

UNIVERSIDADE DE MOGI DAS CRUZES
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
ANÁLISE E IMPLEMENTAÇÃO DE SOFTWARE

PROJETO TIREGRIP

Guilherme Del Pino – RGM 11241102212

Felipe Silva de Castro - RGM 11241105181

Gustavo de Assis – RGM 11241103404

Ralf Alves Junior – RGM 11241104500

Mogi das Cruzes
2025

RESUMO

O presente trabalho descreve o desenvolvimento do TireGrip, uma plataforma web de comércio eletrônico (e-commerce) focada na venda de pneus automotivos. O objetivo principal do projeto foi criar uma solução digital completa que permitisse aos clientes consultar produtos, visualizar informações detalhadas, gerenciar um carrinho de compras e finalizar pedidos de forma eficiente e segura. A plataforma foi concebida para ser responsiva e intuitiva, visando a melhor experiência do usuário no ambiente digital. O escopo do projeto abrangeu a implementação de funcionalidades essenciais, como autenticação de usuários, exibição de catálogo de produtos, detalhamento individual de itens, gerenciamento dinâmico do carrinho e um fluxo de finalização de compra com simulação de pagamento. O desenvolvimento seguiu um conjunto de requisitos funcionais e não funcionais, priorizando a usabilidade, o desempenho e a organização do código. O resultado é um sistema web moderno que atende à demanda por uma solução prática e segura para a aquisição de pneus online.

Palavras-chave: E-commerce. Desenvolvimento Web. Pneus. TireGrip. Análise e Desenvolvimento de Sistemas.

SUMÁRIO

RESUMO INTRODUÇÃO METODOLOGIA DESENVOLVIMENTO	4.1
Estrutura do Sistema.....	4.2
Funcionalidades Implementadas.....	4.3
Tecnologias Utilizadas	
1.ESCOPO	
2.FIGMA	
3.CÓDIGOS REFERÊNCIAS	

INTRODUÇÃO

O comércio eletrônico tem se consolidado como um canal de vendas fundamental em diversos setores, incluindo o automotivo. A crescente demanda por conveniência e acesso rápido a informações detalhadas impulsiona a necessidade de plataformas digitais robustas e especializadas. Neste contexto, o projeto TireGrip surge com o propósito de desenvolver uma solução de e-commerce dedicada à venda de pneus, preenchendo uma lacuna de mercado por uma experiência de compra online simplificada e focada neste tipo de produto.

O objetivo geral deste trabalho é apresentar o processo de Análise e Implementação de Software que resultou na criação da plataforma web TireGrip. Os objetivos específicos incluem:

- 1.Facilitar o processo de compra de pneus online.
- 2.Disponibilizar informações completas e visuais dos produtos.
- 3.Prover uma navegação prática e intuitiva.
- 4.Garantir um fluxo de compra seguro e organizado (endereço, confirmação e pagamento).
- 5.Disponibilizar um carrinho de compras com atualização dinâmica.

O público-alvo são clientes que buscam adquirir pneus automotivos de maneira rápida, segura e prática. O sistema foi estruturado para ser uma aplicação web responsiva, moderna e com funcionalidades essenciais para a jornada de compra, conforme detalhado no Escopo Oficial do Projeto.

METODOLOGIA

A metodologia de desenvolvimento adotada para o projeto TireGrip baseou-se em uma abordagem prática de Análise e Desenvolvimento de Sistemas, focada na implementação de uma aplicação web funcional. O processo seguiu as seguintes etapas principais:

- 1.Definição de Escopo e Requisitos: Inicialmente, foram definidos o objetivo geral, os objetivos específicos e o público-alvo do projeto. Em seguida, foram levantados os Requisitos Funcionais (RF), como login de usuários, exibição de produtos, adição ao carrinho e finalização de compra, e os Requisitos Não Funcionais (RNF), como responsividade da interface, desempenho fluido e segurança básica na autenticação.
- 2.Design e Prototipação (Figma): Utilizou-se a ferramenta Figma para a criação do design e prototipação da interface do usuário (UI/UX). Esta etapa garantiu que a navegação fosse intuitiva e que o layout fosse moderno e responsivo, conforme os RNF.
- 3.Desenvolvimento e Implementação: A fase de implementação consistiu na codificação do front-end da aplicação. O código foi estruturado com foco em boas práticas e organização, utilizando tecnologias web padrão (HTML, CSS, JavaScript/Tailwind CSS, conforme trechos de código apresentados no documento original).
- 4.Testes e Validação: O sistema foi validado com base no Fluxo Geral do Usuário, garantindo que as funcionalidades essenciais (navegação, carrinho, finalização) estivessem operacionais e em conformidade com os requisitos definidos.

A abordagem priorizou a entrega de um produto mínimo viável (MVP) com foco nas funcionalidades centrais de um e-commerce, conforme detalhado no escopo do projeto.

DESENVOLVIMENTO

O desenvolvimento do projeto TireGrip resultou em uma aplicação web estruturada em módulos, conforme as funcionalidades definidas no escopo.

4.1 Estrutura do Sistema

O sistema é composto por uma aplicação front-end que gerencia a interação do usuário com o catálogo de produtos e o processo de compra. A estrutura de arquivos e o código-fonte (parcialmente apresentados no documento original) demonstram a organização do projeto.

4.2 Funcionalidades Implementadas

As principais funcionalidades desenvolvidas incluem:

- Tela de Login: Implementação de autenticação de usuário com validação de campos e tratamento de erros.
- Home: Exibição do catálogo de produtos com imagens, preços e navegação principal (Header e Footer).
- Tela Individual de Produto: Apresentação detalhada do produto, incluindo galeria de imagens, descrição técnica, preço e a funcionalidade de "Adicionar ao Carrinho".
- Carrinho de Compras: Funcionalidade dinâmica que permite a listagem, o cálculo total, a remoção e a alteração da quantidade de itens.
- Finalização da Compra: Fluxo guiado para inserção de endereço, revisão do pedido e simulação de métodos de pagamento.

4.3 Tecnologias Utilizadas

O desenvolvimento utilizou tecnologias de front-end modernas, com destaque para o uso de Tailwind CSS (conforme trechos de código CSS apresentados) para garantir a responsividade e a estilização da interface de forma eficiente. A organização do código-fonte, com a separação de componentes e estilos, reflete a aplicação de boas práticas de desenvolvimento de software.

O projeto está hospedado e pode ser consultado no repositório GitHub:

<https://github.com/felipeCastro-21/Projeto-final-tiregrip.git>

1. ESCOPO

ESCOPO OFICIAL DO PROJETO – TIREGRIP

1. Nome do Projeto TireGrip

2. Objetivo Geral

Desenvolver uma plataforma web completa para a loja de pneus TireGrip, permitindo que clientes consultem produtos, obtenham informações detalhadas, adicionem itens ao carrinho e concluam compras de forma simples e eficiente.

3. Objetivos Específicos Facilitar o processo de compra de

pneus online. Disponibilizar informações completas e visuais dos produtos. Permitir navegação

prática e intuitiva. Prover um fluxo de compra seguro e organizado (endereço, confirmação e

pagamento). Disponibilizar carrinho com atualização dinâmica.

4. Público-Alvo Clientes que desejam adquirir pneus automotivos de maneira rápida, segura e prática pelo ambiente digital.

5.

Descrição Geral do Sistema O sistema TireGrip será composto por uma aplicação web responsive e moderna contendo funcionalidades essenciais para exibição e compra de pneus. O usuário poderá navegar pelos produtos, visualizar detalhes, adicionar itens ao carrinho e finalizar sua compra.

6. Funcionalidades do Sistema 6.1 Tela de Login Autenticação de usuário via email e

senha. Validação de campos obrigatórios. Mensagem de erro para login inválido. 6.2 Home

Exibição de produtos com imagens e preços. Header com menu de navegação (Login, Produtos,

Carrinho). Footer com informações de contato, redes sociais e Copyright.

Apresentação

institucional e destaques. 6.3 Tela Individual de Produto Imagem principal + galeria do produto.

Descrição técnica detalhada. Preço e opções de quantidade. Botão “Adicionar ao Carrinho”. 6.4

Carrinho Listagem de produtos adicionados. Cálculo automático de valores totais.

Função para

remover ou alterar a quantidade de itens. Validação no ícone do carrinho exibindo quantidade. 6.5

Finalização da Compra Formulário de endereço completo. Revisão dos itens do pedido.

Confirmação e métodos de pagamento.

7. Requisitos do Sistema 7.1 Requisitos Funcionais (RF)

RF01: Permitir login de usuários. RF02: Exibir produtos na tela inicial. RF03: Exibir detalhes

completos dos produtos. RF04: Permitir adicionar produtos ao carrinho. RF05: Exibir e atualizar

informações do carrinho. RF06: Permitir finalização do pedido com formulário de endereço. RF07:

Processar pagamento (mock/simulação). 7.2 Requisitos Não Funcionais (RNF)

RNF01: Interface deve ser responsiva. RNF02: Sistema deve ter desempenho fluido.

RNF03: Código organizado com boas práticas. RNF04: Segurança básica na autenticação. 8. Fluxo Geral do Usuário Acessa

Home e visualiza produtos. Seleciona um produto para detalhes. Adiciona ao carrinho. Abre o carrinho, revisa e confirma. Insere dados de endereço. Finaliza pagamento. 9. Considerações

Finais O projeto TireGrip fornecerá uma plataforma moderna e funcional para compras online de pneus, garantindo uma experiência completa e eficiente ao usuário.

Github:

<https://github.com/felipeCastro-21/Projeto-final-tiregrip.git>

2. FIGMA

TIREGRIP

ACESSE SUA CONTA

E-MAIL
seu@email.com

SENHA
.....

Esqueceu a senha?

ENTRAR

OU

Não tem uma conta? [Cadastrar-se](#)

TIREGRIP

[INÍCIO](#) [PRODUTOS](#) [SOBRE](#) [CONTATO](#) [CARRINHO](#)

PERFORMANCE MÁXIMA NA ESTRADA

Pneus de alta qualidade com tecnologia de ponta para garantir sua segurança e desempenho em qualquer terreno.

[VER PRODUTOS](#)

GARANTIA TOTAL
12 meses de garantia em todos os produtos

ENTREGA RÁPIDA
Receba em até 48h após a confirmação

MÁXIMA SEGURANÇA
Produtos certificados e testados

NOSSOS PRODUTOS



PERFORMANCE PRO
205/55 R16 91V

R\$ 489,90
à vista no PIX

[ADICIONAR AO CARRINHO](#)



EXTREME GRIP
225/45 R17 94W

R\$ 629,90
à vista no PIX

[ADICIONAR AO CARRINHO](#)



URBAN SPORT
195/60 R15 88H

R\$ 369,90
à vista no PIX

[ADICIONAR AO CARRINHO](#)



RACING EDGE
245/40 R18 97Y

R\$ 899,90
à vista no PIX

[ADICIONAR AO CARRINHO](#)



HEAVY DUTY MAX
265/70 R16 112T

R\$ 749,90
à vista no PIX

[ADICIONAR AO CARRINHO](#)

CONTATO
Email: contato@tiregrip.com
Tel: (11) 9999-9999
WhatsApp: (11) 98888-8888
Seg-Sexta 8h às 18h
Sábado 8h às 14h

LINKS ÚTEIS
FAQ
Sobre Nós
Política de Privacidade
Termos de Uso
Garantia

NEWSLETTER
Receba ofertas exclusivas e novidades

[ASSINAR](#)

REDES SOCIAIS
[Facebook](#) [Instagram](#) [Twitter](#) [YouTube](#)

Siga-nos e fique por dentro das promoções

© 2025 TireGrip. Todos os direitos reservados.

TIREGRIP

INÍCIO PRODUTOS SOBRE CONTATO

MEU CARRINHO



PERFORMANCE PRO
205/55 R16 91V

1

R\$ 489.90

Remover

RESUMO DO PEDIDO

Subtotal	R\$ 489.90
Frete	R\$ 50.00
TOTAL	R\$ 539.90

FINALIZAR COMPRA

CONTINUAR COMPRANDO

TIREGRIP

INÍCIO PRODUTOS SOBRE CONTATO

1 ENDEREÇO 2 RESUMO 3 PAGAMENTO

ENDEREÇO DE ENTREGA

CEP
00000-000

RUA
Nome da rua

NÚMERO
123

COMPLEMENTO
Apto, bloco, etc

BAIRRO
Nome do bairro

CIDADE
Nome da cidade

CONTINUAR

RESUMO

Subtotal	R\$ 489.90
Frete	R\$ 50.00
TOTAL	R\$ 539.90

FINALIZAR PEDIDO

1 ENDEREÇO 2 RESUMO 3 PAGAMENTO

RESUMO DO PEDIDO



PERFORMANCE PRO
205/55 R16 91V
Qty: 1

R\$ 489.90

VOLTAR **CONTINUAR**

RESUMO

Subtotal	R\$ 489.90
Frete	R\$ 50.00
TOTAL	R\$ 539.90

TIREGRIP

INÍCIO PRODUTOS SOBRE CONTATO

FINALIZAR PEDIDO

ENDEREÇO RESUMO PAGAMENTO

FORMA DE PAGAMENTO

Cartão de Crédito Em até 10x sem juros

PIX 5% de desconto

VOLTAR FINALIZAR PEDIDO

RESUMO

Subtotal	R\$ 489,90
Frete	R\$ 50,00
Desconto PIX (5%)	- R\$ 27,00
TOTAL	R\$ 512,90

TIREGRIP

INÍCIO PRODUTOS SOBRE CONTATO

PERFORMANCE PRO

205/55 R16 91V

Preço à vista no PIX

R\$ 489,90

ou 10x de R\$ 48,99 sem juros

COMPRAR AGORA

ESPECIFICAÇÕES TÉCNICAS

Largura	205mm
Perfil	55
Aro	16"
Índice de Carga	91
Índice de Velocidade	V (240 km/h)
Tipo	Radial

CARACTERÍSTICAS

- ✓ Tecnologia de drenagem de água avançada
- ✓ Composto de borracha de alta aderência
- ✓ Baixo nível de ruído durante a condução
- ✓ Maior durabilidade e resistência ao desgaste
- ✓ Economia de combustível otimizada
- ✓ Desempenho em todas as estações

CONTATO

Email: contato@tiregrip.com
Tel: (11) 9999-9999
WhatsApp: (11) 98888-8888

LINKS ÚTEIS

FAQ
Sobre Nós
Política de Privacidade

NEWSLETTER

Receba ofertas exclusivas e novidades

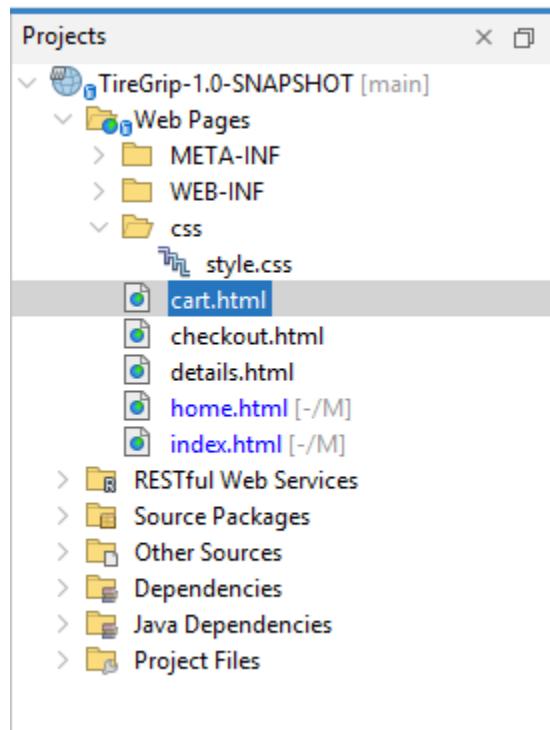
Seu e-mail

REDES SOCIAIS

Siga-nos e fique por dentro das promoções

3. CÓDIGOS

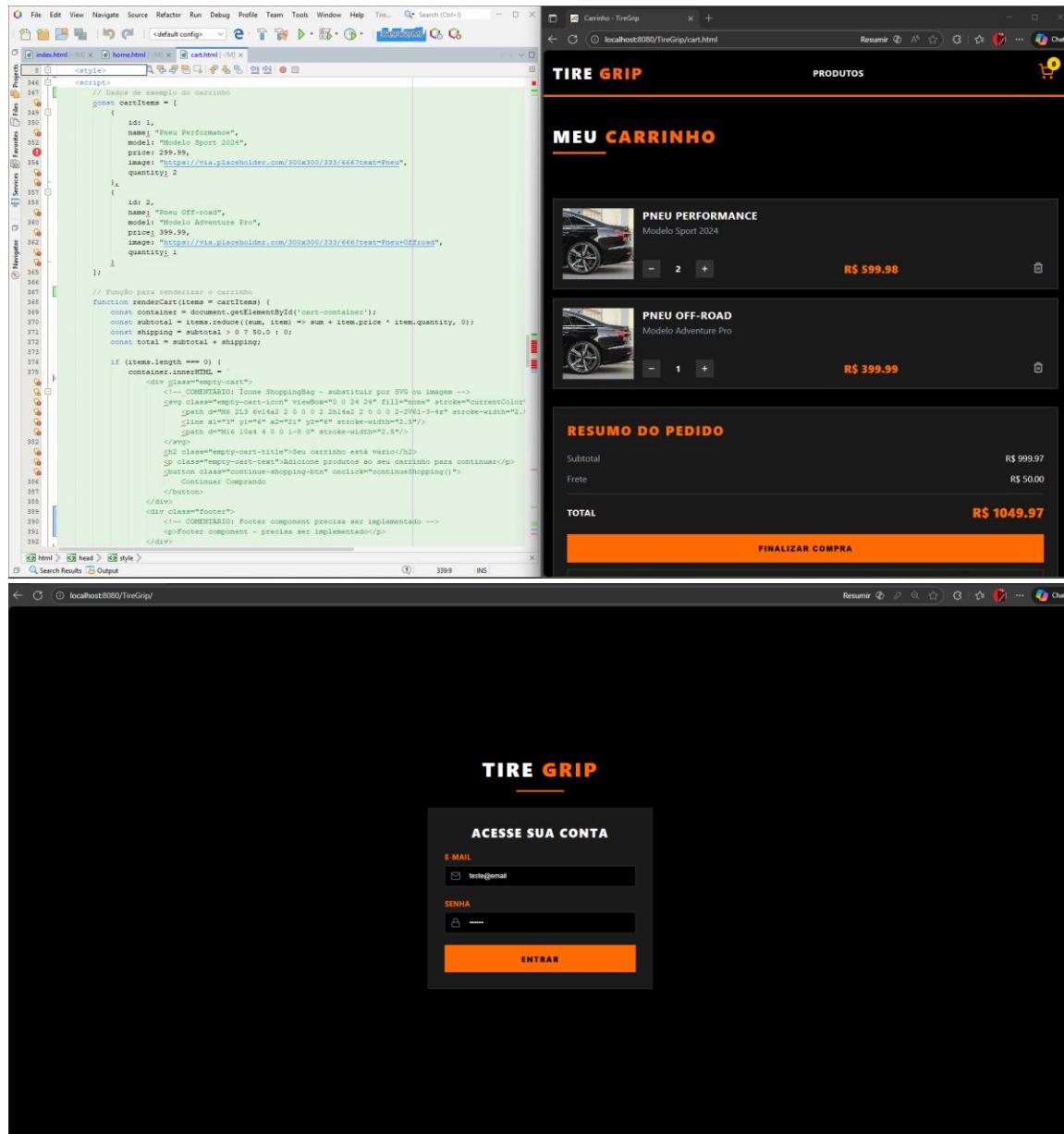
Estrutura dos Arquivos



Front-End

Codigos:

Tela de Login



The screenshot shows a browser window displaying the login page for a website named 'TIRE GRIP'. The page has a dark background with orange and white text. At the top, it says 'TIKE GRIP' and 'ACESSE SUA CONTA'. Below that is a form with fields for 'E MAIL' (containing 'teste@email') and 'SENHA' (containing '*****'). There is a large orange 'ENTRAR' button at the bottom of the form.

