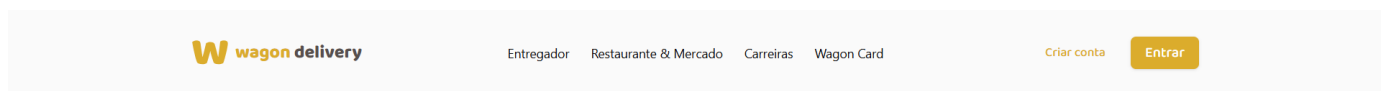




# PONTIFÍCIA UNIVERSIDADE CATÓLICA DE CAMPINAS (PUC-CAMPINAS)

<b>Curso:</b> Engenharia de Computação		<b>Disciplina:</b> PI - Desenvolvimento WEB (218040)		
<b>Período:</b> 01	<b>Turma:</b> 01	<b>Sala:</b> H15   110	<b>Turno:</b> Integral	<b>Data:</b> 11/11/2024
<b>Integrante(s):</b> Pedro Henrique Galembeck (24005794), João Henrique Moura dos Santos (24006456), Arthur Camilotti de Souza (24002213) e Murilo Henrique de Arruda Prada (24014995).				
<b>Orientador(a):</b> José Matias Lemes Filho				

## WAGON DELIVERY | PROJETO INTEGRADOR



### Peça o que quiser com Wagon

Entregamos tudo o que você deseja diretamente na porta da sua casa! De cafés especiais e selecionados a deliciosos pratos dos melhores restaurantes parceiros, estamos prontos para lhe atender a qualquer hora do dia.

Ver restaurantes próximos

Tradicional

Americano

Cremoso

Gelado

Com leite

Alcoólico

Especial

Brasileira

Árabe

Doces & Bolos

Lanches

Saudável

Japonesa

Carnes

Pizza

Italiana

Vegetariana

[Coffee Wagon | Explorar →](#)[Food Wagon | Explorar →](#)

# SUMÁRIO

<b>1. Introdução .....</b>	<b>3</b>
• Visão geral do projeto .....	
• Objetivos do projeto .....	
• Público-alvo e uso pretendido .....	
<b>2. Tecnologias de desenvolvimento .....</b>	
<b>3. Arquitetura do projeto .....</b>	
• Estrutura de pastas & organização do código .....	
• Componentes principais/reutilizáveis .....	
• Interface de usuário .....	
<b>4. Principais funcionalidades .....</b>	
• Cadastro & autenticação de usuários .....	
• Listagem de produtos (cafés & pratos) .....	
<b>5. Detalhes de implementação .....</b>	
• Variáveis de ambiente .....	
<b>6. Desafios e soluções .....</b>	
• Principais desafios técnicos encontrados .....	
• Alternativas consideradas .....	
• Soluções implementadas .....	
<b>7. Melhorias e expansões futuras .....</b>	
• Funcionalidades planejadas .....	
• Integrações adicionais .....	
<b>8. Conclusão .....</b>	

# CAPÍTULO 1

## INTRODUÇÃO

### 1.1 Visão geral do projeto:

O projeto **Wagon Delivery** foi criado com o propósito de oferecer uma solução digital inovadora para o setor de logística e entrega, neste caso, de cafés e pratos de restaurantes. Em um mercado cada vez mais competitivo, a eficiência no gerenciamento de entregas se tornou essencial para empresas que buscam atender às crescentes expectativas dos clientes em relação à rapidez e confiabilidade. Nesse contexto, o **Wagon Delivery** foi desenvolvido para otimizar e automatizar processos logísticos, permitindo um controle simplificado de pedidos e entregas.

A plataforma utiliza tecnologias modernas como **React**, **Next.js** e **TailwindCSS** para oferecer uma **interface** de usuário **intuitiva, responsiva** e de **alto desempenho**. O **React** proporciona uma experiência de usuário dinâmica, enquanto o **Next.js** garante SEO aprimorado e carregamento rápido de páginas através da renderização no servidor (Server-Side Rendering). O **TailwindCSS**, por sua vez, foi empregado para garantir uma estilização elegante e flexível, permitindo que o design seja facilmente adaptado para diferentes dispositivos e tamanhos de tela. Posteriormente, neste relatório, as tecnologias utilizadas no desenvolvimento serão tratadas e aprofundadas.

Para gerenciar a autenticação e segurança dos usuários, o projeto incorpora a **Clerk**, uma solução de autenticação moderna que facilita o registro, login e gerenciamento de sessões que facilita o registro, login e gerenciamento de sessões de maneira segura. O **Shadcn-UI** foi utilizado como biblioteca de componentes visuais, permitindo uma personalização mais refinada e uma identidade visual coesa ao longo da aplicação.

Entre as principais **funcionalidades** do **Wagon Delivery** estão: o cadastro e autenticação de usuários (necessário para finalização da compra na página de carrinho), paginação dinâmica para as páginas de café e comida, carrinho de compras global, entre outras. Além disso, a plataforma foi projetada para ser escalável, permitindo a inclusão de novas

funcionalidades e a integração com outros sistemas no futuro, como APIs de rastreamento e otimização de rotas, por exemplo.

Este relatório, por sua vez, tem como objetivo principal **documentar** o **processo** de **desenvolvimento** do **Wagon Delivery**, detalhando desde a concepção inicial até a implementação de cada funcionalidade, os desafios enfrentados pela equipe durante o desenvolvimento e as soluções adotadas. Também serão exploradas as decisões tecnológicas e as melhores práticas empregadas para garantir uma solução robusta, de fácil manutenção e preparada para a expansão. Ao final, discutiremos as perspectivas futuras para o projeto, incluindo possíveis melhorias e integrações adicionais que podem vir a ser adicionadas para ampliar ainda mais o valor da plataforma para o setor de entregas e logística.

## 1.2 Objetivos do projeto:

O projeto **Wagon Delivery** foi desenvolvido para fornecer uma **interface** de usuário **intuitiva** e **eficiente**, destinada à navegação e realização de pedidos em uma plataforma de e-commerce. A solução visa simplificar a experiência de compra, com foco nas etapas de seleção de produtos, gerenciamento de carrinho e finalização de pedidos. Os principais objetivos do projeto são:

**1. Criar uma experiência de compra intuitiva e atraente:** desenvolver uma interface de fácil utilização e visualmente atraente, que permita ao usuário navegar pela listagem de produtos, selecionar itens e adicionar ao carrinho de forma prática e intuitiva.

