UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Gabriel Luiz de Lima Soares

Portal Web para Otimização de Processos e Expansão Digital: Um Caso Aplicado à Ice Point Sorveteria

Uberlândia, Brasil

2026

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Gabriel Luiz de Lima Soares

Portal Web para Otimização de Processos e Expansão Digital: Um Caso Aplicado à Ice Point Sorveteria

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Orientador: Fabiano Azevedo Dorça

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2026

Gabriel Luiz de Lima Soares

Portal Web para Otimização de Processos e Expansão Digital: Um Caso Aplicado à Ice Point Sorveteria

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, agosto de 2026:

Fabiano Azevedo Dorça
Orientador

Renan Gonçalves Cattelan
Convidado 1

Jefferson Rodrigo de Souza
Convidado 2

Uberlândia, Brasil
2026

Aos meus pais, Valter e Elisângela,
pelo amor incondicional e por cada sacrifício que tornou possível a realização dos meus sonhos.
À minha irmã, Natália, por ser a alegria da família e por deixar meus dias mais leves.
À minha namorada, Eduarda, pelo companheirismo, suporte e amor durante toda essa jornada.
E aos meus amigos de faculdade, pela parceria fundamental, tanto nos desafios quanto nas alegrias.

Agradecimentos

Aos meus pais, Valter e Elisângela, expresso a minha mais profunda e eterna gratidão. Vocês são a base do meu caráter e o pilar principal que sustentou cada passo desta caminhada. Todos os sacrifícios que fizeram, as renúncias silenciosas e o amor incondicional depositado em mim não foram em vão. Este título e esta conquista não são apenas meus, mas, essencialmente, pertencem a vocês.

À minha família como um todo, agradeço pelo incentivo inesgotável em acreditar no meu potencial, mesmo quando as incertezas surgiam. Em especial, à minha irmã, Natália, que nos inunda de alegria todos os dias.

À minha namorada, Eduarda, meu muito obrigado por ser o meu maior alicerce. A sua parceria e o apoio emocional constante foram o refúgio seguro nas horas de maior pressão e esgotamento. O seu amor foi o que me ajudou a manter a serenidade e a perseverar até a linha de chegada.

Aos meus amigos de faculdade e da vida, agradeço por tornarem esta jornada tão memorável. Os obstáculos acadêmicos e os desafios práticos pareceram muito menores diante da nossa união. O companheirismo, as madrugadas de estudos e as conquistas divididas fizeram com que a carga da graduação se tornasse mais leve e repleta de sorrisos e boas lembranças.

Por fim, direciono meus mais sinceros agradecimentos ao corpo docente desta instituição, por toda a paciência, dedicação ao ensino e suporte que moldaram minha visão crítica e acadêmica. Estendo meu genuíno reconhecimento aos profissionais e líderes que cruzaram a minha trajetória no mercado de trabalho, com um destaque afetuoso às equipes da Algar Tech e da Algar Telecom, que acreditaram no meu talento e contribuíram de forma ímpar para o meu amadurecimento e consolidação técnica como desenvolvedor de *software*. A todos vocês, a minha reverência e gratidão por ajudarem a construir a pessoa e o profissional que sou hoje.

Resumo

Pequenas e Médias Empresas (PMEs) enfrentam desafios significativos na gestão de suas operações logísticas, frequentemente dependendo de processos manuais suscetíveis a falhas humanas e retrabalho. Este trabalho apresenta o desenvolvimento e a implementação de um sistema Web completo para a Ice Point Sorveteria, estruturado sob o padrão arquitetural MVC (*Model-View-Controller*). A aplicação engloba um *e-commerce* para o consumidor final (B2C) e um painel gerencial interno, com o objetivo de mitigar erros na locação de estoques limitados e automatizar o cálculo logístico. A infraestrutura tecnológica foi construída utilizando o *framework* Vue.js no *frontend*, NestJS no *backend* e Supabase atuando como *Backend-as-a-Service* (BaaS) para persistência em PostgreSQL e autenticação segura (SSO). Dentre as automações de regras de negócio validadas, destacam-se a trava mínima de atacado (acima de oitenta unidades), o intervalo logístico de doze horas para higienização de frotas e o cálculo dinâmico de fretes integrado à API do *Google Maps*. A qualidade do sistema é assegurada por uma suíte de testes unitários automatizados com o *framework* Jest, cobrindo onze cenários críticos de regras de negócio, complementada por um *pipeline* de Integração e Entrega Contínuas (CI/CD) via GitHub Actions que automatiza a verificação e a implantação a cada atualização do código-fonte. Os resultados indicam a total eliminação de despachos deficitários e uma padronização rastreável de receitas e agendamentos, consolidando a plataforma como um marco tecnológico das operações de *delivery* da empresa.

Palavras-chave: Engenharia de Software. *E-commerce*. Vue.js. NestJS. Automação Logística. CI/CD.

Abstract

Small and Medium Enterprises (SMEs) face significant challenges in managing their logistics operations, often relying on manual processes prone to human error and rework. This work presents the development and implementation of a comprehensive web system for Ice Point Sorveteria, structured under the MVC (*Model-View-Controller*) architectural pattern. The application encompasses an e-commerce platform for the end consumer (B2C) and an internal management dashboard, aiming to mitigate errors in renting limited stock and automate logistics calculations. The technological infrastructure was built using the Vue.js framework for the frontend, NestJS for the backend, and Supabase acting as a Backend-as-a-Service (BaaS) for PostgreSQL persistence and secure authentication (SSO). Among the validated automated business rules are the wholesale minimum threshold (over eighty units), the twelve-hour logistical interval for fleet sanitization, and dynamic freight calculation integrated with the Google Maps API. Software quality is ensured by an automated unit testing suite using the Jest framework, covering eleven critical business rule scenarios, complemented by a Continuous Integration and Continuous Delivery (CI/CD) pipeline via GitHub Actions that automates verification and deployment with every code update. The results indicate the complete elimination of deficit dispatches and a traceable standardization of revenues and scheduling, consolidating the platform as a technological milestone in the company's delivery operations.

Keywords: Software Engineering. E-commerce. Vue.js. NestJS. Logistics Automation. CI/CD.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
B2B	*Business-to-Business*
B2C	*Business-to-Consumer*
BaaS	*Backend-as-a-Service*
CD	*Continuous Delivery* (Entrega Contínua)
CEP	Código de Endereçamento Postal
CI	*Continuous Integration* (Integração Contínua)
CLS	*Cumulative Layout Shift*
CSS	*Cascading Style Sheets*
DER	Diagrama de Entidade-Relacionamento
DTO	*Data Transfer Object*
FCP	*First Contentful Paint*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IHC	Interação Humano-Computador
JSON	*JavaScript Object Notation*
JWT	*JSON Web Token*
LCP	*Largest Contentful Paint*
MVC	*Model-View-Controller*
NF-e	Nota Fiscal Eletrônica
PIX	Pagamento Instantâneo Brasileiro
PME	Pequenas e Médias Empresas
RBAC	*Role-Based Access Control* (Controle de Acesso Baseado em Perfil)

REST	*Representational State Transfer*
SLA	*Service Level Agreement* (Acordo de Nível de Serviço)
SPA	*Single Page Application*
SQL	*Structured Query Language*
SSO	*Single Sign-On*
TCC	Trabalho de Conclusão de Curso
UAT	*User Acceptance Testing* (Teste de Aceitação do Usuário)
UFU	Universidade Federal de Uberlândia
UI	*User Interface*
UML	*Unified Modeling Language*
URI	*Uniform Resource Identifier*
UUID	*Universally Unique Identifier*
UX	*User Experience*
VPS	*Virtual Private Server* (Servidor Privado Virtual)
WWW	*World Wide Web*

Sumário

1 Introdução

1.1 Contextualização

Na conjuntura contemporânea, a transformação digital tem se consolidado como fator determinante para a competitividade das pequenas e médias empresas (PMEs). Com o rápido avanço da tecnologia, a transição de processos puramente manuais para ambientes digitalizados deixou de ser apenas um diferencial de mercado para se tornar uma necessidade real de sobrevivência e crescimento. Estudos recentes apontam que as PMEs que investem em tecnologias digitais ampliam significativamente a sua produtividade (MICROSOFT, 2023) e conseguem alinhar melhor a tecnologia com seus objetivos de negócio (McKinsey & Company, 2023).

A adoção de portais Web e sistemas de informação ajuda não só a otimizar a rotina de trabalho interno, mas também melhora a relação com o cliente, garantindo uma experiência mais segura e organizada. Na área de Engenharia de Software, desenvolver essas soluções incentiva o uso de arquiteturas mais modernas e escaláveis. A própria literatura técnica destaca os benefícios de usar *frameworks* como o Vue.js e NestJS (IBRAHIM, 2023). Quando essas ferramentas são integradas a serviços de *Backend-as-a-Service* (BaaS), elas oferecem uma base sólida para a criação de sistemas completos de *e-commerce* e gestão.

Nesse contexto, o presente trabalho foi desenvolvido com o propósito de resolver problemas operacionais concretos da Ice Point Sorveteria por meio da criação de um portal Web de gestão e *e-commerce*.

1.2 Problema e Justificativa

A Ice Point Sorveteria é uma empresa real focada na fabricação e venda de sorvetes e picolés a baixo custo. Pelo seu perfil de negócio, ela não trabalha com entregas de varejo (unidades isoladas), preferindo concentrar suas encomendas em grandes quantidades, como o aluguel de carrinhos de picolé para festas, o fornecimento de potes de sorvete acima de 10 litros e pedidos com mais de 80 picolés. Historicamente, o principal foco financeiro dessas demandas sempre foi o aluguel dos carrinhos.

Antes do sistema proposto neste TCC, a captação de pedidos ocorria predominantemente de forma analógica, via telefone ou WhatsApp. No entanto, devido ao crescente volume de vendas e à complexidade das especificações de cada evento (como cores dos carrinhos e separação de sabores), a gestão descentralizada baseada em registros físicos passou a apresentar gargalos operacionais. A sobrecarga no atendimento dificultava o

controle de disponibilidade em tempo real, gerando esporádicos choques de agenda que demandavam readequações logísticas de última hora.

A ideia de desenvolver este portal surgiu exatamente dessa constatação empírica: o atendimento manual não escala de forma sustentável junto com o aumento da demanda. Empresas que operam nesse modelo perdem informações vitais e enfrentam gargalos no fluxo de aprovação. O desenvolvimento deste trabalho se justifica pela necessidade de modernizar a infraestrutura de vendas, pois, conforme evidenciado pelo mercado, PMEs que se digitalizam ampliam sua produtividade e o alinhamento estratégico, permitindo que a sorveteria suporte seu crescimento e absorva uma demanda maior de eventos sem perder a excelência e agilidade no atendimento. A criação de um sistema computacional centralizado elimina a latência da comunicação manual, automatizando o controle de disponibilidade e garantindo que as informações corretas sejam preenchidas no momento da encomenda, sendo uma solução real de escalabilidade para a Ice Point.

Ademais, para garantir a robustez dessa digitalização, optou-se pela utilização da arquitetura *Model-View-Controller* (MVC). Essa escolha justifica-se pela capacidade do padrão MVC em promover um total desacoplamento entre a interface com o usuário (*frontend*) e as regras de negócio (*backend*), através da comunicação via API RESTful estruturada em JSON. Isso não apenas facilita a manutenibilidade do código, separando as responsabilidades de interação visual (View), orquestração de requisições (Controller) e persistência de dados (Model), mas também possibilita que a plataforma evolua de forma modular no futuro.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo principal deste trabalho é projetar, desenvolver e implementar um sistema Web integrado (portal de gestão e *e-commerce*) para a Ice Point Sorveteria, visando estruturar e automatizar o fluxo de encomendas de carrinhos de picolé, eliminar a perda de dados no atendimento manual e fornecer uma plataforma escalável que amplie a inserção digital da empresa.