Code snippets from the browser's developer tools are overlaid on the screenshot:

```
// Dados de exemplo do carrinho
const cartItems = [
  {
    id: 1,
    name: "Pneu Performance",
    model: "Modelo Sport 2024",
    price: 399.99,
    image: "https://via.placeholder.com/300x300/333/666?text=Pneu",
    quantity: 2
  },
  {
    id: 2,
    name: "Pneu Off-road",
    model: "Modelo Adventure Pro",
    price: 399.99,
    image: "https://via.placeholder.com/300x300/333/666?text=PneuOffroad",
    quantity: 1
  }
]

// Função para renderizar o carrinho
function renderCart(items = cartItems) {
  const subtotal = items.reduce((sum, item) => sum + item.price * item.quantity, 0);
  const shipping = subtotal > 0 ? 50.0 : 0;
  const total = subtotal + shipping;

  if (items.length === 0) {
    container.innerHTML =
      `<!-- COMENTARIO: Ibase Shoppingbag - substituir por SVG ou imagem -->
      <div class="empty-cart">
        <!-- COMENTARIO: Ibase Shoppingbag - substituir por SVG ou imagem -->
        <img alt="Icone de carrinho vazio" width="24" height="24" style="color: ${currentColor};"/>
        <div> Seu carrinho está vazio! </div>
        <p>Adicione produtos ao seu carrinho para continuar!</p>
        <button class="continue-shopping-btn" onclick="continueShopping()> Continuar Comprando </button>
      </div>`;
  } else {
    <h2>MEU CARRINHO</h2>
    <table>
      <thead>
        <tr>
          <th>PRODUTO</th>
          <th>IMAGEM</th>
          <th>QUANTIDADE</th>
          <th>VALOR</th>
        </tr>
      <tbody>
        <tr>
          <td>PNEU PERFORMANCE</td>
          <td><img alt="Imagem do Pneu Performance" style="width: 100px; height: auto;"/></td>
          <td>2</td>
          <td>R$ 599.98</td>
        </tr>
        <tr>
          <td>PNEU OFF-ROAD</td>
          <td><img alt="Imagem do Pneu Off-road" style="width: 100px; height: auto;"/></td>
          <td>1</td>
          <td>R$ 399.99</td>
        </tr>
      </tbody>
    </table>
    <h3>RESUMO DO PEDIDO</h3>
    <table>
      <tbody>
        <tr>
          <td>Subtotal</td>
          <td>R$ 999.97</td>
        </tr>
        <tr>
          <td>Frete</td>
          <td>R$ 50.00</td>
        </tr>
        <tr>
          <td>TOTAL</td>
          <td>R$ 1049.97</td>
        </tr>
      </tbody>
    </table>
    <button class="finalizar-compra-btn" type="button">FINALIZAR COMPRA</button>
  
```

```
<div class="login-card">
  <h2 class="login-title">Acesse sua Conta</h2>

  <form class="login-form" id="login-form">
    <!-- Email Field -->
    <div class="form-group">
      <label for="email" class="form-label">E-mail</label>
      <div class="input-wrapper">
        <!-- COMENTÁRIO: Ícone Mail - substituir por SVG ou imagem -->
        <svg class="input-icon" viewBox="0 0 24 24" fill="none" stroke="currentColor">
          <path d="M4 4h16c1.1 0 2 .9 2 2v12c0 1.1-.9 2-2 2H4c-1.1 0-2-.9-2-2V6c0-1.1.9-2 2-2z" stroke-width="2"/>
          <polyline points="22,6 12,13 2,6" stroke-width="2"/>
        </svg>
        <input
          type="email"
          id="email"
          class="form-input"
          placeholder="seu@email.com"
          required
        />
      </div>
    </div>

    <!-- Password Field -->
    <div class="form-group">
      <label for="password" class="form-label">Senha</label>
      <div class="input-wrapper">
        <!-- COMENTÁRIO: Ícone Lock - substituir por SVG ou imagem -->
        <svg class="input-icon" viewBox="0 0 24 24" fill="none" stroke="currentColor">
          <rect x="3" y="11" width="18" height="11" rx="2" ry="2" stroke-width="2"/>
          <path d="M7 11V7a5 5 0 0 1 10 0v4" stroke-width="2"/>
        </svg>
        <input
          type="password"
          id="password"
          class="form-input"
          placeholder="*****"
          required
        />
      </div>
    </div>

    <!-- Submit Button -->
    <a href=".home.html" class="submit-button">
      Entrar
    </a>
  </form>
</div>
```

Menu

```

<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Performance Máxima - Pneus</title>
    <link rel="stylesheet" href="css/style.css">
  </head>
  <header class="header">
    <div class="container">
      <div class="header-content">
        <!-- Logo -->
        <button class="logo-button" id="home-button">
          <span class="logotext-white">TIRE
          <span class="logotext-orange">GRIP
        </button>

        <!-- Navigation -->
        <nav>
          <a href="#">Produtos</a>
          <a href="#">Contato</a>
        </nav>

        <!-- Cart Icon -->
        <a href="#">
          <img alt="Cart icon" class="cart-button" id="cart-button">
          <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
          <img alt="Cart icon" viewBox="0 0 24 24" fill="none" stroke="currentColor">
            <path d="M12 10.5L18 14.5L14 18Z" stroke-width="2.5" />
            <circle cx="20" cy="21" r="1" stroke-width="2.5" />
            <path d="M1 14.5 12 10.5 23 14.5" stroke-width="2.5" />
          </img>
          <span class="cart-badge" id="cart-badge">0</span>
        </a>
      </div>
      <!-- Mobile Menu Button -->
      <button class="mobile-menu-button">
        <img alt="Mobile menu icon" data-icon="menu" alt="Icone Menu" />
        <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
        <img alt="Mobile menu icon" viewBox="0 0 24 24" fill="none" stroke="currentColor">
          <line x1="3" y1="6" x2="21" y2="6" stroke-width="2.5" />
          <line x1="3" y1="12" x2="21" y2="12" stroke-width="2.5" />
          <line x1="3" y1="18" x2="21" y2="18" stroke-width="2.5" />
        </img>
      </button>
    </div>
  </header>

```

TIRE GRIP

PRODUTOS | CONTATO

PERFORMANCE MÁXIMA NA ESTRADA

Pneus de alta qualidade com tecnologia de ponta para garantir sua segurança e desempenho em qualquer terreno.

[VER PRODUTOS](#)


GARANTIA TOTAL
 12 meses de garantia em todos os produtos


ENTREGA RÁPIDA
 Receba em até 48h após a confirmação


MÁXIMA SEGURANÇA
 Produtos certificados e testados

NOSSOS PRODUTOS



TIRE GRIP

PRODUTOS | CONTATO

PERFORMANCE MÁXIMA NA ESTRADA

Pneus de alta qualidade com tecnologia de ponta para garantir sua segurança e desempenho em qualquer terreno.

[VER PRODUTOS](#)


GARANTIA TOTAL
 12 meses de garantia em todos os produtos


ENTREGA RÁPIDA
 Receba em até 48h após a confirmação


MÁXIMA SEGURANÇA
 Produtos certificados e testados

NOSSOS PRODUTOS



```

1  <!DOCTYPE html>
2  <html lang="pt-BR">
3
4    <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Portfólio Maxina - E-commerce</title>
8      <link rel="stylesheet" href="css/style.css">
9    </head>
10   <header class="header">
11     <div class="header-content">
12       <div class="header-content">
13         <!-- Logo -->
14         <button class="logo-button" id="home-button">
15           <span class="logo-text-white">TIRE
16           <span class="logo-text-orange">KTF
17         </button>
18
19         <!-- Navigation -->
20         <a style="text-decoration: none" class="nav-button" href="#products-section">ProdutosContato0</span>
34     </a>
35
36     <!-- Mobile Menu Button -->
37     <button class="mobile-menu-button">
38       <!-- COMENTÁRIO: Icône Hamburger - substituir por SVG ou imagem -->
39       <img class="menu-icon" viewbox="0 0 24 24" fill="none" stroke="currentColor">
40         <line x1="3" y1="21" x2="21" y2="6" stroke-width="2.5" />
41         <line x1="3" y1="12" x2="21" y2="12" stroke-width="2.5" />
42         <line x1="3" y1="18" x2="21" y2="18" stroke-width="2.5" />
43       </svg>
44     </button>
45   </div>
46 </header>
47
48 <body>
49   <!-- Hero Section -->
50   <section class="hero">
51     <div class="container">
52       <div class="hero-content">
53         <h1>PERFORMANCE<br>MAXIMA NA ESTRADA</h1>
54         <p>Pneus de alta qualidade com tecnologia de ponta para garantir sua segurança e desempenho em qualquer terreno.</p>
55         <a href="#products-section" class="cta-button">
56           Ver Produtos
57         </a>
58       </div>
59     </div>
60   </section>
61
62   <!-- Features Section -->
63   <section class="features">
64     <div class="features-grid">
65       <!-- Feature 1 -->
66       <div class="feature-card">
67         <div class="feature-icon orange">
68           <!-- COMENTÁRIO: Icône de garantia - substituir por SVG ou imagem -->
69           <img class="feature-icon" viewbox="0 0 24 24" fill="none" stroke="currentColor">
70             <path d="M12 22a8-4 8-10V5l-8-3-8 3v7a6 6 10 8 10x" stroke-width="2.5" />
71           </svg>
72         </div>
73         <div class="feature-content">
74           <h3>Garantia Total</h3>
75           <p>12 meses de garantia em todos os produtos</p>
76         </div>
77       </div>
78
79       <!-- Feature 2 -->
80       <div class="feature-card">
81         <div class="feature-icon orange">
82           <!-- COMENTÁRIO: Icône de rodagem - substituir por SVG ou imagem -->
83           <img class="feature-icon" viewbox="0 0 24 24" fill="none" stroke="currentColor">
84             <polyline points="13 2 3 14 12 14 11 22 21 10 12 10 13 2" stroke-width="2.5" />
85           </svg>
86         </div>
87         <div class="feature-content">
88
89       </div>
90     </div>
91   </section>
92
93 </body>

```

```
309 // ~~~~~
310 <script>
311     //Aqui da para passar id do produto.
312     document.addEventListener('DOMContentLoaded', function() {
313         const addToCartButtons = document.querySelectorAll('.add-to-cart');
314         const cartBadge = document.getElementById('cart-badge');
315
316         if (addToCartButtons.length > 0 && cartBadge) {
317             addToCartButtons.forEach(button => {
318                 button.addEventListener('click', function () {
319                     console.log("TESTE - Botão clicado");
320                     let currentCount = parseInt(cartBadge.textContent) || 0;
321                     cartBadge.textContent = currentCount + 1;
322
323                     this.style.backgroundColor = '#ff6a0080';
324                     setTimeout(() => {
325                         this.style.backgroundColor = '';
326                     }, 300);
327                 });
328             });
329         } else {
330             console.log('Botões ou badge não encontrados!');
331         }
332     });
333     </script>
334
335 </html>
```

Carrinho

```

<script>
  // Dados de exemplo do carrinho
  const carritoItems = [
    {
      id: 1,
      name: "Pneu Performance",
      model: "Sport 2024",
      price: 399.99,
      image: "https://via.placeholder.com/300x300/333/666?text=Pneu",
      quantity: 2
    },
    {
      id: 2,
      name: "Pneu Off-road",
      model: "Adventure Pro",
      price: 399.99,
      image: "https://via.placeholder.com/300x300/333/666?text=Pneu Offroad",
      quantity: 1
    }
  ];

  // Função para renderizar o carrinho
  function renderCart(items = carritoItems) {
    const container = document.getElementById('cart-contained');
    const subtotal = items.reduce((sum, item) => sum + item.price * item.Quantity, 0);
    const shipping = subtotal > 0 ? 50.0 : 0;
    const total = subtotal + shipping;

    if (items.length === 0) {
      container.innerHTML =
        `<div class="empty-cart">
          <!-- COMENTÁRIO: Icone ShoppingBag - substituir por SVG ou imagem -->
          <p>Carrozinho vazio</p>
          <img alt="Icone de sacola com 2 carros" data-width="24" data-height="24" data-currentColor="#FFF" data-dasharray="0 0 0 2.3h2a2 2 0 0 0 2.2v2d-3-4c stroke-width=2,5"/>
          <div class="empty-cart-text">Seu carrinho está vazio!</div>
          <p class="empty-cart-text">Adicione produtos ao seu carrinho para continuar.</p>
          <button class="continue-shopping-btn" onclick="continueShopping()>
            Continuar Comprando
          </button>
        </div>`;
    } else {
      <!-- COMENTÁRIO: Footer component precisa ser implementado -->
      <p>Footer component - precisa ser implementado</p>
    }
  }

```

TIRE GRIP PRODUTOS

MEU CARRINHO



PNEU PERFORMANCE
Modelo Sport 2024

R\$ 399.98



PNEU OFF-ROAD
Modelo Adventure Pro

R\$ 399.99

RESUMO DO PEDIDO

Subtotal	R\$ 999.97
Frete	R\$ 50.00
TOTAL	R\$ 1049.97

FINALIZAR COMPRA

CONTATO

Email: contato@tiregrip.com
Tel: (11) 9999-9999
WhatsApp: (11) 98888-8888
Seg-Sex: 8h às 18h
Sáb: 8h às 14h

LINKS ÚTEIS

FAQ
Sobre Nós
Política de Privacidade
Termos de Uso
Garantia

NEWSLETTER

Receba ofertas exclusivas e novidades

ASSINAR

localhost:8080/TireGrip/cart.html

TIRE GRIP

PRODUTOS



SEU CARRINHO ESTÁ VAZIO

Adicione produtos ao seu carrinho para continuar

CONTINUAR COMPRANDO

CONTATO

Email: contato@tiregrip.com
Tel: (11) 9999-9999
WhatsApp: (11) 98888-8888
Seg-Sext: 8h às 18h
Sáb: 8h às 14h

LINKS ÚTEIS

FAQ
Sobre Nós
Política de Privacidade
Termos de Uso
Garantia

NEWSLETTER

Receba ofertas exclusivas e novidades

Seu e-mail

ASSINAR

© 2025 TireGrip. Todos os direitos reservados.

localhost:8080/TireGrip/checkout.html

TIRE GRIP

PRODUTOS



FINALIZAR PEDIDO

1 ENDEREÇO 2 RESUMO 3 PAGAMENTO

ENDEREÇO DE ENTREGA

CEP: 00000-000
RUA: Nome da rua
NÚMERO: 123 COMPLEMENTO: Apto, bloco, etc.
BAIRRO: Nome do bairro CIDADE: Nome da cidade