**2. Facilitar o processo de finalização de pedidos:** oferecer um fluxo de finalização de compras simplificado e direto, com informações claras e organizadas, proporcionando ao usuário uma experiência de checkout rápida e sem complicações.

**3. Implementar uma interface responsiva e flexível:** garantir que a interface se adapte a diferentes dispositivos e tamanhos de tela, utilizando o TailwindCSS para criar uma experiência consistente e responsiva, independente de o usuário estar utilizando um computador, tablet ou smartphone.

**4. Utilizar autenticação segura para experiência personalizada:** integração com o Clerk, a qual permite uma autenticação personalizada ao usuário, de modo

que cada cliente possa gerenciar sua conta e, futuramente, visualizar seus pedidos com segurança, mantendo suas informações protegidas.

**5. Garantia de escalabilidade e desempenho:** desenvolver a aplicação utilizando React e Next.js, garantindo que o projeto seja escalável e que a performance seja mantida mesmo com um aumento no número de produtos e/ou usuários.

**6. Personalizar e melhorar a interface com componentes consistentes:** utilizar a biblioteca Shadcn-UI e a criação de componentes reutilizáveis do React facilita a manutenção e a evolução da interface ao longo do tempo.

**7. Aumentar a satisfação do usuário:** melhorar a experiência geral do usuário ao oferecer uma interface que facilita o processo de compra e checkout, aumentando as chances de conversão e fidelização dos clientes.

### 1.3 Público-alvo e uso pretendido:

O projeto **Wagon Delivery** foi idealizado para atender empresas e pequenos negócios de e-commerce que buscam uma solução prática e moderna para melhorar a experiência de compra de seus clientes. A plataforma é voltada para estabelecimentos que desejam oferecer uma interface de usuário intuitiva e eficiente para seus clientes, permitindo uma navegação facilitada, seleção de produtos, gerenciamento de carrinho e um processo de finalização de compra otimizado.

- **Público-alvo:**

- **Empresas de e-commerce:** lojas virtuais de pequeno e médio porte que necessitam de uma plataforma confiável para gerenciar a experiência de compra de seus clientes.

- **Pequenos negócios:** empreendedores e comerciantes que buscam digitalizar suas operações e oferecer aos clientes uma experiência de compra agradável e ágil.

- **Usuários finais de e-commerce:** consumidores que utilizam plataformas de e-commerce e valorizam uma interface de fácil navegação e funcionalidades úteis, como filtros por tags para encontrar produtos específicos de maneira rápida e eficiente.

- **Uso pretendido:**

O **Wagon Delivery** foi projetado para ser uma interface visual que facilita o processo de compra do usuário, desde a busca por produtos até a finalização do pedido. A plataforma permite:

- **Navegação e seleção de produtos:** com funcionalidades de listagem de produtos, o usuário pode localizar facilmente o que procura e explorar opções de maneira eficiente.

- **Gerenciamento de carrinho de compras:** o sistema permite que o usuário adicione itens ao carrinho, visualize os produtos selecionados e faça ajustes antes de finalizar a compra.

- **Checkout simplificado:** um fluxo de checkout otimizado para garantir uma experiência de compra rápida, minimizando o abandono de carrinho e maximizando a conversão.

A solução visa tornar o processo de compra mais agradável e simples para o usuário, ao mesmo que ajuda os estabelecimentos a aumentar a satisfação dos seus clientes, proporcionando uma plataforma que alia praticidade e modernidade.

## CAPÍTULO 2

# TECNOLOGIAS DE DESENVOLVIMENTO

O projeto **Wagon Delivery** foi desenvolvido utilizando uma combinação de tecnologias modernas que visam oferecer uma interface de usuário de alto desempenho, escalável e segura. A seguir, são detalhadas as principais tecnologias empregadas no desenvolvimento:

### 1. **React** (<https://react.dev/>)

A biblioteca **React** foi utilizada para a construção de interfaces de usuário dinâmicas e interativas. Com o **React**, a aplicação é organizada em componentes reutilizáveis, facilitando a manutenção e escalabilidade do código.

### 2. **Next.js** (<https://nextjs.org/>)

Utilizado como framework principal, o **Next.js** fornece funcionalidades avançadas para o desenvolvimento de aplicações web com **React**, como renderização no servidor (SSR) e geração de sites estáticos (SSG). Isso garante carregamento rápido de páginas e melhora o SEO da plataforma, proporcionando uma experiência de usuário mais eficiente e responsiva.

### 3. **TypeScript** (<https://www.typescriptlang.org/>)

A escolha do **TypeScript** adiciona tipagem estática ao JavaScript, ajudando a evitar erros comuns durante o desenvolvimento e tornando o código mais legível e fácil de manter. TypeScript também melhora a produtividade ao oferecer sugestões de código e identificar problemas antes da execução.

### 4. **TailwindCSS** (<https://tailwindcss.com/>)

Para estilização, o **TailwindCSS** foi empregado, proporcionando uma maneira rápida e flexível de criar interfaces responsivas e esteticamente agradáveis. O TailwindCSS permite a personalização do design diretamente nos componentes, mantendo a consistência visual em diferentes dispositivos e tamanhos de tela.

## 5. Context API (<https://legacy.reactjs.org/docs/context.html>)

Utilizado para gerenciar o estado global da aplicação, o **Context API** facilita a passagem de dados entre componentes sem a necessidade de "prop drilling". Isso é particularmente útil para informações compartilhadas como dados do carrinho de compras e preferências do usuário.

## 6. Local Storage (<https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>)

O **Local Storage** é utilizado para armazenar temporariamente dados do usuário, como o conteúdo do carrinho de compras. Dessa forma, o usuário pode continuar de onde parou caso a página seja atualizada ou se ele retornar posteriormente, oferecendo uma experiência mais fluida e conveniente.