1.3.2 Objetivos Específicos

Para a viabilização do objetivo geral, estabelecem-se os seguintes objetivos específicos:

- Realizar o levantamento de requisitos a partir do mapeamento dos processos manuais e deficiências operacionais relatadas pelos proprietários da empresa;

- Projetar a interface do sistema aplicando heurísticas de usabilidade (IHC) para garantir uma navegação intuitiva tanto para o cliente quanto para a gestão interna;
- Implementar o *frontend* e o *backend* empregando tecnologias modernas como Vue.js e NestJS, embasados em arquitetura *Model-View-Controller* (MVC) e boas práticas de Engenharia de Software;
- Desenvolver e estruturar a persistência e relacionamento de dados em nuvem, utilizando PostgreSQL integrado via plataforma Supabase;
- Garantir a qualidade e a confiabilidade do *software* por meio da implementação de testes unitários automatizados (Jest) sobre as regras de negócio críticas e da adoção de um *pipeline* de Integração e Entrega Contínuas (CI/CD) via GitHub Actions, assegurando barreiras de verificação automatizadas antes de cada implantação em produção.

1.4 Estrutura do Trabalho

A fim de apresentar o desenvolvimento metodológico e técnico deste projeto, o documento encontra-se estruturado da seguinte maneira: O Capítulo 2 revisa a Fundamentação Teórica, abordando os conceitos de desenvolvimento Web, comércio eletrônico e as tecnologias arquiteturais empregadas. O Capítulo 3 discute Trabalhos Relacionados, estabelecendo paralelos entre o presente sistema e soluções similares reportadas na literatura. O Capítulo 4 detalha a Engenharia de Requisitos e Modelagem, englobando o levantamento funcional, diagramas de caso de uso e modelagem de dados. No Capítulo 5, descrevem-se a Implementação e Resultados, evidenciando os detalhes técnicos do *frontend*, *backend* e os desafios superados durante o desenvolvimento. Por fim, o Capítulo 6 apresenta a Conclusão, analisando os impactos do sistema para a empresa, limitações do escopo e perspectivas para trabalhos futuros.

2 Fundamentação Teórica

2.1 Desenvolvimento Web: SPAs e APIs RESTful

A arquitetura do desenvolvimento Web evoluiu consideravelmente ao longo dos anos. Inicialmente estruturada em aplicações monolíticas que renderizavam páginas inteiras no servidor a cada requisição, a Web moderna migrou para o paradigma de interfaces ricas e sistemas distribuídos. Nesse contexto, destacam-se as *Single Page Applications* (SPAs) e as *Application Programming Interfaces* (APIs).

Conforme as definições consolidadas na literatura técnica de desenvolvimento *frontend*, (Mozilla Developer Network, 2024), as SPAs são aplicações Web que carregam uma única página HTML e atualizam o conteúdo dinamicamente à medida que o usuário interage. Essa abordagem elimina a necessidade de recarregar a página por completo a cada transição. O resultado desse modelo arquitetural é uma navegação fluida e uma experiência do usuário (UX) com níveis de interatividade e resposta similares aos de aplicativos nativos. Para prover os dados que alimentam a interface, as SPAs comunicam-se via protocolo HTTP com o servidor por meio de APIs RESTful. Uma API RESTful, por sua vez, expõe os recursos do sistema sob URLs específicas, respondendo requisições com dados estruturados em JSON. É essa comunicação assíncrona que garante o desacoplamento total entre as camadas de *frontend* e *backend*.

2.2 Comercialização Digital e E-commerce

No âmbito corporativo, a web impulsionou o surgimento do *e-commerce* (comércio eletrônico). O modelo predominante abordado neste trabalho é o *Business-to-Consumer* (B2C), no qual as transações ocorrem diretamente entre a empresa e o consumidor final. A arquitetura de um portal B2C exige não apenas uma interface convidativa, que exponha produtos de forma clara, mas também processos seguros de autenticação de usuários, gerenciamento de carrinhos e consolidação de pagamentos ou encomendas (TURBAN et al., 2017). Para que as PMEs atinjam sucesso nesse modelo, a confiabilidade da infraestrutura, a disponibilidade de integrações com redes sociais e a usabilidade do catálogo online são fatores fundamentais.

2.3 Tecnologias de Frontend: Vue.js e Ecossistema

Para a construção da camada de visualização, optou-se pelo *framework* Vue.js. O Vue.js destaca-se por sua reatividade baseada em *proxies* e por sua arquitetura voltada

à componentização, que permite encapsular estrutura (HTML), estilo (CSS) e lógica (JavaScript) em componentes independentes e reutilizáveis (VUE.JS, 2024).

No presente projeto, o desenvolvimento foi alicerçado na *Composition API* do Vue 3, utilizando a sintaxe `<script setup>`. Essa abordagem oferece maior flexibilidade para organizar o código baseado em domínios lógicos, em detrimento da rígida separação por propriedades da *Options API* clássica. Em conjunto, utilizou-se a linguagem TypeScript para garantir tipagem estática e detecção de erros em tempo de compilação, elevando a robustez do código. Além disso, o gerenciamento de estado global (como o controle de usuários logados e itens do carrinho) foi construído utilizando a biblioteca Pinia, o padrão oficial do ecossistema Vue para escalabilidade do fluxo de dados *frontend*.

2.4 Tecnologias de Backend: NestJS e Arquitetura

O *backend* do sistema foi desenvolvido com NestJS, um *framework* voltado para Node.js que impõe uma arquitetura sólida e previsível para o lado do servidor, fortemente inspirada pelo Angular (NESTJS, 2024). Sua base em TypeScript e a utilização extensiva de *Decorators* permitem a criação de um código limpo e altamente testável.

A arquitetura do NestJS baseia-se em conceitos fundamentais do padrão *Model-View-Controller* (MVC) adaptados para APIs, dividindo as responsabilidades em:

- **Controllers:** Responsáveis por interceptar as requisições HTTP, definir as rotas (*endpoints*) e delegar a lógica da aplicação para os provedores corretos.
- **Services (Providers):** Contêm as regras de negócio puras. Utilizam o padrão de projeto Injeção de Dependência (*Dependency Injection*) fornecido pelo NestJS para comunicar-se com outras partes do sistema.
- **Entities:** Classes que mapeiam estruturalmente as tabelas do banco de dados relacional.
- **DTOs (*Data Transfer Objects*):** Objetos tipados que definem o contrato de entrada de cada rota. Em conjunto com a biblioteca `class-validator` e o `ValidationPipe` em escopo global, os DTOs asseguram que apenas os campos explicitamente definidos sejam aceitos, mitigando ataques de *Mass Assignment*.

Para a comunicação com o banco de dados, integrou-se o *Object-Relational Mapper* (ORM) TypeORM, o qual traduz a lógica orientada a objetos das Entidades diretamente para *queries* SQL, facilitando operações de Cadastro, Leitura, Atualização e Exclusão (CRUD).

2.5 Persistência de Dados e Nuvem: PostgreSQL e Supabase

O armazenamento de informações adotou o PostgreSQL (GROUP, 2024), um banco de dados relacional de código aberto reconhecido por sua alta confiabilidade e consistência transacional. Contudo, em vez de configurar e hospedar a infraestrutura de banco de dados do zero, utilizou-se o Supabase, uma plataforma que opera no conceito de *Backend-as-a-Service* (BaaS) (SUPABASE, 2026).

O modelo BaaS terceiriza a infraestrutura subjacente de servidores, oferecendo bancos de dados gerenciados, APIs instantâneas e, de forma crucial, ferramentas de autenticação *built-in*. Justifica-se o uso do Supabase não apenas pela aceleração do *deployment*, mas, sobretudo, pela segurança garantida em transações e controle de acesso. O serviço já fornece integrações nativas para autenticação *OAuth* (como *login* via Google ou Facebook), com criptografia avançada de senhas em repouso. Dessa forma, as credenciais sensíveis dos clientes da sorveteria permanecem sob o sigilo e a gestão de um provedor de nuvem confiável, sem onerar o *backend* do sistema com lógicas complexas de criptografia manual.

2.6 Interação Humano-Computador (IHC)

A área de Interação Humano-Computador (IHC) estuda os princípios fundamentais para garantir que os sistemas sejam intuitivos, acessíveis e que proporcionem uma experiência do usuário satisfatória (NIELSEN, 1994). Como o sistema da Ice Point abrange o cliente final, a usabilidade precisa ser o foco central no *design* das telas.

Para nortear as escolhas de interface e validação de protótipos, aplicaram-se as dez heurísticas de usabilidade desenvolvidas por Nielsen (NIELSEN, 1994), sistematizadas em seu trabalho seminal como um conjunto de princípios gerais que orientam o projeto e a avaliação de interfaces interativas. A Tabela 1 apresenta as heurísticas de maior relevância para o presente sistema e os respectivos pontos de aplicação na interface.

Tabela 1 – Heurísticas de Nielsen aplicadas no portal Ice Point Sorveteria

Heurística	**Descrição**	**Aplicação no Sistema**
H1: Visibilidade do estado do sistema	Manter o usuário informado sobre o que está acontecendo	Indicador de etapas e *feedbacks* visuais no fluxo de *Checkout*
H2: Correspondência com o mundo real	Usar linguagem e conceitos familiares ao usuário	Terminologia "Meus Pedidos", "Carrinho" e "Sabores" em vez de termos técnicos
H3: Controle e liberdade do usuário	Oferecer saídas claramente sinalizadas de ações indesejadas	Botão de cancelamento de pedido e navegação sem becos sem saída
H5: Prevenção de erros	Projetar o sistema para evitar que problemas ocorram	Validação em tempo real de formulários, bloqueio de datas inválidas e CEPs fora de Uberaba
H8: Estética e *design* minimalista	Não exibir informações irrelevantes; cada unidade extra compete com as relevantes	Interface limpa com hierarquia visual clara nos catálogos e painéis administrativos
H9: Ajudar a reconhecer, diagnosticar e corrigir erros	Mensagens de erro em linguagem simples, que indiquem o problema e sugiram solução	Mensagens descritivas ao bloquear pedidos (ex.: "Pedido mínimo de 80 picolés por encomenda")

Fonte: Adaptado de NIELSEN (1994). Elaborado pelo autor (2026).

Tais conceitos garantem que a transição do processo manual para o digital seja natural e sem atritos para os utilizadores, reduzindo a curva de aprendizado tanto para o cliente final quanto para os colaboradores internos da sorveteria.

2.7 Qualidade de Software: Testes Automatizados

A garantia de qualidade em sistemas computacionais modernos baseia-se fortemente na automação de testes. Segundo SOMMERVILLE (SOMMERVILLE, 2011), os testes de *software* constituem o principal mecanismo para verificar se um sistema atende às suas especificações e para identificar defeitos antes que estes impactem o ambiente de produção.

Dentre as categorias de testes, os **testes unitários** representam o nível mais granular da pirâmide de testes: verificam unidades individuais de código, tipicamente um método ou função, de forma isolada de suas dependências externas (BECK, 2002). Para viabilizar esse isolamento, recorre-se à técnica de ***mocking*** (ou dublê de teste), na qual dependências reais (como bancos de dados, serviços de *e-mail* e APIs externas) são substituídas por implementações controladas que simulam comportamentos específicos durante a execução do teste. Essa abordagem garante que os testes sejam *determinísticos*, rápidos e livres de efeitos colaterais em qualquer ambiente (BECK, 2002).

No ecossistema NestJS e Node.js, o *framework* **Jest** consolidou-se como o padrão para implementação de testes unitários. O NestJS oferece suporte nativo ao Jest por meio do pacote `@nestjs/testing`, que disponibiliza o utilitário `TestingModule`: ele permite instanciar os *Services* da aplicação com todas as dependências substituídas por *mocks*

controlados, replicando fielmente os princípios de Injeção de Dependência que estruturam o *framework* (NESTJS, 2024). Dessa forma, é possível exercitar isoladamente as regras de negócio mais críticas do sistema sem depender de conectividade com o banco de dados ou com serviços externos.

2.8 Integração e Entrega Contínuas (CI/CD)

No contexto do desenvolvimento de *software* moderno, as práticas de Integração Contínua (*Continuous Integration* - CI) e Entrega Contínua (*Continuous Delivery* - CD) consolidaram-se como pilares da Engenharia de *Software* de alta qualidade (FOWLER; FOEMMEL, 2006). A CI consiste na prática de integrar frequentemente o trabalho dos desenvolvedores a um repositório compartilhado, com a execução automática de compilações e testes a cada submissão de código. A CD estende esse conceito ao automatizar as etapas de empacotamento e implantação (*deploy*), de modo que qualquer revisão de código validada pelos testes possa ser entregue ao ambiente de produção de forma rápida e confiável (HUMBLE; FARLEY, 2010).