RESUMO

Subtotal	R\$ 999.97
Frete	R\$ 50.00
TOTAL	R\$ 1049.97

CONTINUAR

CONTATO

LINKS ÚTEIS

NEWSLETTER

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

TimGrip-1.0-SNAPSHOT - Apache NetBeans IDE 28

index.html [M] x home.html [M] x cart.html [M] x checkout.html [M] x

```

456 //body
457 <div id="checkout-container">
458   <!-- O conteúdo sera carregado via JavaScript -->
459 </div>
460
461 <script>
462   // Criação de exemplo do carrinho
463   const checkoutItems = [
464     {
465       id: 1,
466       name: "Pneu Performance",
467       model: "Mobile Sport 2024",
468       price: 299.99,
469       image: "https://via.placeholder.com/300x300/333/666?text=Pneu",
470       quantity: 2
471     },
472     {
473       id: 2,
474       name: "Pneu Off-road",
475       model: "Mobile Adventure Pro",
476       price: 399.99,
477       image: "https://via.placeholder.com/300x300/333/666?text=PneuOffroad",
478       quantity: 1
479     }
480   ];
481
482   // Estado do checkout
483   let currentStep = 1;
484   let paymentMethod = null;
485
486   // Função para calcular totais
487   function calculateTotals() {
488     const subtotal = checkoutItems.reduce((sum, item) => sum + item.price * item.quantity, 0);
489     const shipping = 50.0;
490     const total = subtotal + shipping;
491     const discount = paymentMethod === 'pix' ? total * 0.05 : 0;
492     const finalTotal = total - discount;
493
494     return { subtotal, shipping, total, discount, finalTotal };
495   }
496
497   // Função para renderizar o checkout
498   function renderCheckout() {
499     const container = document.getElementById('checkout-container');
500     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
501
502     const steps = [
503       { number: 1, label: 'Endereço' },
504       { number: 2, label: 'Resumo' },
505       { number: 3, label: 'Pagamento' },
506     ];
507
508     container.innerHTML =
509       `<header class="header">
510         <div class="header-content">
511           <!-- Logo -->
512           <img alt="Logo da loja" />
513           <span class="logo-text white">TIRE</span>
514           <span class="logo-text orange">TIRE</span>
515         </div>
516       </header>
517       <div class="steps">
518         <ul>
519           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
520         </ul>
521       </div>
522       <div class="content">
523         <!-- Cart Icon -->
524         <a href="#" class="cart-button" id="cart-button">
525           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
526           <img alt="Icone de carrinho de compras" />
527           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
528           <span class="cart-badge" id="cart-badge">0</span>
529         </a>
530
531         <!-- Mobile Menu Button -->
532         <button class="mobile-menu-button">
533           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
534           <img alt="Icone de menu" />
535         </button>
536       </div>
537     `;
538   }
539
540   // Função para renderizar o checkout
541   function renderCheckout() {
542     const container = document.getElementById('checkout-container');
543     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
544
545     const steps = [
546       { number: 1, label: 'Endereço' },
547       { number: 2, label: 'Resumo' },
548       { number: 3, label: 'Pagamento' },
549     ];
550
551     container.innerHTML =
552       `<header class="header">
553         <div class="header-content">
554           <!-- Logo -->
555           <img alt="Logo da loja" />
556           <span class="logo-text white">TIRE</span>
557           <span class="logo-text orange">TIRE</span>
558         </div>
559       </header>
560       <div class="steps">
561         <ul>
562           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
563         </ul>
564       </div>
565       <div class="content">
566         <!-- Cart Icon -->
567         <a href="#" class="cart-button" id="cart-button">
568           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
569           <img alt="Icone de carrinho de compras" />
570           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
571           <span class="cart-badge" id="cart-badge">0</span>
572         </a>
573
574         <!-- Mobile Menu Button -->
575         <button class="mobile-menu-button">
576           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
577           <img alt="Icone de menu" />
578         </button>
579       </div>
580     `;
581   }
582
583   // Função para renderizar o checkout
584   function renderCheckout() {
585     const container = document.getElementById('checkout-container');
586     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
587
588     const steps = [
589       { number: 1, label: 'Endereço' },
590       { number: 2, label: 'Resumo' },
591       { number: 3, label: 'Pagamento' },
592     ];
593
594     container.innerHTML =
595       `<header class="header">
596         <div class="header-content">
597           <!-- Logo -->
598           <img alt="Logo da loja" />
599           <span class="logo-text white">TIRE</span>
600           <span class="logo-text orange">TIRE</span>
601         </div>
602       </header>
603       <div class="steps">
604         <ul>
605           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
606         </ul>
607       </div>
608       <div class="content">
609         <!-- Cart Icon -->
610         <a href="#" class="cart-button" id="cart-button">
611           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
612           <img alt="Icone de carrinho de compras" />
613           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
614           <span class="cart-badge" id="cart-badge">0</span>
615         </a>
616
617         <!-- Mobile Menu Button -->
618         <button class="mobile-menu-button">
619           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
620           <img alt="Icone de menu" />
621         </button>
622       </div>
623     `;
624   }
625
626   // Função para renderizar o checkout
627   function renderCheckout() {
628     const container = document.getElementById('checkout-container');
629     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
630
631     const steps = [
632       { number: 1, label: 'Endereço' },
633       { number: 2, label: 'Resumo' },
634       { number: 3, label: 'Pagamento' },
635     ];
636
637     container.innerHTML =
638       `<header class="header">
639         <div class="header-content">
640           <!-- Logo -->
641           <img alt="Logo da loja" />
642           <span class="logo-text white">TIRE</span>
643           <span class="logo-text orange">TIRE</span>
644         </div>
645       </header>
646       <div class="steps">
647         <ul>
648           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
649         </ul>
650       </div>
651       <div class="content">
652         <!-- Cart Icon -->
653         <a href="#" class="cart-button" id="cart-button">
654           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
655           <img alt="Icone de carrinho de compras" />
656           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
657           <span class="cart-badge" id="cart-badge">0</span>
658         </a>
659
660         <!-- Mobile Menu Button -->
661         <button class="mobile-menu-button">
662           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
663           <img alt="Icone de menu" />
664         </button>
665       </div>
666     `;
667   }
668
669   // Função para renderizar o checkout
670   function renderCheckout() {
671     const container = document.getElementById('checkout-container');
672     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
673
674     const steps = [
675       { number: 1, label: 'Endereço' },
676       { number: 2, label: 'Resumo' },
677       { number: 3, label: 'Pagamento' },
678     ];
679
680     container.innerHTML =
681       `<header class="header">
682         <div class="header-content">
683           <!-- Logo -->
684           <img alt="Logo da loja" />
685           <span class="logo-text white">TIRE</span>
686           <span class="logo-text orange">TIRE</span>
687         </div>
688       </header>
689       <div class="steps">
690         <ul>
691           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
692         </ul>
693       </div>
694       <div class="content">
695         <!-- Cart Icon -->
696         <a href="#" class="cart-button" id="cart-button">
697           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
698           <img alt="Icone de carrinho de compras" />
699           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
700           <span class="cart-badge" id="cart-badge">0</span>
701         </a>
702
703         <!-- Mobile Menu Button -->
704         <button class="mobile-menu-button">
705           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
706           <img alt="Icone de menu" />
707         </button>
708       </div>
709     `;
710   }
711
712   // Função para renderizar o checkout
713   function renderCheckout() {
714     const container = document.getElementById('checkout-container');
715     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
716
717     const steps = [
718       { number: 1, label: 'Endereço' },
719       { number: 2, label: 'Resumo' },
720       { number: 3, label: 'Pagamento' },
721     ];
722
723     container.innerHTML =
724       `<header class="header">
725         <div class="header-content">
726           <!-- Logo -->
727           <img alt="Logo da loja" />
728           <span class="logo-text white">TIRE</span>
729           <span class="logo-text orange">TIRE</span>
730         </div>
731       </header>
732       <div class="steps">
733         <ul>
734           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
735         </ul>
736       </div>
737       <div class="content">
738         <!-- Cart Icon -->
739         <a href="#" class="cart-button" id="cart-button">
740           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
741           <img alt="Icone de carrinho de compras" />
742           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
743           <span class="cart-badge" id="cart-badge">0</span>
744         </a>
745
746         <!-- Mobile Menu Button -->
747         <button class="mobile-menu-button">
748           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
749           <img alt="Icone de menu" />
750         </button>
751       </div>
752     `;
753   }
754
755   // Função para renderizar o checkout
756   function renderCheckout() {
757     const container = document.getElementById('checkout-container');
758     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
759
760     const steps = [
761       { number: 1, label: 'Endereço' },
762       { number: 2, label: 'Resumo' },
763       { number: 3, label: 'Pagamento' },
764     ];
765
766     container.innerHTML =
767       `<header class="header">
768         <div class="header-content">
769           <!-- Logo -->
770           <img alt="Logo da loja" />
771           <span class="logo-text white">TIRE</span>
772           <span class="logo-text orange">TIRE</span>
773         </div>
774       </header>
775       <div class="steps">
776         <ul>
777           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
778         </ul>
779       </div>
780       <div class="content">
781         <!-- Cart Icon -->
782         <a href="#" class="cart-button" id="cart-button">
783           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
784           <img alt="Icone de carrinho de compras" />
785           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
786           <span class="cart-badge" id="cart-badge">0</span>
787         </a>
788
789         <!-- Mobile Menu Button -->
790         <button class="mobile-menu-button">
791           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
792           <img alt="Icone de menu" />
793         </button>
794       </div>
795     `;
796   }
797
798   // Função para renderizar o checkout
799   function renderCheckout() {
800     const container = document.getElementById('checkout-container');
801     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
802
803     const steps = [
804       { number: 1, label: 'Endereço' },
805       { number: 2, label: 'Resumo' },
806       { number: 3, label: 'Pagamento' },
807     ];
808
809     container.innerHTML =
810       `<header class="header">
811         <div class="header-content">
812           <!-- Logo -->
813           <img alt="Logo da loja" />
814           <span class="logo-text white">TIRE</span>
815           <span class="logo-text orange">TIRE</span>
816         </div>
817       </header>
818       <div class="steps">
819         <ul>
820           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
821         </ul>
822       </div>
823       <div class="content">
824         <!-- Cart Icon -->
825         <a href="#" class="cart-button" id="cart-button">
826           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
827           <img alt="Icone de carrinho de compras" />
828           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
829           <span class="cart-badge" id="cart-badge">0</span>
830         </a>
831
832         <!-- Mobile Menu Button -->
833         <button class="mobile-menu-button">
834           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
835           <img alt="Icone de menu" />
836         </button>
837       </div>
838     `;
839   }
840
841   // Função para renderizar o checkout
842   function renderCheckout() {
843     const container = document.getElementById('checkout-container');
844     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
845
846     const steps = [
847       { number: 1, label: 'Endereço' },
848       { number: 2, label: 'Resumo' },
849       { number: 3, label: 'Pagamento' },
850     ];
851
852     container.innerHTML =
853       `<header class="header">
854         <div class="header-content">
855           <!-- Logo -->
856           <img alt="Logo da loja" />
857           <span class="logo-text white">TIRE</span>
858           <span class="logo-text orange">TIRE</span>
859         </div>
860       </header>
861       <div class="steps">
862         <ul>
863           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
864         </ul>
865       </div>
866       <div class="content">
867         <!-- Cart Icon -->
868         <a href="#" class="cart-button" id="cart-button">
869           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
870           <img alt="Icone de carrinho de compras" />
871           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
872           <span class="cart-badge" id="cart-badge">0</span>
873         </a>
874
875         <!-- Mobile Menu Button -->
876         <button class="mobile-menu-button">
877           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
878           <img alt="Icone de menu" />
879         </button>
880       </div>
881     `;
882   }
883
884   // Função para renderizar o checkout
885   function renderCheckout() {
886     const container = document.getElementById('checkout-container');
887     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
888
889     const steps = [
890       { number: 1, label: 'Endereço' },
891       { number: 2, label: 'Resumo' },
892       { number: 3, label: 'Pagamento' },
893     ];
894
895     container.innerHTML =
896       `<header class="header">
897         <div class="header-content">
898           <!-- Logo -->
899           <img alt="Logo da loja" />
900           <span class="logo-text white">TIRE</span>
901           <span class="logo-text orange">TIRE</span>
902         </div>
903       </header>
904       <div class="steps">
905         <ul>
906           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
907         </ul>
908       </div>
909       <div class="content">
910         <!-- Cart Icon -->
911         <a href="#" class="cart-button" id="cart-button">
912           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
913           <img alt="Icone de carrinho de compras" />
914           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
915           <span class="cart-badge" id="cart-badge">0</span>
916         </a>
917
918         <!-- Mobile Menu Button -->
919         <button class="mobile-menu-button">
920           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
921           <img alt="Icone de menu" />
922         </button>
923       </div>
924     `;
925   }
926
927   // Função para renderizar o checkout
928   function renderCheckout() {
929     const container = document.getElementById('checkout-container');
930     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
931
932     const steps = [
933       { number: 1, label: 'Endereço' },
934       { number: 2, label: 'Resumo' },
935       { number: 3, label: 'Pagamento' },
936     ];
937
938     container.innerHTML =
939       `<header class="header">
940         <div class="header-content">
941           <!-- Logo -->
942           <img alt="Logo da loja" />
943           <span class="logo-text white">TIRE</span>
944           <span class="logo-text orange">TIRE</span>
945         </div>
946       </header>
947       <div class="steps">
948         <ul>
949           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
950         </ul>
951       </div>
952       <div class="content">
953         <!-- Cart Icon -->
954         <a href="#" class="cart-button" id="cart-button">
955           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
956           <img alt="Icone de carrinho de compras" />
957           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
958           <span class="cart-badge" id="cart-badge">0</span>
959         </a>
960
961         <!-- Mobile Menu Button -->
962         <button class="mobile-menu-button">
963           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
964           <img alt="Icone de menu" />
965         </button>
966       </div>
967     `;
968   }
969
970   // Função para renderizar o checkout
971   function renderCheckout() {
972     const container = document.getElementById('checkout-container');
973     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
974
975     const steps = [
976       { number: 1, label: 'Endereço' },
977       { number: 2, label: 'Resumo' },
978       { number: 3, label: 'Pagamento' },
979     ];
980
981     container.innerHTML =
982       `<header class="header">
983         <div class="header-content">
984           <!-- Logo -->
985           <img alt="Logo da loja" />
986           <span class="logo-text white">TIRE</span>
987           <span class="logo-text orange">TIRE</span>
988         </div>
989       </header>
990       <div class="steps">
991         <ul>
992           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
993         </ul>
994       </div>
995       <div class="content">
996         <!-- Cart Icon -->
997         <a href="#" class="cart-button" id="cart-button">
998           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
999           <img alt="Icone de carrinho de compras" />
1000           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
1001           <span class="cart-badge" id="cart-badge">0</span>
1002         </a>
1003
1004         <!-- Mobile Menu Button -->
1005         <button class="mobile-menu-button">
1006           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
1007           <img alt="Icone de menu" />
1008         </button>
1009       </div>
1010     `;
1011   }
1012
1013   // Função para renderizar o checkout
1014   function renderCheckout() {
1015     const container = document.getElementById('checkout-container');
1016     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
1017
1018     const steps = [
1019       { number: 1, label: 'Endereço' },
1020       { number: 2, label: 'Resumo' },
1021       { number: 3, label: 'Pagamento' },
1022     ];
1023
1024     container.innerHTML =
1025       `<header class="header">
1026         <div class="header-content">
1027           <!-- Logo -->
1028           <img alt="Logo da loja" />
1029           <span class="logo-text white">TIRE</span>
1030           <span class="logo-text orange">TIRE</span>
1031         </div>
1032       </header>
1033       <div class="steps">
1034         <ul>
1035           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
1036         </ul>
1037       </div>
1038       <div class="content">
1039         <!-- Cart Icon -->
1040         <a href="#" class="cart-button" id="cart-button">
1041           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
1042           <img alt="Icone de carrinho de compras" />
1043           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
1044           <span class="cart-badge" id="cart-badge">0</span>
1045         </a>
1046
1047         <!-- Mobile Menu Button -->
1048         <button class="mobile-menu-button">
1049           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
1050           <img alt="Icone de menu" />
1051         </button>
1052       </div>
1053     `;
1054   }
1055
1056   // Função para renderizar o checkout
1057   function renderCheckout() {
1058     const container = document.getElementById('checkout-container');
1059     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
1060
1061     const steps = [
1062       { number: 1, label: 'Endereço' },
1063       { number: 2, label: 'Resumo' },
1064       { number: 3, label: 'Pagamento' },
1065     ];
1066
1067     container.innerHTML =
1068       `<header class="header">
1069         <div class="header-content">
1070           <!-- Logo -->
1071           <img alt="Logo da loja" />
1072           <span class="logo-text white">TIRE</span>
1073           <span class="logo-text orange">TIRE</span>
1074         </div>
1075       </header>
1076       <div class="steps">
1077         <ul>
1078           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
1079         </ul>
1080       </div>
1081       <div class="content">
1082         <!-- Cart Icon -->
1083         <a href="#" class="cart-button" id="cart-button">
1084           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
1085           <img alt="Icone de carrinho de compras" />
1086           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
1087           <span class="cart-badge" id="cart-badge">0</span>
1088         </a>
1089
1090         <!-- Mobile Menu Button -->
1091         <button class="mobile-menu-button">
1092           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
1093           <img alt="Icone de menu" />
1094         </button>
1095       </div>
1096     `;
1097   }
1098
1099   // Função para renderizar o checkout
1100   function renderCheckout() {
1101     const container = document.getElementById('checkout-container');
1102     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
1103
1104     const steps = [
1105       { number: 1, label: 'Endereço' },
1106       { number: 2, label: 'Resumo' },
1107       { number: 3, label: 'Pagamento' },
1108     ];
1109
1110     container.innerHTML =
1111       `<header class="header">
1112         <div class="header-content">
1113           <!-- Logo -->
1114           <img alt="Logo da loja" />
1115           <span class="logo-text white">TIRE</span>
1116           <span class="logo-text orange">TIRE</span>
1117         </div>
1118       </header>
1119       <div class="steps">
1120         <ul>
1121           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
1122         </ul>
1123       </div>
1124       <div class="content">
1125         <!-- Cart Icon -->
1126         <a href="#" class="cart-button" id="cart-button">
1127           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
1128           <img alt="Icone de carrinho de compras" />
1129           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
1130           <span class="cart-badge" id="cart-badge">0</span>
1131         </a>
1132
1133         <!-- Mobile Menu Button -->
1134         <button class="mobile-menu-button">
1135           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
1136           <img alt="Icone de menu" />
1137         </button>
1138       </div>
1139     `;
1140   }
1141
1142   // Função para renderizar o checkout
1143   function renderCheckout() {
1144     const container = document.getElementById('checkout-container');
1145     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
1146
1147     const steps = [
1148       { number: 1, label: 'Endereço' },
1149       { number: 2, label: 'Resumo' },
1150       { number: 3, label: 'Pagamento' },
1151     ];
1152
1153     container.innerHTML =
1154       `<header class="header">
1155         <div class="header-content">
1156           <!-- Logo -->
1157           <img alt="Logo da loja" />
1158           <span class="logo-text white">TIRE</span>
1159           <span class="logo-text orange">TIRE</span>
1160         </div>
1161       </header>
1162       <div class="steps">
1163         <ul>
1164           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
1165         </ul>
1166       </div>
1167       <div class="content">
1168         <!-- Cart Icon -->
1169         <a href="#" class="cart-button" id="cart-button">
1170           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
1171           <img alt="Icone de carrinho de compras" />
1172           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
1173           <span class="cart-badge" id="cart-badge">0</span>
1174         </a>
1175
1176         <!-- Mobile Menu Button -->
1177         <button class="mobile-menu-button">
1178           <!-- COMENTÁRIO: Icône Menu - substituir por SVG ou imagem -->
1179           <img alt="Icone de menu" />
1180         </button>
1181       </div>
1182     `;
1183   }
1184
1185   // Função para renderizar o checkout
1186   function renderCheckout() {
1187     const container = document.getElementById('checkout-container');
1188     const { subtotal, shipping, total, discount, finalTotal } = calculateTotals();
1189
1190     const steps = [
1191       { number: 1, label: 'Endereço' },
1192       { number: 2, label: 'Resumo' },
1193       { number: 3, label: 'Pagamento' },
1194     ];
1195
1196     container.innerHTML =
1197       `<header class="header">
1198         <div class="header-content">
1199           <!-- Logo -->
1200           <img alt="Logo da loja" />
1201           <span class="logo-text white">TIRE</span>
1202           <span class="logo-text orange">TIRE</span>
1203         </div>
1204       </header>
1205       <div class="steps">
1206         <ul>
1207           ${steps.map(step => `<li>${step.number}. ${step.label}</li>`).join('')}
1208         </ul>
1209       </div>
1210       <div class="content">
1211         <!-- Cart Icon -->
1212         <a href="#" class="cart-button" id="cart-button">
1213           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
1214           <img alt="Icone de carrinho de compras" />
1215           <!-- COMENTÁRIO: Icône ShoppingCart - substituir por SVG ou imagem -->
1216           <span class="cart-badge" id="cart-badge">0</span>
1217         </a>
1218
1219         <!-- Mobile Menu Button -->
122
```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

TreeGrip-1.0-SNAPSHOT - Apache NetBeans IDE 28

index.html home.html cart.html checkout.html [-M] x

Source History Favorites Files Projects

```
<!-- Content -->
<div class="checkout-content">
    <!-- Left Column - Form -->
    <div>
        <!-- Step 1: Address -->
        $currentStep == 1 .
        <div class="form-section">
            <h2 class="section-title">Endereço de Entrega</h2>
            <form class="checkout-form" id="address-form">
                <div class="form-group">
                    <label class="form-label">CEP:</label>
                    <input type="text" class="form-input" placeholder="00000-000" required>
                </div>
                <div class="form-group full-width">
                    <label class="form-label">Rua:</label>
                    <input type="text" class="form-input" placeholder="Nome da rua" required>
                </div>
                <div class="form-group">
                    <label class="form-label">Número:</label>
                    <input type="text" class="form-input" placeholder="123" required>
                </div>
                <div class="form-group">
                    <label class="form-label">Complemento:</label>
                    <input type="text" class="form-input" placeholder="Apto, bloco, etc.">
                </div>
                <div class="form-group">
                    <label class="form-label">Bairro:</label>
                    <input type="text" class="form-input" placeholder="Nome do bairro" required>
                </div>
                <div class="form-group">
                    <label class="form-label">Cidade:</label>
                    <input type="text" class="form-input" placeholder="Nome da cidade" required>
                </div>
            </div>
            <button type="button" class="btn btn-primary btn-full" onclick="goToStep(2)">
                Continuar
            </button>
        </form>
    </div>
    <!-- I -->
    <!-- Step 2: Order Summary -->
    $currentStep == 2 .
    <div class="form-section">
        <h2 class="section-title">Resumo do Pedido</h2>
        <div class="order-items">

```

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

TreeGrip-1.0-SNAPSHOT - Apache NetBeans IDE 28

index.html home.html cart.html checkout.html [-M] x

Source History Favorites Files Projects

```
        </form>
    <!-- I -->
    <!-- Step 2: Order Summary -->
    $currentStep == 2 .
    <div class="form-section">
        <h2 class="section-title">Resumo do Pedido</h2>
        <div class="order-items">
            $checkoutItems.map(item =>
                <div class="order-item">
                    <div class="item-image">
                        <img alt="${item.name}" onerror="this.src='https://via.placeholder.com/300x300/333/666?ext=Image&Indisponivel'">
                    </div>
                    <div class="item-details">
                        <h3 class="item-name">${item.name}</h3>
                        <p class="item-info"> ${item.modelo}</p>
                        <p class="item-info"> ${item.quantity}</p>
                    </div>
                    <div class="item-price">
                        R$ ${item.price * item.quantity}.toFixed(2)
                    </div>
                </div>
            ).join(''))
        </div>
        <div class="button-group">
            <button type="button" class="btn btn-secondary" onclick="goToStep(1)">
                Voltar
            </button>
            <button type="button" class="btn btn-primary" onclick="goToStep(3)">
                Continuar
            </button>
        </div>
    </div>
    <!-- I -->
    <!-- Step 3: Payment -->
    $currentStep == 3 .
    <div class="form-section">
        <h2 class="section-title">Forma de Pagamento</h2>
        <div class="payment-methods">
            <!-- Credit Card -->
            <button class="payment-method" $paymentMethod == 'credit' ? 'selected' : ''>
                onclick="paymentMethod('credit')"
            </div>
            <div class="payment-icon credit" $paymentMethod == 'credit' ? 'selected' : ''>
                <!-- COMENTARIO: Icone CreditCard -->
                <img alt="Icone de Cartão de Crédito" onerror="this.src='https://via.placeholder.com/300x300/333/666?ext=Image&Indisponivel'">
            </div>

```

Detalhe

```
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Search (Ctrl+F)
default config
index.html (M) x home.html (M) x cart.html (M) x checkout.html (M) x details.html (M) x
Source History Find Replace Go Back Go Forward Stop Refresh Reload Chat
Project Navigator Files Favorites Services
details.html
246 </head>
247 </body>
248 <header class="header">
249   <div class="header__container">
250     <!-- Logo -->
251     <button class="log-button" id="home-button">
252       <span class="log-text-white">TIRE
253       <span class="log-text-orange">GRIP
254     </button>
255
256     <!-- Navegação -->
257     <nav class="decription-nav">
258       <a href="#" style="text-decoration: none" class="nav-button" href="home.html">Produtos
259     </nav>
260
261     <!-- Carrinho -->
262     <a href="#" class="cart-button" id="cart-button">
263       <!-- COMENTÁRIO: ícone Shopping Cart - substituir por SVG ou imagem -->
264       <svg class="cart-icon" viewBox="0 0 24 24" fill="none" stroke="currentColor">
265         <circle cx="12" cy="12" r="11" stroke-width="2.5" />
266         <path d="M1 lmt12.66 13.39z2 0 0 0 2 1.6lb9.7za2 2 0 0 0 2-1.6l23 4t6" stroke-width="2.5" />
267       </svg>
268       <span class="cart-badge" id="cart-badge">0</span>
269     </button>
270
271     <!-- Mobile Menu Button -->
272     <button class="mobile-menu-button">
273       <!-- COMENTÁRIO: ícone menu móvel - substituir por SVG ou imagem -->
274       <svg class="menu-icon" viewBox="0 0 24 24" fill="none" stroke="currentColor">
275         <line x1="8" y1="6" x2="21" y2="12" stroke-width="2.5" />
276         <line x1="8" y1="12" x2="21" y2="18" stroke-width="2.5" />
277         <line x1="8" y1="18" x2="21" y2="12" stroke-width="2.5" />
278       </svg>
279     </button>
280   </div>
281 </header>
282 <div id="product-detail-container">
283   <!-- O conteúdo será carregado via JavaScript -->
284 </div>
285
286 <script>
287   // Dados do produto (exemplo)
288   const product = {
289     nome: "Pneu Performance Max",
290     preco: 599.99,
291     descricao: "Modelo Sport 2024 - Alta Performance",
292     especificacoes: {
293       largura: "205mm",
294       perfil: "55",
295       aro: "16\"},
296       indiceCarga: 91,
297       indiceVelocidade: "V (240 km/h)",
298       tipo: "Radial"
299     },
300     caracteristicas: [
301       "Tecnologia de drenagem de água avançada"
302     ]
303   }
304 </script>
305
306 </body>
307 </html>
```



TIRE GRIP PRODUTOS

PNEU PERFORMANCE MAX

Modelo Sport 2024 - Alta Performance

Preço à vista no PIX

R\$ 599.99

ou 10x de R\$ 60.00 sem juros

COMPRAR AGORA

ESPECIFICAÇÕES TÉCNICAS

Largura	205mm
Perfil	55
Aro	16"
Índice de Carga	91
Índice de Velocidade	V (240 km/h)
Tipo	Radial

CARACTERÍSTICAS

Tecnologia de drenagem de água avançada

The image shows two side-by-side screenshots of a software interface, likely a code editor or browser developer tools, comparing two different versions of an HTML file. The left screenshot displays the original code, while the right screenshot shows the modified code. Both versions include the same header, navigation bar, and product detail container, but differ in the script section.

Left Screenshot (Original Code):

```
<!-- Header -->
<header>
  <div class="container">
    <div class="header-content">
      <button class="log-in-button" id="home-button">
        <span class="log-in-text-white">TIRE</span>
        <span class="log-in-text-orange">GRIP</span>
      </button>

      <!-- Navigation -->
      <nav class="desktop-nav">
        <a style="text-decoration: none" href="#produtos">Produtos</a>
      </nav>

      <!-- Cart Icon -->
      <a href="#" class="cart-icon" id="cart-button">
        <img alt="Icone Shopping Cart - substituir por SVG ou imagem" data-bbox="188 218 558 278"/>
        <span class="cart-badge" id="cart-badge">0</span>
      </a>
    </div>
  </div>
</header>
<div id="product-detail-container">
  <!-- O conteúdo será carregado via JavaScript -->
</div>
<script>
  // Fazendo o produto (exemplo)
  const products = [
    {
      id: 1,
      name: "Pneu PERFORMANCE 2024",
      model: "Mobile Sport 2024 - Alta Performance",
      price: 599.99,
      image: "https://via.placeholder.com/400x400/333/667?text=Pneu+Performance"
    }
  ];

  // Especificações técnicas
  const specifications = [
    { label: "Tamanho", value: "205mm" },
    { label: "Perfis", value: "55" },
    { label: "Peso", value: "1.2kg" },
    { label: "Índice de Carga", value: "91" },
    { label: "Índice de Velocidade", value: "V (240 km/h)" },
    { label: "Tipo", value: "Radial" }
  ];

  // Características
  const features = [
    "Tecnologia de drenagem de água avançada",
    "Composto de borracha de alta aderência",
    "Nível de ruído reduzido ao máximo",
    "Maior durabilidade e resistência ao desgaste",
    "Economia de combustível otimizada",
    "Desempenho em todas as estações"
  ];
</script>
```