## 7. Clerk Authenticator (<https://clerk.com/>)

O **Clerk** é a solução escolhida para autenticação e controle de acesso. Ele facilita o login e o registro de usuários, fornecendo uma camada adicional de segurança e permitindo que o sistema personalize a experiência de cada cliente de maneira segura e confiável.

## 8. React Hook Form (<https://react-hook-form.com/>)

Para o gerenciamento de formulários, o **React Hook Form** foi implementado, tornando o processo de criação e controle de formulários mais eficiente. Essa biblioteca é leve e otimizada, contribuindo para o desempenho da aplicação.

## 9. Validações com Zod (<https://zod.dev/>)

O **Zod** foi utilizado em conjunto com o React Hook Form para validação de formulários. Com ele, é possível definir esquemas de validação para garantir que os dados inseridos pelos usuários sejam válidos e estejam no formato adequado para as informações de entrega, melhorando a segurança e a qualidade dos dados recebidos.

Essas tecnologias foram escolhidas para proporcionar uma experiência de uso intuitiva, segura e otimizada. A combinação de **Next.js**, **React**, **TailwindCSS** permite um desenvolvimento ágil, enquanto **TypeScript** e



**Context API** garantem a escalabilidade e organização do código. A autenticação com **Clerk** e as validações de formulário com **Zod** garantem que a aplicação atenda aos requisitos de segurança e qualidade, entregando uma solução robusta e preparada para crescimento futuro.

## CAPÍTULO 3

# ARQUITETURA DO PROJETO

A arquitetura do **Wagon Delivery** foi cuidadosamente planejada para proporcionar uma estrutura de código organizada, modular e fácil de manter. Abaixo, detalhamos os principais aspectos da organização do projeto, incluindo a estrutura de pastas, componentes reutilizáveis e a interface de usuário.

### 3.1. Estrutura de pastas & organização do código:

O projeto segue uma estrutura de pastas organizada para facilitar o desenvolvimento e manutenção. A estrutura é dividida em duas pastas principais: **“public”** e **“src”**.



**| - public/:** contém todos os assets públicos, como imagens e ícones.

**| - assets/:** dividido em subpastas “coffee” e “food”, onde são armazenadas imagens e ícones específicos para cada categoria/página.

**| - icons/:** pasta para ícones globais que podem ser usados em várias partes da interface.

**| - src/:** é o diretório principal do código-fonte, contendo a “core” do projeto.

**| - app/:** armazena o conteúdo de páginas, como componentes principais de navegação e layout, além das páginas/interfaces de autenticação (“sign-in” & “sign-up”) do Clerk.

**| - components/:** contém os componentes reutilizáveis da interface, como botões, modais, cards de produtos, entre outros, que são utilizados em diferentes partes da aplicação.

**| - config/:** arquivo de configurações gerais que podem ser necessárias em múltiplas partes do código, facilitando a personalização e ajustes da aplicação.

**| - constants/:** armazena constantes reutilizáveis, como strings e configurações que não mudam ao longo da execução da aplicação.

**| - contexts/:** contém os arquivos de contexto, utilizando o Context API para gerenciar o estado global da aplicação.

**| - hooks/:** armazena hooks customizados, encapsulando lógicas que podem ser reutilizadas por vários componentes.

**| - lib/:** configurações específicas da biblioteca Shadcn-UI, utilizada para estilização e componentes de interface.

**| - styles/:** armazena estilos globais da aplicação, geralmente arquivos CSS ou configurações de TailwindCSS.

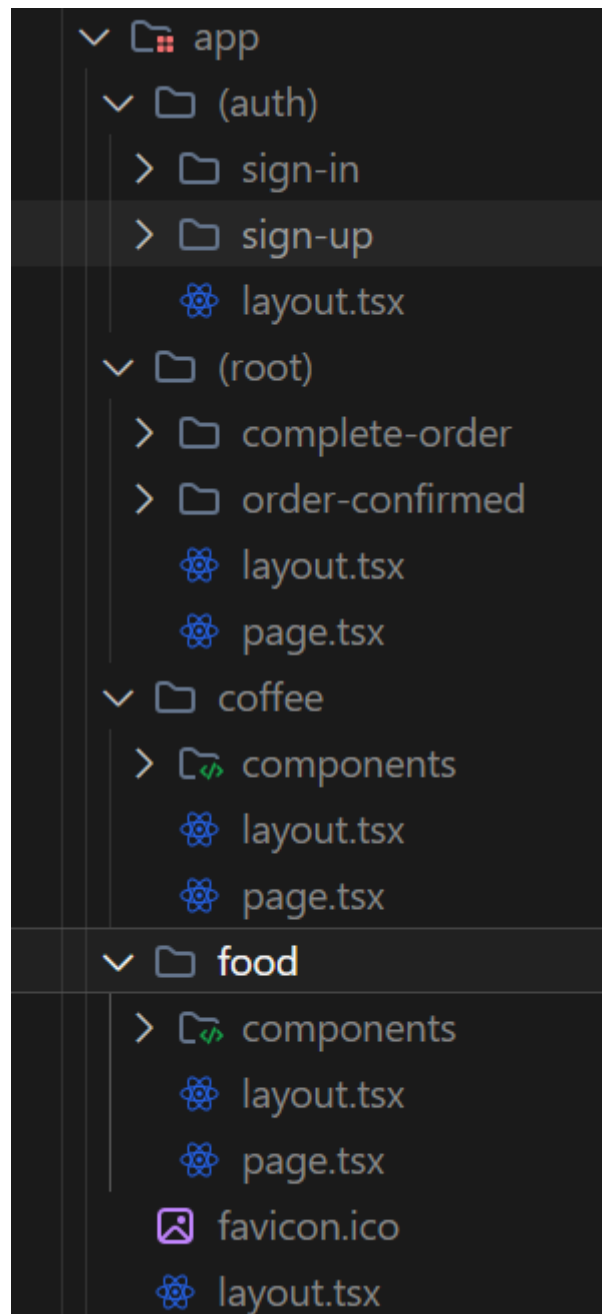
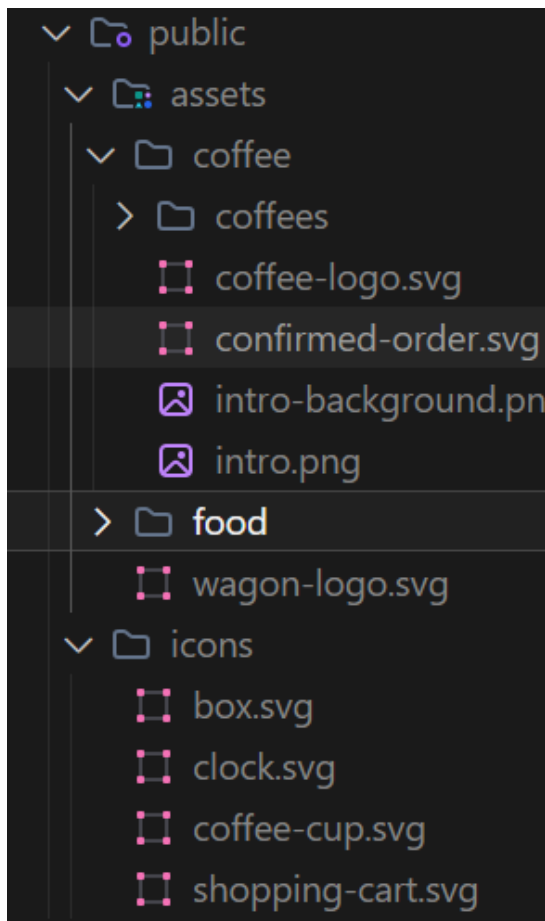
**| - types/:** contém as interfaces TypeScript que são usadas em diferentes componentes, promovendo tipagem consistente e organização.

