutura em nuvem para suportar um número crescente de usuários e transações. Esses testes forneceriam informações valiosas para melhorar o desempenho, minimizando a latência e mantendo a integridade das operações sob pressão.

6.1.7 Uso do *Next.js* para melhoria do *SEO*

Outra melhoria futura recomendada é a utilização do *Next.js* para otimizar o *SEO* (Search Engine Optimization) do sistema. O uso de renderização do lado do servidor (*SSR*) e a geração de páginas estáticas seriam estratégias eficazes para melhorar a indexação pelos motores de busca, proporcionando um melhor ranqueamento nas pesquisas. Além disso, o *Next.js* otimiza automaticamente o carregamento de páginas, o que também contribuiria para um tempo de resposta mais rápido, um fator crucial tanto para *SEO* quanto para a experiência do usuário. A implementação dessa tecnologia traria benefícios significativos em termos de visibilidade e descoberta do *e-commerce* nos motores de busca.

6.1.8 Integração com o sistema dos Correios para cálculo de preços dinâmicos de entrega

Para ampliar ainda mais a funcionalidade do sistema, um trabalho futuro interessante seria a integração com a *API* dos Correios para cálculo dinâmico de preços de entrega. Essa integração permitiria que os usuários recebessem informações em tempo real sobre o custo do frete, com base em dados como peso, dimensões do pacote e destino. A funcionalidade agregaria maior precisão e transparência ao processo de compra, reduzindo dúvidas e promovendo uma experiência de usuário mais eficiente. A implementação dessa integração aumentaria a competitividade do e-commerce ao oferecer opções de entrega personalizadas e preços atualizados automaticamente, facilitando a decisão de compra.

6.1.9 *Cache* de imagens em *CDN*

Um futuro aprimoramento que poderia ser explorado é a utilização de uma *CDN* (Content Delivery Network) para o *cache* de imagens. Esse recurso aumentaria a eficiência no carregamento de imagens, distribuindo-as em servidores geograficamente mais próximos dos usuários. O *cache* de imagens reduziria o tempo de carregamento das páginas e proporcionaria uma navegação mais rápida, especialmente para clientes em regiões distantes ou com conexão mais lenta. Além de melhorar a experiência do usuário, essa abordagem também beneficiaria o *SEO*, já que o tempo de carregamento é um fator de ranqueamento. A implementação de uma *CDN* traria vantagens significativas para a escalabilidade e a performance do sistema, especialmente em momentos de tráfego intenso.

6.1.10 Próximos passos

Com a conclusão e implementação das melhorias propostas, o próximo passo crítico é disponibilizar oficialmente essas atualizações para os clientes. Isso envolve a realização de testes extensivos para garantir a estabilidade e eficácia de todas as novas funcionalidades. Após a verificação da integridade do sistema em ambiente de produção, as melhorias serão incorporadas ao sistema, oferecendo aos usuários uma experiência aprimorada.

Além disso, a integração completa com o *Stripe* em ambiente de produção é essencial para assegurar uma transição suave nas transações financeiras. Esse processo envolve a ativação do ambiente de pagamento para produção, garantindo que todas as transações ocorram de acordo com as configurações e segurança necessárias. A fase de integração com o *Stripe* em produção demandará atenção minuciosa para garantir a segurança das transações financeiras e a conformidade com as normas de pagamento online.

Durante esses próximos passos, também é crucial manter uma comunicação transparente com os clientes, informando sobre as melhorias implementadas, novas funcionalidades disponíveis e a integração efetiva do sistema com o *Stripe* em produção. Esse diálogo transparente não apenas cria confiança, mas também proporciona aos clientes informações valiosas sobre as atualizações realizadas, incentivando a adoção e utilização contínua da plataforma de *e-commerce*.

Finalmente, após a implementação oficial, o monitoramento contínuo do desempenho do sistema e a análise de feedbacks dos usuários serão fundamentais. Isso possibilitará a identificação de eventuais ajustes necessários e a manutenção de uma plataforma sempre atualizada e adaptada às necessidades do público-alvo. O ciclo de *feedback* contínuo garantirá a evolução constante da plataforma, mantendo-a competitiva e alinhada com as expectativas dos clientes.

Referências

Artia. *Gestão de Projetos. O que é e TUDO sobre como gerenciar projetos*. 2019. Disponível em: <https://artia.com/blog/gestao-de-projetos-o-que-e-para-que-serve/>. Citado na página 39.

COOPER, A.; REIMANN, R.; CRONIN, D. *About Face 3: The Essentials of Interaction Design*. [S.l.]: Wiley, 2007. Citado na página 20.

COSTA, P. T. G. C. da et al. E-commerce no brasil: revisão sistemática de literatura de 2011 a 2021 / e-commerce in brazil: systematic literature review from 2011 to 2021. *Brazilian Journal of Business*, v. 3, n. 4, p. 2969–2982, ago. 2021. Disponível em: <https://ojs.brazilianjournals.com.br/ojs/index.php/BJB/article/view/34803>. Citado na página 27.

FILHO, W. d. P. P. *Engenharia de Software*. S.l.: LTC, 2003. v. 2. Citado na página 37.

HAHN, E. *Express in Action: Writing, building, and testing Node.js applications*. S.l.: Manning Publications, 2016. Citado na página 40.

JAVASCRIPT. 2020. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Citado na página 23.

JS, N.; JS, N. *Node.js*. 2020. Citado na página 24.

LAUDON, K. C.; LAUDON, J. P. *Sistemas de Informação Gerencial*. [S.l.]: Pearson Education do Brasil, 2016. Citado na página 18.

MACHADO, F. N. R. *Análise e Gestão de Requisitos de Software: Onde nascem os sistemas*. S.l.: Saraiva Educação SA, 2018. Citado na página 36.

MARCORATTI, J. C. *O processo de Software*. 2014. Disponível em: <http://www.macoratti.net/proc_sw1.htm>. Citado na página 36.

MARTINS, A. C. C. *Projeto de Interfaces Gráficas para Web*. 113 p. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, 2007. Dissertação (Mestrado). Citado na página 21.

MDN. *Começando com React*. 2020. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/Comecando_com_React>. Citado na página 25.

OLIVEIRA, X. S. d. Desenvolvimento web - desenvolvimento e manutenção de um marketplace para a indústria da moda. 2022. Disponível em: <https://iconline.ipleiria.pt/handle/10400.8/8082>. Citado na página 26.

OSTERWALDER, A.; PIGNEUR, Y. *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. [S.l.]: Wiley, 2010. Citado na página 19.

WAZLAWICK, R. *Análise e design orientados a objetos para sistemas de informação: Modelagem com UML, OCL e IFML*. S.l.: Elsevier Brasil, 2016. Citado na página 36.

WAZLAWICK, R. S. *Análise e projeto de sistemas de informação orientados a objetos.* [S.l.]: Elsevier Editora, 2011. Citado na página 38.