Right Screenshot (Modified Code):

```
<!-- Header -->
<header>
  <div class="container">
    <div class="header-content">
      <button class="log-in-button" id="home-button">
        <span class="log-in-text-white">TIRE</span>
        <span class="log-in-text-orange">GRIP</span>
      </button>

      <!-- Navigation -->
      <nav class="desktop-nav">
        <a style="text-decoration: none" href="#produtos">Produtos</a>
      </nav>

      <!-- Cart Icon -->
      <a href="#" class="cart-icon" id="cart-button">
        <img alt="Icone Shopping Cart - substituir por SVG ou imagem" data-bbox="188 218 558 278"/>
        <span class="cart-badge" id="cart-badge">0</span>
      </a>
    </div>
  </div>
</header>
<div id="product-detail-container">
  <!-- O conteúdo será carregado via JavaScript -->
</div>
<script>
  // Fazendo o produto (exemplo)
  const products = [
    {
      id: 1,
      name: "PNEU PERFORMANCE 2024",
      model: "Mobile Sport 2024 - Alta Performance",
      price: 599.99,
      image: "https://via.placeholder.com/400x400/333/667?text=Pneu+Performance"
    }
  ];

  // Especificações técnicas
  const specifications = [
    { label: "Tamanho", value: "205mm" },
    { label: "Perfis", value: "55" },
    { label: "Peso", value: "1.2kg" },
    { label: "Índice de Carga", value: "91" },
    { label: "Índice de Velocidade", value: "V (240 km/h)" },
    { label: "Tipo", value: "Radial" }
  ];

  // Características
  const features = [
    "Tecnologia de drenagem de água avançada",
    "Composto de borracha de alta aderência",
    "Nível de ruído reduzido ao máximo",
    "Maior durabilidade e resistência ao desgaste",
    "Economia de combustível otimizada",
    "Desempenho em todas as estações"
  ];
</script>
```

css

```
:root { --font-size: 16px; --background: #ffffff; --foreground: oklch(0.145 0 0); --card: #ffffff; --card-foreground: oklch(0.145 0 0); --popover: oklch(1 0 0); --popover-foreground: oklch(0.145 0 0); --primary: #030213; --primary-foreground: oklch(1 0 0); --secondary: oklch(0.95 0.0058 264.53); --secondary-foreground: #030213; --muted: #ecef0; --muted-foreground: #717182; --accent: #e9ebef; --accent-foreground: #030213; --destructive: #d4183d; --destructive-foreground: #ffffff; --border: rgba(0, 0, 0, 0.1); --input: transparent; --input-background: #f3f3f5; --switch-background: #cbced4; --font-weight-medium: 500; --font-weight-normal: 400; --ring: oklch(0.708 0 0); --chart-1: oklch(0.646 0.222 41.116); --chart-2: oklch(0.6 0.118 184.704); --chart-3: oklch(0.398 0.07 227.392); --chart-4: oklch(0.828 0.189 84.429); --chart-5: oklch(0.769 0.188 70.08); --radius: 0.625rem; --sidebar: oklch(0.985 0 0); --sidebar-foreground: oklch(0.145 0 0); --sidebar-primary: #030213; --sidebar-primary-foreground: oklch(0.985 0 0); --sidebar-accent: oklch(0.97 0 0); --sidebar-accent-foreground: oklch(0.205 0 0); --sidebar-border: oklch(0.922 0 0); --sidebar-ring: oklch(0.708 0 0); }

.dark { --background: oklch(0.145 0 0); --foreground: oklch(0.985 0 0); --card: oklch(0.145 0 0); --card-foreground: oklch(0.985 0 0); --popover: oklch(0.145 0 0); --popover-foreground: oklch(0.985 0 0); --primary: oklch(0.985 0 0); --primary-foreground: oklch(0.205 0 0); --secondary: oklch(0.269 0 0); --secondary-foreground: oklch(0.985 0 0); --muted: oklch(0.269 0 0); --muted-foreground: oklch(0.708 0 0); --accent: oklch(0.269 0 0); --accent-foreground: oklch(0.985 0 0); --destructive: oklch(0.396 0.141 25.723); --destructive-foreground: oklch(0.637 0.237 25.331); --border: oklch(0.269 0 0); --input: oklch(0.269 0 0); --ring: oklch(0.439 0 0); --font-weight-medium: 500; --font-weight-normal: 400; --chart-1: oklch(0.488 0.243 264.376); --chart-2: oklch(0.696 0.17 162.48); --chart-3: oklch(0.769 0.188 70.08); --chart-4: oklch(0.627 0.265 303.9); --chart-5: oklch(0.645 0.246 16.439); --sidebar: oklch(0.205 0 0); --sidebar-foreground: oklch(0.985 0 0); --sidebar-primary: oklch(0.488 0.243 264.376); --sidebar-primary-foreground: oklch(0.985 0 0); --sidebar-accent: oklch(0.269 0 0); --sidebar-accent-foreground: oklch(0.985 0 0); --sidebar-border: oklch(0.269 0 0); --sidebar-ring: oklch(0.439 0 0); }

@theme inline { --color-background: var(--background); --color-foreground: var(--foreground); --color-card: var(--card); --color-card-foreground: var(--card-foreground); --color-popover: var(--popover); --color-popover-foreground: var(--popover-foreground); --color-primary: var(--primary); --color-primary-foreground: var(--primary-foreground); --color-secondary: var(--secondary); --color-secondary-foreground: var(--secondary-foreground); --color-muted: var(--muted); --color-muted-foreground: var(--muted-foreground); --color-accent: var(--accent); --color-accent-foreground: var(--accent-foreground); --color-destructive: var(--destructive); --color-destructive-foreground: var(--destructive-foreground); --color-border: var(--border); --color-input: var(--input); --color-input-background: var(--input-background); --color-switch-background: var(--switch-background); --color-ring: var(--ring); --color-chart-1: var(--chart-1); --color-chart-2: var(--chart-2); --color-chart-3: var(--chart-3); --color-chart-4: var(--chart-4); --color-chart-5: var(--chart-5); --radius-sm: calc(var(--radius) - 4px); --radius-md: calc(var(--radius) - 2px); --radius-lg: var(--radius); --radius-xl: calc(var(--radius) + 4px); --color-sidebar: var(--
```

```
sidebar); --color-sidebar-foreground: var(--sidebar-foreground); --color-sidebar-primary:  
var(--sidebar-primary); --color-sidebar-primary-foreground: var(--sidebar-primary-  
foreground); --color-sidebar-accent: var(--sidebar-accent); --color-sidebar-accent-  
foreground: var(--sidebar-accent-foreground); --color-sidebar-border: var(--sidebar-  
border); --color-sidebar-ring: var(--sidebar-ring); }
```

```
@layer base { * { @apply border-border outline-ring/50; }
```

```
body {  
    @apply bg-background text-foreground;  
}
```

```
}
```

```
/**
```

- Base typography. This is not applied to elements which have an ancestor with a Tailwind text class.

```
/ @layer base { :where(:not(:has([class='text-']))), :not(:has([class^='text-'])))) { h1 { font-  
size: var(--text-2xl); font-weight: var(--font-weight-medium); line-height: 1.5; } }
```

```
h2 {  
    font-size: var(--text-xl);  
    font-weight: var(--font-weight-medium);  
    line-height: 1.5;  
}
```

```
h3 {  
    font-size: var(--text-lg);  
    font-weight: var(--font-weight-medium);  
    line-height: 1.5;  
}
```

```
h4 {  
    font-size: var(--text-base);  
    font-weight: var(--font-weight-medium);  
    line-height: 1.5;  
}
```

```
p {  
    font-size: var(--text-base);  
    font-weight: var(--font-weight-normal);  
    line-height: 1.5;
```

```
        }

    label {
        font-size: var(--text-base);
        font-weight: var(--font-weight-medium);
        line-height: 1.5;
    }

    button {
        font-size: var(--text-base);
        font-weight: var(--font-weight-medium);
        line-height: 1.5;
    }

    input {
        font-size: var(--text-base);
        font-weight: var(--font-weight-normal);
        line-height: 1.5;
    }
}

html { font-size: var(--font-size); }

/* Reset e configurações gerais */

• { margin: 0; padding: 0; box-sizing: border-box; }

body { font-family: 'Inter', system-ui, sans-serif; background-color: black; color: white; min-height: 100vh; }

/* Container */ .container { max-width: 96rem; margin: 0 auto; padding: 0 1rem; }

/* Hero Section */ .hero { background-color: #1A1A1A; border-bottom: 4px solid #FF6B00; padding: 4rem 0; }

.hero-content { max-width: 48rem; }

.hero h1 { color: white; text-transform: uppercase; letter-spacing: 0.1em; margin-bottom: 1rem; font-weight: 800; font-size: 3rem; line-height: 1.2; }

.hero .highlight { color: #FF6B00; }
```

```
.hero p { color: #D1D5DB; margin-bottom: 2rem; font-size: 1.125rem; }

.cta-button { background-color: #FF6B00; color: black; padding: 1rem 2rem; text-transform: uppercase; letter-spacing: 0.1em; font-weight: 800; border: none; cursor: pointer; display: inline-block; text-decoration: none; transition: background-color 0.3s; }

.cta-button:hover { background-color: #FFD000; }

/* Features Section */ .features { border-bottom: 2px solid #333333; padding: 3rem 0; }

.features-grid { display: grid; grid-template-columns: 1fr; gap: 1.5rem; }

@media (min-width: 768px) { .features-grid { grid-template-columns: repeat(3, 1fr); } }

.feature-card { background-color: #1A1A1A; border: 2px solid #333333; padding: 1.5rem; display: flex; gap: 1rem; align-items: flex-start; }

.feature-icon { padding: 0.75rem; height: fit-content; }

.feature-icon.orange { background-color: #FF6B00; }

.feature-icon.yellow { background-color: #FFD000; }

.feature-icon svg { width: 2rem; height: 2rem; }

.feature-content h3 { color: white; text-transform: uppercase; letter-spacing: 0.05em; margin-bottom: 0.5rem; font-weight: 700; }

.feature-content p { color: #9CA3AF; }

/* Products Section */ .products { padding: 3rem 0; }

.section-header { margin-bottom: 2rem; }

.section-header h2 { color: white; text-transform: uppercase; letter-spacing: 0.1em; margin-bottom: 0.5rem; font-weight: 800; font-size: 2rem; }

.section-header .highlight { color: #FF6B00; }

.divider { height: 0.25rem; width: 6rem; background-color: #FF6B00; }

.products-grid { display: grid; grid-template-columns: 1fr; gap: 1.5rem; }

@media (min-width: 768px) { .products-grid { grid-template-columns: repeat(2, 1fr); } }

@media (min-width: 1024px) { .products-grid { grid-template-columns: repeat(3, 1fr); } }
```

```
@media (min-width: 1280px) { .products-grid { grid-template-columns: repeat(5, 1fr); } }

/* Product Card - Placeholder */ .product-card { background-color: #1A1A1A; border: 1px solid #333; text-align: center; }

.content { padding: 1rem; }

.content-text { display: flex; flex-direction: column; align-items: start; }

.product-card img { width: 100%; height: auto; margin-bottom: 1rem; }

.product-card h3 { color: white; margin-bottom: 0.5rem; }

.product-card .price { color: #FF6B00; font-weight: bold; font-size: 2rem; margin-bottom: 1rem; } /* Define um tamanho fixo para as imagens, sem distorcer */
.product-card img { width: 100%; height: 320px; } /* ajuste como quiser */
object-fit: cover; transition: transform 0.3s ease; /* animação suave */

/* Zoom ao passar o mouse */
.product-card:hover img { transform: scale(1.05); }

/* Opcional: leve destaque no card */
.product-card { overflow: hidden; } /* evita que o zoom ultrapasse o card */
transition: transform 0.3s ease, box-shadow 0.3s ease;

.product-buttons { display: flex; align-items: center; justify-content: space-between; gap: 0.5rem; }

.product-buttons button { flex: 1; padding: 0.5rem; border: none; cursor: pointer; height: 70px; font-size: 15px; }

.view-btn { background-color: #333; color: white; }

.cart-btn { background-color: #FF6B00; color: black; font-weight: 800; width: 80%; height: 30px; transition: background-color 0.3s; display: flex; align-items: center; justify-content: space-between; }
.cart-btn:hover { background-color: #FF6B00; color: black; font-weight: 800; width: 80%; height: 30px; }

/* Footer Styles */
.footer { background-color: black; border-top: 4px solid #FF6B00; margin-top: 4rem; padding: 3rem 0; }

.footer-grid { display: flex; justify-content: space-between; }

@media (min-width: 768px) { .footer-grid { grid-template-columns: repeat(4, 1fr); } }

.footer-section { display: flex; flex-direction: column; }

.footer-title { color: #FF6B00; text-transform: uppercase; letter-spacing: 0.1em; margin-bottom: 1rem; font-family: 'Inter', system-ui, sans-serif; font-weight: 800; }
```

```
.contact-info { display: flex; flex-direction: column; gap: 0.5rem; color: #D1D5DB; }

.contact-info .business-hours { padding-top: 0.5rem; }

.footer-links { list-style: none; display: flex; flex-direction: column; gap: 0.5rem; }

.footer-link { color: #D1D5DB; text-decoration: none; transition: color 0.3s; }

.footer-link:hover { color: #FF6B00; }

.newsletter-text { color: #D1D5DB; margin-bottom: 1rem; }

.newsletter-form { display: flex; flex-direction: column; gap: 0.5rem; }

.newsletter-input { background-color: #1A1A1A; color: white; padding: 0.5rem 1rem; border: 2px solid #333333; outline: none; transition: border-color 0.3s; }

.newsletter-input:focus { border-color: #FF6B00; }

.newsletter-button { background-color: #FF6B00; color: black; padding: 0.5rem 1.5rem; text-transform: uppercase; letter-spacing: 0.1em; font-family: 'Inter', system-ui, sans-serif; font-weight: 800; border: none; cursor: pointer; transition: background-color 0.3s; }

.newsletter-button:hover { background-color: #FFD000; }

.social-links { display: flex; gap: 1rem; margin-bottom: 1.5rem; }

.social-link { background-color: #1A1A1A; padding: 0.75rem; display: flex; align-items: center; justify-content: center; transition: background-color 0.3s; text-decoration: none; }

.social-link:hover { background-color: #FF6B00; }

.social-icon { width: 1.5rem; height: 1.5rem; color: #FF6B00; transition: color 0.3s; }

.social-link:hover .social-icon { color: black; }

.social-text { color: #9CA3AF; }

.footer-bottom { border-top: 1px solid #333333; margin-top: 2rem; padding-top: 1.5rem; text-align: center; color: #9CA3AF; }

/* Header Styles */
.header { background-color: black; border-bottom: 4px solid #FF6B00; position: sticky; top: 0; z-index: 50; padding: 1rem 0; }

.header-content { display: flex; align-items: center; justify-content: space-between; }
```

```
/* Logo */ .logo-button { background: none; border: none; cursor: pointer; text-transform: uppercase; letter-spacing: 0.1em; font-family: 'Inter', system-ui, sans-serif; font-weight: 800; font-size: 1.75rem; }

.logo-text-white { color: white; }

.logo-text-orange { color: #FF6B00; }

/* Desktop Navigation */ .desktop-nav { display: none; align-items: center; gap: 2rem; }

@media (min-width: 768px) { .desktop-nav { display: flex; } }

.nav-button { background: none; border: none; cursor: pointer; color: white; text-transform: uppercase; letter-spacing: 0.05em; font-family: 'Inter', system-ui, sans-serif; font-weight: 700; transition: color 0.3s; padding: 0.5rem 0; }

.nav-button:hover { color: #FF6B00; }

.nav-button.active { color: #FF6B00; }

/* Cart Button */ .cart-button { position: relative; background: none; border: none; cursor: pointer; padding: 0.5rem; border-radius: 0.25rem; transition: background-color 0.3s; }

.cart-button:hover { background-color: #1A1A1A; }

.cart-btn:hover { background-color: #FFD000; }

.cart-icon { width: 1.75rem; height: 1.75rem; color: #FF6B00; }

.cart-badge { position: absolute; top: -0.25rem; right: -0.25rem; background-color: #FFD000; color: black; border-radius: 50%; width: 1.5rem; height: 1.5rem; display: flex; align-items: center; justify-content: center; font-family: 'Inter', system-ui, sans-serif; font-weight: 800; font-size: 0.75rem; }

.cart-badge.hidden { display: none; }

/* Mobile Menu Button */ .mobile-menu-button { display: block; background: none; border: none; cursor: pointer; color: white; padding: 0.25rem; }

@media (min-width: 768px) { .mobile-menu-button { display: none; } }

.menu-icon { width: 1.5rem; height: 1.5rem; }
```

Banco de Dados

```
-- ===== -- TireGrip E-commerce
Database -- Sistema de venda de pneus automotivos --
=====

-- Criar banco de dados DROP DATABASE IF EXISTS TireGrip; CREATE DATABASE TireGrip
CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci; USE TireGrip;

-- ===== -- Tabela de Usuários --
CREATE TABLE usuarios ( pk_id
INT AUTO_INCREMENT PRIMARY KEY, nome VARCHAR(100) NOT NULL, email
VARCHAR(100) NOT NULL UNIQUE, senha VARCHAR(100) NOT NULL, data_cadastro
TIMESTAMP DEFAULT CURRENT_TIMESTAMP, INDEX idx_email (email) ) ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ===== -- Tabela de Produtos (Pneus) --
CREATE TABLE produtos ( pk_id
INT AUTO_INCREMENT PRIMARY KEY, nome VARCHAR(100) NOT NULL, modelo
VARCHAR(100) NOT NULL, preco DECIMAL(10,2) NOT NULL, descricao TEXT, largura
VARCHAR(20), perfil VARCHAR(20), aro VARCHAR(20), indice_carga VARCHAR(20),
indice_velocidade VARCHAR(20), estoque INT DEFAULT 0, created_at TIMESTAMP
DEFAULT CURRENT_TIMESTAMP, INDEX idx_nome (nome), INDEX idx_preco (preco) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ===== -- Tabela de Carrinho --
CREATE TABLE carrinho ( pk_id
INT AUTO_INCREMENT PRIMARY KEY, usuario_id INT NOT NULL, produto_id INT NOT
NULL, quantidade INT NOT NULL DEFAULT 1, subtotal DECIMAL(10,2) NOT NULL,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, FOREIGN KEY (usuario_id)
REFERENCES usuarios(pk_id) ON DELETE CASCADE, FOREIGN KEY (produto_id)
REFERENCES produtos(pk_id) ON DELETE CASCADE, INDEX idx_usuario (usuario_id),
UNIQUE KEY unique_user_product (usuario_id, produto_id) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ===== -- Tabela de Endereços --
CREATE TABLE enderecos ( pk_id
INT AUTO_INCREMENT PRIMARY KEY, usuario_id INT NOT NULL, cep VARCHAR(10) NOT
NULL, rua VARCHAR(200) NOT NULL, numero VARCHAR(20) NOT NULL, complemento
VARCHAR(100), bairro VARCHAR(100) NOT NULL, cidade VARCHAR(100) NOT NULL,
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP, FOREIGN KEY (usuario_id)
REFERENCES usuarios(pk_id) ON DELETE CASCADE, INDEX idx_usuario (usuario_id) )
ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

```

-- ====== -- Tabela de Pedidos --
===== CREATE TABLE pedidos ( pk_id
INT AUTO_INCREMENT PRIMARY KEY, usuario_id INT NOT NULL, endereco_id INT NOT
NULL, data_pedido TIMESTAMP DEFAULT CURRENT_TIMESTAMP, subtotal DECIMAL(10,2)
NOT NULL, frete DECIMAL(10,2) NOT NULL DEFAULT 50.00, desconto DECIMAL(10,2)
DEFAULT 0.00, total DECIMAL(10,2) NOT NULL, forma_pagamento VARCHAR(20) NOT
NULL, status VARCHAR(20) DEFAULT 'PENDENTE', FOREIGN KEY (usuario_id)
REFERENCES usuarios(pk_id) ON DELETE RESTRICT, FOREIGN KEY (endereco_id)
REFERENCES enderecos(pk_id) ON DELETE RESTRICT, INDEX idx_usuario (usuario_id),
INDEX idx_status (status), INDEX idx_data (data_pedido) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

-- ====== -- Tabela de Itens do Pedido --
===== CREATE TABLE itens_pedido (
pk_id INT AUTO_INCREMENT PRIMARY KEY, pedido_id INT NOT NULL, produto_id INT NOT
NULL, quantidade INT NOT NULL, preco_unitario DECIMAL(10,2) NOT NULL, subtotal
DECIMAL(10,2) NOT NULL, FOREIGN KEY (pedido_id) REFERENCES pedidos(pk_id) ON
DELETE CASCADE, FOREIGN KEY (produto_id) REFERENCES produtos(pk_id) ON DELETE
RESTRICT, INDEX idx_pedido (pedido_id) ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;

-- ====== -- DADOS DE TESTE --
=====

-- Inserir usuários de teste INSERT INTO usuarios (nome, email, senha) VALUES ('João
Silva', 'joao@teste.com', '123456'), ('Maria Santos', 'maria@teste.com', '123456'), ('Admin
TireGrip', 'admin@tiregrip.com', 'admin123');

-- Inserir produtos de teste (Pneus) INSERT INTO produtos (nome, modelo, preco,
descricao, largura, aro, indice_carga, indice_velocidade, estoque) VALUES ('PNEU
PERFORMANCE MAX', 'Modelo Sport 2024 - Alta Performance', 599.99, 'Pneu de alta
performance com tecnologia de drenagem de água avançada, composto de borracha de alta
aderência e baixo nível de ruído durante a condução.', '205', '55', '16"', '91', 'V (240 km/h)',
50),
('PNEU OFF-ROAD PRO', 'Modelo Adventure Pro', 699.99, 'Pneu para terrenos difíceis com
maior durabilidade e resistência ao desgaste. Ideal para aventuras off-road.', '215', '65',
'17"', '95', 'H (210 km/h)', 30),
('PNEU ECONÔMICO PLUS', 'Modelo Eco 2024', 399.99, 'Pneu com economia de combustível
otimizada e desempenho em todas as estações. Excelente custo-benefício.', '185', '60', '15"',
'88', 'T (190 km/h)', 80),

```

('PNEU PREMIUM COMFORT', 'Modelo Luxury Ride', 799.99, 'Pneu premium com máximo conforto e silêncio. Tecnologia de absorção de impactos para viagens longas.', '225', '50', '18"', '98', 'W (270 km/h)', 25),

('PNEU ALL-TERRAIN', 'Modelo Multi-Surface', 649.99, 'Pneu versátil para uso urbano e off-road leve. Aderência superior em diferentes superfícies.', '205', '70', '16"', '92', 'S (180 km/h)', 40),

('PNEU SPORT RACING', 'Modelo Track Pro', 899.99, 'Pneu de alta performance para uso esportivo. Aderência máxima em curvas e frenagens.', '245', '40', '19"', '100', 'Y (300 km/h)', 15);

-- Inserir endereços de teste INSERT INTO enderecos (usuario_id, cep, rua, numero, complemento, bairro, cidade) VALUES (1, '01310-100', 'Avenida Paulista', '1578', 'Apto 101', 'Bela Vista', 'São Paulo'), (1, '04543-907', 'Avenida Brigadeiro Faria Lima', '2232', NULL, 'Jardim Paulistano', 'São Paulo'), (2, '22640-102', 'Avenida Vieira Souto', '500', 'Cobertura', 'Ipanema', 'Rio de Janeiro');

-- Inserir alguns itens no carrinho de teste INSERT INTO carrinho (usuario_id, produto_id, quantidade, subtotal) VALUES (1, 1, 2, 1199.98), (1, 3, 1, 399.99);

-- Inserir pedido de teste INSERT INTO pedidos (usuario_id, endereco_id, subtotal, frete, desconto, total, forma_pagamento, status) VALUES (2, 3, 1299.98, 50.00, 67.50, 1282.48, 'PIX', 'CONFIRMADO');

-- Inserir itens do pedido de teste INSERT INTO itens_pedido (pedido_id, produto_id, quantidade, preco_unitario, subtotal) VALUES (1, 2, 1, 699.99, 699.99), (1, 5, 1, 649.99, 649.99);

-- ===== -- VIEWS ÚTEIS (Opcional) --
=====

-- View para visualizar carrinho com detalhes do produto CREATE VIEW vw_carrinho_detalhado AS SELECT c.pk_id, c.usuario_id, u.nome as usuario_nome, c.produto_id, p.nome as produto_nome, p.modelo as produto_modelo, p.preco as produto_preco, c.quantidade, c.subtotal FROM carrinho c INNER JOIN usuarios u ON c.usuario_id = u.pk_id INNER JOIN produtos p ON c.produto_id = p.pk_id;

-- View para visualizar pedidos com detalhes CREATE VIEW vw_pedidos_detalhados AS SELECT ped.pk_id, ped.usuario_id, u.nome as usuario_nome, u.email as usuario_email, ped.data_pedido, ped.subtotal, ped.frete, ped.desconto, ped.total, ped.forma_pagamento, ped.status, e.rua, e.numero, e.bairro, e.cidade, e.cep FROM pedidos ped INNER JOIN usuarios u ON ped.usuario_id = u.pk_id INNER JOIN enderecos e ON ped.endereco_id = e.pk_id;