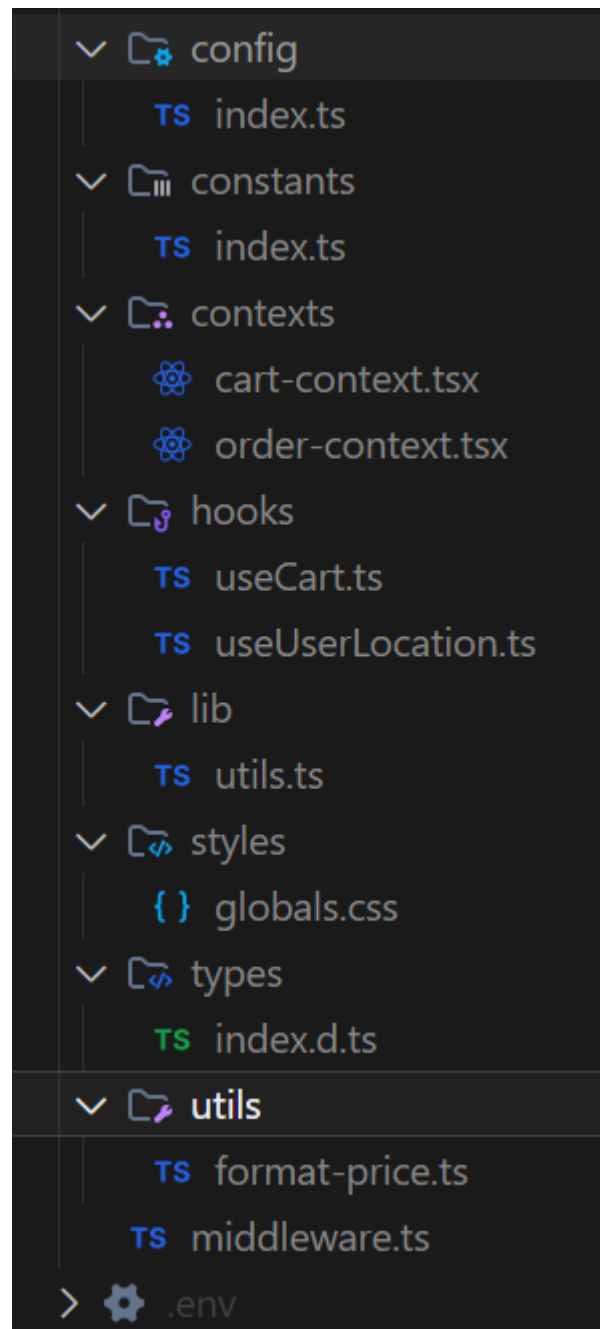
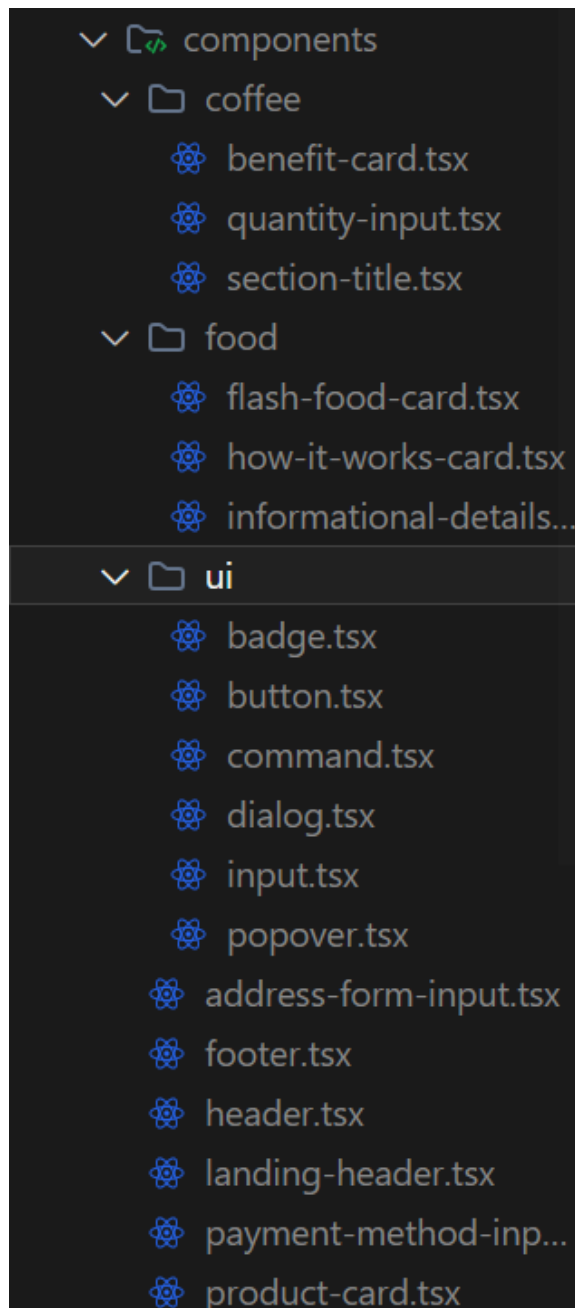
**| - utils/:** armazena utilitários e helpers, como funções auxiliares que facilitam operações comuns dentro da aplicação.

|- **middleware.ts**: arquivo de configuração específico para autenticação com Clerk, garantindo que o fluxo de login e autenticação funcione de forma integrada.

|- **.env**: arquivo de variáveis de ambiente, utilizado para armazenar chaves de API e outras variáveis sensíveis que não devem ser *hardcoded* no código.

|- **README.md**: documentação do projeto, com instruções de instalação, configuração e uso.





### 3.2 Componentes principais/reutilizáveis:

O diretório **components/** contém os componentes principais e reutilizáveis do projeto. Estes componentes foram desenvolvidos para serem independentes e encapsulados, o que permite seu uso em diversas partes da aplicação. Alguns dos componentes mais importantes incluem:

- **Cards de produtos:** exibem as informações principais de cada produto, como imagem, preço e descrição, permitindo que o usuário adicione o item ao carrinho diretamente do card.
- **Botões customizados:** estilizados com TailwindCSS e configurados para suportar diferentes ações e estilos (como primário, secundário, etc).
- **Modais & diálogos:** utilizados para exibir mensagens e opções de confirmação ao usuário, melhorando a interação com a aplicação.
- **Componentes de formulário:** integrados ao React Hook Form para facilitar a coleta de dados, e com validações realizadas pelo Zod.
- **Navegação & header:** componentes de navegação e cabeçalho de aplicação, garantindo que o usuário tenha uma experiência de navegação consistente e acessível.

A modularidade dos componentes é fundamental para a manutenção e escalabilidade da aplicação, além de promover uma experiência de usuário coesa e consistente.

### 3.3 Interface do usuário:

A interface do usuário do **Wagon Delivery** foi projetada com foco em simplicidade e usabilidade. Utilizando o TailwindCSS para estilização, a interface é responsiva e adaptável a diferentes dispositivos. O uso de Shadcn-UI adiciona componentes visuais consistentes e customizáveis, mantendo uma estética agradável e profissional.

Principais características da interface do usuário:

- **Layout responsivo:** adaptado para funcionar bem em dispositivos móveis, tablets e computadores.

- **Categorias e listagem de produtos:** a interface oferece filtros por tags/categorias para ajudar o usuário a encontrar produtos de forma rápida e eficiente, além de uma listagem organizada de produtos.

- **Carrinho de compras:** facilita a adição e remoção de produtos, permitindo ao usuário gerenciar seu carrinho de forma intuitiva.

- **Checkout simplificado:** um fluxo de checkout otimizado para proporcionar uma finalização de compra rápida e eficiente.

- **Autenticação com Clerk:** oferece uma experiência de login e registro segura, integrando a autenticação de forma transparente para o usuário.

Essa combinação de tecnologias e boas práticas de design garante que a interface do **Wagon Delivery** seja agradável, intuitiva e eficaz, focando na experiência do usuário e facilitando a navegação e compra na plataforma.

## CAPÍTULO 4

# PRINCIPAIS FUNCIONALIDADES

### 4.1 Cadastro & autenticação de usuários:

A funcionalidade de cadastro e autenticação de usuários é essencial para personalizar a experiência de compra e garantir a segurança dos dados do cliente. A plataforma utilizar o Clerk para gerenciar o processo de autenticação de maneira eficiente e segura, necessária para a conclusão da compra, facilitando o acesso dos usuários aos recursos da plataforma.

Wagon delivery

Entregador Restaurante & Mercado Carreiras Wagon Card

Criar conta Entrar

### Sign in to Wagon Delivery

Welcome back! Please sign in to continue

GitHub Google

or

Email address

Continue ▶

Don't have an account? Sign up

Secured by clerk

Development mode



## Create your account

Welcome! Please fill in the details to get started.



GitHub



Google

or

First name

Optional

Last name

Optional


Email address

Password



Continue ▶

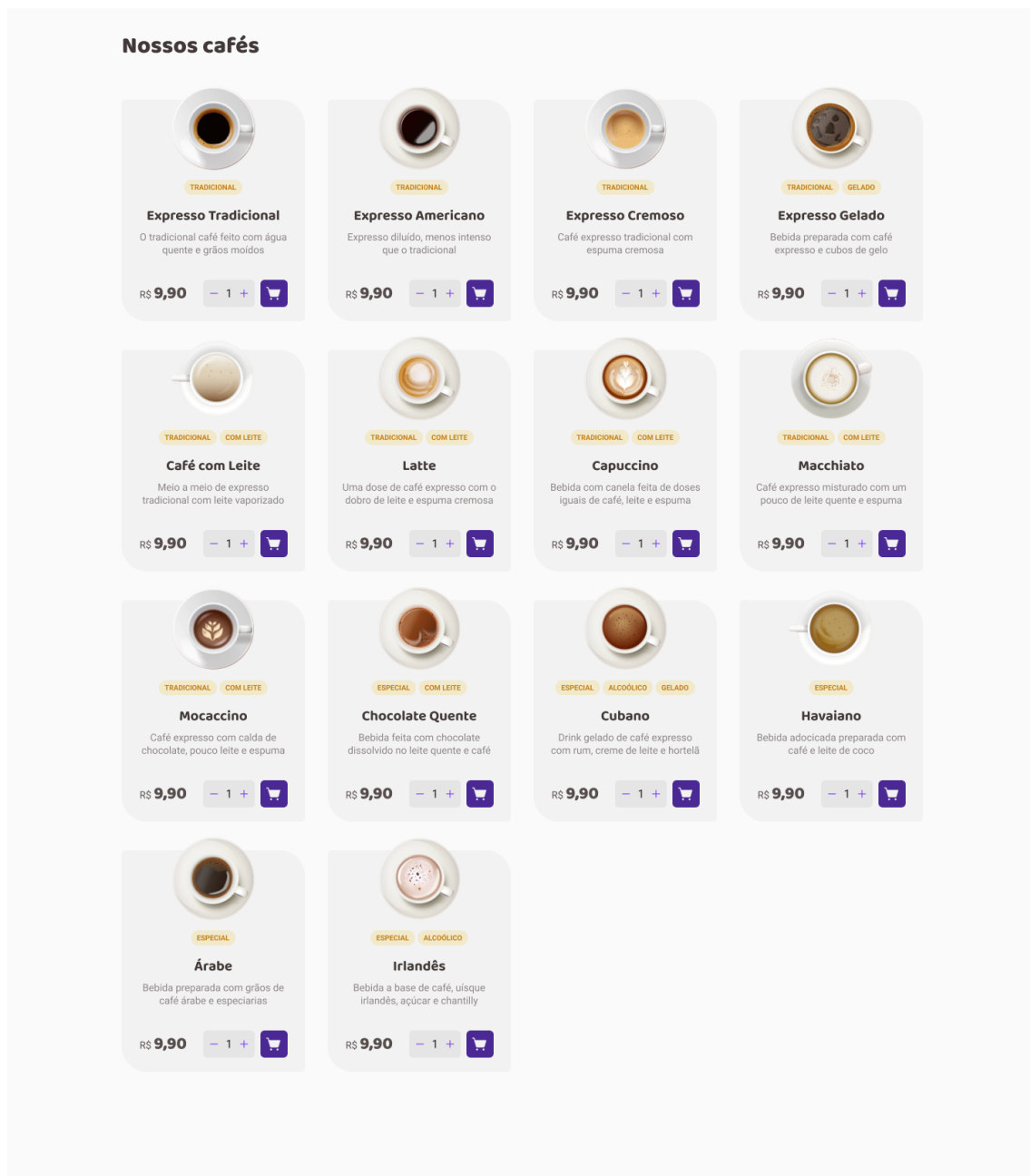
Already have an account? [Sign in](#)

Secured by  clerk

































Development mode

## 4.2 Listagem de produtos:

A Listagem de Produtos é uma funcionalidade central da plataforma **Wagon Delivery** e foi projetada para proporcionar uma navegação fácil e uma visualização organizada dos produtos disponíveis. Essa seção permite que o usuário explore o catálogo completo, encontre produtos por categorias e aplique filtros para refinar sua busca.



## Nossos produtos

 <p>HAMBURGUER</p> <p><b>Cheese Burger</b></p> <p>Pão macio, carne grelhada e queijo derretido.</p> <p>R\$ <b>12,90</b> - 1 + </p>	 <p>HAMBURGUER</p> <p><b>X-Bacon</b></p> <p>Pão macio, carne suculenta, queijo derretido e muito bacon crocante.</p> <p>R\$ <b>14,90</b> - 1 + </p>	 <p>HAMBURGUER</p> <p><b>Chicken Burger</b></p> <p>Peito de frango grelhado, com molho especial, alface e pão macio.</p> <p>R\$ <b>13,90</b> - 1 + </p>	 <p>HAMBURGUER</p> <p><b>Big Cheese</b></p> <p>Três suculentas carnes, muito queijo derretido e um molho especial.</p> <p>R\$ <b>16,90</b> - 1 + </p>
 <p>PIZZA</p> <p><b>Pizza de Mussarela</b></p> <p>Massa fina, molho de tomate e queijo derretido.</p> <p>R\$ <b>63,90</b> - 1 + </p>	 <p>PIZZA</p> <p><b>Pizza de Frango</b></p> <p>Massa crocante, molho de tomate e frango desfiado.</p> <p>R\$ <b>69,90</b> - 1 + </p>	 <p>PIZZA</p> <p><b>Pizza de Pepperoni</b></p> <p>Base crocante e generosas fatias de pepperoni levemente picantes, douradas no forno.</p> <p>R\$ <b>69,90</b> - 1 + </p>	 <p>PIZZA</p> <p><b>Pizza de Chocolate</b></p> <p>Uma delícia doce e irresistível!</p> <p>R\$ <b>74,90</b> - 1 + </p>
 <p>JAPA</p> <p><b>Combo de Sushi</b></p> <p>Combo com os melhores tipos de sushi!</p> <p>R\$ <b>47,90</b> - 1 + </p>	 <p>JAPA</p> <p><b>Temaki</b></p> <p>Salmão irresistível envolto por algas secas!</p> <p>R\$ <b>15,90</b> - 1 + </p>	 <p>JAPA</p> <p><b>Shimeji</b></p> <p>O prato fungi mais adorado da cultura japonesa!</p> <p>R\$ <b>21,90</b> - 1 + </p>	 <p>JAPA</p> <p><b>Porção de Hot Roll</b></p> <p>Uma porção de hot rolls com cream-cheese!</p> <p>R\$ <b>15,90</b> - 1 + </p>
 <p>VEGANO</p> <p><b>Hambúrguer Vegano</b></p> <p>Que tal uma opção de hambúrguer vegano?</p> <p>R\$ <b>17,90</b> - 1 + </p>	 <p>VEGANO</p> <p><b>Sushi Vegano</b></p> <p>Uma opção de sushi vegano para você!</p> <p>R\$ <b>24,90</b> - 1 + </p>	 <p>VEGANO</p> <p><b>Pizza Vegana</b></p> <p>Uma pizza vegana deliciosa!</p> <p>R\$ <b>29,90</b> - 1 + </p>	 <p>VEGANO</p> <p><b>Bolo Vegano</b></p> <p>Um bolo vegano de baunilha com cobertura.</p> <p>R\$ <b>37,90</b> - 1 + </p>

## CAPÍTULO 5

### DETALHES DE IMPLEMENTAÇÃO

A configuração das variáveis de ambiente é um passo essencial para garantir que as chaves de API e outras informações sensíveis sejam gerenciadas de maneira segura e acessível apenas ao ambiente da aplicação. No **Wagon Delivery**, o arquivo “.env” é utilizado para armazenar as chaves do **Clerk**, que é o serviço de autenticação responsável pela segurança e autenticação dos usuários na plataforma.

## 5.1 Variáveis de ambiente:

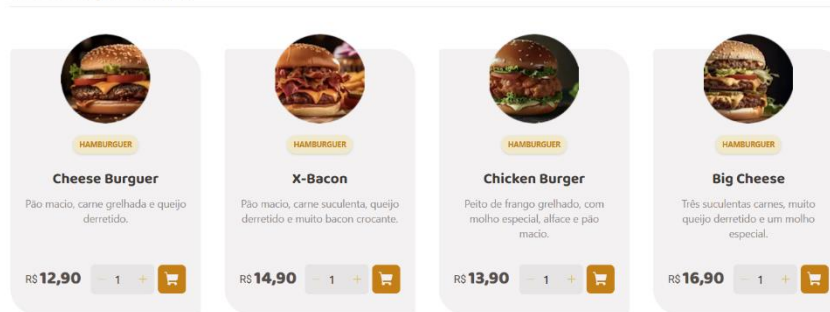
Para configurar as variáveis de ambiente necessárias, o projeto utiliza um arquivo “.env”, no qual as chaves de API do Clerk devem ser definidas. Abaixo, detalhamos duas maneiras de configurar essas chaves para habilitar a autenticação de usuários no projeto.

### • Opção 1: Utilizar chaves fornecidas no projeto

Uma opção é utilizar as chaves de autenticação para o Clerk iguais as que foram usadas durante o desenvolvimento do projeto pela equipe, podendo configurá-las diretamente no arquivo “.env”, como demonstrado abaixo:

```
NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=pk_test_YWNILWdyb3VwZXItODAuY2xlcmsuY
```

#### Nossos produtos



```
WNjb3VudHMuZGV2JA
```

```
CLERK_SECRET_KEY=sk_test_m6Otur9FWDDDGnZjS8cVoUQO32RI077vqBAn9QcJb9
```

Essas duas chaves são essenciais para o funcionamento da autenticação:

- NEXT\_PUBLIC\_CLERK\_PUBLISHABLE\_KEY: A chave pública necessária para identificar o projeto no lado do cliente.
- CLERK\_SECRET\_KEY: A chave secreta que permite o gerenciamento seguro das autenticações e acessos.

Após salvar as chaves no arquivo .env, a aplicação conseguirá acessar esses valores automaticamente ao ser executada, garantindo que o **Clerk** funcione corretamente.

### • Opção 2: Criar uma conta no Clerk e gerar as chaves

Se você não possui uma conta no Clerk ou precisa gerar novas chaves, siga os passos abaixo para configurá-las:

1. Acesse o site do Clerk (<https://clerk.com/>) e crie uma conta (ou faça login, caso já possuir uma).
2. No painel de administração, crie um novo projeto para o Wagon Delivery.
3. No projeto, acesse as configurações de API para obter a Chave Pública (Public Key) e a Chave Secreta (Secret Key).
4. Copie as chaves geradas e cole-as no arquivo ".env" do projeto com os nomes "NEXT\_PUBLIC\_CLERK\_PUBLISHABLE\_KEY" e "CLERK\_SECRET\_KEY", conforme o exemplo acima.
5. Após configurar o arquivo ".env", salve-o e reinicie a aplicação para que as alterações tenham efeito.

Esse procedimento garante que as chaves estejam configuradas corretamente e que o **Clerk** esteja habilitado para autenticação de usuários no **Wagon Delivery**. Certifique-se de não compartilhar o arquivo .env em repositórios públicos, pois ele contém informações sensíveis que podem comprometer a segurança da aplicação.

## CAPÍTULO 6

### DESAFIOS E SOLUÇÕES

Durante o desenvolvimento do **Wagon Delivery**, foram encontrados alguns desafios técnicos que exigiram pesquisa, análise de alternativas e soluções estratégicas para garantir uma experiência de usuário fluida e uma interface visualmente consistente. Abaixo, estão descritos os principais desafios enfrentados e as soluções aplicadas.

## **6.1 Principais desafios técnicos encontrados:**

### **I) Globalização do sistema de carrinho de compras:**

Um dos maiores desafios foi criar um sistema de carrinho de compras global, isto é, acessível a partir de qualquer página da aplicação, sem a necessidade de recarregar ou perder informações ao navegar entre páginas, além da possibilidade de adicionar itens das diferentes páginas em apenas uma compra. Garantir que o carrinho mantivesse seu estado de forma persistente foi fundamental para oferecer uma experiência contínua ao usuário.

### **II) Estilização responsiva:**

Outro desafio foi garantir que a interface da aplicação fosse responsiva, adaptando-se automaticamente a diferentes tamanhos de tela, desde dispositivos móveis até desktops. A estilização responsiva é essencial para assegurar que a interface seja intuitiva e atraente em todos os dispositivos, o que é um requisito básico para uma plataforma de e-commerce moderna.

## **6.2 Alternativas consideradas:**

### **I) Implementar um sistema de carrinho para cada página:**

Uma abordagem inicial considerada foi implementar um sistema de carrinho independente em cada página da aplicação, mantendo o estado do carrinho isolado em cada rota. Embora essa opção oferecesse menos complexidade de implementação, apresentava desvantagens significativas: o usuário teria que recarregar o carrinho a cada troca de página (utilização do “*local storage*”), o que comprometeria a experiência e a praticidade da navegação.

## **II) Frameworks CSS tradicionais para estilização:**

Uma alternativa considerada foi o uso de frameworks CSS tradicionais (como styled-components) para a estilização da interface. Embora esses frameworks ofereçam soluções pré-construídas, elas podem dificultar personalizações avançadas e não são sempre ideais para layouts altamente customizáveis e responsivos.

### **6.3 Soluções implementadas:**

#### **I) Unificação do sistema de carrinho de compras:**

Para resolver o problema do carrinho de compras global, foi implementado um sistema centralizado de estado utilizando o Context API e o "local storage". Isso permitiu que o carrinho fosse acessível e atualizado em tempo real a partir de qualquer página da aplicação, garantindo uma experiência de compra ininterrupta. Além disso, utilizou-se o "local storage" para persistir o conteúdo do carrinho, permitindo que o usuário mantenha seus itens salvos por sessão, mesmo ao recarregar a página ou retornar posteriormente.