Ferramentas como o *GitHub Actions* implementam esses princípios por meio de *pipelines* declarativas, definidas em arquivos YAML, que orquestram sequencialmente etapas de *build*, teste e implantação disparadas por eventos do repositório, como um *push* na *branch* principal. Essa automação elimina o chamado "*deploy* manual", reconhecidamente um dos principais vetores de erros humanos na transição entre desenvolvimento e produção (HUMBLE; FARLEY, 2010), estabelecendo barreiras de qualidade automáticas que asseguram que apenas código verificado e testado alcance os usuários finais.

3 Trabalhos Relacionados

A pesquisa desenvolvida por Kelvin Lehrback Guilherme (GUILHERME, 2023) tem como principal objetivo a criação de uma aplicação mobile Web voltada à comercialização de jogos eletrônicos clássicos, popularmente conhecidos como retro games. A proposta busca atender demandas específicas de colecionadores e entusiastas desse nicho, enfrentando desafios como a dificuldade de encontrar produtos originais, a falta de critérios padronizados de precificação e os riscos associados a negociações em plataformas genéricas. Para isso, o autor adotou uma abordagem estruturada, que incluiu o levantamento de requisitos, modelagem de dados, prototipação da interface e a implementação de um sistema utilizando o *framework* Ionic no *front-end*, Node.js no *back-end* e PostgreSQL como banco de dados. A solução desenvolvida contempla funcionalidades completas, como criação de anúncios, sistema de busca, gerenciamento de vendas, compras, sistema de reputação de vendedores e integração com serviços externos como Imgur e Gmail. Os resultados obtidos indicam que o sistema é funcional, com uma interface amigável e uma estrutura técnica robusta. Nesse sentido, este trabalho se conecta diretamente ao projeto de TCC em desenvolvimento, uma vez que ambos visam a construção de plataformas de *e-commerce* adaptadas para nichos específicos, com ênfase na digitalização de pequenos negócios e no uso de tecnologias modernas e boas práticas de Engenharia de Software.

Já o trabalho de Brenner Borges (BORGES, 2023) tem como foco a modernização e reimplantação do sistema SODD, utilizado pela Faculdade de Computação da Universidade Federal de Uberlândia para organizar a distribuição de disciplinas entre docentes. O projeto foi conduzido com base em metodologias ágeis, como o uso do Kanban e o BDD (Behavior Driven Development), além da aplicação dos princípios SOLID e de uma arquitetura fundamentada em serviços e repositórios. As tecnologias empregadas incluíram Node.js, TypeScript, Docker, PostgreSQL e Heroku, com uma atenção especial à criação de ambientes separados para desenvolvimento, homologação e produção. Os resultados alcançados demonstraram melhorias significativas em termos de escalabilidade, segurança e facilidade de manutenção do sistema, representando um avanço técnico considerável em relação à versão anterior. Diante disso, este trabalho dialoga diretamente com o projeto de TCC aqui proposto, pois ambos compartilham o objetivo de transformar processos institucionais e empresariais através de soluções digitais modernas, priorizando a organização do código, a escalabilidade da aplicação e a experiência do usuário final.

Por fim, Dias e Mello (DIAS; MELLO, 2024) propõem o desenvolvimento de uma plataforma digital voltada à compra e venda de veículos, com ênfase na usabilidade, segurança e análise de dados para apoiar a tomada de decisões estratégicas. A solução foi construída com um *front-end* em Vue.js e Node.js, integrando-se a um *back-end* desenvolvido

em C# com .NET, estruturado sob a arquitetura MVC. Para garantir a qualidade da interface, foram aplicadas as heurísticas de Nielsen, resultando em um sistema acessível e intuitivo. Durante o desenvolvimento, dados fictícios foram utilizados para simular interações reais e analisar métricas como tempo médio de navegação, número de visitas, vendas e receitas mensais. Os testes realizados comprovaram a eficácia das funcionalidades propostas, especialmente na recuperação de desempenho após cenários simulados de instabilidade, evidenciando o papel estratégico da análise de dados na evolução do sistema. Dessa forma, este trabalho se relaciona fortemente com o presente projeto, pois ambos se dedicam à construção de plataformas Web voltadas à digitalização de processos comerciais, utilizando tecnologias modernas, priorizando a experiência do usuário e adotando práticas voltadas à escalabilidade e à inteligência na tomada de decisões.

3.1 Síntese e Comparativo entre Trabalhos Relacionados

Os trabalhos apresentados exploram o desenvolvimento Web sob diferentes óticas, trazendo contribuições relevantes seja na faceta do *e-commerce* para o consumidor final (GUILHERME, 2023; DIAS; MELLO, 2024) ou na gestão de processos internos corporativos e acadêmicos (BORGES, 2023). Em alinhamento com essas iniciativas e buscando expandir tais abordagens, o presente projeto prossegue nessa linha de pesquisa ao propor a construção de uma solução híbrida. Essa solução unifica as melhores práticas do *e-commerce* B2C (catálogo e encomendas) com um portal de gestão administrativa robusto para os funcionários da Ice Point, integrando ambas as vertentes.

A justificativa para essa abordagem híbrida ancora-se na realidade das Pequenas e Médias Empresas (PMEs) que geralmente não dispõem de orçamento ou maturidade técnica para adquirir, licenciar e manter dois sistemas computacionais apartados, um ERP para controle interno de funcionários e uma plataforma de *e-commerce* separada para os clientes. Ao unificar essas duas frentes em um único ecossistema alimentado por uma arquitetura *Backend-as-a-Service* (BaaS), o sistema proposto entrega uma solução de engenharia altamente eficiente e economicamente viável para o perfil da Ice Point Sorveteria.

Dessa forma, a presente pesquisa busca atuar de forma complementar aos estudos da área. Ao dar continuidade aos cenários analisados em provas de conceito (DIAS; MELLO, 2024), este projeto foca na aplicação prática de tais fundamentos na resolução de um gargalo operacional cotidiano de uma empresa de mercado. Adicionalmente, busca-se expandir o leque arquitetural com a adoção do Supabase (BaaS), explorando alternativas emergentes de infraestrutura em nuvem que somam-se aos modelos de hospedagem convencional já consolidados na literatura.

A Tabela 2 sintetiza as principais características dos trabalhos analisados em

contraste com o sistema desenvolvido para a Ice Point Sorveteria.

Tabela 2 – Comparativo entre os trabalhos relacionados e a solução proposta

Trabalho	**Domínio / Foco**	**Stack Principal**	**Infraestrutura / DB**	**Aplicação**
Guilherme (2023)	E-commerce (Nicho)	Ionic, Node.js	PostgreSQL (Convencional)	Empresa Real
Borges (2023)	Gestão Acadêmica	Node.js, TypeScript	Docker, Heroku	Instituição Real
Dias (2024)	E-commerce (Veículos)	Vue.js, .NET (C#)	SGBD Relacional (Convencional)	Simulação Fictícia
Este Trabalho	**Gestão + E-commerce**	**Vue.js, NestJS**	**BaaS (Supabase)**	**Empresa Real**

Fonte: Elaborado pelo autor (2026).

4 Engenharia de Requisitos e Modelagem

A engenharia de requisitos e a modelagem do sistema constituem as bases para garantir que a solução computacional atenda às reais necessidades operacionais do negócio. Neste capítulo, descreve-se a metodologia adotada para o levantamento de dados, detalham-se as regras de negócio implementadas e apresenta-se a modelagem técnica do sistema da Ice Point Sorveteria.

4.1 Metodologia de Desenvolvimento

O projeto foi conduzido por meio de um ciclo iterativo e incremental, pautado na entrega contínua de valor. A primeira fase consistiu no levantamento de requisitos a partir da análise dos processos manuais anteriores, como anotações físicas e extrema dependência da memória humana que geravam gargalos severos no atendimento das encomendas.

Após o mapeamento das necessidades, procedeu-se à etapa de prototipação da interface de usuário (UI). Essa fase de *design* foi conduzida em paralelo e fundamentada pelas práticas da disciplina de Interação Humano-Computador (IHC), permitindo a aplicação de metodologias formais como a definição de *personas* e a avaliação baseada nas heurísticas de Nielsen (NIELSEN, 1994). Para validar os fluxos propostos, os protótipos foram submetidos a testes de usabilidade práticos com outros discentes utilizando o método *Think Aloud* (Pensar Alto). Durante as sessões, os usuários verbalizavam suas ações enquanto percorriam fluxos críticos do sistema, desde a navegação pelo catálogo até a simulação do *checkout* (inserção de endereço local, seleção de data/hora, forma de pagamento e aceite de termos).

Essa abordagem empírica revelou *insights* valiosos de usabilidade antes da codificação definitiva em Vue.js. Identificaram-se, por exemplo, falhas de legibilidade e contraste de cores quando o sistema operacional do usuário forçava o modo escuro (*dark mode*), erros ortográficos pontuais na interface e inconsistências lógicas visuais na etapa de prototipagem, como o cálculo incorreto do total da compra que considerava apenas o valor do frete. A detecção precoce dessas anomalias garantiu o refinamento antecipado das telas, entregando uma interface acessível e de baixa carga cognitiva para o cliente final.

4.2 Requisitos do Sistema e Regras de Negócio

A partir da análise do domínio e das limitações operacionais da empresa, os requisitos foram categorizados em funcionais e não funcionais. A Tabela 3 apresenta os

principais requisitos funcionais mapeados e implementados no código.

Tabela 3 – Requisitos Funcionais (RF) implementados no sistema

ID	**Descrição**
RF01	O sistema deve permitir que clientes realizem encomendas de produtos e carrinhos de picolé.
RF02	O sistema deve travar o fechamento de pedidos fora da faixa operacional estipulada (mínimo de 80 e máximo de 2.000 picolés).
RF03	O sistema deve calcular o frete com base na distância de entrega utilizando a API do Google Maps.
RF04	O sistema deve restringir a modalidade de entrega exclusivamente a CEPs da cidade de Uberaba.
RF05	O sistema deve conceder dinamicamente 10% de desconto para pagamentos via PIX ou dinheiro na entrega.
RF06	O sistema deve respeitar um intervalo logístico mínimo de 12 horas entre locações do mesmo carrinho.
RF07	O sistema deve integrar a API ViaCEP para autopreenchimento de endereço, otimizando a usabilidade.
RF08	O sistema deve restringir agendamentos (entregas e retiradas) para o prazo mínimo de um dia útil (D+1).
RF09	O sistema deve validar a alocação volumétrica, obrigando a adição de um carrinho para cada 250 picolés.
RF10	O sistema deve validar a maioridade legal do usuário (mínimo de 18 anos completos) no fluxo de finalização de encomenda.
RF11	O sistema deve bloquear o cancelamento autônomo via painel caso a entrega esteja a menos de 2 horas.
RF12	O sistema deve condicionar a finalização do pedido ao aceite explícito dos Termos e Condições (Checkbox).

Fonte: Elaborado pelo autor (2026).

Além dos requisitos funcionais, a conformidade normativa com a Lei Geral de Proteção de Dados (LGPD) foi elencada como o Requisito Não Funcional (RNF) mais crítico da aplicação, impondo que a coleta e o armazenamento de informações operem sob os preceitos de consentimento explícito e *Privacy by Design*. Adicionalmente, a aplicação das heurísticas de usabilidade de Nielsen e dos princípios de Interação Humano-Computador (IHC) também configuram-se como requisitos não funcionais essenciais do sistema, assegurando uma navegação acessível e fluida para os usuários finais.