```
-- ====== CONSULTAS ÚTEIS PARA
TESTES -- ======
-- Verificar todos os usuários -- SELECT * FROM usuarios;
-- Verificar todos os produtos -- SELECT * FROM produtos ORDER BY preco;
-- Verificar carrinho de um usuário -- SELECT * FROM vw_carrinho_detalhado WHERE
usuario_id = 1;
-- Verificar pedidos de um usuário -- SELECT * FROM vw_pedidos_detalhados WHERE
usuario_id = 2;
-- Verificar itens de um pedido -- SELECT ip.*, p.nome, p.modelo -- FROM itens_pedido ip --
INNER JOIN produtos p ON ip.producto_id = p.pk_id -- WHERE ip.pedido_id = 1;
-- ====== FIM DO SCRIPT --
=====
```

Back-End

conectadb package config;

```
import java.sql.*;
public class ConectaDB {
    private static final String DB_URL = "jdbc:mysql:
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "P@ssw0rd";
    private static final String DB_DRIVER = "com.mysql.cj.jdbc.Driver";

    public static Connection conectar() throws ClassNotFoundException {
        Connection conn = null;
        try {
            Class.forName(DB_DRIVER);
            conn = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
```

```
    } catch (SQLException ex) {

        System.out.println("Erro - SQL: " + ex.getMessage());
        ex.printStackTrace();
    }

    return conn;
}

public static void fecharConexao(Connection conn) {
    if (conn != null) {
        try {
            conn.close();

        } catch (SQLException ex) {
            System.out.println("Erro ao fechar conexão: " +
ex.getMessage());
        }
    }
}

public static void fecharStatement(Statement stmt) {
    if (stmt != null) {
        try {
            stmt.close();

        } catch (SQLException ex) {
            System.out.println("Erro ao fechar statement: " +
ex.getMessage());
        }
    }
}

public static void fecharResultSet(ResultSet rs) {
    if (rs != null) {
        try {
            rs.close();
        }
    }
}
```

```

        } catch (SQLException ex) {
            System.out.println("Erro ao fechar resultset: " +
ex.getMessage());
        }
    }

}

package model.DAO;

import java.sql.*; import java.util.ArrayList; import java.util.List; import model.Carrinho;
import config.ConectaDB;

/**

```

- Classe DAO para operações do Carrinho de Compras
-
- @author TireGrip Team
- @version 1.0

```

*/ public class CarrinhoDAO {

public boolean adicionar(Carrinho obj) throws ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "INSERT INTO carrinho (usuario_id, produto_id,
quantidade, subtotal) VALUES (?, ?, ?, ?)";
        pstmt = conn.prepareStatement(sql);

        pstmt.setInt(1, obj.getUsuarioId());
        pstmt.setInt(2, obj.getProdutoId());
        pstmt.setInt(3, obj.getQuantidade());

```

```

        pstmt.setFloat(4, obj.getSubtotal()));

        pstmt.executeUpdate();
        return true;

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return false;
    } finally {
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

public List<Carrinho> consulta_por_usuario(int usuarioId) throws
ClassNotFoundException {
    List<Carrinho> lst = new ArrayList<>();
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "SELECT c.*, p.nome as produto_nome, p.modelo as
produto_modelo, p.preco as produto_preco " +
                    "FROM carrinho c " +
                    "INNER JOIN produtos p ON c.produto_id = p.pk_id "
+
                    "WHERE c.usuario_id = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, usuarioId);

        rs = pstmt.executeQuery();

        int n_reg = 0;
        while (rs.next()) {
            Carrinho obj = new Carrinho();
            obj.setId(rs.getInt("pk_id"));
            obj.setUsuarioId(rs.getInt("usuario_id"));
            obj.setProdutoId(rs.getInt("produto_id"));
            obj.setQuantidade(rs.getInt("quantidade"));
            obj.setSubtotal(rs.getFloat("subtotal"));
            obj.setProdutoNome(rs.getString("produto_nome"));
    }
}

```

```

        obj.setProdutoModelo(rs.getString("produto_modelo"));
        obj.setProdutoPreco(rs.getFloat("produto_preco"));

        lst.add(obj);
        n_reg++;
    }

    return (n_reg == 0) ? null : lst;
}

} catch (SQLException ex) {
    System.out.println("Erro - SQL: " + ex.getMessage());
    return null;
} finally {
    ConectaDB.fecharResultSet(rs);
    ConectaDB.fecharStatement(pstmt);
    ConectaDB.fecharConexao(conn);
}
}

public Carrinho consulta_item(int usuarioId, int produtoId) throws
ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "SELECT * FROM carrinho WHERE usuario_id = ? AND
produto_id = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, usuarioId);
        pstmt.setInt(2, produtoId);

        rs = pstmt.executeQuery();

        if (rs.next()) {
            Carrinho obj = new Carrinho();
            obj.setId(rs.getInt("pk_id"));
            obj.setUsuarioId(rs.getInt("usuario_id"));
            obj.setProdutoId(rs.getInt("produto_id"));
            obj.setQuantidade(rs.getInt("quantidade"));
            obj.setSubtotal(rs.getFloat("subtotal"));
            return obj;
        }
    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return null;
    } finally {
        ConectaDB.fecharResultSet(rs);
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}
}
```

```

        } else {
            return null;
        }

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return null;
    } finally {
        ConectaDB.fecharResultSet(rs);
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

public boolean atualizar_quantidade(int id, int quantidade, float subtotal) throws ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "UPDATE carrinho SET quantidade = ?, subtotal = ? WHERE pk_id = ?";
        pstmt = conn.prepareStatement(sql);

        pstmt.setInt(1, quantidade);
        pstmt.setFloat(2, subtotal);
        pstmt.setInt(3, id);

        pstmt.executeUpdate();
        return true;

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return false;
    } finally {
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

public boolean remover(int id) throws ClassNotFoundException {
    Connection conn = null;

```

```
PreparedStatement pstmt = null;

try {
    conn = ConectaDB.conectar();
    String sql = "DELETE FROM carrinho WHERE pk_id = ?";
    pstmt = conn.prepareStatement(sql);
    pstmt.setInt(1, id);

    int result = pstmt.executeUpdate();
    return (result > 0);

} catch (SQLException ex) {
    System.out.println("Erro - SQL: " + ex.getMessage());
    return false;
} finally {
    ConectaDB.fecharStatement(pstmt);
    ConectaDB.fecharConexao(conn);
}
}

public boolean limpar_carrinho(int usuarioId) throws
ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "DELETE FROM carrinho WHERE usuario_id = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, usuarioId);

        pstmt.executeUpdate();
        return true;

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return false;
    } finally {
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}
```

```
public float calcular_subtotal(int usuarioId) throws
ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "SELECT SUM(subtotal) as total FROM carrinho WHERE
usuario_id = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, usuarioId);

        rs = pstmt.executeQuery();

        if (rs.next()) {
            return rs.getFloat("total");
        } else {
            return 0.0f;
        }
    }

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return 0.0f;
    } finally {
        ConectaDB.fecharResultSet(rs);
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

}
```

```
package model.DAO;
```

```
import java.sql.*; import java.util.ArrayList; import java.util.List; import model.Endereco;
import config.ConectaDB;

/***
• Classe DAO para operações CRUD da entidade Endereco
•
• @author TireGrip Team
• @version 1.0

*/ public class EnderecoDAO {

    public boolean cadastrar(Endereco obj) throws ClassNotFoundException {
        Connection conn = null;
        PreparedStatement pstmt = null;

        try {
            conn = ConectaDB.conectar();
            String sql = "INSERT INTO enderecos (usuario_id, cep, rua,
numero, complemento, bairro, cidade) VALUES (?, ?, ?, ?, ?, ?, ?)";
            pstmt = conn.prepareStatement(sql);

            pstmt.setInt(1, obj.getUsuarioId());
            pstmt.setString(2, obj.getCep());
            pstmt.setString(3, obj.getRua());
            pstmt.setString(4, obj.getNumero());
            pstmt.setString(5, obj.getComplemento());
            pstmt.setString(6, obj.getBairro());
            pstmt.setString(7, obj.getCidade());

            pstmt.executeUpdate();
            return true;

        } catch (SQLException ex) {
            System.out.println("Erro - SQL: " + ex.getMessage());
            return false;
        } finally {
            ConectaDB.fecharStatement(pstmt);
            ConectaDB.fecharConexao(conn);
        }
    }

    public List<Endereco> consulta_por_usuario(int usuarioId) throws
ClassNotFoundException {
```

```

List<Endereco> lst = new ArrayList<>();
Connection conn = null;
PreparedStatement pstmt = null;
ResultSet rs = null;

try {
    conn = ConectaDB.conectar();
    String sql = "SELECT * FROM enderecos WHERE usuario_id = ?"
    ORDER BY pk_id DESC";
    pstmt = conn.prepareStatement(sql);
    pstmt.setInt(1, usuarioId);

    rs = pstmt.executeQuery();

    int n_reg = 0;
    while (rs.next()) {
        Endereco obj = new Endereco();
        obj.setId(rs.getInt("pk_id"));
        obj.setUsuarioId(rs.getInt("usuario_id"));
        obj.setCep(rs.getString("cep"));
        obj.setRua(rs.getString("rua"));
        obj.setNumero(rs.getString("numero"));
        obj.setComplemento(rs.getString("complemento"));
        obj.setBairro(rs.getString("bairro"));
        obj.setCidade(rs.getString("cidade"));

        lst.add(obj);
        n_reg++;
    }

    return (n_reg == 0) ? null : lst;
}

} catch (SQLException ex) {
    System.out.println("Erro - SQL: " + ex.getMessage());
    return null;
} finally {
    ConectaDB.fecharResultSet(rs);
    ConectaDB.fecharStatement(pstmt);
    ConectaDB.fecharConexao(conn);
}
}

public Endereco consulta_id(Endereco obj) throws ClassNotFoundException

```

```

{
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "SELECT * FROM enderecos WHERE pk_id = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, obj.getId());

        rs = pstmt.executeQuery();

        if (rs.next()) {
            obj.setId(rs.getInt("pk_id"));
            obj.setUsuarioId(rs.getInt("usuario_id"));
            obj.setCep(rs.getString("cep"));
            obj.setRua(rs.getString("rua"));
            obj.setNumero(rs.getString("numero"));
            obj.setComplemento(rs.getString("complemento"));
            obj.setBairro(rs.getString("bairro"));
            obj.setCidade(rs.getString("cidade"));
            return obj;
        } else {
            return null;
        }
    }

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return null;
    } finally {
        ConectaDB.fecharResultSet(rs);
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

public boolean alterar(Endereco obj) throws ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConectaDB.conectar();

```

```

        String sql = "UPDATE enderecos SET cep = ?, rua = ?, numero = ?,
        complemento = ?, bairro = ?, cidade = ? WHERE pk_id = ?";
        pstmt = conn.prepareStatement(sql);

        pstmt.setString(1, obj.getCep());
        pstmt.setString(2, obj.getRua());
        pstmt.setString(3, obj.getNumero());
        pstmt.setString(4, obj.getComplemento());
        pstmt.setString(5, obj.getBairro());
        pstmt.setString(6, obj.getCidade());
        pstmt.setInt(7, obj.getId());

        pstmt.executeUpdate();
        return true;

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return false;
    } finally {
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

public boolean excluir(int id) throws ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "DELETE FROM enderecos WHERE pk_id = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, id);

        int result = pstmt.executeUpdate();
        return (result > 0);

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return false;
    } finally {
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

```

```
        }
    }

}

package model.DAO;

import java.sql.*; import java.util.ArrayList; import java.util.List; import model.ItemPedido;
import config.ConectaDB;

/**
 * 
 * • Classe DAO para operações da entidade ItemPedido
 * • 
 * • @author TireGrip Team
 * • @version 1.0
 */

public class ItemPedidoDAO {

    public boolean cadastrar(ItemPedido obj) throws ClassNotFoundException
    {
        Connection conn = null;
        PreparedStatement pstmt = null;

        try {
            conn = ConectaDB.conectar();
            String sql = "INSERT INTO itens_pedido (pedido_id, produto_id,
quantidade, preco_unitario, subtotal) VALUES (?, ?, ?, ?, ?)";
            pstmt = conn.prepareStatement(sql);

            pstmt.setInt(1, obj.getPedidoId());
            pstmt.setInt(2, obj.getProdutoId());
            pstmt.setInt(3, obj.getQuantidade());
        }
    }
}
```

```

        pstmt.setFloat(4, obj.getPrecoUnitario());
        pstmt.setFloat(5, obj.getSubtotal());

        pstmt.executeUpdate();
        return true;

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return false;
    } finally {
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

public List<ItemPedido> consulta_por_pedido(int pedidoId) throws
ClassNotFoundException {
    List<ItemPedido> lst = new ArrayList<>();
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "SELECT ip.*, p.nome as produto_nome, p.modelo as
produto_modelo " +
                    "FROM itens_pedido ip " +
                    "INNER JOIN produtos p ON ip.produto_id = p.pk_id " +
                    "WHERE ip.pedido_id = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, pedidoId);

        rs = pstmt.executeQuery();

        int n_reg = 0;
        while (rs.next()) {
            ItemPedido obj = new ItemPedido();
            obj.setId(rs.getInt("pk_id"));
            obj.setPedidoId(rs.getInt("pedido_id"));
            obj.setProdutoId(rs.getInt("produto_id"));
            obj.setQuantidade(rs.getInt("quantidade"));
            obj.setPrecoUnitario(rs.getFloat("preco_unitario"));
    }
}

```

```

        obj.setSubtotal(rs.getFloat("subtotal"));
        obj.setProdutoNome(rs.getString("produto_nome"));
        obj.setProdutoModelo(rs.getString("produto_modelo"));

        lst.add(obj);
        n_reg++;
    }

    return (n_reg == 0) ? null : lst;
}

} catch (SQLException ex) {
    System.out.println("Erro - SQL: " + ex.getMessage());
    return null;
} finally {
    ConectaDB.fecharResultSet(rs);
    ConectaDB.fecharStatement(pstmt);
    ConectaDB.fecharConexao(conn);
}
}

public boolean cadastrar_lote(List<ItemPedido> itens) throws
ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "INSERT INTO itens_pedido (pedido_id, produto_id,
quantidade, preco_unitario, subtotal) VALUES (?, ?, ?, ?, ?)";
        pstmt = conn.prepareStatement(sql);

        for (ItemPedido obj : itens) {
            pstmt.setInt(1, obj.getPedidoId());
            pstmt.setInt(2, obj.getProdutoId());
            pstmt.setInt(3, obj.getQuantidade());
            pstmt.setFloat(4, obj.getPrecoUnitario());
            pstmt.setFloat(5, obj.getSubtotal());
            pstmt.addBatch();
        }

        pstmt.executeBatch();
        return true;
    }
}
```

```

        } catch (SQLException ex) {
            System.out.println("Erro - SQL: " + ex.getMessage());
            return false;
        } finally {
            ConectaDB.fecharStatement(pstmt);
            ConectaDB.fecharConexao(conn);
        }
    }

}

package model.DAO;

import java.sql.*; import java.util.ArrayList; import java.util.List; import model.Pedido;
import config.ConectaDB;

/**
 * 
 * • Classe DAO para operações da entidade Pedido
 * • 
 * • @author TireGrip Team
 * • @version 1.0
 */
public class PedidoDAO {

    public int cadastrar(Pedido obj) throws ClassNotFoundException {
        Connection conn = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;

        try {
            conn = ConectaDB.conectar();
            String sql = "INSERT INTO pedidos (usuario_id, endereco_id, subtotal, frete, desconto, total, forma_pagamento, status) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
            pstmt = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);

            pstmt.setInt(1, obj.getUsuarioId());
            pstmt.setInt(2, obj.getEnderecoId());
            pstmt.setFloat(3, obj.getSubtotal());
            pstmt.setFloat(4, obj.getFrete());
        }
    }
}

```

```

        pstmt.setFloat(5, obj.getDesconto());
        pstmt.setFloat(6, obj.getTotal());
        pstmt.setString(7, obj.getFormaPagamento());
        pstmt.setString(8, obj.getStatus());

        pstmt.executeUpdate();

        rs = pstmt.getGeneratedKeys();
        if (rs.next()) {
            return rs.getInt(1);
        } else {
            return 0;
        }

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return 0;
    } finally {
        ConectaDB.fecharResultSet(rs);
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

public List<Pedido> consulta_por_usuario(int usuarioId) throws
ClassNotFoundException {
    List<Pedido> lst = new ArrayList<>();
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "SELECT p.*, u.nome as usuario_nome, " +
                    "CONCAT(e.rua, ', ', e.numero, ' - ', e.bairro, ' - "
        ', e.cidade) as endereco_completo " +
                    "FROM pedidos p " +
                    "INNER JOIN usuarios u ON p.usuario_id = u.pk_id "
        +
                    "INNER JOIN enderecos e ON p.endereco_id = e.pk_id
        " +
                    "WHERE p.usuario_id = ? " +

```

```

        "ORDER BY p.data_pedido DESC";
pstmt = conn.prepareStatement(sql);
pstmt.setInt(1, usuarioId);

rs = pstmt.executeQuery();

int n_reg = 0;
while (rs.next()) {
    Pedido obj = new Pedido();
    obj.setId(rs.getInt("pk_id"));
    obj.setUsuarioId(rs.getInt("usuario_id"));
    obj.setEnderecoId(rs.getInt("endereco_id"));
    obj.setDataPedido(rs.getTimestamp("data_pedido"));
    obj.setSubtotal(rs.getFloat("subtotal"));
    obj.setFrete(rs.getFloat("frete"));
    obj.setDesconto(rs.getFloat("desconto"));
    obj.setTotal(rs.getFloat("total"));
    obj.setFormaPagamento(rs.getString("forma_pagamento"));
    obj.setStatus(rs.getString("status"));
    obj.setUsuarioNome(rs.getString("usuario_nome"));
    obj.setEnderecoCompleto(rs.getString("endereco_completo"));

    lst.add(obj);
    n_reg++;
}

return (n_reg == 0) ? null : lst;

} catch (SQLException ex) {
    System.out.println("Erro - SQL: " + ex.getMessage());
    return null;
} finally {
    ConectaDB.fecharResultSet(rs);
    ConectaDB.fecharStatement(pstmt);
    ConectaDB.fecharConexao(conn);
}
}

public Pedido consulta_id(Pedido obj) throws ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

```

```

try {
    conn = ConectaDB.conectar();
    String sql = "SELECT p.*, u.nome as usuario_nome, " +
        "CONCAT(e.rua, ', ', e.numero, ' - ', e.bairro, ' - "
    ', e.cidade) as endereco_completo " +
        "FROM pedidos p " +
        "INNER JOIN usuarios u ON p.usuario_id = u.pk_id "
+
        "INNER JOIN enderecos e ON p.endereco_id = e.pk_id
" +
        "WHERE p.pk_id = ?";
    pstmt = conn.prepareStatement(sql);
    pstmt.setInt(1, obj.getId());

    rs = pstmt.executeQuery();

    if (rs.next()) {
        obj.setId(rs.getInt("pk_id"));
        obj.setUsuarioId(rs.getInt("usuario_id"));
        obj.setEnderecoId(rs.getInt("endereco_id"));
        obj.setDataPedido(rs.getTimestamp("data_pedido"));
        obj.setSubtotal(rs.getFloat("subtotal"));
        obj.setFrete(rs.getFloat("frete"));
        obj.setDesconto(rs.getFloat("desconto"));
        obj.setTotal(rs.getFloat("total"));
        obj.setFormaPagamento(rs.getString("forma_pagamento"));
        obj.setStatus(rs.getString("status"));
        obj.setUsuarioNome(rs.getString("usuario_nome"));
        obj.setEnderecoCompleto(rs.getString("endereco_completo"));
        return obj;
    } else {
        return null;
    }

} catch (SQLException ex) {
    System.out.println("Erro - SQL: " + ex.getMessage());
    return null;
} finally {
    ConectaDB.fecharResultSet(rs);
    ConectaDB.fecharStatement(pstmt);
    ConectaDB.fecharConexao(conn);
}
}

```

```

public boolean atualizar_status(int pedidoId, String status) throws
ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "UPDATE pedidos SET status = ? WHERE pk_id = ?";
        pstmt = conn.prepareStatement(sql);

        pstmt.setString(1, status);
        pstmt.setInt(2, pedidoId);

        pstmt.executeUpdate();
        return true;

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return false;
    } finally {
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

public List<Pedido> consulta_geral() throws ClassNotFoundException {
    List<Pedido> lst = new ArrayList<>();
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;

    try {
        conn = ConectaDB.conectar();
        stmt = conn.createStatement();
        String sql = "SELECT p.*, u.nome as usuario_nome, " +
                    "CONCAT(e.rua, ', ', e.numero, ' - ', e.bairro, ' - ',
                    e.cidade) as endereco_completo " +
                    "FROM pedidos p " +
                    "INNER JOIN usuarios u ON p.usuario_id = u.pk_id "
+
                    "INNER JOIN enderecos e ON p.endereco_id = e.pk_id
" +

```

```

        "ORDER BY p.data_pedido DESC";
rs = stmt.executeQuery(sql);

int n_reg = 0;
while (rs.next()) {
    Pedido obj = new Pedido();
    obj.setId(rs.getInt("pk_id"));
    obj.setUsuarioId(rs.getInt("usuario_id"));
    obj.setEnderecoId(rs.getInt("endereco_id"));
    obj.setDataPedido(rs.getTimestamp("data_pedido"));
    obj.setSubtotal(rs.getFloat("subtotal"));
    obj.setFrete(rs.getFloat("frete"));
    obj.setDesconto(rs.getFloat("desconto"));
    obj.setTotal(rs.getFloat("total"));
    obj.setFormaPagamento(rs.getString("forma_pagamento"));
    obj.setStatus(rs.getString("status"));
    obj.setUsuarioNome(rs.getString("usuario_nome"));
    obj.setEnderecoCompleto(rs.getString("endereco_completo"));

    lst.add(obj);
    n_reg++;
}

return (n_reg == 0) ? null : lst;