#### **II) Adoção do TailwindCSS para estilização responsiva:**

A escolha do TailwindCSS foi fundamental para a implementação de uma interface responsiva e estilizada. Com o TailwindCSS, foi possível criar uma estilização adaptável e flexível, aplicando classes utilitárias diretamente nos componentes, o que facilita ajustes para diferentes tamanhos de tela sem a necessidade de CSS adicional. Isso simplificou o desenvolvimento de uma interface consistente, garantindo que todos os elementos da interface fossem responsivos e proporcionando uma experiência visual uniforme entre dispositivos.

## **CAPÍTULO 7**

### **MELHORIAS E EXPANSÕES FUTURAS**



Para garantir que o **Wagon Delivery** continue a oferecer uma experiência aprimorada e atenda às crescentes demandas dos usuários, novas funcionalidades e integrações estão planejadas para as próximas versões. Essas melhorias visam expandir as possibilidades da plataforma, tornando-a mais completa e funcional.

## **7.1 Funcionalidades planejadas:**

**I) Opções avançadas de filtragem de produtos:** atualmente, o sistema permite a filtragem de produtos por tags básicas, mas a ideia é adicionar um sistema de filtragem avançada que permita ao usuário selecionar múltiplos filtros simultaneamente (como categorias, faixa de preço, preferências alimentares e disponibilidade), possibilitando uma busca mais precisa, facilitando a localização de produtos específicos.

**II) Sistema de avaliações e comentários dos produtos:** uma funcionalidade planejada é permitir que os usuários avaliem os produtos e deixem comentários sobre suas experiências, ajudando novos clientes a tomar decisões de compra e promoverá um senso de comunidade entre os usuários, além de fornecer feedback direto aos fornecedores.

**III) Histórico de compras e recomendações personalizadas:** pretende-se implementar um sistema que registre o histórico de compras dos usuários e sugira produtos com base em compras anteriores e preferências pessoais, agregando valor ao cliente e incentivando novas compras através de recomendações customizadas.

**IV) Sistema de acompanhamento de entrega:** semelhante ao iFood, pretende-se implementar um sistema de acompanhamento do status da entrega em tempo real, com atualizações da localização do pedido juntamente ao entregador.

## **7.2 Integrações adicionais:**

**I) Integração com gateways de pagamento avançados:** para oferecer mais opções de pagamento, está prevista a integração com gateways adicionais, como Stripe e PayPal, permitindo que os usuários

escolham entre uma variedade de métodos de pagamento pela própria plataforma, incluindo cartões de crédito, débito e carteiras digitais, tornando o checkout mais conveniente.

**II) Conexão com APIs de fornecedores para atualização de estoque:** integrar a plataforma com APIs de fornecedores permitirá que o estoque de produtos seja atualizado em tempo real, evitando problemas com a venda de produtos fora de estoque e melhorando a precisão da disponibilidade de produtos na plataforma.

**III) Notificações em tempo real:** para melhorar a comunicação com o usuário, planeja-se a integração com serviços de notificação em tempo real, como Firebase ou Pusher, permitindo enviar atualizações instantâneas sobre o status de pedidos, promoções especiais e outras informações importantes diretamente para o dispositivo do usuário.

Essas melhorias e integrações adicionais são parte do plano de expansão contínua do **Wagon Delivery**, que visa elevar a experiência do usuário, oferecendo uma plataforma de e-commerce cada vez mais completa, eficiente e adaptada às necessidades dos consumidores.

## CAPÍTULO 8

### CONCLUSÃO

O desenvolvimento do **Wagon Delivery** representa um avanço significativo na criação de plataformas de e-commerce focadas na experiência do usuário e na eficiência operacional. Utilizando uma combinação de tecnologias modernas e uma arquitetura bem estruturada, o projeto alcançou seu objetivo de proporcionar uma interface de compra prática, responsiva e intuitiva, onde os usuários podem explorar produtos, adicionar itens ao carrinho e finalizar suas compras de maneira simplificada.

A escolha de ferramentas como **React**, **Next.js** e **TailwindCSS** demonstrou ser essencial para a construção de uma interface robusta e responsiva, que se adapta aos mais diversos dispositivos, garantindo uma experiência visual e funcional de alta qualidade tanto em desktops quanto em dispositivos móveis. A implementação de um sistema de autenticação seguro e integrado, utilizando o **Clerk**, assegurou a proteção de dados dos usuários e o gerenciamento eficaz das sessões, aprimorando a confiabilidade e a segurança do sistema.

Durante o processo de desenvolvimento, desafios como a **globalização do sistema de carrinho de compras** e a **estilização responsiva** foram enfrentados e solucionados com estratégias como o uso do **Context API** para gerenciamento centralizado de estado e a adoção de **TailwindCSS** para uma estilização responsiva, permitindo que a aplicação se mantivesse coerente e acessível em diferentes contextos de uso. Essas soluções ajudaram a criar um sistema coeso e acessível, que atende às necessidades dos usuários e promove uma navegação prática.

O projeto também considerou extensões futuras e melhorias importantes, como o **sistema de avaliações e comentários**, **filtros avançados** e **integrações com gateways de pagamento adicionais**, o que demonstra o compromisso contínuo em melhorar a experiência de compra e a eficiência da plataforma. A introdução de novas funcionalidades, aliada à integração com APIs de fornecedores e sistemas de notificações em tempo real, permitirá que a aplicação se mantenha atualizada com as necessidades e expectativas dos consumidores modernos, promovendo uma experiência de compra mais dinâmica e personalizada.

Em síntese, o **Wagon Delivery** é mais que uma plataforma de e-commerce; é um sistema pensado para oferecer uma experiência completa

e prática ao usuário, garantindo segurança, agilidade e personalização. A flexibilidade e escalabilidade do projeto o tornam preparado para futuras expansões e adaptações, destacando-o como uma solução eficiente e moderna para o mercado de vendas online. Com base nos resultados já alcançados e nas possibilidades de desenvolvimento, o **Wagon Delivery** está posicionado para crescer e evoluir continuamente, oferecendo valor tanto aos consumidores quanto aos fornecedores, ao se adaptar rapidamente às mudanças e inovações no cenário do comércio eletrônico.