A estruturação das regras de negócio (Tabela 3) reflete diretamente as limitações operacionais da Ice Point. Na gestão de capacidade produtiva, estabeleceu-se um limite mínimo e máximo (RF02), onde pedidos inferiores a 80 picolés são bloqueados sistemicamente

por inviabilizarem a logística dos equipamentos. Em contrapartida, encomendas massivas superiores a 2.000 unidades acionam um redirecionamento do cliente para o atendimento personalizado via WhatsApp, visto que demandas dessa magnitude exigem reconfigurações drásticas na linha de fábrica. Ainda no escopo físico, o sistema implementa um algoritmo de Alocação Volumétrica (RF09): a capacidade de cada carrinho é parametrizada no banco de dados para 250 picolés. A interface do *checkout* trava a finalização até que o usuário selecione carrinhos suficientes para comportar seu pedido (ex: um pedido de 501 picolés exige a seleção obrigatória de 3 carrinhos).

No âmbito da logística temporal, a regra de frota (RF06) impõe um intervalo de 12 horas para a higienização dos carrinhos. Complementarmente, aplica-se a regra de Agendamento D+1 (RF08), na qual a camada de interface do usuário impõe o bloqueio computacional da seleção de datas retroativas ou para o mesmo dia, assegurando à equipe prazo hábil para organizar a pauta de despachos. Caso o usuário opte por desistir de uma reserva, implementou-se a Trava de Cancelamento Emergencial (RF11): se o agendamento estiver a menos de 2 horas da concretização, tanto a camada de interface (*frontend*) quanto a API do *backend* bloqueiam a ação sistêmica e exibem um alerta modal exigindo contato telefônico com a equipe.

Em relação à geolocalização e precificação (RF03, RF04 e RF05), a validação de localidade é inflexível aos limites da cidade-sede. Para maximizar a proteção jurídica e a experiência do usuário (UX), a idade é submetida a uma Validação de Maioridade no cadastro (RF10). O formulário de logradouro integra-se ao ViaCEP (RF07) para autopreenchimento imediato a partir do CEP inserido. Por fim, o processo condiciona legalmente a compra através da aceitação obrigatória dos Termos e Condições via *checkbox* (RF12) e aplica a Precificação Dinâmica exibindo instantaneamente o desconto em caso de pagamentos realizados no ato da entrega (dinheiro ou PIX).

4.3 Arquitetura do Sistema: Implementação MVC

O sistema foi construído sob uma arquitetura de *software* em camadas. O *frontend* (Vue.js) atua como a *View*, gerenciando as rotas interativas e o consumo das APIs. O *backend* (NestJS), por sua vez, engloba o *Controller* e o *Model*, orquestrando a lógica de negócios e as operações de banco de dados.

Na implementação do NestJS, as requisições HTTP oriundas da *View* são interceptadas pelos *Controllers*, responsáveis por realizar a validação estrutural do corpo da requisição (*payload*) e mapear as rotas. Esta validação é reforçada por um `ValidationPipe` global configurado com as opções `whitelist: true` e `forbidNonWhitelisted: true`: o *pipe* rejeita automaticamente qualquer propriedade que não esteja declarada no DTO correspondente, eliminando vetores de *Mass Assignment Attack* sem exigir filtragem manual

nos *Services*. Os controladores delegam a execução para os *Services*, camadas que concentram puramente as regras de negócio intrínsecas, como a validação de disponibilidade no período de 12 horas entre as encomendas e o cálculo de fretes, e abstêm-se de lidar com a rede. Finalmente, os *Services* comunicam-se com a persistência de dados por intermédio das Entidades (*Entities*), que traduzem o paradigma orientado a objetos para transações SQL no banco de dados PostgreSQL.

4.4 Modelagem UML e de Banco de Dados

A representação visual do domínio foi documentada utilizando artefatos de *Unified Modeling Language* (UML) e diagramação relacional. Os modelos aqui dispostos concentram-se estritamente nos fluxos operacionais de *e-commerce* e gestão de encomendas já codificados, excluindo funcionalidades projetadas para implementações futuras.

4.4.1 Diagrama de Casos de Uso

O Diagrama de Casos de Uso estabelece as fronteiras do sistema e as interações primárias dos atores logados. Conforme ilustra a Figura 1, as permissões são categorizadas. O Cliente atua nas rotinas de visualização e finalização de encomendas, os Funcionários possuem alçada para visualizar encomendas pendentes e realizar o gerenciamento básico do catálogo de produtos e o Admin (proprietário) detém a prerrogativa de gerenciamento amplo, incluindo relatórios financeiros e gerenciamento de usuários.

Figura 1 – Diagrama de Casos de Uso do sistema Ice Point

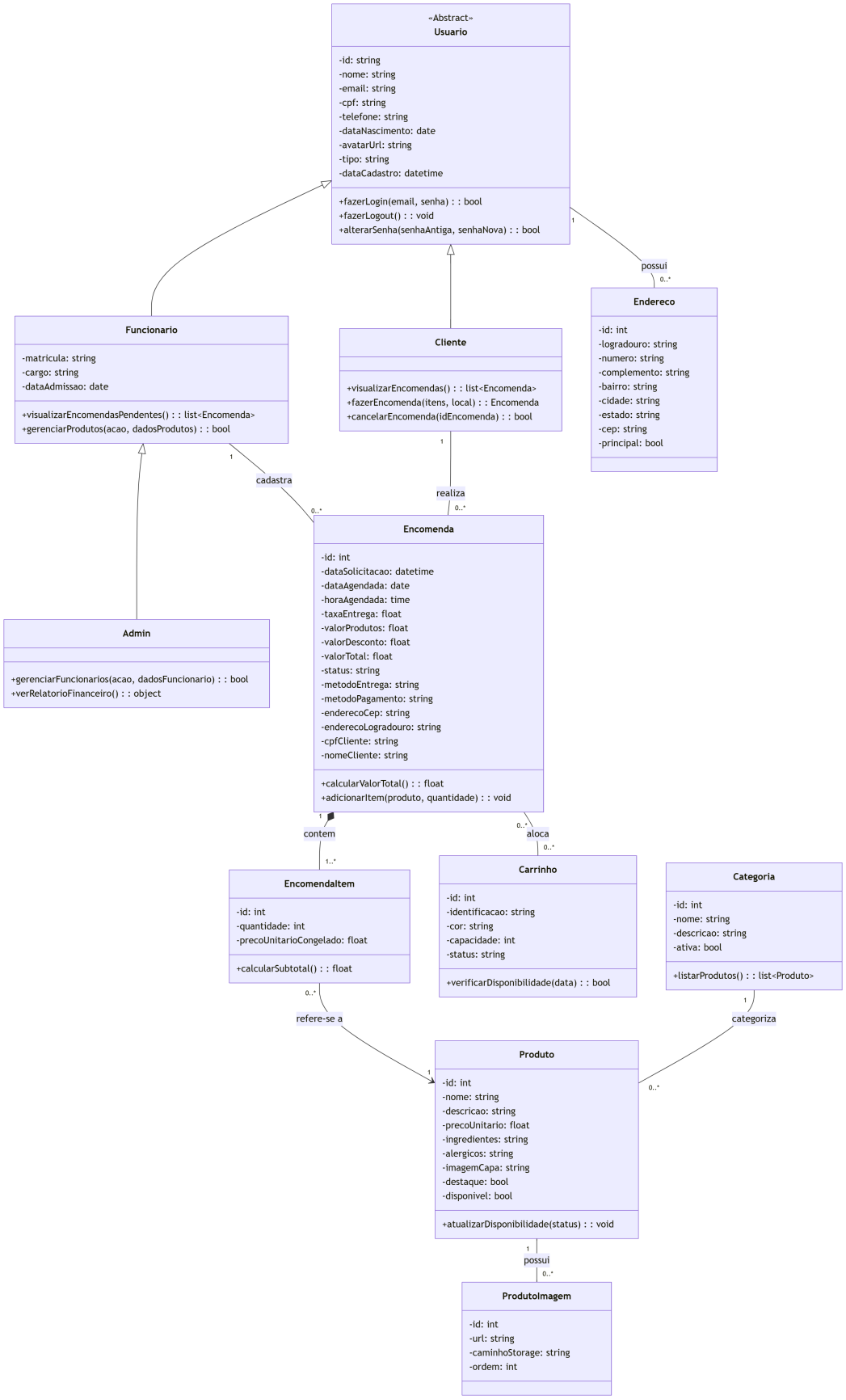
Fonte: Elaborado pelo autor (2026).

4.4.2 Diagrama de Classes

A estrutura orientada a objetos da aplicação é retratada no Diagrama de Classes (Figura 2). A modelagem aplica herança a partir da classe abstrata `Usuario`, que se

especifica em `Cliente`, `Funcionario` e `Admin`. Destaca-se a presença de restrições de negócio no encapsulamento de atributos, como `cpf` e `avatarUrl`. A classe `Encomenda` funciona como a entidade agregadora central: ela contém a composição de `EncomendaItem` e associa-se diretamente a uma ou mais instâncias de `Carrinho`, delegando a estes a responsabilidade de verificar sua própria disponibilidade de datas.

Figura 2 – Diagrama de Classes detalhando as entidades do sistema



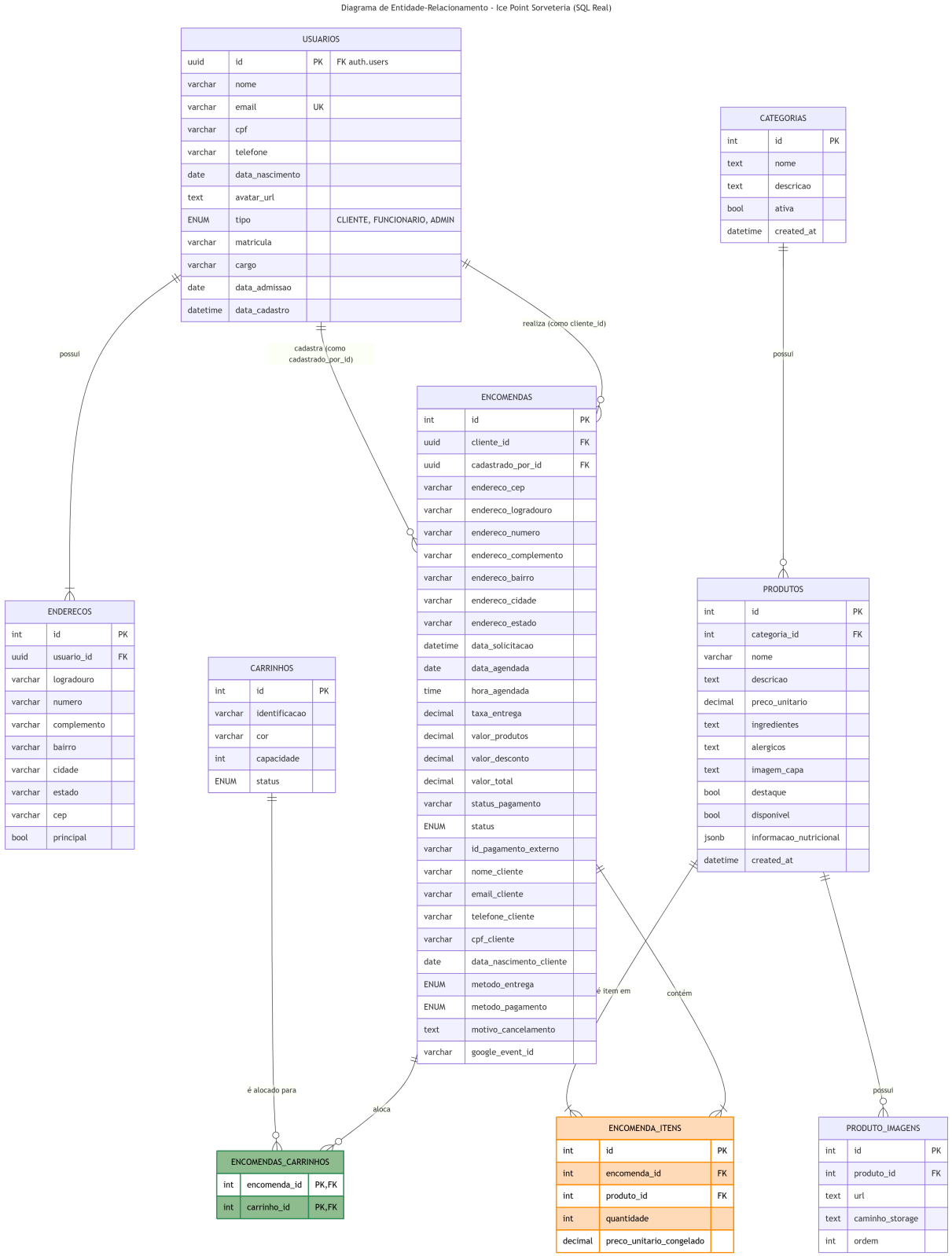
Fonte: Elaborado pelo autor (2026).

4.4.3 Diagrama de Entidade-Relacionamento (DER)

A modelagem lógica e física do banco de dados relacional está expressa no Diagrama de Entidade-Relacionamento (Figura 3), que reflete as escolhas de arquitetura de dados, notadamente a desnormalização de tabelas para preservação de histórico. A tabela `ENCOMENDAS` incorpora diretamente os dados instantâneos do usuário (ex: `nome_cliente`, `endereco_logradouro`, `cpf_cliente`) para "congelar" a informação no ato da compra, garantindo-se que, caso o usuário altere seu cadastro posteriormente, o histórico fiscal do pedido permaneça intacto e auditável.

Outro pilar estrutural é a utilização de chaves primárias do tipo `UUID` na tabela `USUARIOS`, alinhando a base de dados com a plataforma de autenticação do Supabase. Adicionalmente, a relação de cardinalidade múltipla (N:M) entre pedidos e frota de locação é resolvida fisicamente pela tabela associativa `ENCOMENDAS_CARRINHOS`.

Figura 3 – Diagrama de Entidade-Relacionamento (DER) refletindo a modelagem SQL implementada

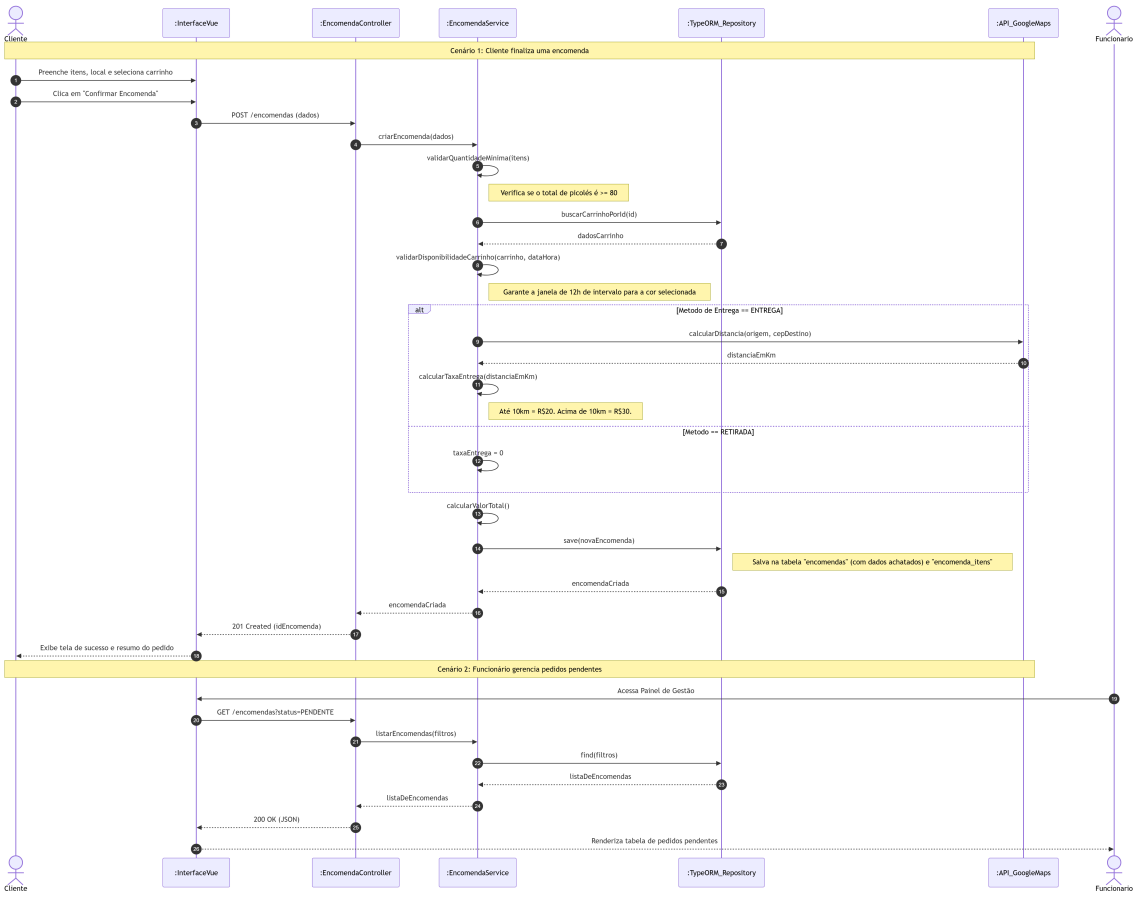


Fonte: Elaborado pelo autor (2026).

4.4.4 Diagrama de Sequência

O aspecto comportamental, síncrono e de requisições de rede é evidenciado pelo Diagrama de Sequência (Figura 4), que traça o percurso de mensagens durante a funcionalidade primária: o fechamento de uma encomenda. A modelagem rastreia as invocações desde a navegação na interface, submissão de itens para a validação no Controlador, os disparos assíncronos para microsserviços e APIs externas (para o cálculo geoespacial do frete), até a devolução do *status* de persistência para o usuário.

Figura 4 – Diagrama de Sequência demonstrando o fluxo assíncrono de finalização de encomendas



Fonte: Elaborado pelo autor (2026).

5 Implementação e Resultados

Este capítulo detalha a transposição da modelagem teórica para o ambiente de produção. Dessa maneira, são apresentadas a infraestrutura configurada, as interfaces gráficas validadas pelo cliente final e a implementação técnica das regras complexas de negócio utilizando o ecossistema Node.js e ferramentas nativas de geolocalização.

5.1 Configuração do Ambiente e Tecnologias Adotadas

A arquitetura do sistema foi consolidada em um ecossistema distribuído de três camadas estruturais: *frontend*, *backend* e persistência de dados (nuvem).

A interface de usuário (*frontend*) foi desenvolvida como uma *Single Page Application* (SPA) utilizando o *framework* Vue.js (versão 3). A adoção da *Composition API* proporcionou a criação de componentes reativos e altamente modulares. A navegação cliente-servidor foi delegada ao Vue Router, enquanto o estado global crítico da aplicação (como o estado do carrinho de compras e os dados do usuário autenticado) foi gerenciado de forma reativa pelo Pinia.

O *backend*, responsável pela blindagem das regras de negócio e garantia da integridade transacional, foi orquestrado em TypeScript com a adoção do NestJS. Este *framework* impôs uma estrutura modular (separando domínios como `users`, `products`, `encomendas` e `shipping`), expondo as funcionalidades de negócio através de uma API RESTful robusta.

Já o armazenamento e a segurança foram terceirizados estrategicamente para o Supabase, atuando como um provedor de *Backend-as-a-Service* (BaaS). Além de instanciar o banco de dados PostgreSQL para persistência relacional das entidades mapeadas no capítulo anterior, o Supabase proveu os serviços unificados de autenticação (SSO via Google, Facebook e E-mail), suprimindo a necessidade de manipulação local de senhas sensíveis e gerenciamento de sessões (*Tokens JWT*).

5.2 Interfaces do Usuário (Frontend)

A implementação da *View* traduziu os requisitos não funcionais de acessibilidade e usabilidade propostos na fase de prototipação para um ambiente Web responsivo.

5.2.1 Visão do Cliente (Catálogo e Checkout)

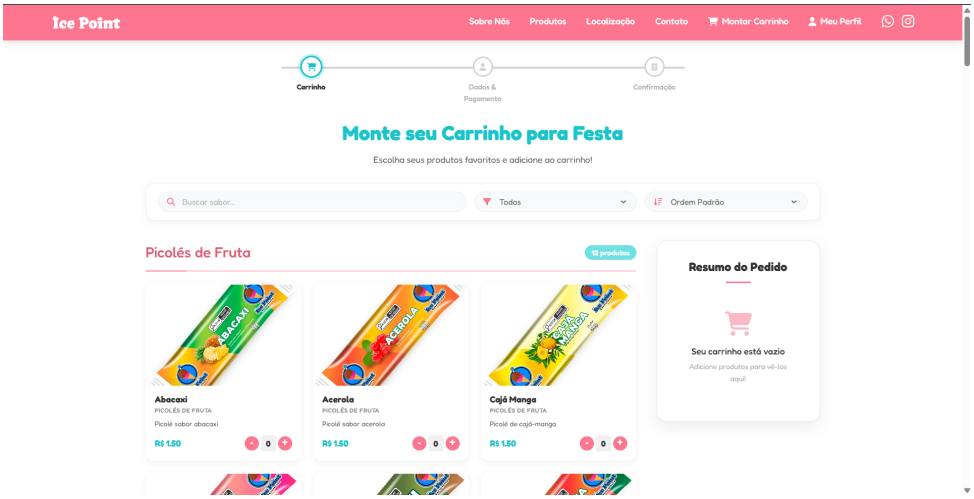
A experiência do consumidor final (*Business-to-Consumer* ou B2C) inicia-se na tela principal do portal e avança para a exploração do catálogo digital. A Figura 5 exibe o

painel de entrada, enquanto a Figura 6 demonstra o sistema de filtragem e exibição do catálogo de picolés e carrinhos.

Figura 5 – Tela Inicial do portal Ice Point

Fonte: Elaborado pelo autor (2026).

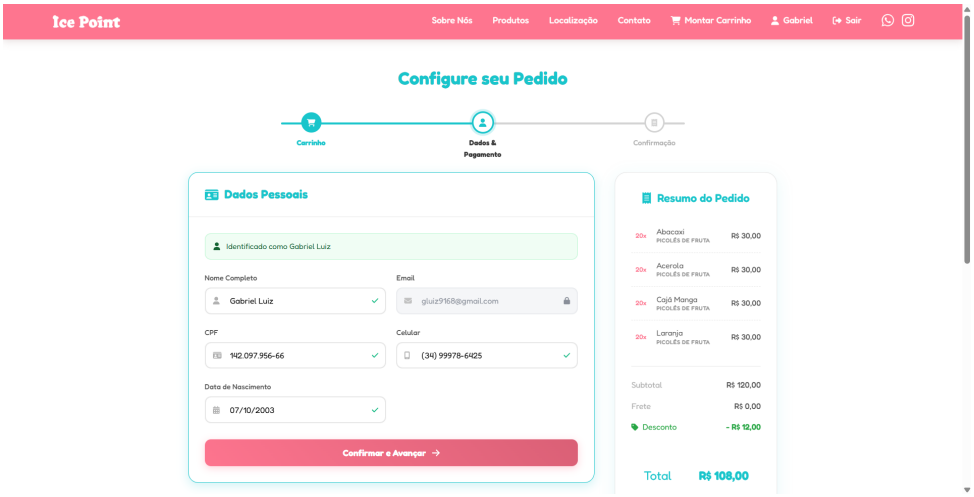
Figura 6 – Catálogo digital detalhado com produtos e disponibilidade



Fonte: Elaborado pelo autor (2026).

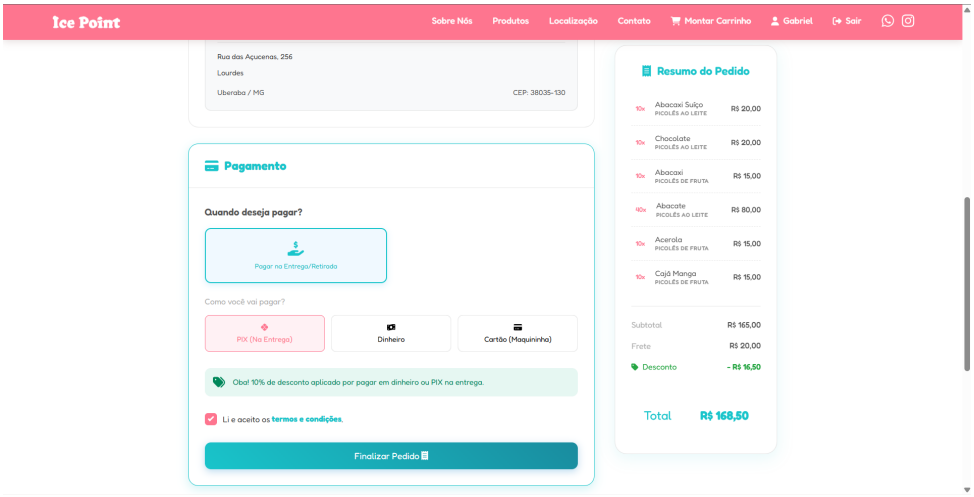
O fluxo culmina no *Checkout*, processo de fechamento de encomendas onde a plataforma exige as informações de preenchimento de endereço e calcula as taxas de frete de forma dinâmica, conforme exibido na Figura 7. A revisão minuciosa dos valores e confirmação da solicitação são ilustradas pela Figura 8. Por fim, a transparência e rastreabilidade para o cliente são asseguradas por meio do histórico unificado ilustrado na Figura 9.

Figura 7 – Preenchimento dos dados de entrega no Checkout



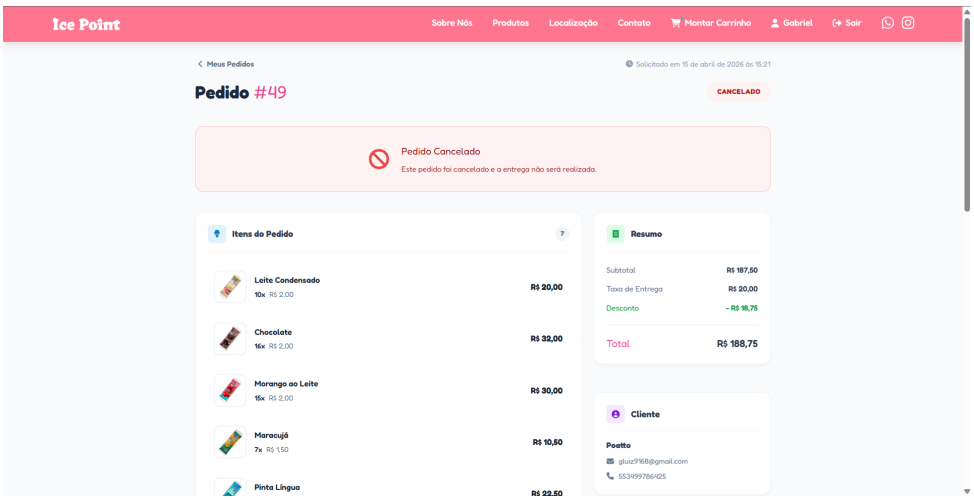
Fonte: Elaborado pelo autor (2026).

Figura 8 – Revisão do valor e fechamento da encomenda



Fonte: Elaborado pelo autor (2026).

Figura 9 – Visualização do histórico de compras (Meus Pedidos)



Fonte: Elaborado pelo autor (2026).

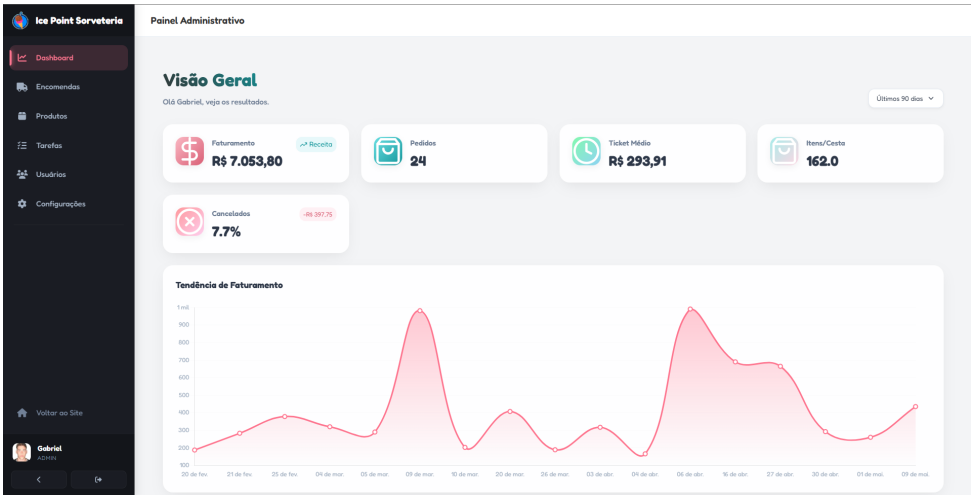
Um diferencial relevante de acessibilidade implementado é o *Guest Checkout*, por meio do qual o portal permite que o consumidor finalize uma encomenda sem possuir cadastro prévio, com a utilização de apenas um endereço de *e-mail* válido. O pedido é registrado no sistema associado àquele endereço eletrônico, podendo ser consultado e rastreado pelo cliente após a criação de uma conta com o mesmo *e-mail* utilizado. Esta abordagem elimina a fricção do cadastro obrigatório como barreira de conversão.

O ecossistema de comunicação transacional é gerenciado integralmente via *e-mail*. O serviço Resend, integrado ao *backend*, orquestra os seguintes disparos automáticos: confirmação de pedido imediatamente após o fechamento da encomenda, notificações de atualização de *status* nos eventos "Em Preparação", "Saiu para Entrega" e "Entregue" e *e-mails* de segurança para a redefinição de senha. Neste último fluxo, o usuário recebe um *link* único, de uso temporário e com prazo definido, para acessar a rota exclusiva de atualização de credencial, garantindo que a operação seja executada apenas pelo titular da conta.

5.2.2 Visão Administrativa (Gestão Interna)

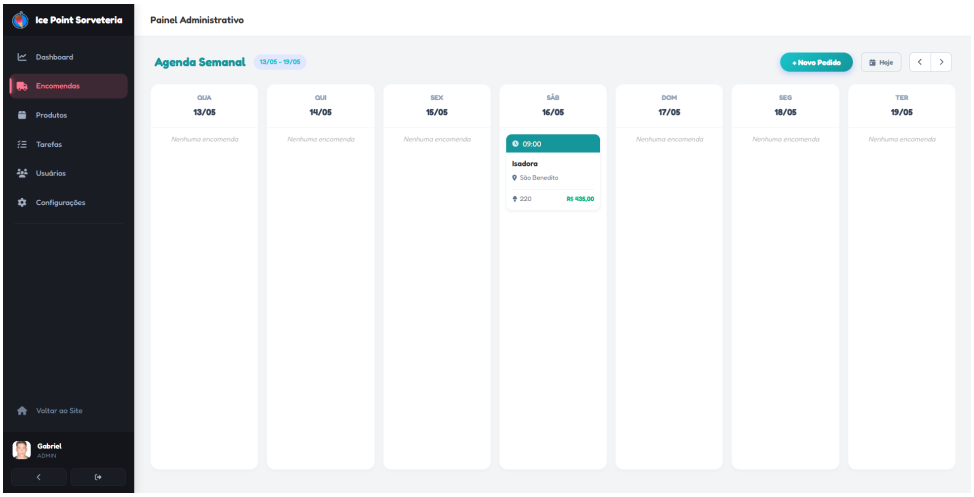
A operação administrativa e logística da Ice Point é centralizada em telas restritas aos perfis de Funcionário e Admin. A tela de *Dashboard* (Figura 10) compila indicadores gerenciais. A mitigação dos problemas práticos de perda de pedidos se consolidou com a esteira digital de Pedidos Pendentes (Figura 11), enquanto o catálogo B2C é refletido e editado instantaneamente através da interface ilustrada na Figura 12.

Figura 10 – Painel Administrativo (*Dashboard*)



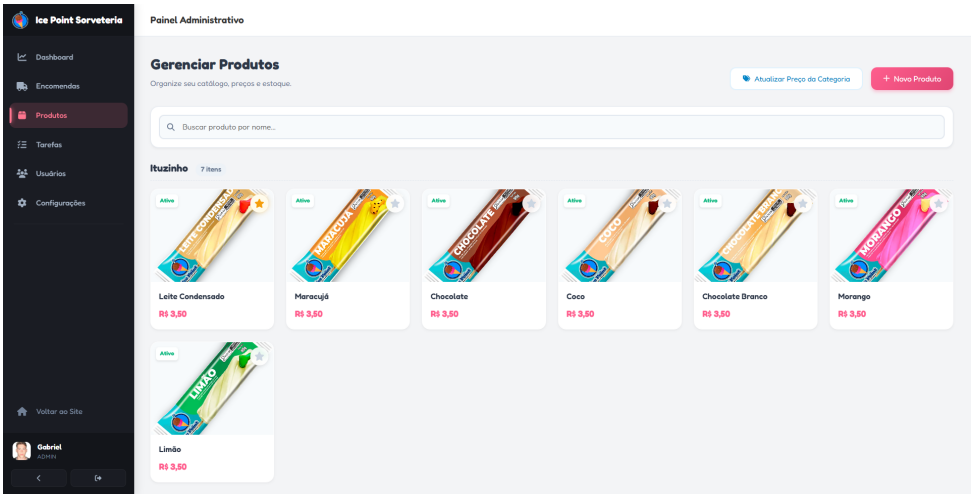
Fonte: Elaborado pelo autor (2026).

Figura 11 – Tela de controle de Pedidos Pendentes



Fonte: Elaborado pelo autor (2026).

Figura 12 – Painel de edição e gestão de produtos do catálogo



Fonte: Elaborado pelo autor (2026).

5.3 Análise de Código: Lógica Geoespacial de Frete

Dada a natureza operacional do negócio, uma das soluções técnicas adotadas foi a substituição do cálculo de fretes empírico por um sistema algorítmico parametrizado. O módulo `ShippingService` foi desenvolvido no *backend* NestJS para automatizar o distanciamento logístico via API do *Google Maps*.

O algoritmo injeta a origem fixa da sorveteria (CEP 38035-230) e a *string* normalizada de destino (montada com rua, bairro e CEP informados pelo cliente). A requisição HTTP segura (*server-to-server*) aciona o serviço *Distance Matrix* do Google. A distância devolvida em metros é convertida para quilômetros para então retroalimentar a lógica de negócios: raios geográficos inferiores a 10 km são precificados em R$ 20, ao passo que distâncias superiores incorrem em uma tarifa de R$ 30.

O Trecho de Código a seguir demonstra a codificação da lógica geoespacial de frete, evidenciando o tratamento de exceções (quando o endereço é inconsistente) e a apuração do valor de entrega no servidor, o qual recalcula a tarifa de forma independente do cliente, impedindo adulterações de preço no fluxo padrão de *checkout*.

```
async calculateShippingFee(
  addressData: CalculateShippingDto,
): Promise<{ distance: string; fee: number }> {
  const apiKey = this.configService.get<string>(
    'GOOGLE_MAPS_SHIPPING_API_KEY',
  );
```

```
  const destinationString = [
    `${addressData.street}, ${addressData.number}`,
    addressData.neighborhood ? addressData.neighborhood : '',
    `${addressData.city} - ${addressData.state}`,
    `CEP ${addressData.cep}`,
    'Brasil',
  ].filter(Boolean).join(', ');

  const distanceInMeters = await this.getDistanceMatrix(
    this.STORE_ORIGIN,
    destinationString,
    apiKey,
  );

  if (distanceInMeters === null) {
    throw new BadRequestException(
      'Endereço não localizado com precisão. Verifique o número e o CEP.'
    );
  }

  const distanceInKm = distanceInMeters / 1000;
  const price = this.calculatePriceLogic(distanceInKm);

  return { distance: distanceInKm.toFixed(2) + ' km', fee: price };
}

private calculatePriceLogic(km: number): number {
  if (km < 10) return 20;
  return 30;
}
```

5.4 Privacidade e Conformidade com a LGPD

Alinhado às restrições mapeadas na etapa de requisitos não funcionais, o sistema incorpora em seu núcleo arquitetural mecanismos ativos para garantir o direito à eliminação de dados, conforme previsto no Artigo 18, inciso VI, da Lei Geral de Proteção de Dados (LGPD). Quando um cliente solicita a exclusão de sua conta pela interface, o fluxo não realiza uma simples inativação, mas aciona uma rotina rigorosa de expurgo de dados sem comprometer a integridade dos relatórios financeiros da empresa.

Esse objetivo é atingido aplicando o conceito de *Hard Delete* em conjunto com a anonimização (*Soft Delete* indireto). Os dados pessoais identificáveis na tabela de usuários (nome, CPF, endereço, telefone e chaves de acesso) são erradicados definitivamente do banco de dados, impossibilitando qualquer rastreio reverso à identidade civil do cliente. Entretanto, os registros transacionais das encomendas previamente concluídas (valores pagos, data e produtos alugados) perdem a chave estrangeira (*foreign key*) que os vinculava ao usuário, tendo sua autoria desvinculada para proteger a privacidade do indivíduo. Dessa forma, as transações continuam contabilizadas no painel gerencial (*Dashboard*), de modo que a remoção total da identidade do indivíduo é consolidada sem corromper as métricas de vendas e de faturamento da empresa. O fluxo completo estabelece um Acordo de Nível de Serviço (SLA) de até sete dias úteis para a remoção total dos registros mediante envio de e-mail automatizado à administração para confirmação.

5.5 Infraestrutura de Implantação e Monitoramento

A arquitetura de produção (*deploy*) do sistema foi projetada para garantir autonomia, baixo custo inicial e alta disponibilidade, superando as limitações operacionais enfrentadas durante a fase de testes. Inicialmente, o *backend* foi hospedado em serviços de *tier* gratuito, porém, a latência gerada pela hibernação obrigatória dos servidores após períodos de ociosidade (*cold start*) inviabilizou a agilidade no processamento dos pedidos reais.

Para solucionar este entrave e preparar o sistema para futuras integrações (como o uso de APIs do WhatsApp), a hospedagem foi migrada para uma *Virtual Private Server* (VPS) dedicada na Hostinger, que demonstrou a melhor relação custo-benefício entre as alternativas avaliadas. Dentro deste ambiente privado, a infraestrutura foi inteiramente conteinerizada com Docker e orquestrada pelo Coolify, abrigando tanto a base de dados em Supabase quanto a aplicação servidora em NestJS.

A agilidade na manutenção deste ambiente é garantida por um *pipeline* automatizado de Integração e Entrega Contínuas (CI/CD), implementado via **GitHub Actions**. A cada novo *push* na *branch* principal (`main`), o fluxo automatizado executa sequencialmente quatro estágios de verificação: (i) instalação limpa das dependências via `npm ci`, garantindo reprodutibilidade exata do ambiente ao instalar versões travadas no *lockfile*; (ii) compilação completa do projeto com o compilador TypeScript do NestJS (*build gate*), que bloqueia o *deploy* em caso de erros de tipagem ou sintaxe; (iii) execução da suíte de testes unitários Jest (*quality gate*), que valida automaticamente as onze regras de negócio implementadas antes de qualquer alteração atingir a produção; e (iv) disparo autenticado do *deploy* para o Coolify via *webhook* HTTPS com token de autorização Bearer. Este fluxo garante que nenhum código quebrado ou com violação de regra de negócio possa ser implantado em

produção de forma inadvertida (HUMBLE; FARLEY, 2010).

O Trecho de Código a seguir exibe a implementação completa da *pipeline* CI/CD, evidenciando a sequência de estágios e a lógica de disparo condicional do *deploy*. Nota-se que as credenciais sensíveis (a URL do *webhook* e o token de autorização do Coolify) são armazenadas como *Secrets* criptografados no repositório GitHub, nunca expostos no código-fonte, prática fundamental de segurança:

```
name: CI/CD Pipeline - Ice Point Backend

on:
  push:
    branches:
      - main

jobs:
  build-test-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout do Codigo
        uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '20'
          cache: 'npm'

      - name: Instalar Dependencias
        run: npm ci

      - name: Build do Projeto (NestJS)
        run: npm run build

      - name: Rodar Testes (Jest)
        run: npm run test

      - name: Disparar Deploy no Coolify
        run: |
          curl --request GET "${{ secrets.COOLIFY_WEBHOOK_URL }}" \
```

```
--header "Authorization: Bearer ${{ secrets.COOLIFY_TOKEN }}"
```

O *frontend*, por sua vez, foi implantado na plataforma Vercel. A escolha foi motivada não apenas por sua profunda integração com o ecossistema Vue.js, mas pela disponibilidade do *Vercel Analytics* e do monitoramento de *Web Vitals*. A análise ativa destas métricas, verificada durante a Fase Piloto (maio de 2026), revelou *insights* vitais para o negócio: mais de 70% do tráfego de usuários provém de dispositivos móveis, validando a priorização do *design responsivo* adotada desde a fase de prototipação. Além disso, métricas de performance vitais como *First Contentful Paint* (FCP) e *Largest Contentful Paint* (LCP) mantêm-se em índices de alta performance (abaixo de 1,3s), embora o rastreamento contínuo tenha sinalizado oportunidades pontuais de otimização no *Cumulative Layout Shift* (CLS). O domínio oficial da empresa (`icepoint.com.br`) também é gerenciado via Vercel. Para finalizar a malha de infraestrutura, integrou-se o serviço Resend, garantindo o disparo transacional gratuito de *e-mails*.

5.5.1 Segurança da API

Além da validação estrutural via `ValidationPipe`, a API NestJS implementa duas camadas adicionais de segurança em nível de *middleware*, configuradas na inicialização da aplicação. A primeira é o módulo **Helmet**, que injeta automaticamente um conjunto de cabeçalhos HTTP de segurança, como *Content-Security-Policy*, *X-Frame-Options* e *X-XSS-Protection*, mitigando classes de ataques conhecidas como *Cross-Site Scripting* (XSS) e *Clickjacking*. A segunda é o módulo **Throttler** (*rate limiting*), que limita a taxa de requisições aceitas por endereço IP em uma janela de tempo configurável, protegendo os *endpoints* de autenticação e *checkout* contra ataques automatizados de força bruta e enumeração. Essas camadas de proteção atuam de forma transparente, sem impactar a experiência do usuário legítimo, e são parte integrante de uma postura de *security by design* adotada na arquitetura do *backend*.

5.6 Privacidade e Conformidade Legal (LGPD)

Diante do tratamento de dados sensíveis e pessoais de clientes (como nome, endereço e CPF), a aplicação do conceito de *Privacy by Design* tornou-se indispensável. A arquitetura foi desenvolvida visando não apenas a segurança lógica, mas também o rigoroso atendimento à Lei Geral de Proteção de Dados (LGPD).

No ambiente da aplicação (*frontend*), foram criadas rotas explícitas de transparência, notadamente a `/politica-privacidade` e a `/termos`, que informam o usuário, de forma clara, sobre as práticas de retenção e uso de dados da Ice Point. Mais importante, o sistema garante ativamente o direito à eliminação dos dados pessoais, previsto no Art. 18, inciso

VI, da LGPD (Brasil, 2018), por meio da rota `/exclusao-dados`. Nesta página, o cliente é instruído sobre o processo para solicitar a exclusão total de sua presença digital, sendo gerado um alerta por *e-mail* aos administradores do sistema, que possuem um Prazo de Nível de Serviço (SLA) de até 7 dias para concluir a operação.

Visando resguardar a integridade financeira e contábil da empresa, a exclusão dos dados opera sob um modelo híbrido de deleção física (*Hard Delete*) e anonimização (*Soft Delete*). No banco de dados relacional (PostgreSQL), os dados pessoais diretamente identificáveis do cliente (nome, CPF, endereço, telefone) são obliterados manualmente da tabela de usuários. Contudo, os registros associativos de encomendas (valores pagos, quantidades de picolés e carrinhos locados) têm a sua autoria desvinculada e substituída por identificadores genéricos. Essa abordagem sistêmica garante a total exclusão da identidade do indivíduo (atendendo à LGPD) sem corromper as métricas do *Dashboard* gerencial de vendas.

5.7 Resultados Globais Observados

Os testes iniciais demonstraram robustez e a imediata erradicação das principais deficiências operacionais da Ice Point. A transposição das lógicas de negócio empíricas para travas sistêmicas, como os limites rigorosos de encomenda, 80 a 2.000 picolés, e a validação volumétrica da capacidade da frota, 250 picolés por carrinho, tornou sistematicamente inviável o registro de pedidos deficitários ou a superlotação logística no momento de fechamento da compra.

A metodologia de validação adotada no projeto compreendeu múltiplas camadas de teste. Primeiramente, implementaram-se **testes unitários automatizados** com o *framework* Jest, ferramenta integrada nativamente ao ecossistema NestJS. Os testes foram desenvolvidos em dois módulos críticos do sistema, cobrindo ao todo onze cenários distintos de regras de negócio.

No `EncomendasService`, foram implementados três testes unitários cobrindo o método de cancelamento, a regra de negócio de maior sensibilidade logística do sistema. Os cenários validados automaticamente foram: (i) permissão de cancelamento quando a antecedência é superior a 2 horas; (ii) lançamento de exceção (`BadRequestException`) quando o prazo inferior a 2 horas é violado por um cliente comum; e (iii) permissão irrestrita de cancelamento para o perfil ADMIN, demonstrando a implementação de Controle de Acesso Baseado em Perfil (RBAC - *Role-Based Access Control*): o papel `ADMIN` possui autorização irrestrita sobre pedidos de qualquer cliente, podendo operar mesmo dentro do período de bloqueio operacional, enquanto o perfil `CLIENTE` está limitado a suas próprias encomendas e às regras temporais vigentes.

No `CartService`, foram implementados oito testes cobrindo o método `finalize`

`Order`, responsável pelo fechamento de encomendas. Este conjunto de testes cobre as seguintes garantias de negócio e segurança: (i) **inviolabilidade financeira**: o *backend* recalcula o valor total com base nos preços armazenados no banco de dados, ignorando qualquer valor enviado pelo *frontend*, eliminando vetores de adulteração de preço via manipulação de *payload* HTTP; (ii) **fronteira geográfica**: o sistema rejeita pedidos de entrega para cidades distintas de Uberaba, tornando a restrição territorial inviolável mesmo via ferramentas de requisição direta à API; (iii) **volumetria de frota**: o pedido é barrado quando a quantidade de picolés excede a capacidade total dos carrinhos selecionados, aplicando a regra de 250 picolés por unidade de frota; (iv) **pedido mínimo**: pedidos inferiores a 80 picolés são rejeitados no servidor, constituindo uma segunda camada de validação além da interface; (v) **barreira temporal D+1**: agendamentos para o dia atual ou datas passadas são bloqueados sistematicamente, protegendo a organização da linha de produção; (vi) **colisão logística de 12 horas**: carrinhos já alocados dentro da janela de higienização são identificados como indisponíveis e bloqueiam o fechamento do pedido; (vii) **lógica de desconto por forma de pagamento**: o desconto de 10% é aplicado exclusivamente para PIX e pagamento em dinheiro, sendo vetado para cartões de crédito; e (viii) **proteção legal de maioridade (RF10)**: o sistema impede a finalização de pedidos por clientes com idade inferior a 18 anos, resguardando juridicamente a empresa.

Todos os testes unitários isolam completamente a lógica de negócio de dependências externas (banco de dados, serviços de *e-mail* e APIs de terceiros), substituindo-as por *mocks* controlados via `jest.fn()` e `jest.spyOn()`, garantindo execução determinística, reprodutível e sem efeitos colaterais em qualquer ambiente (BECK, 2002).

Em complemento à cobertura automatizada dos testes unitários já descritos, conduziu-se uma bateria de *testes manuais exploratórios* e de *caixa-preta* realizada pelo próprio desenvolvedor, que consistiu em exercitar sistematicamente os fluxos de cadastro, *checkout* e cancelamento, incluindo a tentativa deliberada de burlar as regras de negócio, ou seja, a submissão de pedidos abaixo do mínimo, seleção de datas inválidas e endereços fora de Uberaba. Adicionalmente, os protótipos de interface foram submetidos a testes de usabilidade utilizando o método *Think Aloud* (Pensar Alto) na disciplina de Interação Humano-Computador (IHC), no qual outros graduandos verbalizavam suas ações ao inspecionar as telas com base nas heurísticas de Nielsen, identificando inconsistências de *layout* e oportunidades de melhoria de fluxo que foram corrigidas antes da codificação definitiva. Por fim, o sistema em funcionamento passou por um ciclo de *Testes de Aceitação do Usuário* (UAT - *User Acceptance Testing*), no qual clientes e colaboradores reais da Ice Point interagiram livremente com a plataforma em ambiente de produção (Fase Piloto), validando a aderência funcional e a usabilidade para o perfil real dos usuários finais.

No aspecto da organização temporal, a automação do agendamento restrito (D+1) associada à trava de cancelamento emergencial (bloqueio de cancelamento com menos de

2 horas de antecedência) sanou a problemática crônica de preparações de última hora e perdas materiais no ato da entrega. Secundariamente, a lógica de distanciamento por geolocalização e o preenchimento preditivo via API do ViaCEP forneceram transparência logística aos clientes e extirparam gargalos de endereço incorreto.

O conjunto dessas blindagens operacionais e de validações de formulário, atrelado à segurança jurídica conferida pela validação de maioridade legal e ao consentimento explícito dos Termos de Uso, não apenas reduziu drasticamente a sobrecarga cognitiva dos colaboradores na separação dos pedidos, mas estabeleceu um ecossistema digital maduro, escalável e juridicamente protegido para as operações comerciais da sorveteria.

6 Conclusão

O desenvolvimento do portal Ice Point Sorveteria representou um avanço significativo na modernização e profissionalização das operações logísticas e comerciais da empresa. Este capítulo finaliza o trabalho apresentando as considerações sobre o projeto desenvolvido, as limitações técnicas identificadas na versão atual e o delineamento estrutural para as futuras evoluções do *software*.

6.1 Considerações Finais

A transição de um processo puramente manual e descentralizado para uma arquitetura algorítmica de gestão logística configurou-se como o maior êxito tecnológico deste projeto. A implementação de rotinas como a alocação volumétrica de frota, precificação dinâmica e travas de temporalidade sanou gargalos estruturais históricos de atendimento da empresa, entregando um ecossistema digital *full-stack* funcional e escalável.

Atualmente, o sistema encontra-se em Fase Piloto (Operação Assistida). Esta etapa tem sido crucial para compor a Análise Qualitativa de Aceitação. Sob a ótica externa, o *feedback* dos clientes tem sido unânime quanto à usabilidade e estética da interface, elogiando a fluidez proporcionada pelo *frontend* em Vue.js. Identificou-se apenas um ruído pontual de fluxo: usuários que realizam encomendas no formato "Anônimo" tentaram realizar *login* para rastrear o pedido sem antes concluir o cadastro com o *e-mail* utilizado, evidenciando uma oportunidade futura para refinamentos de UX no fluxo de *Guest Checkout*.

No cenário operacional diário, os resultados da implantação foram expressivamente positivos. A centralização das encomendas no portal resultou em um histórico estruturado e de fácil acesso, extinguindo a fragmentação de informações que ocorria no modelo analógico. Funcionalidades automatizadas, como a geração instantânea de comprovantes e o envio de *e-mails* de confirmação, elevaram o nível de transparência e profissionalismo do atendimento. Além de atuar como uma ferramenta estritamente gerencial, o portal consolidou-se como uma vitrine digital eficaz, permitindo que os clientes explorem antecipadamente o catálogo de produtos e sabores, o que funciona tanto como facilitador de vendas quanto como instrumento de divulgação institucional.

Sob a ótica interna, a introdução do sistema revelou os desafios e as dinâmicas naturais da transformação digital no ambiente corporativo. Durante a fase de transição, observou-se uma adoção híbrida: enquanto usuários operacionais absorveram plenamente a rotina de gerenciar e alimentar o portal com os dados das encomendas, a captação

inicial com o cliente final ainda ocorre frequentemente via aplicativos de mensagens. Essa divisão orgânica de tarefas demonstra que o fluxo de trabalho da equipe está se adaptando gradativamente à nova ferramenta. A necessidade pontual de transcrição manual de dados externos para o portal ilustra uma barreira sociotécnica comum, comprovando que a adoção tecnológica é um processo contínuo de readequação de hábitos, o que motiva a concepção de futuras integrações conversacionais para otimizar essa etapa de captação.

6.2 Limitações do Sistema

Apesar da estabilidade operacional alcançada, a versão presente do sistema exibe limitações inerentes ao recorte de escopo definido para esta primeira iteração:

- **Gestão de Encomendas Lançadas:** O painel administrativo restringe-se à visualização e avanço de *status* das encomendas. O ecossistema atual não provê a funcionalidade de "Edição de Pedidos" retroativa (ex: adicionar mais um carrinho a um pedido já finalizado), obrigando o cancelamento sistêmico e um novo lançamento manual em casos de retificação por parte do administrador.
- **Ausência de Emissão Fiscal:** O ecossistema gera um espelho gerencial da transação para controle interno, não possuindo, contudo, valor contábil direto. Constata-se a lacuna de integrações com APIs especializadas na emissão automática de Notas Fiscais Eletrônicas (NF-e) perante a Secretaria da Fazenda.
- **Escopo Logístico Parametrizado (Catálogo Restrito):** A lógica de alocação de frota e métricas de distanciamento está fortemente acoplada ao domínio de "Carrinhos de Picolé". A iminente expansão corporativa para a venda de potes de sorvete de grande volume imporá variáveis de resfriamento e transporte logístico distintas, exigindo abstrações sistêmicas nas entidades de persistência do sistema.

6.3 Trabalhos Futuros

A arquitetura modular concebida sobre o *NestJS* conferiu ao projeto alta capacidade de extensão vertical. Nesse cenário, delineiam-se os seguintes avanços arquiteturais priorizados para os próximos ciclos iterativos:

- **Gateways de Pagamento (Mercado Pago):** Integração formal com *gateways* financeiros e plataformas de *checkout* consolidadas, possibilitando a leitura de *QR Codes* dinâmicos para pagamentos via PIX e validação de faturamento direto na plataforma, automatizando a conciliação bancária da Ice Point.

- **Automação Conversacional (Chatbot) — Protótipo em Desenvolvimento:** Um módulo de assistente conversacional integrado ao WhatsApp já se encontra em fase de prototipação ativa. A implementação inicial, operando em ambiente de testes com número próprio do desenvolvedor, demonstra a viabilidade técnica da integração. O *bot* consumirá a API RESTful estruturada no *backend* para processar encomendas diretamente pelo *chat* do WhatsApp, eliminando o gargalo de lançamentos manuais do administrador e permitindo que os pedidos recebidos via aplicativo de mensagens sejam registrados diretamente no banco de dados da plataforma.
- **Painel Operacional de Equipe (Tarefas):** Expansão dos privilégios de acesso do Funcionário para o gerenciamento colaborativo de tarefas operacionais diárias (como higienização de frotas e separação de fardos), impulsionando a interface além de um *e-commerce* para um módulo de Planejamento de Recursos Empresariais (ERP).
- **Múltiplas Categorias de Estoque (Potes de Sorvete):** Modelagem de uma nova esteira de processamento digital, parametrizando as lógicas de frete e acondicionamento para o fornecimento fracionado de potes de sorvete para clientes em atacado e varejo.
- **Refatoração de Pedidos e Integração Fiscal:** Implementação de transações lógicas que garantam o CRUD (*Create, Read, Update, Delete*) pleno dos pedidos confirmados, e a conexão assíncrona dessas movimentações a *Webhooks* fiscais geradores de Notas Fiscais Eletrônicas (NF-e).

A construção do portal corporativo validou empiricamente a hipótese de que a engenharia rigorosa de *software* suprime gargalos logísticos em PMEs. Amparada por regras de conformidade (LGPD), automações transacionais (CI/CD) e travas de segurança robustas em *frontend* e *backend*, este trabalho não entregou uma simples plataforma de demonstração, mas sim o motor digital capaz de suportar, modernizar e catalisar as operações comerciais de longo prazo da Ice Point Sorveteria.

Referências

BECK, K. **Test Driven Development: By Example**. Boston: Addison-Wesley, 2002. ISBN 978-0-321-14653-3. Citado 2 vezes nas páginas 19 e 45.

BORGES, B. de S. Monografia (Trabalho de Conclusão de Curso de Bacharelado em Ciência da Computação), **Desenvolvimento do Back-end e Implantação do Sistema Online de Distribuição de Disciplinas para a FACOM**. 2023. Orientador: Prof. Dr. Bruno Augusto Nassif Travençolo. Citado 2 vezes nas páginas 21 e 22.

Brasil. **Lei nº 13.709, de 14 de agosto de 2018: Lei Geral de Proteção de Dados Pessoais (LGPD)**. Brasília, DF: [s.n.], 2018. Diário Oficial da União. Disponível em: <https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm>. Acesso em: 29 jun. 2026. Citado na página 44.

DIAS, L. P.; MELLO, J. G. de. Monografia (Trabalho de Conclusão de Curso de Bacharelado em Ciência da Computação), **Desenvolvimento e Análise de Dados de uma Plataforma E-commerce de Veículos**. 2024. Orientadores: Prof. Elvio Gilberto da Silva e Prof. Roque Maitino Neto. Citado 2 vezes nas páginas 21 e 22.

FOWLER, M.; FOEMMEL, M. **Continuous Integration**. 2006. Disponível em: <https://martinfowler.com/articles/continuousIntegration.html>. Acesso em: 29 jun. 2026. Citado na página 20.

GROUP, P. G. D. **PostgreSQL Documentation**. 2024. Disponível em: <https://www.postgresql.org/>. Citado na página 18.

GUILHERME, K. L. Monografia (Trabalho de Conclusão de Curso de Bacharelado em Sistemas de Informação), **RETRO COMMERCE: Desenvolvimento de uma Plataforma E-commerce para o Mercado de Retro Games**. 2023. Orientador: Prof. Me. Daniel Ribeiro Trindade. Citado 2 vezes nas páginas 21 e 22.

HUMBLE, J.; FARLEY, D. **Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation**. Boston: Addison-Wesley, 2010. ISBN 978-0-321-60191-9. Citado 2 vezes nas páginas 20 e 42.

IBRAHIM, F. **DEVELOPMENT OF THE WEB APPLICATION "EVENT REGISTRATION PLATFORM"**. 2023. Bachelor's Thesis, National Research Tomsk State University (NR TSU), Research and Education Center "Higher IT School" (HITs). Main Educational program 09.03.04 - Software Engineering, Specialization "Software Engineering", Tomsk, Russia. Approved by Professor Dr. G.O.A. Zmeev. [Online; 2023]. Citado na página 13.

McKinsey & Company. Transformações digitais no brasil: insights sobre o nível de maturidade das empresas. **McKinsey Insights**, 2023. Disponível em: <https://www.mckinsey.com/br/our-insights/transformacoes-digitais-no-brasil>. Citado na página 13.

MICROSOFT. 98% das mpmes em transformação digital reconhecem o impacto positivo da tecnologia nos negócios. **Microsoft**, 2023. Disponível em: <https:

//news.microsoft.com/pt-br/98-das-mpmes-em-transformacao-digital-reconhecem-o-impacto-positivo-da-tecnologia-nos-negocios/>. Citado na página 13.

Mozilla Developer Network. **SPA (Single-page application)**. 2024. MDN Web Docs Glossary. Disponível em: <https://developer.mozilla.org/en-US/docs/Glossary/SPA>. Acesso em: 27 jul. 2026. Citado na página 16.

NESTJS. **NestJS Documentation**. 2024. Disponível em: <https://nestjs.com/>. Citado 2 vezes nas páginas 17 e 20.

NIELSEN, J. **Usability Engineering**. San Diego: Academic Press, 1994. Citado 3 vezes nas páginas 18, 19 e 24.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011. ISBN 978-85-7936-108-1. Citado na página 19.

SUPABASE. **Supabase Documentation**. 2026. Disponível em: <https://www.supabase.com/>. Citado na página 18.

TURBAN, E.; WHITESIDE, J.; KING, D.; OUTLAND, J. **Introduction to Electronic Commerce and Social Commerce**. [S.l.]: Springer, 2017. Citado na página 16.

VUE.JS. **Vue.js Documentation**. 2024. Disponível em: <https://vuejs.org/>. Citado na página 17.