} catch (SQLException ex) {
    System.out.println("Erro - SQL: " + ex.getMessage());
    return null;
} finally {
    ConectaDB.fecharResultSet(rs);
    ConectaDB.fecharStatement(stmt);
    ConectaDB.fecharConexao(conn);
}
}

}
```

```
package model.DAO;

import java.sql.*; import java.util.ArrayList; import java.util.List; import model.Produto;
import config.ConectaDB;

/**
 * 
 * • Classe DAO para operações CRUD da entidade Produto
 * •
 * • @author TireGrip Team
 * • @version 1.0
 */

*/ public class ProdutoDAO {

    public boolean cadastrar(Produto obj) throws ClassNotFoundException {
        Connection conn = null;
        PreparedStatement pstmt = null;

        try {
            conn = ConectaDB.conectar();
            String sql = "INSERT INTO produtos (nome, modelo, preco,
descricao, largura, perfil, aro, indice_carga, indice_velocidade,
```

```

estoque) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
pstmt = conn.prepareStatement(sql);

pstmt.setString(1, obj.getNome());
pstmt.setString(2, obj.getModelo());
pstmt.setFloat(3, obj.getPreco());
pstmt.setString(4, obj.getDescricao());
pstmt.setString(5, obj.getLargura());
pstmt.setString(6, obj.getPerfil());
pstmt.setString(7, obj.getAro());
pstmt.setString(8, obj.getIndiceCarga());
pstmt.setString(9, obj.getIndiceVelocidade());
pstmt.setInt(10, obj.getEstoque());

pstmt.executeUpdate();
return true;

} catch (SQLException ex) {
System.out.println("Erro - SQL: " + ex.getMessage());
return false;
} finally {
    ConectaDB.fecharStatement(pstmt);
    ConectaDB.fecharConexao(conn);
}
}

public List<Produto> consulta_geral() throws ClassNotFoundException {
List<Produto> lst = new ArrayList<>();
Connection conn = null;
Statement stmt = null;
ResultSet rs = null;

try {
    conn = ConectaDB.conectar();
    stmt = conn.createStatement();
    String sql = "SELECT * FROM produtos ORDER BY pk_id ASC";
    rs = stmt.executeQuery(sql);

    int n_reg = 0;
    while (rs.next()) {
        Produto obj = new Produto();
        obj.setId(rs.getInt("pk_id"));
        obj.setNome(rs.getString("nome"));

```

```

        obj.setModelo(rs.getString("modelo"));
        obj.setPreco(rs.getFloat("preco"));
        obj.setDescricao(rs.getString("descricao"));
        obj.setLargura(rs.getString("largura"));
        obj.setPerfil(rs.getString("perfil"));
        obj.setAro(rs.getString("aro"));
        obj.setIndiceCarga(rs.getString("indice_carga"));
        obj.setIndiceVelocidade(rs.getString("indice_velocidade"));
        obj.setEstoque(rs.getInt("estoque"));

        lst.add(obj);
        n_reg++;
    }

    return (n_reg == 0) ? null : lst;
}

} catch (SQLException ex) {
    System.out.println("Erro - SQL: " + ex.getMessage());
    return null;
} finally {
    ConectaDB.fecharResultSet(rs);
    ConectaDB.fecharStatement(stmt);
    ConectaDB.fecharConexao(conn);
}
}

public Produto consulta_id(Produto obj) throws ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "SELECT * FROM produtos WHERE pk_id = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, obj.getId());

        rs = pstmt.executeQuery();

        if (rs.next()) {
            obj.setId(rs.getInt("pk_id"));
            obj.setNome(rs.getString("nome"));
            obj.setModelo(rs.getString("modelo"));

```

```

        obj.setPreco(rs.getFloat("preco"));
        obj.setDescricao(rs.getString("descricao"));
        obj.setLargura(rs.getString("largura"));
        obj.setPerfil(rs.getString("perfil"));
        obj.setAro(rs.getString("aro"));
        obj.setIndiceCarga(rs.getString("indice_carga"));
        obj.setIndiceVelocidade(rs.getString("indice_velocidade"));
        obj.setEstoque(rs.getInt("estoque"));
        return obj;
    } else {
        return null;
    }
}

} catch (SQLException ex) {
    System.out.println("Erro - SQL: " + ex.getMessage());
    return null;
} finally {
    ConectaDB.fecharResultSet(rs);
    ConectaDB.fecharStatement(pstmt);
    ConectaDB.fecharConexao(conn);
}
}

public List<Produto> consulta_nome(String nome) throws
ClassNotFoundException {
    List<Produto> lst = new ArrayList<>();
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "SELECT * FROM produtos WHERE nome LIKE ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, "%" + nome + "%");

        rs = pstmt.executeQuery();

        int n_reg = 0;
        while (rs.next()) {
            Produto obj = new Produto();
            obj.setId(rs.getInt("pk_id"));
            obj.setNome(rs.getString("nome"));

```

```

        obj.setModelo(rs.getString("modelo"));
        obj.setPreco(rs.getFloat("preco"));
        obj.setDescricao(rs.getString("descricao"));
        obj.setLargura(rs.getString("largura"));
        obj.setPerfil(rs.getString("perfil"));
        obj.setAro(rs.getString("aro"));
        obj.setIndiceCarga(rs.getString("indice_carga"));
        obj.setIndiceVelocidade(rs.getString("indice_velocidade"));
        obj.setEstoque(rs.getInt("estoque"));

        lst.add(obj);
        n_reg++;
    }

    return (n_reg == 0) ? null : lst;
}

} catch (SQLException ex) {
    System.out.println("Erro - SQL: " + ex.getMessage());
    return null;
} finally {
    ConectaDB.fecharResultSet(rs);
    ConectaDB.fecharStatement(pstmt);
    ConectaDB.fecharConexao(conn);
}
}

public boolean alterar(Produto obj) throws ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "UPDATE produtos SET nome = ?, modelo = ?, preco = ?,
        descricao = ?, largura = ?, perfil = ?, aro = ?, indice_carga = ?,
        indice_velocidade = ?, estoque = ? WHERE pk_id = ?";
        pstmt = conn.prepareStatement(sql);

        pstmt.setString(1, obj.getNome());
        pstmt.setString(2, obj.getModelo());
        pstmt.setFloat(3, obj.getPreco());
        pstmt.setString(4, obj.getDescricao());
        pstmt.setString(5, obj.getLargura());
        pstmt.setString(6, obj.getPerfil());
    }
}

```

```

        pstmt.setString(7, obj.getAro());
        pstmt.setString(8, obj.getIndiceCarga());
        pstmt.setString(9, obj.getIndiceVelocidade());
        pstmt.setInt(10, obj.getEstoque());
        pstmt.setInt(11, obj.getId());

        pstmt.executeUpdate();
        return true;

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return false;
    } finally {
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

public boolean excluir(int id) throws ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "DELETE FROM produtos WHERE pk_id = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, id);

        int result = pstmt.executeUpdate();
        return (result > 0);

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return false;
    } finally {
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

public boolean atualizarEstoque(int produtoId, int quantidade) throws
ClassNotFoundException {
    Connection conn = null;

```

```
PreparedStatement pstmt = null;

try {
    conn = ConectaDB.conectar();
    String sql = "UPDATE produtos SET estoque = estoque - ? WHERE
pk_id = ?";
    pstmt = conn.prepareStatement(sql);
    pstmt.setInt(1, quantidade);
    pstmt.setInt(2, produtoId);

    pstmt.executeUpdate();
    return true;

} catch (SQLException ex) {
    System.out.println("Erro - SQL: " + ex.getMessage());
    return false;
} finally {
    ConectaDB.fecharStatement(pstmt);
    ConectaDB.fecharConexao(conn);
}
}

}
```

```
package model.DAO;

import java.sql.*; import java.util.ArrayList; import java.util.List; import model.Usuario;
import config.ConectaDB;

/**
 * 
 * • Classe DAO para operações CRUD da entidade Usuario
 * •
 * • @author TireGrip Team
 * • @version 1.0
 */

*/ public class UsuarioDAO {

    /**
     * Cadastra um novo usuário no banco de dados
     *
     * @param obj objeto Usuario com os dados a serem cadastrados
     * @return true se cadastrado com sucesso, false caso contrário
     * @throws ClassNotFoundException se o driver JDBC não for encontrado
     */
    public boolean cadastrar(Usuario obj) throws ClassNotFoundException {
        Connection conn = null;
        PreparedStatement pstmt = null;

        try {
            conn = ConectaDB.conectar();
            String sql = "INSERT INTO usuarios (nome, email, senha) VALUES (?, ?, ?)";
            pstmt = conn.prepareStatement(sql);

            pstmt.setString(1, obj.getNome());
            pstmt.setString(2, obj.getEmail());
            pstmt.setString(3, obj.getSenha());

            pstmt.executeUpdate();
            return true;
        }
    }
}
```

```

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return false;
    } finally {
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

/**
 * Consulta todos os usuários cadastrados
 *
 * @return List de Usuario ou null se não houver registros
 * @throws ClassNotFoundException se o driver JDBC não for encontrado
 */
public List<Usuario> consulta_geral() throws ClassNotFoundException {
    List<Usuario> lst = new ArrayList<>();
    Connection conn = null;
    Statement stmt = null;
    ResultSet rs = null;

    try {
        conn = ConectaDB.conectar();
        stmt = conn.createStatement();
        String sql = "SELECT * FROM usuarios ORDER BY pk_id ASC";
        rs = stmt.executeQuery(sql);

        int n_reg = 0;
        while (rs.next()) {
            Usuario obj = new Usuario();
            obj.setId(rs.getInt("pk_id"));
            obj.setNome(rs.getString("nome"));
            obj.setEmail(rs.getString("email"));
            obj.setSenha(rs.getString("senha"));
            obj.setDataCadastro(rs.getTimestamp("data_cadastro"));

            lst.add(obj);
            n_reg++;
        }
    }

    return (n_reg == 0) ? null : lst;
}

```

```

        } catch (SQLException ex) {
            System.out.println("Erro - SQL: " + ex.getMessage());
            return null;
        } finally {
            ConectaDB.fecharResultSet(rs);
            ConectaDB.fecharStatement(stmt);
            ConectaDB.fecharConexao(conn);
        }
    }

    /**
     * Consulta um usuário específico por ID
     *
     * @param obj objeto Usuario com o ID a ser consultado
     * @return Usuario encontrado ou null se não existir
     * @throws ClassNotFoundException se o driver JDBC não for encontrado
     */
    public Usuario consulta_id(Usuario obj) throws ClassNotFoundException {
        Connection conn = null;
        PreparedStatement pstmt = null;
        ResultSet rs = null;

        try {
            conn = ConectaDB.conectar();
            String sql = "SELECT * FROM usuarios WHERE pk_id = ?";
            pstmt = conn.prepareStatement(sql);
            pstmt.setInt(1, obj.getId());

            rs = pstmt.executeQuery();

            if (rs.next()) {
                obj.setId(rs.getInt("pk_id"));
                obj.setNome(rs.getString("nome"));
                obj.setEmail(rs.getString("email"));
                obj.setSenha(rs.getString("senha"));
                obj.setDataCadastro(rs.getTimestamp("data_cadastro"));
                return obj;
            } else {
                return null;
            }
        } catch (SQLException ex) {
            System.out.println("Erro - SQL: " + ex.getMessage());
        }
    }
}

```

```

        return null;
    } finally {
        ConectaDB.fecharResultSet(rs);
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

/**
 * Consulta um usuário por email
 *
 * @param email email do usuário a ser consultado
 * @return Usuário encontrado ou null se não existir
 * @throws ClassNotFoundException se o driver JDBC não for encontrado
 */
public Usuario consulta_email(String email) throws
ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "SELECT * FROM usuarios WHERE email = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, email);

        rs = pstmt.executeQuery();

        if (rs.next()) {
            Usuario obj = new Usuario();
            obj.setId(rs.getInt("pk_id"));
            obj.setNome(rs.getString("nome"));
            obj.setEmail(rs.getString("email"));
            obj.setSenha(rs.getString("senha"));
            obj.setDataCadastro(rs.getTimestamp("data_cadastro"));
            return obj;
        } else {
            return null;
        }
    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
    }
}

```

```

        return null;
    } finally {
        ConectaDB.fecharResultSet(rs);
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

/***
 * Autentica um usuário verificando email e senha
 *
 * @param email email do usuário
 * @param senha senha do usuário
 * @return Usuario se credenciais válidas, null caso contrário
 * @throws ClassNotFoundException se o driver JDBC não for encontrado
 */
public Usuario autenticar(String email, String senha) throws
ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "SELECT * FROM usuarios WHERE email = ? AND senha
= ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, email);
        pstmt.setString(2, senha);

        rs = pstmt.executeQuery();

        if (rs.next()) {
            Usuario obj = new Usuario();
            obj.setId(rs.getInt("pk_id"));
            obj.setNome(rs.getString("nome"));
            obj.setEmail(rs.getString("email"));
            obj.setSenha(rs.getString("senha"));
            obj.setDataCadastro(rs.getTimestamp("data_cadastro"));
            return obj;
        } else {
            return null;
        }
    }
}

```

```

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return null;
    } finally {
        ConectaDB.fecharResultSet(rs);
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

/**
 * Altera os dados de um usuário existente
 *
 * @param obj objeto Usuario com os dados atualizados
 * @return true se alterado com sucesso, false caso contrário
 * @throws ClassNotFoundException se o driver JDBC não for encontrado
 */
public boolean alterar(Usuario obj) throws ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "UPDATE usuarios SET nome = ?, email = ?, senha = ? WHERE pk_id = ?";
        pstmt = conn.prepareStatement(sql);

        pstmt.setString(1, obj.getNome());
        pstmt.setString(2, obj.getEmail());
        pstmt.setString(3, obj.getSenha());
        pstmt.setInt(4, obj.getId());

        pstmt.executeUpdate();
        return true;
    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return false;
    } finally {
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

```

```
}

/**
 * Exclui um usuário do banco de dados
 *
 * @param id ID do usuário a ser excluído
 * @return true se excluído com sucesso, false caso contrário
 * @throws ClassNotFoundException se o driver JDBC não for encontrado
 */
public boolean excluir(int id) throws ClassNotFoundException {
    Connection conn = null;
    PreparedStatement pstmt = null;

    try {
        conn = ConectaDB.conectar();
        String sql = "DELETE FROM usuarios WHERE pk_id = ?";
        pstmt = conn.prepareStatement(sql);
        pstmt.setInt(1, id);

        int result = pstmt.executeUpdate();
        return (result > 0);

    } catch (SQLException ex) {
        System.out.println("Erro - SQL: " + ex.getMessage());
        return false;
    } finally {
        ConectaDB.fecharStatement(pstmt);
        ConectaDB.fecharConexao(conn);
    }
}

}
```

```
package model;

/**
 * Classe Model para representar um item do Carrinho de Compras
 *
 * @author TireGrip Team
 * @version 1.0
 */
public class Carrinho {

    private int id;
    private int usuarioId;
    private int produtoId;
    private int quantidade;
    private float subtotal;

    private String produtoNome;
    private String produtoModelo;
    private float produtoPreco;

    public Carrinho() {
    }

    /**
     * @return o ID do item do carrinho
     */
}
```

```
public int getId() {
    return this.id;
}

/**
 * @param p_id o ID a ser definido
 */
public void setId(int p_id) {
    this.id = p_id;
}

/**
 * @return o ID do usuário
 */
public int getUsuarioId() {
    return this.usuarioId;
}

/**
 * @param p_usuarioId o ID do usuário a ser definido
 */
public void setUsuarioId(int p_usuarioId) {
    this.usuarioId = p_usuarioId;
}

/**
 * @return o ID do produto
 */
public int getProdutoId() {
    return this.produtoId;
}

/**
 * @param p_produtoId o ID do produto a ser definido
 */
public void setProdutoId(int p_produtoId) {
    this.produtoId = p_produtoId;
}

/**
 * @return a quantidade do produto
 */
public int getQuantidade() {
```

```
        return this.quantidade;
    }

    /**
     * @param p_quantidade a quantidade a ser definida
     */
    public void setQuantidade(int p_quantidade) {
        this.quantidade = p_quantidade;
    }

    /**
     * @return o subtotal do item (preço * quantidade)
     */
    public float getSubtotal() {
        return this.subtotal;
    }

    /**
     * @param p_subtotal o subtotal a ser definido
     */
    public void setSubtotal(float p_subtotal) {
        this.subtotal = p_subtotal;
    }

    /**
     * @return o nome do produto
     */
    public String getProdutoNome() {
        return this.produtoNome;
    }

    /**
     * @param p_produtoNome o nome do produto a ser definido
     */
    public void setProdutoNome(String p_produtoNome) {
        this.produtoNome = p_produtoNome;
    }

    /**
     * @return o modelo do produto
     */
```

```
public String getProdutoModelo() {
    return this.produtoModelo;
}

/**
 * @param p_produtoModelo o modelo do produto a ser definido
 */
public void setProdutoModelo(String p_produtoModelo) {
    this.produtoModelo = p_produtoModelo;
}

/**
 * @return o preço unitário do produto
 */
public float getProdutoPreco() {
    return this.produtoPreco;
}

/**
 * @param p_produtoPreco o preço do produto a ser definido
 */
public void setProdutoPreco(float p_produtoPreco) {
    this.produtoPreco = p_produtoPreco;
}

@Override
public String toString() {
    return "Carrinho{" +
        "id=" + id +
        ", usuarioId=" + usuarioId +
        ", produtoId=" + produtoId +
        ", quantidade=" + quantidade +
        ", subtotal=" + subtotal +
        '}';
}

}
```

```
package model;

/**
 * 
 * • Classe Model para representar um Endereço de entrega
 * •
 * • @author TireGrip Team
 * • @version 1.0
 */

public class Endereco {

    private int id;
    private int usuarioId;
    private String cep;
    private String rua;
    private String numero;
    private String complemento;
    private String bairro;
    private String cidade;

    public Endereco() {
    }

    /**
     * @return o ID do endereço
     */
    public int getId() {
        return this.id;
    }
}
```

```
/**  
 * @param p_id o ID a ser definido  
 */  
public void setId(int p_id) {  
    this.id = p_id;  
}  
  
/**  
 * @return o ID do usuário  
 */  
public int getUsuarioId() {  
    return this.usuarioId;  
}  
  
/**  
 * @param p_usuarioId o ID do usuário a ser definido  
 */  
public void setUsuarioId(int p_usuarioId) {  
    this.usuarioId = p_usuarioId;  
}  
  
/**  
 * @return o CEP  
 */  
public String getCep() {  
    return this.cep;  
}  
  
/**  
 * @param p_cep o CEP a ser definido  
 */  
public void setCep(String p_cep) {  
    this.cep = p_cep;  
}  
  
/**  
 * @return o nome da rua  
 */  
public String getRua() {  
    return this.rua;  
}  
  
/**
```

```
* @param p_rua o nome da rua a ser definido
*/
public void setRua(String p_rua) {
    this.rua = p_rua;
}

/**
 * @return o número
 */
public String getNumero() {
    return this.numero;
}

/**
 * @param p_numero o número a ser definido
 */
public void setNumero(String p_numero) {
    this.numero = p_numero;
}

/**
 * @return o complemento
 */
public String getComplemento() {
    return this.complemento;
}

/**
 * @param p_complemento o complemento a ser definido
 */
public void setComplemento(String p_complemento) {
    this.complemento = p_complemento;
}

/**
 * @return o bairro
 */
public String getBairro() {
    return this.bairro;
}

/**
 * @param p_bairro o bairro a ser definido
```

```
/*
public void setBairro(String p_bairro) {
    this.bairro = p_bairro;
}

/** 
 * @return a cidade
 */
public String getCidade() {
    return this.cidade;
}

/** 
 * @param p_cidade a cidade a ser definida
 */
public void setCidade(String p_cidade) {
    this.cidade = p_cidade;
}

/** 
 * Retorna o endereço formatado completo
 *
 * @return endereço formatado
 */
public String getEnderecoCompleto() {
    StringBuilder sb = new StringBuilder();
    sb.append(rua).append(", ").append(numero);
    if (complemento != null && !complemento.isEmpty()) {
        sb.append(" - ").append(complemento);
    }
    sb.append(" - ").append(bairro);
    sb.append(" - ").append(cidade);
    sb.append(" - CEP: ").append(cep);
    return sb.toString();
}

@Override
public String toString() {
    return "Endereco{" +
        "id=" + id +
        ", usuarioId=" + usuarioId +
        ", rua='" + rua + '\'' +
        ", numero='" + numero + '\'' +
```

```
        ", bairro='" + bairro + '\'' +
        ", cidade='" + cidade + '\'' +
        ", cep='" + cep + '\'' +
        '}';
    }

}
```

```
package model;
```

```
/**
```

- Classe Model para representar um Item de Pedido
-
- @author TireGrip Team
- @version 1.0

```
*/ public class ItemPedido {
```

```
    private int id;
    private int pedidoId;
    private int produtoId;
    private int quantidade;
    private float precoUnitario;
    private float subtotal;
```

```
    private String produtoNome;
    private String produtoModelo;
```

```
    public ItemPedido() {
    }
```

```
    /**
     * @return o ID do item do pedido
     */
    public int getId() {
        return this.id;
    }
```

```
/**  
 * @param p_id o ID a ser definido  
 */  
public void setId(int p_id) {  
    this.id = p_id;  
}  
  
/**  
 * @return o ID do pedido  
 */  
public int getPedidoId() {  
    return this.pedidoId;  
}  
  
/**  
 * @param p_pedidoId o ID do pedido a ser definido  
 */  
public void setPedidoId(int p_pedidoId) {  
    this.pedidoId = p_pedidoId;  
}  
  
/**  
 * @return o ID do produto  
 */  
public int getProdutoId() {  
    return this.produtoId;  
}  
  
/**  
 * @param p_produtoId o ID do produto a ser definido  
 */  
public void setProdutoId(int p_produtoId) {  
    this.produtoId = p_produtoId;  
}  
  
/**  
 * @return a quantidade do produto  
 */  
public int getQuantidade() {  
    return this.quantidade;  
}
```

```
/**  
 * @param p_quantidade a quantidade a ser definida  
 */  
public void setQuantidade(int p_quantidade) {  
    this.quantidade = p_quantidade;  
}  
  
/**  
 * @return o preço unitário no momento da compra  
 */  
public float getPrecoUnitario() {  
    return this.precoUnitario;  
}  
  
/**  
 * @param p_precoUnitario o preço unitário a ser definido  
 */  
public void setPrecoUnitario(float p_precoUnitario) {  
    this.precoUnitario = p_precoUnitario;  
}  
  
/**  
 * @return o subtotal do item (preço unitário * quantidade)  
 */  
public float getSubtotal() {  
    return this.subtotal;  
}  
  
/**  
 * @param p_subtotal o subtotal a ser definido  
 */  
public void setSubtotal(float p_subtotal) {  
    this.subtotal = p_subtotal;  
}  
  
/**  
 * @return o nome do produto  
 */  
public String getProdutoNome() {  
    return this.produtoNome;  
}
```

```
/**  
 * @param p_produtoNome o nome do produto a ser definido  
 */  
public void setProdutoNome(String p_produtoNome) {  
    this.produtoNome = p_produtoNome;  
}  
  
/**  
 * @return o modelo do produto  
 */  
public String getProdutoModelo() {  
    return this.produtoModelo;  
}  
  
/**  
 * @param p_produtoModelo o modelo do produto a ser definido  
 */  
public void setProdutoModelo(String p_produtoModelo) {  
    this.produtoModelo = p_produtoModelo;  
}  
  
@Override  
public String toString() {  
    return "ItemPedido{" +  
        "id=" + id +  
        ", pedidoId=" + pedidoId +  
        ", produtoId=" + produtoId +  
        ", quantidade=" + quantidade +  
        ", precoUnitario=" + precoUnitario +  
        ", subtotal=" + subtotal +  
        '}';  
}  
}
```

```
package model;

import java.util.Date;

/***
 * 
 * • Classe Model para representar um Pedido do sistema
 * • 
 * • @author TireGrip Team
 * • @version 1.0
 */

*/ public class Pedido {

    private int id;
    private int usuarioId;
    private int enderecoId;
    private Date dataPedido;
    private float subtotal;
    private float frete;
    private float desconto;
    private float total;
    private String formaPagamento;
    private String status;

    private String usuarioNome;
    private String enderecoCompleto;

    public Pedido() {
    }

    /**
     * @return o ID do pedido
     */
    public int getId() {
        return this.id;
```

```
}

/**
 * @param p_id o ID a ser definido
 */
public void setId(int p_id) {
    this.id = p_id;
}

/**
 * @return o ID do usuário
 */
public int getUsuarioId() {
    return this.usuarioId;
}

/**
 * @param p_usuarioId o ID do usuário a ser definido
 */
public void setUsuarioId(int p_usuarioId) {
    this.usuarioId = p_usuarioId;
}

/**
 * @return o ID do endereço
 */
public int getEnderecoId() {
    return this.enderecoId;
}

/**
 * @param p_enderecoId o ID do endereço a ser definido
 */
public void setEnderecoId(int p_enderecoId) {
    this.enderecoId = p_enderecoId;
}

/**
 * @return a data do pedido
 */
public Date getDataPedido() {
    return this.dataPedido;
}
```

```
/**  
 * @param p_dataPedido a data do pedido a ser definida  
 */  
public void setDataPedido(Date p_dataPedido) {  
    this.dataPedido = p_dataPedido;  
}  
  
/**  
 * @return o subtotal do pedido (soma dos produtos)  
 */  
public float getSubtotal() {  
    return this.subtotal;  
}  
  
/**  
 * @param p_subtotal o subtotal a ser definido  
 */  
public void setSubtotal(float p_subtotal) {  
    this.subtotal = p_subtotal;  
}  
  
/**  
 * @return o valor do frete  
 */  
public float getFrete() {  
    return this.frete;  
}  
  
/**  
 * @param p_frete o valor do frete a ser definido  
 */  
public void setFrete(float p_frete) {  
    this.frete = p_frete;  
}  
  
/**  
 * @return o valor do desconto  
 */  
public float getDesconto() {  
    return this.desconto;  
}
```

```
/**  
 * @param p_desconto o valor do desconto a ser definido  
 */  
public void setDesconto(float p_desconto) {  
    this.desconto = p_desconto;  
}  
  
/**  
 * @return o valor total do pedido  
 */  
public float getTotal() {  
    return this.total;  
}  
  
/**  
 * @param p_total o valor total a ser definido  
 */  
public void setTotal(float p_total) {  
    this.total = p_total;  
}  
  
/**  
 * @return a forma de pagamento (CREDITO ou PIX)  
 */  
public String getFormaPagamento() {  
    return this.formaPagamento;  
}  
  
/**  
 * @param p_formaPagamento a forma de pagamento a ser definida  
 */  
public void setFormaPagamento(String p_formaPagamento) {  
    this.formaPagamento = p_formaPagamento;  
}  
  
/**  
 * @return o status do pedido (PENDENTE, CONFIRMADO, ENVIADO, ENTREGUE)  
 */  
public String getStatus() {  
    return this.status;  
}  
  
/**
```

```
* @param p_status o status a ser definido
*/
public void setStatus(String p_status) {
    this.status = p_status;
}

/**
 * @return o nome do usuário
 */
public String getUsuarioNome() {
    return this.usuarioNome;
}

/**
 * @param p_usuarioNome o nome do usuário a ser definido
 */
public void setUsuarioNome(String p_usuarioNome) {
    this.usuarioNome = p_usuarioNome;
}

/**
 * @return o endereço completo formatado
 */
public String getEnderecoCompleto() {
    return this.enderecoCompleto;
}

/**
 * @param p_enderecoCompleto o endereço completo a ser definido
 */
public void setEnderecoCompleto(String p_enderecoCompleto) {
    this.enderecoCompleto = p_enderecoCompleto;
}

@Override
public String toString() {
    return "Pedido{" +
        "id=" + id +
        ", usuarioId=" + usuarioId +
        ", dataPedido=" + dataPedido +
        ", total=" + total +
```

```
        ", formaPagamento='" + formaPagamento + '\'' +
        ", status='" + status + '\'' +
        '}';
    }

}
```

```
package model;
```

```
/**
```

- Classe Model para representar um Produto (Pneu) do sistema
-
- @author TireGrip Team
- @version 1.0

```
*/ public class Produto {
```

```
    private int id;
    private String nome;
    private String modelo;
    private float preco;
    private String descricao;
    private String largura;
    private String perfil;
    private String aro;
    private String indiceCarga;
```

```
private String indiceVelocidade;
private int estoque;

public Produto() {
}

/**
 * @return o ID do produto
 */
public int getId() {
    return this.id;
}

/**
 * @param p_id o ID a ser definido
 */
public void setId(int p_id) {
    this.id = p_id;
}

/**
 * @return o nome do produto
 */
public String getNome() {
    return this.nome;
}

/**
 * @param p_nome o nome a ser definido
 */
public void setNome(String p_nome) {
    this.nome = p_nome;
}

/**
 * @return o modelo do produto
 */
public String getModelo() {
    return this.modelo;
}
```

```
/**  
 * @param p_modelo o modelo a ser definido  
 */  
public void setModelo(String p_modelo) {  
    this.modelo = p_modelo;  
}  
  
/**  
 * @return o preço do produto  
 */  
public float getPreco() {  
    return this.preco;  
}  
  
/**  
 * @param p_preco o preço a ser definido  
 */  
public void setPreco(float p_preco) {  
    this.preco = p_preco;  
}  
  
/**  
 * @return a descrição do produto  
 */  
public String getDescricao() {  
    return this.descricao;  
}  
  
/**  
 * @param p_descricao a descrição a ser definida  
 */  
public void setDescricao(String p_descricao) {  
    this.descricao = p_descricao;  
}  
  
/**  
 * @return a largura do pneu  
 */  
public String getLargura() {  
    return this.largura;  
}
```

```
/**  
 * @param p_largura a largura a ser definida  
 */  
public void setLargura(String p_largura) {  
    this.largura = p_largura;  
}  
  
/**  
 * @return o perfil do pneu  
 */  
public String getPerfil() {  
    return this.perfil;  
}  
  
/**  
 * @param p_perfil o perfil a ser definido  
 */  
public void setPerfil(String p_perfil) {  
    this.perfil = p_perfil;  
}  
  
/**  
 * @return o aro do pneu  
 */  
public String getAro() {  
    return this.aro;  
}  
  
/**  
 * @param p_ar o aro a ser definido  
 */  
public void setAro(String p_ar) {  
    this.aro = p_ar;  
}  
  
/**  
 * @return o índice de carga do pneu  
 */  
public String getIndiceCarga() {  
    return this.indiceCarga;  
}  
  
/**
```

```
* @param p_indiceCarga o índice de carga a ser definido
*/
public void setIndiceCarga(String p_indiceCarga) {
    this.indiceCarga = p_indiceCarga;
}

/**
 * @return o índice de velocidade do pneu
*/
public String getIndiceVelocidade() {
    return this.indiceVelocidade;
}

/**
 * @param p_indiceVelocidade o índice de velocidade a ser definido
*/
public void setIndiceVelocidade(String p_indiceVelocidade) {
    this.indiceVelocidade = p_indiceVelocidade;
}

/**
 * @return a quantidade em estoque
*/
public int getEstoque() {
    return this.estoque;
}

/**
 * @param p_estoque a quantidade em estoque a ser definida
*/
public void setEstoque(int p_estoque) {
    this.estoque = p_estoque;
}

@Override
public String toString() {
    return "Produto{" +
        "id=" + id +
        ", nome='" + nome + '\'' +
        ", modelo='" + modelo + '\'' +
        ", preco=" + preco +
        ", estoque=" + estoque +
        '}';
}
```

```
}
```

```
}
```

```
package model;
```

```
import java.util.Date;
```

```
/**
```

- Classe Model para representar um Usuário do sistema
-
- @author TireGrip Team
- @version 1.0

```
 */ public class Usuario {
```

```
private int id;
private String nome;
private String email;
private String senha;
private Date dataCadastro;

public Usuario() {
}

/**
 * @return o ID do usuário
 */
public int getId() {
    return this.id;
}

/**
 * @param p_id o ID a ser definido
 */
public void setId(int p_id) {
    this.id = p_id;
}

/**
 * @return o nome do usuário
 */
public String getNome() {
    return this.nome;
}

/**
 * @param p_nome o nome a ser definido
 */
public void setNome(String p_nome) {
    this.nome = p_nome;
}

/**
 * @return o email do usuário
 */
```

```
public String getEmail() {
    return this.email;
}

/**
 * @param p_email o email a ser definido
 */
public void setEmail(String p_email) {
    this.email = p_email;
}

/**
 * @return a senha do usuário
 */
public String getSenha() {
    return this.senha;
}

/**
 * @param p_senha a senha a ser definida
 */
public void setSenha(String p_senha) {
    this.senha = p_senha;
}

/**
 * @return a data de cadastro do usuário
 */
public Date getDataCadastro() {
    return this.dataCadastro;
}

/**
 * @param p_dataCadastro a data de cadastro a ser definida
 */
public void setDataCadastro(Date p_dataCadastro) {
    this.dataCadastro = p_dataCadastro;
}

@Override
public String toString() {
    return "Usuario{" +
        "id=" + id +
```

```
    ", nome='" + nome + '\'' +
    ", email='" + email + '\'' +
    ", dataCadastro=" + dataCadastro +
    '}';
}

}
```

```
<%@page import="model.Usuario"%>

<%@page import="model.DAO.UsuarioDAO"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Login - TireGrip</title>

<style>

* {

margin: 0;
```

```
padding: 0;  
box-sizing: border-box;  
}  
  
body {  
font-family: 'Inter', system-ui, sans-serif;  
background-color: black;  
color: white;  
min-height: 100vh;  
display: flex;  
align-items: center;  
justify-content: center;  
}  
  
.container {  
max-width: 28rem;  
width: 100%;  
padding: 2rem;  
}  
  
.message {  
background-color: #1A1A1A;  
border: 2px solid #333333;  
padding: 2rem;  
text-align: center;  
}  
  
.success {
```

```
border-color: #FFD000;  
color: #FFD000;  
}  
  
.error {  
border-color: #EF4444;  
color: #EF4444;  
}  
  
.message h2 {  
margin-bottom: 1rem;  
text-transform: uppercase;  
letter-spacing: 0.1em;  
}  
  
.message p {  
margin-bottom: 1.5rem;  
}  
  
.btn {  
background-color: #FF6B00;  
color: black;  
padding: 1rem 2rem;  
text-transform: uppercase;  
letter-spacing: 0.1em;  
font-weight: 800;  
border: none;  
cursor: pointer;
```

```
text-decoration: none;  
display: inline-block;  
}  
  
.btn:hover {  
background-color: #FFD000;  
}  
  
</style>  
  
</head>  
  
<body>  
  
<div class="container">  
  
<%  
  
try {  
  
    // Receber parâmetros do formulário  
  
    String email = request.getParameter("email");  
  
    String senha = request.getParameter("password");  
  
  
    // Validar campos obrigatórios  
  
    if (email == null || email.trim().isEmpty() || senha == null || senha.trim().isEmpty())  
    {  
  
        out.println("<div class='message error'>");  
  
        out.println("<h2>Erro</h2>");  
  
        out.println("<p>Por favor, preencha todos os campos!</p>");  
  
        out.println("<a href='..//index.html' class='btn'>Voltar</a>");  
  
        out.println("</div>");  
  
    } else {
```

```
// Tentar autenticar

UsuarioDAO dao = new UsuarioDAO();

Usuario usuario = dao.autenticar(email, senha);

if (usuario != null) {

    // Login bem-sucedido - criar sessão

    HttpSession sessao = request.getSession();

    sessao.setAttribute("usuarioId", usuario.getId());

    sessao.setAttribute("usuarioNome", usuario.getNome());

    sessao.setAttribute("usuarioEmail", usuario.getEmail());

    // Redirecionar para home

    response.sendRedirect("../home.html");

} else {

    // Credenciais inválidas

    out.println("<div class='message error'>");

    out.println("<h2>Erro de Autenticação</h2>");

    out.println("<p>Email ou senha incorretos!</p>");

    out.println("<a href='../index.html' class='btn'>Tentar Novamente</a>");

    out.println("</div>");

}

}

} catch (Exception ex) {

    out.println("<div class='message error'>");


```

```
out.println("<h2>Erro</h2>");

out.println("<p>Erro ao processar login: " + ex.getMessage() + "</p>");

out.println("<a href='..//index.html' class='btn'>Voltar</a>");

out.println("</div>");

}

%0>

</div>

</body>

</html>
```

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%
// Invalidar a sessão
```

```

HttpSession sessao = request.getSession(false);

if (sessao != null) {
    sessao.invalidate();
}

// Redirecionar para a página de login
response.sendRedirect("../index.html");

%>

<%@page import="model.Carrinho"%> <%@page import="model.Produto"%> <%@page
import="model.DAO.CarrinhoDAO"%> <%@page import="model.DAO.ProdutoDAO"%>
<%@page contentType="application/json" pageEncoding="UTF-8"%> <%
response.setContentType("application/json"); response.setCharacterEncoding("UTF-8");

try {
    // Verificar autenticação
    HttpSession sessao = request.getSession(false);
    if (sessao == null || sessao.getAttribute("usuarioId") == null) {
        response.setStatus(HttpServletRequest.SC_UNAUTHORIZED);
        out.print("{\"success\": false, \"message\": \"Usuário não
autenticado\"}");
        return;
    }

    int usuarioId = (Integer) sessao.getAttribute("usuarioId");

    // Receber parâmetros
    String produtoIdParam = request.getParameter("produtoId");
    String quantidadeParam = request.getParameter("quantidade");

```

```

        if (produtoIdParam == null || quantidadeParam == null) {
            response.setStatus(HttpServletRequest.SC_BAD_REQUEST);
            out.print("{\"success\": false, \"message\": \"Parâmetros
inválidos\"}");
            return;
        }

        int produtoId = Integer.parseInt(produtoIdParam);
        int quantidade = Integer.parseInt(quantidadeParam);

        // Buscar produto para obter o preço
        ProdutoDAO produtoDAO = new ProdutoDAO();
        Produto produto = new Produto();
        produto.setId(produtoId);
        produto = produtoDAO.consulta_id(produto);

        if (produto == null) {
            response.setStatus(HttpServletRequest.SC_NOT_FOUND);
            out.print("{\"success\": false, \"message\": \"Produto não
encontrado\"}");
            return;
        }

        // Calcular subtotal
        float subtotal = produto.getPreco() * quantidade;

        // Verificar se item já existe no carrinho
        CarrinhoDAO carrinhoDAO = new CarrinhoDAO();
        Carrinho itemExistente = carrinhoDAO.consulta_item(usuarioId,
produtoId);

        boolean sucesso;
        if (itemExistente != null) {
            // Atualizar quantidade
            int novaQuantidade = itemExistente.getQuantidade() +
quantidade;
            float novoSubtotal = produto.getPreco() * novaQuantidade;
            sucesso =
carrinhoDAO.atualizar_quantidade(itemExistente.getId(), novaQuantidade,
novoSubtotal);
        } else {
            // Adicionar novo item

```

```
Carrinho novoItem = new Carrinho();
novoItem.setUsuarioId(usuarioId);
novoItem.setProdutoId(produtoId);
novoItem.setQuantidade(quantidade);
novoItem.setSubtotal(subtotal);
sucesso = carrinhoDAO.adicionar(novoItem);
}

if (sucesso) {
    out.print("{\"success\": true, \"message\": \"Produto
adicionado ao carrinho\"}");
} else {

response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
    out.print("{\"success\": false, \"message\": \"Erro ao
adicionar produto\"}");
}

} catch (Exception ex) {
    response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
    out.print("{\"success\": false, \"message\": \"\" + ex.getMessage()
+ \"\"}");
}

%>
```

```

<%@page import="model.Carrinho"%> <%@page import="model.Produto"%> <%@page
import="model.DAO.CarrinhoDAO"%> <%@page import="model.DAO.ProdutoDAO"%>
<%@page contentType="application/json" pageEncoding="UTF-8"%> <%
response.setContentType("application/json"); response.setCharacterEncoding("UTF-8");

try {
    HttpSession sessao = request.getSession(false);
    if (sessao == null || sessao.getAttribute("usuarioId") == null) {
        response.setStatus(HttpServletRequest.SC_UNAUTHORIZED);
        out.print("{\"success\": false, \"message\": \"Não
autenticado\"}");
        return;
    }

    String idParam = request.getParameter("id");
    String quantidadeParam = request.getParameter("quantidade");

    if (idParam == null || quantidadeParam == null) {
        response.setStatus(HttpServletRequest.SC_BAD_REQUEST);
        out.print("{\"success\": false, \"message\": \"Parâmetros
inválidos\"}");
        return;
    }

    int id = Integer.parseInt(idParam);
    int quantidade = Integer.parseInt(quantidadeParam);

    CarrinhoDAO dao = new CarrinhoDAO();

    if (quantidade < 1) {
        // Remover item
        boolean sucesso = dao.remover(id);
        out.print("{\"success\": " + sucesso + ", \"message\": \"Item
removido\"}");
    } else {

```

```

        // Atualizar quantidade - precisa recalcular subtotal
        // Por simplicidade, vamos apenas atualizar (o front-end deve
enviar o subtotal correto)
        String subtotalParam = request.getParameter("subtotal");
        float subtotal = Float.parseFloat(subtotalParam);

        boolean sucesso = dao.atualizar_quantidade(id, quantidade,
subtotal);
        out.print("{\"success\": " + sucesso + ", \"message\":
\"Quantidade atualizada\"}");
    }

} catch (Exception ex) {
    response.setStatus(HttpServletRequest.SC_INTERNAL_SERVER_ERROR);
    out.print("{\"success\": false, \"message\": \""
+ ex.getMessage()
+ "\"}");
}

```

%>

```

<%@page import="model.DAO.CarrinhoDAO"%> <%@page
contentType="application/json" pageEncoding="UTF-8"%> <%
response.setContentType("application/json"); response.setCharacterEncoding("UTF-8");

try {
    HttpSession sessao = request.getSession(false);
    if (sessao == null || sessao.getAttribute("usuarioId") == null) {

```

```

        response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        out.print("{\"error\": \"Usuário não autenticado\"}");
        return;
    }

    int usuarioId = (Integer) sessao.getAttribute("usuarioId");

    CarrinhoDAO dao = new CarrinhoDAO();
    boolean sucesso = dao.limpar_carrinho(usuarioId);

    if (sucesso) {
        out.print("{\"success\": true, \"message\": \"Carrinho limpo com sucesso\"}");
    } else {
        out.print("{\"success\": false, \"message\": \"Erro ao limpar carrinho\"}");
    }

} catch (Exception ex) {
    response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
    out.print("{\"error\": \"\" + ex.getMessage() + \"\"}");
}

```

%>

```

<%@page import="model.DAO.CarrinhoDAO"%> <%@page
contentType="application/json" pageEncoding="UTF-8"%> <%
response.setContentType("application/json"); response.setCharacterEncoding("UTF-8");

```

```

try {
    HttpSession sessao = request.getSession(false);
    if (sessao == null || sessao.getAttribute("usuarioId") == null) {
        response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        out.print("{\"success\": false, \"message\": \"Não
autenticado\"}");
        return;
    }

    String idParam = request.getParameter("id");

    if (idParam == null) {
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
        out.print("{\"success\": false, \"message\": \"ID não
fornecido\"}");
        return;
    }

    int id = Integer.parseInt(idParam);

    CarrinhoDAO dao = new CarrinhoDAO();
    boolean sucesso = dao.remover(id);

    out.print("{\"success\": " + sucesso + ", \"message\": \"Item
removido do carrinho\"}");

} catch (Exception ex) {
    response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
    out.print("{\"success\": false, \"message\": \"\" + ex.getMessage()
+ "\"}");
}
%>
```

```

<%@page import="java.util.List"%> <%@page import="model.Carrinho"%> <%@page
import="model.DAO.CarrinhoDAO"%> <%@page import="com.google.gson.Gson"%>
<%@page contentType="application/json" pageEncoding="UTF-8"%> <%
response.setContentType("application/json"); response.setCharacterEncoding("UTF-8");

try {
    HttpSession sessao = request.getSession(false);
```

```

        if (sessao == null || sessao.getAttribute("usuarioId") == null) {
            response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
            out.print("{\"error\": \"Usuário não autenticado\"}");
            return;
        }

        int usuarioId = (Integer) sessao.getAttribute("usuarioId");

        CarrinhoDAO dao = new CarrinhoDAO();
        List<Carrinho> itens = dao.consulta_por_usuario(usuarioId);
        float subtotalTotal = dao.calcular_subtotal(usuarioId);

        Gson gson = new Gson();
        String json = "{\"itens\": " + gson.toJson(itens) + ",
\"subtotal\": " + subtotalTotal + "}";
        out.print(json);

    } catch (Exception ex) {
        response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
        out.print("{\"error\": \"\" + ex.getMessage() + \"\"}");
    }
}

%>
```

```

<%@page import="model.Endereco"%> <%@page import="model.DAO.EnderecoDAO"%>
<%@page contentType="application/json" pageEncoding="UTF-8"%> <%
response.setContentType("application/json"); response.setCharacterEncoding("UTF-8");

try {
    HttpSession sessao = request.getSession(false);
    if (sessao == null || sessao.getAttribute("usuarioId") == null) {
        response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        out.print("{\"success\": false, \"message\": \"Não
autenticado\"}");
        return;
    }
}
```

```
int usuarioId = (Integer) sessao.getAttribute("usuarioId");

// Receber parâmetros
String cep = request.getParameter("cep");
String rua = request.getParameter("rua");
String numero = request.getParameter("numero");
String complemento = request.getParameter("complemento");
String bairro = request.getParameter("bairro");
String cidade = request.getParameter("cidade");

// Validar campos obrigatórios
if (cep == null || rua == null || numero == null || bairro == null
|| cidade == null) {
    response.setStatus(HttpServletRequest.SC_BAD_REQUEST);
    out.print("{\"success\": false, \"message\": \"Campos
obrigatórios não preenchidos\"}");
    return;
}

// Criar objeto Endereco
Endereco endereco = new Endereco();
endereco.setUsuarioId(usuarioId);
endereco.setCep(cep);
endereco.setRua(rua);
endereco.setNumero(numero);
endereco.setComplemento(complemento);
endereco.setBairro(bairro);
endereco.setCidade(cidade);

// Cadastrar
EnderecoDAO dao = new EnderecoDAO();
boolean sucesso = dao.cadastrar(endereco);

if (sucesso) {
    out.print("{\"success\": true, \"message\": \"Endereço
cadastrado com sucesso\"}");
} else {

response.setStatus(HttpServletRequest.SC_INTERNAL_SERVER_ERROR);
    out.print("{\"success\": false, \"message\": \"Erro ao
cadastrar endereço\"}");
}
```

```
    } catch (Exception ex) {
        response.setStatus(HttpServletRequest.SC_INTERNAL_SERVER_ERROR);
        out.print("{\"success\": false, \"message\": \""
        + ex.getMessage()
        + "\"}");
    }

%>
```

```
<%@page import="java.util.List"%> <%@page import="model.Endereco"%> <%@page
import="model.DAO.EnderecoDAO"%> <%@page import="com.google.gson.Gson"%>
<%@page contentType="application/json" pageEncoding="UTF-8"%> <%
response.setContentType("application/json"); response.setCharacterEncoding("UTF-8");

try {
    HttpSession sessao = request.getSession(false);
    if (sessao == null || sessao.getAttribute("usuarioId") == null) {
        response.setStatus(HttpServletRequest.SC_UNAUTHORIZED);
        out.print("{\"error\": \"Não autenticado\"}");
        return;
    }

    int usuarioId = (Integer) sessao.getAttribute("usuarioId");

    EnderecoDAO dao = new EnderecoDAO();
    List<Endereco> enderecos = dao.consulta_por_usuario(usuarioId);

    Gson gson = new Gson();
```

```

        String json = gson.toJson(enderecos);
        out.print(json);

    } catch (Exception ex) {
        response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
        out.print("{\"error\": \"" + ex.getMessage() + "\"}");
    }

%>

```

```

<%@page import="java.util.List"%> <%@page import="model.Pedido"%> <%@page
import="model.ItemPedido"%> <%@page import="model.DAO.PedidoDAO"%> <%@page
import="model.DAO.ItemPedidoDAO"%> <%@page import="com.google.gson.Gson"%>
<%@page contentType="application/json" pageEncoding="UTF-8"%> <%
response.setContentType("application/json"); response.setCharacterEncoding("UTF-8");

try {
    HttpSession sessao = request.getSession(false);
    if (sessao == null || sessao.getAttribute("usuarioId") == null) {
        response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        out.print("{\"error\": \"Não autenticado\"}");
        return;
    }

    String pedidoIdParam = request.getParameter("pedidoId");

    if (pedidoIdParam == null) {
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
        out.print("{\"error\": \"ID do pedido não fornecido\"}");
        return;
    }

    int pedidoId = Integer.parseInt(pedidoIdParam);

```

```
// Buscar pedido
PedidoDAO pedidoDAO = new PedidoDAO();
Pedido pedido = new Pedido();
pedido.setId(pedidoId);
pedido = pedidoDAO.consulta_id(pedido);

if (pedido == null) {
    response.setStatus(HttpServletRequest.SC_NOT_FOUND);
    out.print("{\"error\": \"Pedido não encontrado\"}");
    return;
}

// Buscar itens do pedido
ItemPedidoDAO itemPedidoDAO = new ItemPedidoDAO();
List<ItemPedido> itens =
itemPedidoDAO.consulta_por_pedido(pedidoId);

// Montar resposta JSON
Gson gson = new Gson();
String json = "{\"pedido\": " + gson.toJson(pedido) + ", \"itens\":"
+ gson.toJson(itens) + "}";
out.print(json);

} catch (Exception ex) {
    response.setStatus(HttpServletRequest.SC_INTERNAL_SERVER_ERROR);
    out.print("{\"error\": \"\" + ex.getMessage() + \"\"}");
}
%>
```

```

<%@page import="java.util.List"%> <%@page import="java.util.ArrayList"%> <%@page
import="model.Carrinho"%> <%@page import="model.Pedido"%> <%@page
import="model.ItemPedido"%> <%@page import="model.DAO.CarrinhoDAO"%>
<%@page import="model.DAO.PedidoDAO"%> <%@page
import="model.DAO.ItemPedidoDAO"%> <%@page contentType="application/json"
pageEncoding="UTF-8"%> <% response.setContentType("application/json");
response.setCharacterEncoding("UTF-8");

try {
    // Verificar autenticação
    HttpSession sessao = request.getSession(false);
    if (sessao == null || sessao.getAttribute("usuarioId") == null) {
        response.setStatus(HttpServletRequest.SC_UNAUTHORIZED);
        out.print("{\"success\": false, \"message\": \"Não
autenticado\"}");
        return;
    }

    int usuarioId = (Integer) sessao.getAttribute("usuarioId");

    // Receber parâmetros
    String enderecoIdParam = request.getParameter("enderecoId");
    String formaPagamento = request.getParameter("formaPagamento");

    if (enderecoIdParam == null || formaPagamento == null) {
        response.setStatus(HttpServletRequest.SC_BAD_REQUEST);
        out.print("{\"success\": false, \"message\": \"Parâmetros
inválidos\"}");
        return;
    }

    int enderecoId = Integer.parseInt(enderecoIdParam);

    // Buscar itens do carrinho
    CarrinhoDAO carrinhoDAO = new CarrinhoDAO();
    List<Carrinho> itensCarrinho =
carrinhoDAO.consulta_por_usuario(usuarioId);

    if (itensCarrinho == null || itensCarrinho.isEmpty()) {
        response.setStatus(HttpServletRequest.SC_BAD_REQUEST);

```

```
        out.print("{\"success\": false, \"message\": \"Carrinho\nvazio\"}");\n        return;\n    }\n\n    // Calcular valores\n    float subtotal = carrinhoDAO.calcular_subtotal(usuarioId);\n    float frete = 50.00f;\n    float desconto = 0.0f;\n\n    // Aplicar desconto de 5% se for PIX\n    if ("PIX".equalsIgnoreCase(formaPagamento)) {\n        desconto = (subtotal + frete) * 0.05f;\n    }\n\n    float total = subtotal + frete - desconto;\n\n    // Criar pedido\n    Pedido pedido = new Pedido();\n    pedido.setUsuarioId(usuarioId);\n    pedido.setEnderecoId(enderecoId);\n    pedido.setSubtotal(subtotal);\n    pedido.setFrete(frete);\n    pedido.setDesconto(desconto);\n    pedido.setTotal(total);\n    pedido.setFormaPagamento(formaPagamento.toUpperCase());\n    pedido.setStatus("PENDENTE");\n\n    PedidoDAO pedidoDAO = new PedidoDAO();\n    int pedidoId = pedidoDAO.cadastrar(pedido);\n\n    if (pedidoId == 0) {\n\n        response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);\n        out.print("{\"success\": false, \"message\": \"Erro ao criar\npedido\"}");\n        return;\n    }\n\n    // Criar itens do pedido\n    List<ItemPedido> itensPedido = new ArrayList<>();\n    for (Carrinho itemCarrinho : itensCarrinho) {\n        ItemPedido item = new ItemPedido();
```

```

        item.setPedidoId(pedidoId);
        item.setProdutoId(itemCarrinho.getProdutoId());
        item.setQuantidade(itemCarrinho.getQuantidade());
        item.setPrecoUnitario(itemCarrinho.getProdutoPreco());
        item.setSubtotal(itemCarrinho.getSubtotal());
        itensPedido.add(item);
    }

    ItemPedidoDAO itemPedidoDAO = new ItemPedidoDAO();
    boolean itensAdicionados =
itemPedidoDAO.cadastrar_lote(itensPedido);

    if (!itensAdicionados) {

response.setStatus(HttpServletRequest.SC_INTERNAL_SERVER_ERROR);
        out.print("{\"success\": false, \"message\": \"Erro ao
adicionar itens do pedido\"}");
        return;
    }

    // Limpar carrinho
carrinhoDAO.limpar_carrinho(usuarioId);

    // Retornar sucesso
    out.print("{\"success\": true, \"message\": \"Pedido finalizado com
sucesso\", \"pedidoId\": " + pedidoId + ", \"total\": " + total + "}");

} catch (Exception ex) {
    response.setStatus(HttpServletRequest.SC_INTERNAL_SERVER_ERROR);
    out.print("{\"success\": false, \"message\": \"\" + ex.getMessage()
+ \"\"}");
}

```

%>

```

<%@page import="java.util.List"%> <%@page import="model.Pedido"%> <%@page
import="model.DAO.PedidoDAO"%> <%@page import="com.google.gson.Gson"%>
<%@page contentType="application/json" pageEncoding="UTF-8"%> <%
response.setContentType("application/json"); response.setCharacterEncoding("UTF-8");

try {
    HttpSession sessao = request.getSession(false);
    if (sessao == null || sessao.getAttribute("usuarioId") == null) {
        response.setStatus(HttpServletRequest.SC_UNAUTHORIZED);
        out.print("{\"error\": \"Não autenticado\"}");
        return;
    }

    int usuarioId = (Integer) sessao.getAttribute("usuarioId");

    PedidoDAO dao = new PedidoDAO();
    List<Pedido> pedidos = dao.consulta_por_usuario(usuarioId);

    Gson gson = new Gson();
    String json = gson.toJson(pedidos);
    out.print(json);

} catch (Exception ex) {
    response.setStatus(HttpServletRequest.SC_INTERNAL_SERVER_ERROR);
    out.print("{\"error\": \"\" + ex.getMessage() + \"\"}");
}
%>

```

```

<%@page import="model.Produto"%> <%@page import="model.DAO.ProdutoDAO"%>
<%@page import="com.google.gson.Gson"%> <%@page contentType="application/json"
pageEncoding="UTF-8"%> <% response.setContentType("application/json");
response.setCharacterEncoding("UTF-8");

try {
    String idParam = request.getParameter("id");

    if (idParam == null || idParam.trim().isEmpty()) {
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
        out.print("{\"error\": \"ID do produto não fornecido\"}");
    } else {
        int produtoId = Integer.parseInt(idParam);

        ProdutoDAO dao = new ProdutoDAO();
        Produto produto = new Produto();
        produto.setId(produtoId);
        produto = dao.consulta_id(produto);

        if (produto != null) {
            // Converter para JSON usando Gson
            Gson gson = new Gson();
            String json = gson.toJson(produto);
            out.print(json);
        } else {
            response.setStatus(HttpServletResponse.SC_NOT_FOUND);
            out.print("{\"error\": \"Produto não encontrado\"}");
        }
    }

} catch (NumberFormatException ex) {
    response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
    out.print("{\"error\": \"ID inválido\"}");
} catch (Exception ex) {
    response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
    out.print("{\"error\": \"\" + ex.getMessage() + \"\"}");
}

%>

<%@page import="java.util.List"%> <%@page import="model.Produto"%> <%@page
import="model.DAO.ProdutoDAO"%> <%@page import="com.google.gson.Gson"%>
```

```
<%@page contentType="application/json" pageEncoding="UTF-8"%> <%
response.setContentType("application/json"); response.setCharacterEncoding("UTF-8");

try {
    ProdutoDAO dao = new ProdutoDAO();
    List<Produto> produtos = dao.consulta_geral();

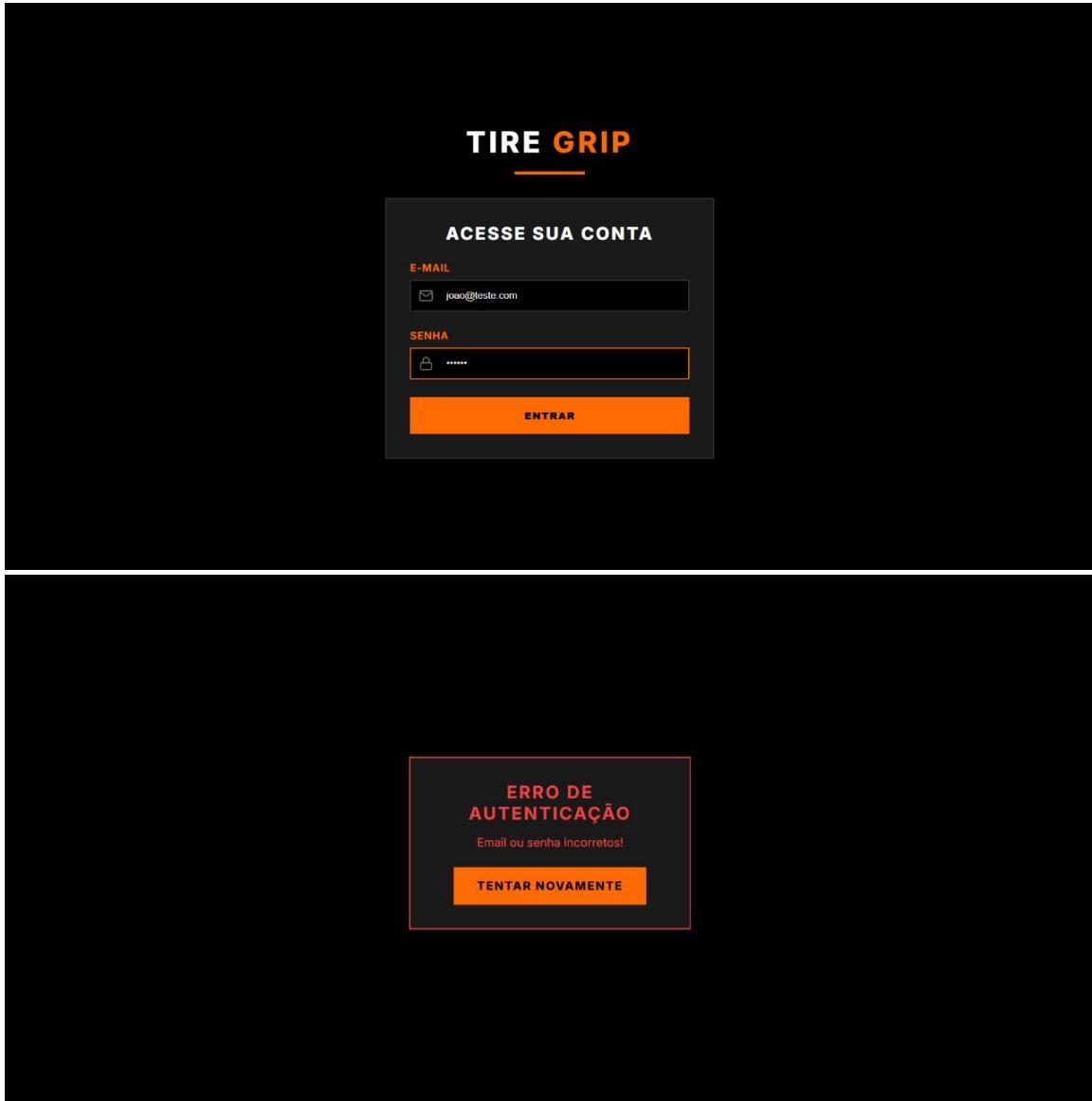
    // Converter para JSON usando Gson
    Gson gson = new Gson();
    String json = gson.toJson(produtos);

    out.print(json);

} catch (Exception ex) {
    response.setStatus(HttpServletRequest.SC_INTERNAL_SERVER_ERROR);
    out.print("{\"error\": \"" + ex.getMessage() + "\"}");
}

%>
```


4. EVIDÊNCIAS DO SISTEMA EM AÇÃO



PERFORMANCE MÁXIMA NA ESTRADA

Pneus de alta qualidade com tecnologia de ponta para garantir sua segurança e desempenho em qualquer terreno.

[VER PRODUTOS](#)**GARANTIA TOTAL**

12 meses de garantia em todos os produtos

**ENTREGA RÁPIDA**

Receba em até 48h após a confirmação

**MÁXIMA SEGURANÇA**

Produtos certificados e testados

NOSSOS PRODUTOS

NOSSOS PRODUTOS

**PNEU PERFORMANCE MAX**

Modelo Sport 2024 - Alta Performance

R\$ 599.99

ADICIONAR AO CARRINHO

**PNEU OFF-ROAD PRO**

Modelo Adventure Pro

R\$ 699.99

ADICIONAR AO CARRINHO

**PNEU ECONÔMICO PLUS**

Modelo Eco 2024

R\$ 399.99

ADICIONAR AO CARRINHO

**PNEU PREMIUM COMFORT**

Modelo Luxury Ride

R\$ 799.99

ADICIONAR AO CARRINHO

**PNEU ALL-TERRAIN**

Modelo Multi-Surface

R\$ 649.99

ADICIONAR AO CARRINHO

TIRE GRIP

PRODUTOS



[← Voltar](#)



PNEU PERFORMANCE MAX

Modelo Sport 2024 - Alta Performance

Preço à vista no PIX
R\$ 599,99
ou 10x de R\$ 60,00 sem juros

 **COMPRAR AGORA**

ESPECIFICAÇÕES TÉCNICAS

Largura	205
Perfil	55
Aro	16"
Índice de Carga	91
Índice de Velocidade	V (240 km/h)

TIRE GRIP

PRODUTOS



Largura	215
Perfil	65
Aro	17"
Índice de Carga	95
Índice de Velocidade	H (210 km/h)
Tipo	Radial

CARACTERÍSTICAS

- ✓ Tecnologia de drenagem de água avançada
- ✓ Composto de borracha de alta aderência
- ✓ Baixo nível de ruído durante a condução
- ✓ Maior durabilidade e resistência ao desgaste
- ✓ Economia de combustível otimizada
- ✓ Desempenho em todas as estações

TIRE GRIP

PRODUTOS



[← Voltar](#)



PNEU PERFORMANCE MAX
Modelo Sport 2024 - Alta Performance

Preço à vista no PIX
R\$ 599,99
ou 10x de R\$ 60,00 sem juros

 **COMPRAR AGORA**

ESPECIFICAÇÕES TÉCNICAS

Largura	205
Perfil	55
Aro	16"
Índice de Carga	91
Índice de Velocidade	V (240 km/h)

TIRE GRIP

PRODUTOS



MEU CARRINHO

 **PNEU ECONÔMICO PLUS**
Modelo Eco 2024

Quantidade: 1   

R\$ 399,99

RESUMO DO PEDIDO

Subtotal	R\$ 399,99
Frete	R\$ 50,00
TOTAL	R\$ 449,99

 **FINALIZAR COMpra**

CONTINUAR COMPRANDO

CONTATO
Email: contato@tiregrip.com

LINKS ÚTEIS
FAQ

NEWSLETTER
Receba ofertas exclusivas e novidades

TIRE GRIP

PRODUTOS

ENDEREÇO 2 RESUMO 3 PAGAMENTO

RESUMO DO PEDIDO

Pneu Performance Modelo Sport 2024 Qtd: 2	R\$ 599.98
Pneu Off-road Modelo Adventure Pro Qtd: 1	R\$ 399.99
VOLTAR CONTINUAR	

RESUMO

Subtotal	R\$ 999.97
Frete	R\$ 50.00
TOTAL	R\$ 1049.97

CONTATO
Email: contato@tiregrip.com
Tel: (11) 9999-9999
WhatsApp: (11) 98888-8888

LINKS ÚTEIS
FAQ
Sobre Nós
Política de Privacidade

NEWSLETTER
Receba ofertas exclusivas e novidades

TIRE GRIP

PRODUTOS

ENDEREÇO 2 RESUMO 3 PAGAMENTO

FORMA DE PAGAMENTO

Cartão de Crédito Em até 10x sem juros
PIX 5% de desconto
VOLTAR FINALIZAR PEDIDO

RESUMO

Subtotal	R\$ 999.97
Frete	R\$ 50.00
TOTAL	R\$ 1049.97

CONTATO
Email: contato@tiregrip.com
Tel: (11) 9999-9999
WhatsApp: (11) 98888-8888

LINKS ÚTEIS
FAQ
Sobre Nós
Política de Privacidade

NEWSLETTER
Receba ofertas exclusivas e novidades

TIRE GRIP

PRODUTOS

0

ENDEREÇO

RESUMO

PAGAMENTO

FORMA DE PAGAMENTO

Cartão de Crédito
Em até 10x sem juros

PIX
5% de desconto

VOLTAR

FINALIZAR PEDIDO

RESUMO

Subtotal	R\$ 999.97
Frete	R\$ 50.00
Desconto PIX (5%)	- R\$ 52.50
TOTAL	R\$ 997.47

CONTATO

Email: contato@tiregrip.com
Tel: (11) 9999-9999
WhatsApp: (11) 99999-9999

LINKS ÚTEIS

FAQ
Sobre Nós
Política de Privacidade

NEWSLETTER

Receba ofertas exclusivas e novidades

Seu e-mail:

