

Centro Universitário SENAC

Desenvolvimento de Sistemas Orientados a Web

Projeto: Sistema de E-commerce (Cafeteria)

Integrantes:

Nicolas Oliveira Nascimento – RA: 1142020874

Paulo Eduardo Messias Grispan – RA: 1142944524

Allan Ribeiro de Souza – RA: 1141581353

Wallace Araújo da Silva – RA: 1142955905

Arthur Vitalino Santos – RA: 114259183

Micael Cadete da Silva Cosme – RA: 1142019443

São Paulo – 2025

SUMÁRIO

1. INTRODUÇÃO (ESCOPO).....	3
2. DESENVOLVIMENTO DO PROJETO.....	4
2.1 Funcionalidades Implementadas	4
2.2 Telas e Códigos (Front-End)	5
2.3 Códigos (Back-End).....	33
2.3.1 Controller.....	33
2.3.2 Models.....	46
2.3.3 Enums	51
2.3.4 Repositories	52
2.3.5 Services.....	54
2.3.6 Config	59
2.3.7 dtos	64
2.3.8 Security	65
2.4 Design Pattern Adotado	69
2.5 Diagrama de Casos de Uso.....	70
2.6 Modelo Lógico.....	71
3. CONCLUSÃO	72

1. INTRODUÇÃO (ESCOPO)

O presente documento tem como objetivo apresentar o desenvolvimento de um sistema de e-commerce voltado para aprimorar a interação entre vendedores e clientes no ambiente digital.

O sistema permite que vendedores anunciem seus produtos de forma prática, com funcionalidades de criação, edição, exclusão e atualização de anúncios (operações CRUD).

Já os clientes têm a possibilidade de visualizar os produtos, adicioná-los ao carrinho de compras e realizar pedidos de maneira rápida e segura.

Além disso, foi implementada a aba de sugestões, onde os clientes podem enviar feedbacks e recomendações diretamente aos vendedores, visando melhorar a qualidade do atendimento e dos produtos oferecidos.

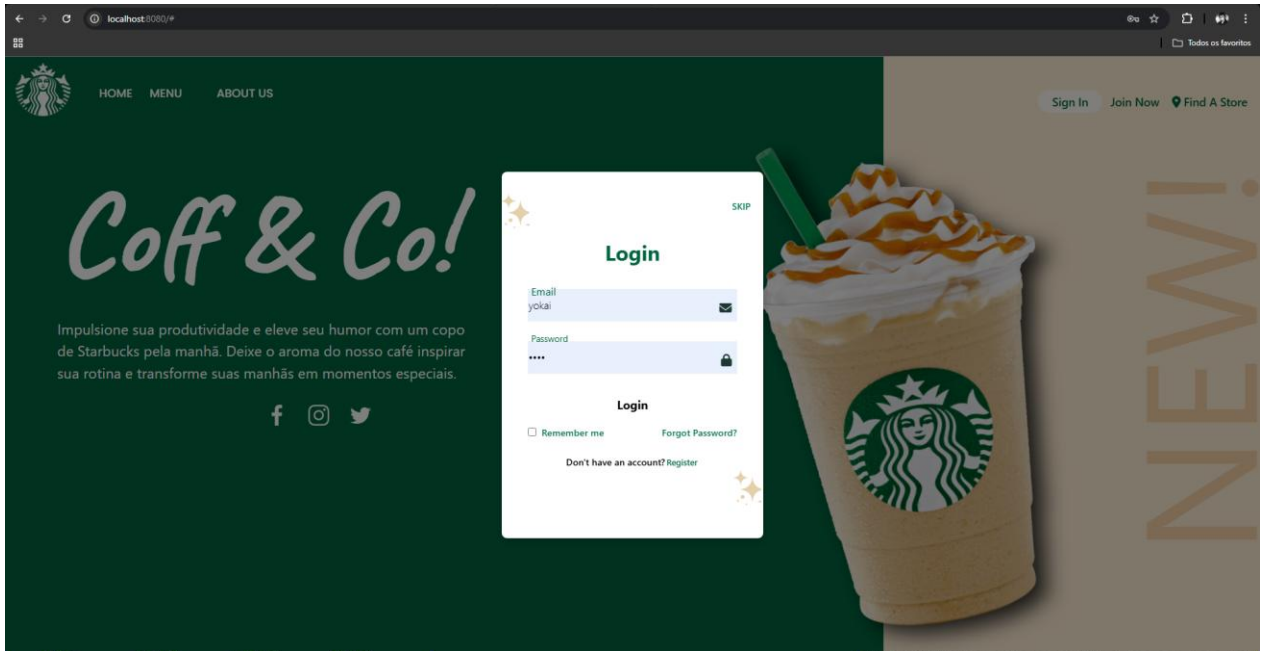
2. DESENVOLVIMENTO DO PROJETO

2.1 Funcionalidades Implementadas

- Usuário deve conseguir se cadastrar como Funcionário ou Cliente
- Usuário deve conseguir fazer login em sua área exclusiva
- Funcionário deve conseguir visualizar seu perfil
- Funcionário deve conseguir realizar o cadastro de produtos
- Funcionário deve conseguir realizar a edição de produtos
- Funcionário deve conseguir realizar a exclusão de produtos
- Funcionário deve ser capaz de gerenciar pedidos
- Funcionário deve ser capaz de visualizar sugestões
- Cliente deve conseguir adicionar produtos ao carrinho
- Cliente deve conseguir personalizar seu pedido
- Cliente deve conseguir concluir sua respectiva compra
- Cliente deve conseguir visualizar seu perfil
- Clientes deve conseguir efetuar a sugestão de produtos

2.2 Telas e Códigos (Front-End)

Tela de login + Tela inicial



```
<!DOCTYPE html>
<html lang="pt-BR" xmlns:th="http://www.thymeleaf.org">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login - Cafeteria</title>

  <link rel="icon" type="image/x-icon" th:href="@{/favicon.ico}">
  <link rel="stylesheet" th:href="@{/css/style.css}">
  <link
href="https://fonts.googleapis.com/css?family=Roboto:300,400,500&display=swap"
rel="stylesheet">
  <link
href="https://fonts.googleapis.com/css?family=Noto+Sans&display=swap"
rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome@6.2.0/css/all.min.css"
integrity="sha512-
xh60/CkQoPOWDDdYTDqRdPCVd1SpvCA9XXcUnZS2FmJNp1coAFzvtCN9BmamE+4aHK8yyUHUSCcJHg
XloTyT2A=="
crossorigin="anonymous" referrerpolicy="no-referrer" />
</head>

<body>
  <div class="wrapper">
    <div class="nav__logo">
```

```

        <a th:href="@{/}"></a>
    </div>
    <header class="nav">
        <nav>
            <ul class="nav__links">
                <li class="nav__link"><a href="#">Home</a></li>
                <li></li><a class="nav-link-custom"
th:href="@{/menu}">Menu</li>
                <li class="nav__link"><a th:href="@{/about}">About
Us</a></li>
            </ul>
            <div class="navbar_auth">
                <button id="signInBtn" class="sign-in">Sign In</button>
                <a id="joinNowBtn" href="#">Join Now</a>
                <a href="#" id="findStoreBtn">
                    <i class="fa-solid fa-location-dot"></i>
                    <span>Find A Store</span>
                </a>
            </div>
        </nav>
    </header>
    <div class="mockup_content">
        <div class="mockup_texts">
            <h2 class="mockup_title">Coff & Co!</h2>
            <p class="mockup_paragrafo">Impulsione sua produtividade e
eleve seu humor com um copo de Starbucks
                pela manhã.
                Deixe o aroma do nosso café inspirar sua rotina e
transforme suas manhãs em momentos especiais.</p>
            <div class="social_midias">
                <div class="social_midia">
                    <a href="#"></a>
                </div>
                <div class="social_midia">
                    <a href="#"></a>
                </div>
                <div class="social_midia">
                    <a href="#"></a>
                </div>
            </div>
            <div class="mockup_image"></div>
        </div>

        <!-- Popup de Cadastro - Oculto inicialmente -->
        <div class="cdk-overlay-pane" id="locationPopup" style="display:
none;">
            <div class="mat-dialog-container">
                <div class="gps-alert-box">

```

```

        <div class="form-box">
            <h2>Sign-up</h2>
        </div>

        <!-- Mensagem de feedback para cadastro -->
        <div id="cadastroMensagem" style="display: none; margin:
10px; padding: 10px; border-radius: 4px;">
        </div>

        <div class="card-remove-wrapper2">
            <form th:action="@{/cadastro}" method="post"
id="registerForm">
                <div class="input-box">
                    <span class="icon"><i class="fa-solid fa-
user"></i></span>
                    <input type="text" id="name" name="nome"
required>
                    <label for="name">Name</label>
                </div>

                <div class="input-box">
                    <span class="icon"><i class="fa-solid fa-
envelope"></i></span>
                    <input type="email" id="email" name="email"
required placeholder=" ">
                    <label for="email">Email</label>
                </div>

                <div class="input-box">
                    <span class="icon"><i class="fa-solid fa-
lock"></i></span>
                    <input type="password" id="senha" name="senha"
required>
                    <label for="senha">Password</label>
                </div>

                <div class="input-box">
                    <span class="icon"><i class="fa-solid fa-map-
marker-alt"></i></span>
                    <input type="text" id="endereco"
name="endereco" required>
                    <label for="endereco">Address</label>
                </div>

                <button type="submit"
class="btn">Register</button>
            </form>
        </div>
        <div class="close">
            <button type="button" class="close-btn"
id="closePopupBtn">Skip</button>
        </div>
    </div>
</div>

    <!-- Popup de Login - Oculto inicialmente -->
    <div class="cdk-overlay-pane2" id="locationPopup2" style="display:
none;">

```

```

<div class="mat-dialog-container2">
  <div class="gps-alert-box2">

    
    
    <div class="form-box">
      <h2>Login</h2>
    </div>

    <!-- Mensagem de erro do Thymeleaf -->
    <div th:if="{param.error}" class="alert alert-danger"
style="margin: 10px; padding: 10px; background:
#f8d7da; color: #721c24; border-radius: 4px;">
      Email ou senha inválidos!
    </div>

    <!-- Mensagem de sucesso do cadastro -->
    <div th:if="{sucessoCadastro}" class="alert alert-
success"
style="margin: 10px; padding: 10px; background:
#d4edda; color: #155724; border-radius: 4px;">
      <span th:text="{sucessoCadastro}"></span>
    </div>

    <div class="close2">
      <button type="button" class="close-btn2"
id="closePopupBtn2">Skip</button>
    </div>

    <div class="card-remove-wrapper3">
      <form th:action="@{/login}" method="post">
        <div class="input-box2">
          <span class="icon"><i class="fa-solid fa-
envelope"></i></span>
          <input type="email" id="username"
name="username" required placeholder=" ">
          <label for="username">Email</label>
        </div>

        <div class="input-box2">
          <span class="icon"><i class="fa-solid fa-
lock"></i></span>
          <input type="password" id="password"
name="password" required>
          <label for="password">Password</label>
        </div>

        <button type="submit" class="btn2">Login</button>

        <div class="remember">
          <label><input type="checkbox" name="remember-
me"> Remember me</label>
          <a href="#">Forgot Password?</a>
        </div>

        <div class="login-registeter">
          <p>Don't have an account?<a id="registerLink"

```

```

href="#"
class="register-link">Register</a></p>
    </div>
  </form>
</div>
</div>
</div>
</div>
</div>

<div id="modalOverlay" style="display: none;"></div>
</div>

<script th:src="@{/js/script.js}"></script>

<script>
  // Script para controlar a exibição dos popups e validação
  document.addEventListener('DOMContentLoaded', function () {
    const signInBtn = document.getElementById('signInBtn');
    const joinNowBtn = document.getElementById('joinNowBtn');
    const loginPopup = document.getElementById('loginPopup2');
    const registerPopup = document.getElementById('registerPopup');
    const closePopupBtn = document.getElementById('closePopupBtn');
    const closePopupBtn2 = document.getElementById('closePopupBtn2');
    const modalOverlay = document.getElementById('modalOverlay');
    const registerLink = document.getElementById('registerLink');
    const registerForm = document.getElementById('registerForm');
    const cadastroMensagem =
document.getElementById('cadastroMensagem');

    // Abrir popup de CADASTRO ao clicar no botão "Sign In"
    if (signInBtn && registerPopup) {
      signInBtn.addEventListener('click', function () {
        registerPopup.style.display = 'block';
        modalOverlay.style.display = 'block';
        // Limpar mensagens anteriores
        cadastroMensagem.style.display = 'none';
      });
    }

    // Função para fechar todos os popups
    function closeAllPopups() {
      if (loginPopup) loginPopup.style.display = 'none';
      if (registerPopup) registerPopup.style.display = 'none';
      if (modalOverlay) modalOverlay.style.display = 'none';
    }

    // Abrir popup de LOGIN ao clicar no botão "Join Now"
    if (joinNowBtn && loginPopup) {
      joinNowBtn.addEventListener('click', function () {
        loginPopup.style.display = 'block';
        modalOverlay.style.display = 'block';
      });
    }

    // Alternar entre popups (do login para cadastro)
    if (registerLink && registerPopup && loginPopup) {
      registerLink.addEventListener('click', function (e) {
        e.preventDefault();
        loginPopup.style.display = 'none';
      });
    }
  });

```

```

        registerPopup.style.display = 'block';
        // Limpar mensagens anteriores
        cadastroMensagem.style.display = 'none';
    });
}

// Fechar popup de cadastro
if (closePopupBtn && registerPopup) {
    closePopupBtn.addEventListener('click', function () {
        registerPopup.style.display = 'none';
        modalOverlay.style.display = 'none';
    });
}

// Fechar popup de login
if (closePopupBtn2 && loginPopup) {
    closePopupBtn2.addEventListener('click', function () {
        loginPopup.style.display = 'none';
        modalOverlay.style.display = 'none';
    });
}

// Fechar ao clicar fora do popup
if (modalOverlay) {
    modalOverlay.addEventListener('click', function () {
        if (loginPopup && loginPopup.style.display === 'block') {
            loginPopup.style.display = 'none';
        }
        if (registerPopup && registerPopup.style.display ===
'block') {
            registerPopup.style.display = 'none';
        }
        modalOverlay.style.display = 'none';
    });
}

// Fechar popups com a tecla ESC
document.addEventListener('keydown', function (e) {
    if (e.key === 'Escape') {
        closeAllPopups();
    }
});

// Validação simples do formulário de cadastro
if (registerForm) {
    registerForm.addEventListener('submit', function (e) {
        const nome = document.getElementById('name').value;
        const email = document.getElementById('email').value;
        const senha = document.getElementById('senha').value;
        const endereco =
document.getElementById('endereco').value;

        // Validação básica
        if (!nome || !email || !senha || !endereco) {
            e.preventDefault();
            mostrarMensagem('Por favor, preencha todos os
campos.', 'erro');
            return false;
        }
    });
}

```

```

        // Validação de email simples
        if (!validarEmail(email)) {
            e.preventDefault();
            mostrarMensagem('Por favor, insira um email válido.',
'erro');

            return false;
        }

        // Se tudo estiver correto, o formulário será enviado
        mostrarMensagem('Cadastro realizado com sucesso!',
'sucesso');
    });
}

// Função para validar email
function validarEmail(email) {
    const re = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    return re.test(email);
}

// Função para mostrar mensagens de feedback
function mostrarMensagem(mensagem, tipo) {
    cadastroMensagem.textContent = mensagem;
    cadastroMensagem.style.display = 'block';

    if (tipo === 'erro') {
        cadastroMensagem.style.backgroundColor = '#f8d7da';
        cadastroMensagem.style.color = '#721c24';
        cadastroMensagem.style.border = '1px solid #f5c6cb';
    } else {
        cadastroMensagem.style.backgroundColor = '#d4edda';
        cadastroMensagem.style.color = '#155724';
        cadastroMensagem.style.border = '1px solid #c3e6cb';
    }

    // Rolagem suave para a mensagem
    cadastroMensagem.scrollIntoView({ behavior: 'smooth', block:
'center' });
}

    });
</script>
</body>
</html>

```

Tela de cadastro produtos

The screenshot shows a web browser at localhost:8080/admin/produtos/novo. The page has a dark green header with the 'Cafeteria' logo and navigation links: 'Início', 'Cardápio', and 'Dashboard'. A user profile 'funcionario@cafe.com' is logged in. The main content area is titled 'Cadastrar Novo Produto' and contains a form with the following fields: 'Nome do Produto *' (text input), 'Preço *' (number input), 'Imagem do Produto' (file upload with 'Escolher arquivo' and 'Nenhum arquivo escolhido' buttons), and a 'Descrição' (text area). At the bottom of the form are 'Salvar Produto' and 'Cancelar' buttons. The footer is dark green and contains a 'Menu' (Início, Cardápio, Cadastro), 'Siga-nos' (Facebook, Instagram, Twitter), and copyright information: '© 2025 Cafeteria — Todos os direitos reservados' and 'Feito com ❤️ • Política de privacidade'.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Novo Produto</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
  <div th:replace="~{fragments/header}"></div>

  <div class="container mt-5">
    <h1>Cadastrar Novo Produto</h1>

    <form th:action="@{/admin/produtos/novo}" method="post"
enctype="multipart/form-data" class="mt-4">
      <div class="row">
        <div class="col-md-6">
          <div class="mb-3">
            <label for="nome" class="form-label">Nome do Produto
*</label>
            <input type="text" class="form-control" id="nome"
name="nome" required>
          </div>

          <div class="mb-3">
            <label for="preco" class="form-label">Preço *</label>
            <input type="number" step="0.01" class="form-control"
id="preco" name="preco" required>
          </div>
        </div>
        <div class="col-md-6">
          <div class="mb-3">
            <label for="descricao" class="form-label">Descrição
*</label>
            <input type="text" class="form-control" id="descricao"
name="descricao" required>
          </div>
        </div>
      </div>
      <div class="d-flex justify-content-end gap-2">
        <button type="submit" class="btn btn-primary">Salvar Produto</button>
        <button type="button" class="btn btn-secondary">Cancelar</button>
      </div>
    </form>
  </div>
</body>
</html>
```

```

        </div>

        <div class="mb-3">
            <label for="imagem" class="form-label">Imagem do
Produto</label>
            <input type="file" class="form-control" id="imagem"
name="imagem" accept="image/*">
        </div>
    </div>

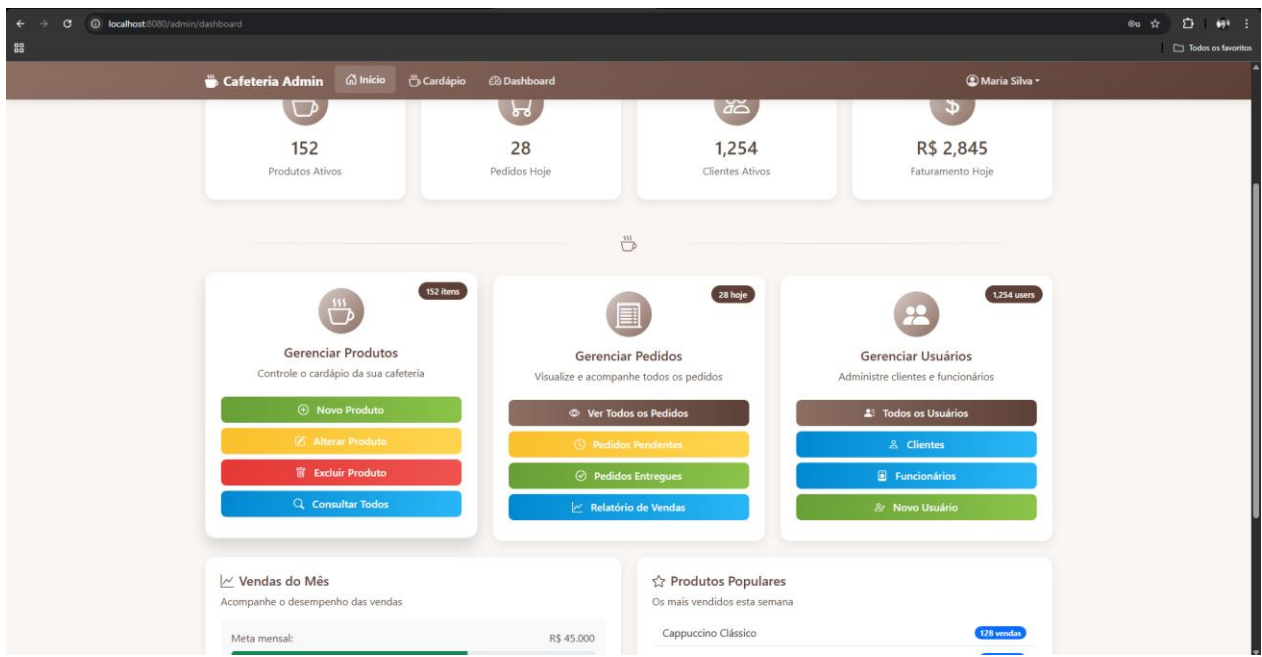
    <div class="col-md-6">
        <div class="mb-3">
            <label for="descricao" class="form-
label">Descrição</label>
            <textarea class="form-control" id="descricao"
name="descricao" rows="5"></textarea>
        </div>
    </div>
</div>

    <button type="submit" class="btn btn-success">Salvar
Produto</button>
    <a th:href="@{/admin/dashboard}" class="btn btn-
secondary">Cancelar</a>
</form>
</div>

<div th:replace="~{fragments/footer}"></div>
</body>
</html>

```

Tela de gerenciamento



```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sistema de Cafeteria</title>
  <!-- Bootstrap 5 -->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <!-- Ícones Bootstrap -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap-icons/font/bootstrap-
icons.css" rel="stylesheet">
  <style>
    :root {
      --primary-color: #8d6e63;
      --secondary-color: #d7ccc8;
      --accent-color: #5d4037;
      --light-color: #f5f5f5;
      --dark-color: #4e342e;
    }

    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background-color: #f9f6f3;
      color: #5a4c42;
      padding-top: 80px;
    }

    .navbar-custom {
```

```
        background: linear-gradient(135deg, var(--primary-color) 0%, var(--accent-color) 100%);
        box-shadow: 0 2px 15px rgba(0,0,0,0.1);
        padding: 0.5rem 1rem;
    }

    .navbar-brand {
        font-weight: 700;
        color: white !important;
        display: flex;
        align-items: center;
    }

    .nav-link {
        color: rgba(255, 255, 255, 0.85) !important;
        font-weight: 500;
        transition: all 0.3s ease;
        padding: 0.5rem 1rem !important;
        border-radius: 5px;
        margin: 0 2px;
        display: flex;
        align-items: center;
    }

    .nav-link:hover, .nav-link.active {
        color: white !important;
        background-color: rgba(255, 255, 255, 0.15);
        transform: translateY(-2px);
    }

    .nav-link i {
        margin-right: 8px;
        font-size: 1.1rem;
    }

    .navbar-toggler {
        border: none;
        color: white !important;
    }

    .dashboard-card {
        background: white;
        border-radius: 15px;
        overflow: hidden;
        transition: transform 0.3s ease, box-shadow 0.3s ease;
        border: none;
        box-shadow: 0 5px 15px rgba(0,0,0,0.08);
        height: 100%;
    }

    .dashboard-card:hover {
        transform: translateY(-5px);
        box-shadow: 0 12px 25px rgba(0,0,0,0.15);
    }

    .card-icon {
        font-size: 2.5rem;
        background: linear-gradient(135deg, var(--secondary-color) 0%,
var(--primary-color) 100%);
```

```
        color: white;
        width: 70px;
        height: 70px;
        border-radius: 50%;
        display: flex;
        align-items: center;
        justify-content: center;
        margin: 0 auto 1rem;
    }

    .card-title {
        font-weight: 600;
        color: var(--accent-color);
        margin-bottom: 0.5rem;
    }

    .btn-dashboard {
        border-radius: 8px;
        padding: 0.5rem 1rem;
        font-weight: 500;
        transition: all 0.3s ease;
        border: none;
        width: 100%;
        margin-bottom: 0.5rem;
    }

    .btn-insert {
        background: linear-gradient(to right, #689f38, #8bc34a);
        color: white;
    }

    .btn-update {
        background: linear-gradient(to right, #fbc02d, #ffd54f);
        color: white;
    }

    .btn-delete {
        background: linear-gradient(to right, #e53935, #ef5350);
        color: white;
    }

    .btn-view {
        background: linear-gradient(to right, #0288d1, #29b6f6);
        color: white;
    }

    .btn-manage {
        background: linear-gradient(to right, var(--primary-color), var(--
accent-color));
        color: white;
    }

    .btn-dashboard:hover {
        transform: translateY(-2px);
        box-shadow: 0 5px 15px rgba(0,0,0,0.2);
        color: white;
    }

    .stats-badge {
```

```
        position: absolute;
        top: 15px;
        right: 15px;
        background: var(--accent-color);
        color: white;
        border-radius: 20px;
        padding: 0.25rem 0.75rem;
        font-size: 0.8rem;
        font-weight: 500;
    }

    .coffee-divider {
        display: flex;
        align-items: center;
        justify-content: center;
        margin: 2rem 0;
    }

    .coffee-divider:before,
    .coffee-divider:after {
        content: "";
        flex: 1;
        height: 1px;
        background: linear-gradient(to right, transparent, var(--
secondary-color), transparent);
    }

    .coffee-divider i {
        margin: 0 1rem;
        color: var(--primary-color);
        font-size: 1.5rem;
    }

    .footer {
        background: var(--dark-color);
        color: white;
        padding: 2rem 0;
        margin-top: 3rem;
    }

    .user-info {
        display: flex;
        align-items: center;
        color: white;
        margin-left: 15px;
        padding: 5px 12px;
        background: rgba(255, 255, 255, 0.15);
        border-radius: 20px;
    }

    .user-info i {
        margin-right: 8px;
    }

    @media (max-width: 991.98px) {
        .user-info {
            margin: 10px 0;
            justify-content: center;
        }
    }
```

```

        .navbar-nav {
            padding: 10px 0;
        }

        .nav-link {
            justify-content: center;
        }

        body {
            padding-top: 70px;
        }
    }
</style>
</head>
<body>
    <!-- Header/Navbar -->
    <nav class="navbar navbar-expand-lg navbar-custom fixed-top">
        <div class="container">
            <a class="navbar-brand" href="/">
                <i class="bi bi-cup-hot-fill me-2"></i>Cafeteria Admin
            </a>

            <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarContent">
                <span class="navbar-toggler-icon"></span>
            </button>

            <div class="collapse navbar-collapse" id="navbarContent">
                <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                    <li class="nav-item">
                        <a class="nav-link active" th:href="@{/}">
                            <i class="bi bi-house-door me-1"></i> Início
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" th:href="@{/menu}">
                            <i class="bi bi-cup-hot me-1"></i> Cardápio
                        </a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" th:href="@{/admin/dashboard}">
                            <i class="bi bi-speedometer2 me-1"></i> Dashboard
                        </a>
                    </li>
                </ul>

                <ul class="navbar-nav ms-auto">
                    <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#"
id="navbarDropdown" role="button" data-bs-toggle="dropdown">
                            <i class="bi bi-person-circle me-1"></i> Maria
Silva
                        </a>
                        <ul class="dropdown-menu dropdown-menu-end">
                            <li><a class="dropdown-item" href="/perfil"><i
class="bi bi-person me-2"></i> Perfil</a></li>
                            <li><a class="dropdown-item" href="/pedidos"><i

```

```

class="bi bi-clock-history me-2"></i> Meus Pedidos</a></li>
      <li><hr class="dropdown-divider"></li>
      <li><a class="dropdown-item" href="/logout"><i
class="bi bi-box-arrow-right me-2"></i> Sair</a></li>
    </ul>
  </li>
</ul>
</div>
</div>
</nav>

<!-- Conteúdo Principal -->
<div class="container mt-5">
  <!-- Dashboard Header -->
  <div class="row mb-4">
    <div class="col-12">
      <div class="p-4 text-center rounded" style="background:
linear-gradient(135deg, var(--primary-color) 0%, var(--accent-color) 100%);
color: white;">
        <h1 class="display-5 fw-bold"><i class="bi bi-speedometer2
me-2"></i>Dashboard Administrativo</h1>
        <p class="lead">Gerencie produtos, pedidos e usuários da
sua cafeteria</p>
      </div>
    </div>
  </div>
</div>

<!-- Stats Overview -->
<div class="row mb-4">
  <div class="col-md-3 mb-3">
    <div class="dashboard-card p-3 text-center">
      <div class="card-icon">
        <i class="bi bi-cup-hot"></i>
      </div>
      <h3>152</h3>
      <p class="text-muted">Produtos Ativos</p>
    </div>
  </div>
  <div class="col-md-3 mb-3">
    <div class="dashboard-card p-3 text-center">
      <div class="card-icon">
        <i class="bi bi-cart"></i>
      </div>
      <h3>28</h3>
      <p class="text-muted">Pedidos Hoje</p>
    </div>
  </div>
  <div class="col-md-3 mb-3">
    <div class="dashboard-card p-3 text-center">
      <div class="card-icon">
        <i class="bi bi-people"></i>
      </div>
      <h3>1,254</h3>
      <p class="text-muted">Clientes Ativos</p>
    </div>
  </div>
  <div class="col-md-3 mb-3">
    <div class="dashboard-card p-3 text-center">
      <div class="card-icon">

```

```

        <i class="bi bi-currency-dollar"></i>
      </div>
      <h3>R$ 2,845</h3>
      <p class="text-muted">Faturamento Hoje</p>
    </div>
  </div>
</div>

<div class="coffee-divider">
  <i class="bi bi-cup-hot"></i>
</div>

<!-- Main Dashboard Cards -->
<div class="row g-4">
  <!-- Produtos -->
  <div class="col-md-4">
    <div class="dashboard-card">
      <div class="card-body p-4 text-center position-relative">
        <span class="stats-badge">152 itens</span>
        <div class="card-icon">
          <i class="bi bi-cup-hot"></i>
        </div>
        <h5 class="card-title">Gerenciar Produtos</h5>
        <p class="card-text">Controle o cardápio da sua
cafeteria</p>
        <div class="mt-4">
          <a href="/admin/produtos/novo" class="btn btn-
dashboard btn-insert">
            <i class="bi bi-plus-circle me-2"></i> Novo
Produto
          </a>
          <a href="/admin/produtos/alterar" class="btn btn-
dashboard btn-update">
            <i class="bi bi-pencil-square me-2"></i>
Alterar Produto
          </a>
          <a href="/admin/produtos/excluir" class="btn btn-
dashboard btn-delete">
            <i class="bi bi-trash me-2"></i> Excluir
Produto
          </a>
          <a href="/admin/produtos" class="btn btn-dashboard
btn-view">
            <i class="bi bi-search me-2"></i> Consultar
Todos
          </a>
        </div>
      </div>
    </div>
  </div>
</div>

<!-- Pedidos -->
<div class="col-md-4">
  <div class="dashboard-card">
    <div class="card-body p-4 text-center position-relative">
      <span class="stats-badge">28 hoje</span>
      <div class="card-icon">
        <i class="bi bi-receipt-cutoff"></i>
      </div>
    </div>
  </div>
</div>

```

```

        <h5 class="card-title">Gerenciar Pedidos</h5>
        <p class="card-text">Visualize e acompanhe todos os
pedidos</p>

        <div class="mt-4">
            <a href="/admin/pedidos" class="btn btn-dashboard
btn-manage">

                <i class="bi bi-eye me-2"></i> Ver Todos os
Pedidos

                </a>
            <a href="/admin/pedidos/pendentes" class="btn btn-
dashboard btn-update">

                <i class="bi bi-clock me-2"></i> Pedidos
Pendentes

                </a>
            <a href="/admin/pedidos/entregues" class="btn btn-
dashboard btn-insert">

                <i class="bi bi-check-circle me-2"></i>
Pedidos Entregues

                </a>
            <a href="/admin/pedidos/relatorio" class="btn btn-
dashboard btn-view">

                <i class="bi bi-graph-up me-2"></i> Relatório
de Vendas

                </a>
        </div>
    </div>
</div>
</div>
<!-- Usuários -->
<div class="col-md-4">
    <div class="dashboard-card">
        <div class="card-body p-4 text-center position-relative">
            <span class="stats-badge">1,254 users</span>
            <div class="card-icon">
                <i class="bi bi-people-fill"></i>
            </div>
            <h5 class="card-title">Gerenciar Usuários</h5>
            <p class="card-text">Administre clientes e
funcionários</p>

            <div class="mt-4">
                <a href="/admin/usuarios" class="btn btn-dashboard
btn-manage">

                    <i class="bi bi-person-lines-fill me-2"></i>
Todos os Usuários

                    </a>
                <a href="/admin/usuarios/clientes" class="btn btn-
dashboard btn-view">

                    <i class="bi bi-person me-2"></i> Clientes
                    </a>
                <a href="/admin/usuarios/funcionarios" class="btn
btn-dashboard btn-view">

                    <i class="bi bi-person-badge me-2"></i>
Funcionários

                    </a>
                <a href="/admin/usuarios/novo" class="btn btn-
dashboard btn-insert">

                    <i class="bi bi-person-plus me-2"></i> Novo
Usuário

```

```

        </a>
      </div>
    </div>
  </div>
</div>

<!-- Additional Cards Row -->
<div class="row mt-4">
  <div class="col-md-6 mb-4">
    <div class="dashboard-card">
      <div class="card-body p-4">
        <h5 class="card-title"><i class="bi bi-graph-up me-2"></i>Vendas do Mês</h5>
        <p class="card-text">Acompanhe o desempenho das vendas</p>

        <div class="mt-3 bg-light p-3 rounded">
          <div class="d-flex justify-content-between mb-2">
            <span>Meta mensal:</span>
            <span>R$ 45.000</span>
          </div>
          <div class="progress mb-3" style="height: 10px;">
            <div class="progress-bar bg-success"
              role="progressbar" style="width: 65%" aria-valuenow="65" aria-valuemin="0"
              aria-valuemax="100"></div>
          </div>
          <div class="d-flex justify-content-between">
            <span>Atual:</span>
            <span>R$ 29.250</span>
          </div>
        </div>
        <a href="#" class="btn btn-sm btn-outline-primary mt-3">Ver Relatório Completo</a>
      </div>
    </div>
  </div>
  <div class="col-md-6 mb-4">
    <div class="dashboard-card">
      <div class="card-body p-4">
        <h5 class="card-title"><i class="bi bi-star me-2"></i>Produtos Populares</h5>
        <p class="card-text">Os mais vendidos esta semana</p>
        <ul class="list-group list-group-flush mt-3">
          <li class="list-group-item d-flex justify-content-between align-items-center">
            Cappuccino Clássico
            <span class="badge bg-primary rounded-pill">128 vendas</span>
          </li>
          <li class="list-group-item d-flex justify-content-between align-items-center">
            Croissant de Chocolate
            <span class="badge bg-primary rounded-pill">97 vendas</span>
          </li>
          <li class="list-group-item d-flex justify-content-between align-items-center">
            Café Expresso
            <span class="badge bg-primary rounded-pill">85

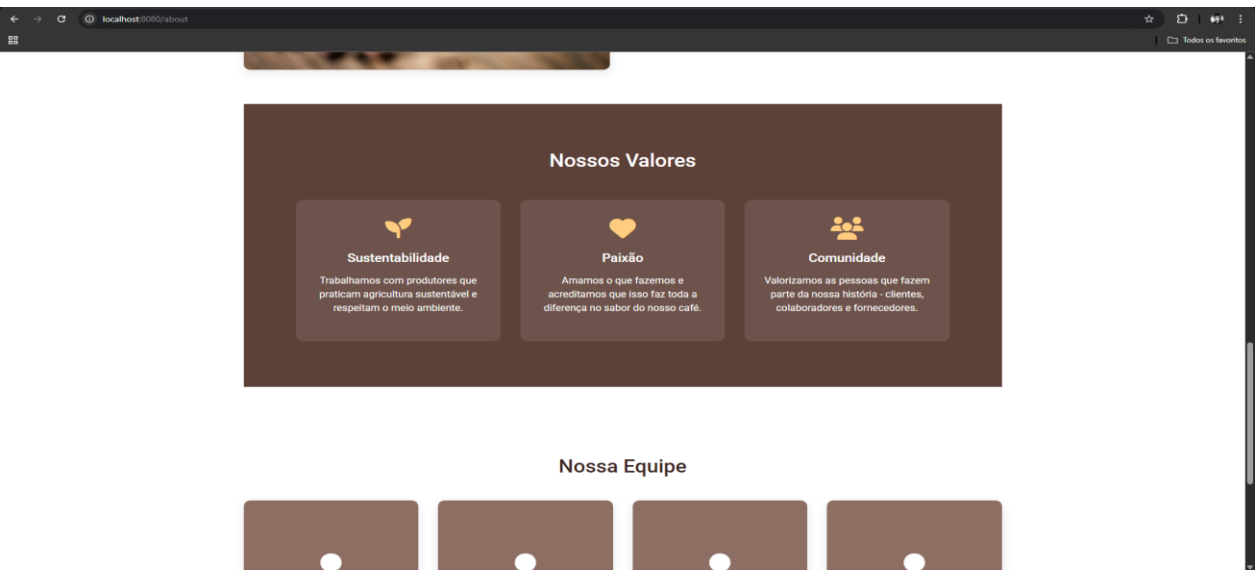
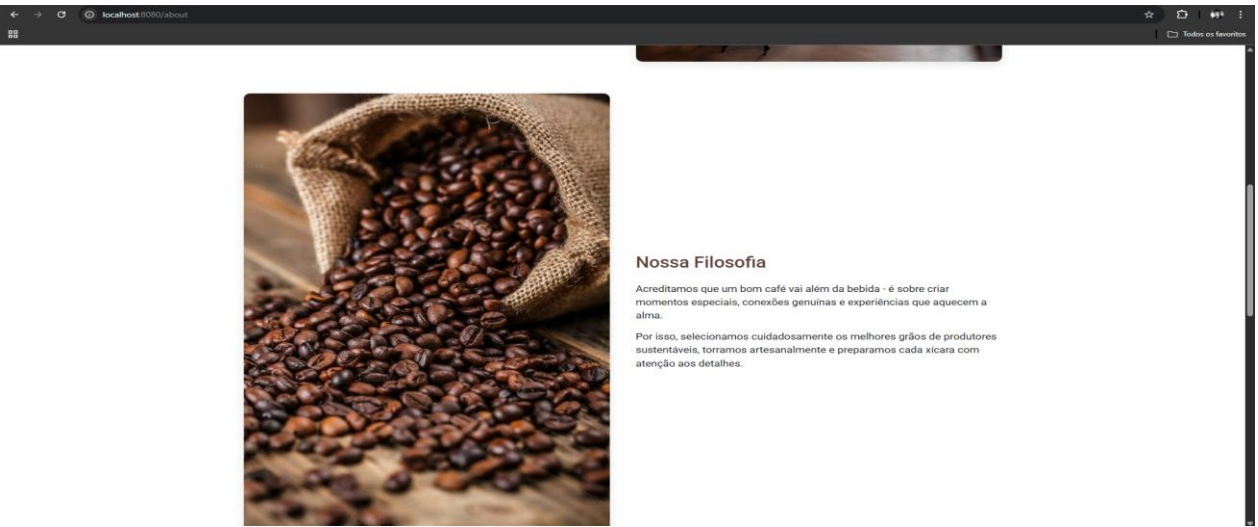
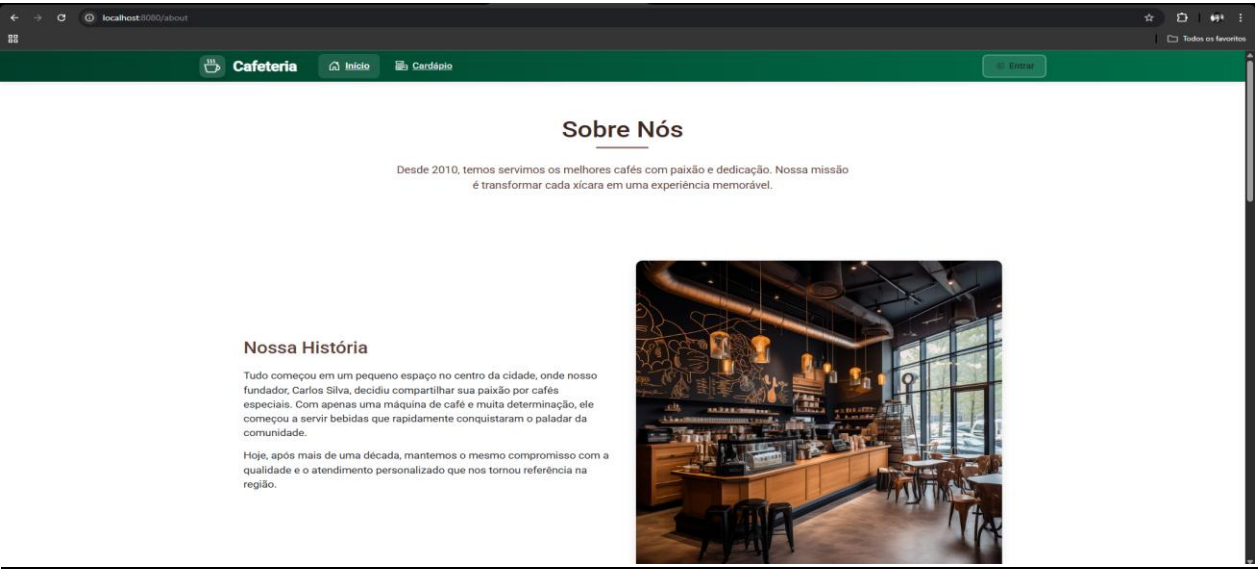
```

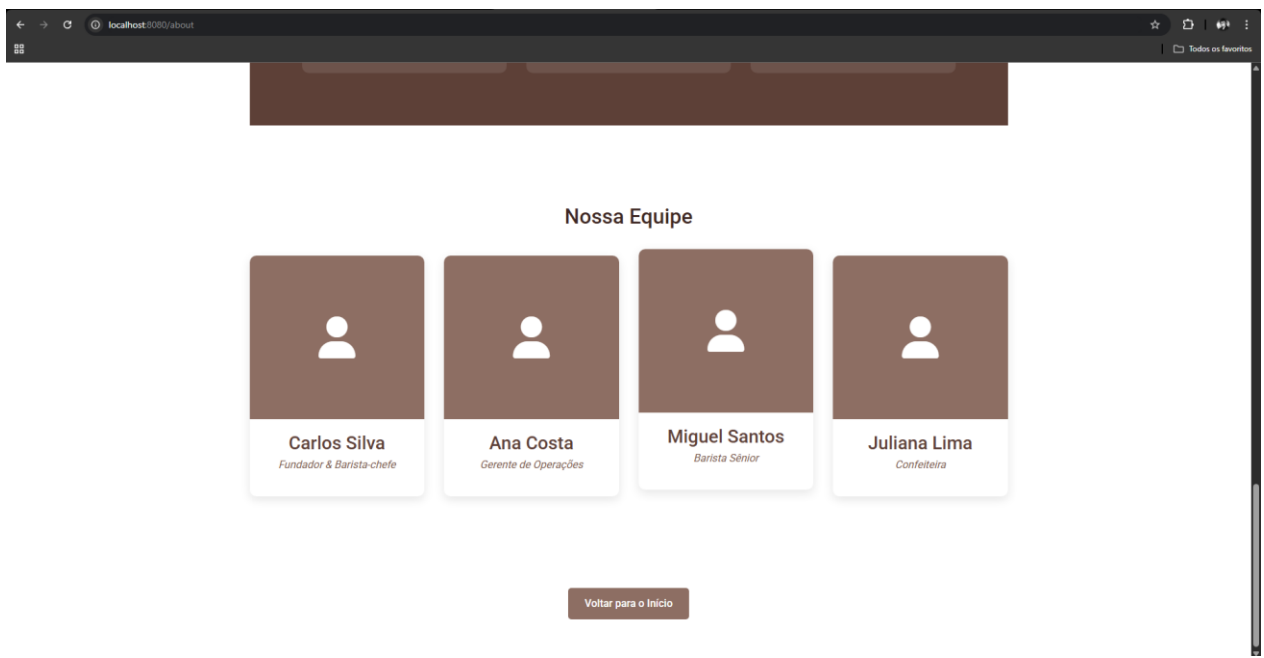
```
vendas</span>
                                </li>
                                <li class="list-group-item d-flex justify-content-
between align-items-center">
                                    Mocha com Chantilly
                                    <span class="badge bg-primary rounded-pill">76
vendas</span>
                                </li>
                            </ul>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Footer -->
    <footer class="footer">
        <div class="container">
            <div class="row">
                <div class="col-md-6">
                    <h5><i class="bi bi-cup-hot-fill me-2"></i> Cafeteria
Admin</h5>
                    <p>Sistema de gerenciamento para cafeterias</p>
                </div>
                <div class="col-md-6 text-md-end">
                    <p>&copy; 2023 Cafeteria Admin. Todos os direitos
reservados.</p>
                    <p>v1.2.0</p>
                </div>
            </div>
        </div>
    </footer>

    <!-- Bootstrap JS -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min
.js"></script>
</body>
</html>
```

Tela “Sobre a Empresa”





```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sobre Nós - Cafeteria</title>
  <style>
    /* Reset e estilos base */
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
      font-family: 'Roboto', 'Noto Sans', sans-serif;
    }

    body {
      background-color: #f8f5f0;
      color: #333;
      line-height: 1.6;
    }

    /* Container principal */
    .about-container {
      max-width: 1200px;
      margin: 0 auto;
      padding: 20px;
    }

    /* Cabeçalho */
    .about-header {
      text-align: center;
      padding: 40px 0;
    }

    .about-header h1 {
```

```

        font-size: 2.5rem;
        color: #3e2723;
        margin-bottom: 15px;
        position: relative;
        display: inline-block;
    }

    .about-header h1::after {
        content: '';
        position: absolute;
        bottom: -10px;
        left: 50%;
        transform: translateX(-50%);
        width: 80px;
        height: 3px;
        background-color: #8d6e63;
    }

    .about-header p {
        font-size: 1.1rem;
        color: #5d4037;
        max-width: 700px;
        margin: 20px auto;
    }

    /* Seções */
    .about-section {
        display: flex;
        flex-wrap: wrap;
        align-items: center;
        margin: 60px 0;
        gap: 40px;
    }

    .about-section:nth-child(even) {
        flex-direction: row-reverse;
    }

    .about-image {
        flex: 1;
        min-width: 300px;
        border-radius: 10px;
        overflow: hidden;
        box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
    }

    .about-image img {
        width: 100%;
        height: auto;
        display: block;
        transition: transform 0.5s ease;
    }

    .about-image:hover img {
        transform: scale(1.05);
    }

    .about-content {
        flex: 1;

```

```
        min-width: 300px;
    }

    .about-content h2 {
        font-size: 1.8rem;
        color: #5d4037;
        margin-bottom: 20px;
    }

    .about-content p {
        margin-bottom: 15px;
        font-size: 1.05rem;
    }

    /* Equipe */
    .team-section {
        padding: 60px 0;
        text-align: center;
    }

    .team-section h2 {
        font-size: 2rem;
        color: #3e2723;
        margin-bottom: 40px;
    }

    .team-grid {
        display: grid;
        grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
        gap: 30px;
        margin-top: 30px;
    }

    .team-member {
        background: white;
        border-radius: 10px;
        overflow: hidden;
        box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
        transition: transform 0.3s ease;
    }

    .team-member:hover {
        transform: translateY(-10px);
    }

    .team-member-img {
        width: 100%;
        height: 250px;
        background-color: #8d6e63;
        display: flex;
        align-items: center;
        justify-content: center;
        color: white;
        font-size: 4rem;
    }

    .team-member-info {
        padding: 20px;
    }
```

```
.team-member-info h3 {
  color: #5d4037;
  margin-bottom: 5px;
}

.team-member-info p {
  color: #795548;
  font-style: italic;
}

/* Valores */
.values-section {
  background-color: #5d4037;
  color: white;
  padding: 80px 0;
  margin: 60px 0;
}

.values-container {
  max-width: 1000px;
  margin: 0 auto;
  text-align: center;
}

.values-container h2 {
  font-size: 2rem;
  margin-bottom: 50px;
}

.values-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 30px;
}

.value-item {
  padding: 30px 20px;
  background: rgba(255, 255, 255, 0.1);
  border-radius: 10px;
  transition: transform 0.3s ease;
}

.value-item:hover {
  transform: translateY(-5px);
  background: rgba(255, 255, 255, 0.15);
}

.value-item i {
  font-size: 2.5rem;
  margin-bottom: 20px;
  color: #ffcc80;
}

.value-item h3 {
  margin-bottom: 15px;
  font-size: 1.3rem;
}
```

```

/* Responsividade */
@media (max-width: 768px) {

    .about-section,
    .about-section:nth-child(even) {
        flex-direction: column;
    }

    .about-header h1 {
        font-size: 2rem;
    }

    .team-grid,
    .values-grid {
        grid-template-columns: 1fr;
    }
}

/* Botão de voltar */
.back-button {
    display: inline-block;
    margin-top: 40px;
    padding: 12px 25px;
    background-color: #8d6e63;
    color: white;
    text-decoration: none;
    border-radius: 5px;
    font-weight: 500;
    transition: background-color 0.3s ease;
}

.back-button:hover {
    background-color: #6d4c41;
}
</style>
<link
href="https://fonts.googleapis.com/css?family=Roboto:300,400,500&display=swap"
rel="stylesheet">
<link
href="https://fonts.googleapis.com/css?family=Noto+Sans&display=swap"
rel="stylesheet">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome/6.2.0/css/all.min.css">
</head>

<body>
    <div th:replace="~{fragments/header}"></div>

    <div class="about-container">
        <div class="about-header">
            <h1>Sobre Nós</h1>
            <p>Desde 2010, temos servimos os melhores cafés com paixão e
dedicação. Nossa missão é transformar cada
                xícara em uma experiência memorável.</p>
        </div>

        <div class="about-section">
            <div class="about-image">
                <div

```

```

        style="background-color: #8d6e63; height: 100%; display:
flex; align-items: center; justify-content: center; color: white; font-size:
1.5rem; ">
        <a th:href="@{/about}"></a>

    </div>
</div>
<div class="about-content">
    <h2>Nossa História</h2>
    <p>Tudo começou em um pequeno espaço no centro da cidade, onde
nosso fundador, Carlos Silva, decidiu
        compartilhar sua paixão por cafés especiais. Com apenas
uma máquina de café e muita determinação,
        ele começou a servir bebidas que rapidamente conquistaram
o paladar da comunidade.</p>
    <p>Hoje, após mais de uma década, mantemos o mesmo compromisso
com a qualidade e o atendimento
        personalizado que nos tornou referência na região.</p>
    </div>
</div>

<div class="about-section">
    <div class="about-image">
        <div
            style="background-color: #6d4c41; height: 100%; display:
flex; align-items: center; justify-content: center; color: white; font-size:
1.5rem; ">
            <a th:href="@{/about}"></a>
        </div>
    </div>
    <div class="about-content">
        <h2>Nossa Filosofia</h2>
        <p>Acreditamos que um bom café vai além da bebida - é sobre
criar momentos especiais, conexões genuínas
            e experiências que aquecem a alma.</p>
        <p>Por isso, selecionamos cuidadosamente os melhores grãos de
produtores sustentáveis, torramos
            artesanalmente e preparamos cada xícara com atenção aos
detalhes.</p>
    </div>
</div>

<div class="values-section">
    <div class="values-container">
        <h2>Nossos Valores</h2>
        <div class="values-grid">
            <div class="value-item">
                <i class="fas fa-seedling"></i>
                <h3>Sustentabilidade</h3>
                <p>Trabalhamos com produtores que praticam agricultura
sustentável e respeitam o meio ambiente.
                </p>
            </div>
            <div class="value-item">
                <i class="fas fa-heart"></i>
                <h3>Paixão</h3>
                <p>Amamos o que fazemos e acreditamos que isso faz

```

```

toda a diferença no sabor do nosso café.</p>
</div>
<div class="value-item">
  <i class="fas fa-users"></i>
  <h3>Comunidade</h3>
  <p>Valorizamos as pessoas que fazem parte da nossa
história - clientes, colaboradores e
      fornecedores.</p>
</div>
</div>
</div>
</div>

<div class="team-section">
  <h2>Nossa Equipe</h2>
  <div class="team-grid">
    <div class="team-member">
      <div class="team-member-img">
        <i class="fas fa-user"></i>
      </div>
      <div class="team-member-info">
        <h3>Carlos Silva</h3>
        <p>Fundador & Barista-chefe</p>
      </div>
    </div>
    <div class="team-member">
      <div class="team-member-img">
        <i class="fas fa-user"></i>
      </div>
      <div class="team-member-info">
        <h3>Ana Costa</h3>
        <p>Gerente de Operações</p>
      </div>
    </div>
    <div class="team-member">
      <div class="team-member-img">
        <i class="fas fa-user"></i>
      </div>
      <div class="team-member-info">
        <h3>Miguel Santos</h3>
        <p>Barista Sênior</p>
      </div>
    </div>
    <div class="team-member">
      <div class="team-member-img">
        <i class="fas fa-user"></i>
      </div>
      <div class="team-member-info">
        <h3>Juliana Lima</h3>
        <p>Confeiteira</p>
      </div>
    </div>
  </div>
</div>
</div>

<div style="text-align: center; margin: 40px 0;">
  <a th:href="@{/}" class="back-button">Voltar para o Início</a>

```

```
        </div>
    </div>
</body>
</html>
```

2.3 Códigos (Back-End)

2.3.1 Controller

Classe AboutController

```
package com.senac.cafeteria.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class AboutController {

    @GetMapping("/about")
    public String about() {
        return "public/about";
    }
}
```

Classe AdminController

```
package com.senac.cafeteria.controller;

import com.senac.cafeteria.models.ItemPedido;
import com.senac.cafeteria.models.Pedido;
import com.senac.cafeteria.models.Produto;
import com.senac.cafeteria.models.enums.StatusPedido;
import com.senac.cafeteria.services.PedidoService;
import com.senac.cafeteria.services.ProdutoService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.propertyeditors.CustomNumberEditor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.WebDataBinder;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.io.IOException;
import java.math.BigDecimal;
import java.time.LocalDate;

import java.util.Base64;
import java.util.List;
import java.util.stream.Collectors;

/*
 * Controller responsável pelas páginas e ações administrativas (produtos,
 * pedidos, dashboard).
 * Contém handlers para CRUD de produtos, listagem/alteração de pedidos e
 * geração de dados do dashboard.
 */
@Controller
@RequestMapping("/admin")
public class AdminController {
```

```

        // Serviço para operações relacionadas a produtos (salvar, listar,
        atualizar, excluir)
        @Autowired
        private ProdutoService produtoService;

        // Serviço para operações relacionadas a pedidos (listar, atualizar
        status, excluir)
        @Autowired
        private PedidoService pedidoService;

        /*
        * Configurações do binder para conversão de tipos vindos do formulário.
        * Aqui é registrado um editor para BigDecimal e é proibida a binding
        direta do campo 'imagem'
        * para evitar sobrescrever o array de bytes por binding automático.
        */
        @InitBinder
        public void initBinder(WebDataBinder binder) {
            binder.registerCustomEditor(BigDecimal.class, new
            CustomNumberEditor(BigDecimal.class, true));
            binder.setDisallowedFields("imagem");
        }

        // ===== PRODUTOS =====

        // Formulário para criar um novo produto
        @GetMapping("/produtos/novo")
        public String novoProdutoForm(Model model) {
            model.addAttribute("produto", new Produto()); // adiciona objeto vazio
            ao template
            return "admin/novo-produto";
        }

        // Salva um novo produto recebido do formulário (inclui upload de imagem)
        @PostMapping("/produtos/novo")
        public String salvarProduto(@RequestParam String nome,
                                   @RequestParam String descricao,
                                   @RequestParam BigDecimal preco,
                                   @RequestParam("imagem") MultipartFile imagem)
            throws IOException {

            Produto produto = new Produto();
            produto.setNome(nome);
            produto.setDescricao(descricao);
            produto.setPreco(preco);

            // Delega ao serviço a persistência e tratamento da imagem
            produtoService.salvarProduto(produto, imagem);
            return "redirect:/admin/produtos";
        }

        // Lista todos os produtos mostrando imagem como Base64 para exibição no
        template
        @GetMapping("/produtos")
        public String listarProdutos(Model model) {
            List<Produto> produtos = produtoService.listarTodos();

            for (Produto produto : produtos) {

```

```

        if (produto.getImagem() != null) {
            String base64Image =
Base64.getEncoder().encodeToString(produto.getImagem());
            produto.setImagemBase64(base64Image); // seta campo
transitório para view
        }
    }

    model.addAttribute("produtos", produtos);
    return "admin/listar-produtos";
}

// Formulário para editar um produto existente (carrega produto por id)
@GetMapping("/produtos/editar/{id}")
public String editarProdutoForm(@PathVariable Long id, Model model) {
    Produto produto = produtoService.buscarPorId(id);

    if (produto.getImagem() != null) {
        String base64Image =
Base64.getEncoder().encodeToString(produto.getImagem());
        produto.setImagemBase64(base64Image);
    }

    model.addAttribute("produto", produto);
    return "admin/editar-produto";
}

// Atualiza um produto a partir do formulário (pode incluir nova imagem)
@PostMapping("/produtos/editar/{id}")
public String atualizarProduto(@PathVariable Long id,
                               @ModelAttribute Produto produto,
                               @RequestParam("imagem") MultipartFile
imagem) throws IOException {
    produtoService.atualizarProduto(id, produto, imagem);
    return "redirect:/admin/produtos";
}

// Exclui um produto por id
@GetMapping("/produtos/excluir/{id}")
public String excluirProduto(@PathVariable Long id) {
    produtoService.excluirProduto(id);
    return "redirect:/admin/produtos";
}

// ===== PEDIDOS =====

/*
 * Lista pedidos filtrando por status opcional.
 * Adiciona logs simples para auxiliar debug local.
 */
@GetMapping("/pedidos")
public String listarPedidos(@RequestParam(required = false) StatusPedido
status, Model model) {
    System.out.println("=== LISTAR PEDIDOS CHAMADO ===");
    System.out.println("Status filtro: " + status);

    List<Pedido> pedidos;

    if (status != null) {

```

```

        pedidos = pedidoService.listarPedidosPorStatus(status);
        model.addAttribute("filtroAtivo", status);
        System.out.println("Pedidos filtrados por " + status + ": " +
pedidos.size());
    } else {
        pedidos = pedidoService.listarTodosPedidos();
        System.out.println("Todos os pedidos: " + pedidos.size());
    }

    // Log para debug
    pedidos.forEach(p -> System.out.println("Pedido " + p.getId() + " -
Status: " + p.getStatus()));

    model.addAttribute("pedidos", pedidos);
    return "admin/listar-pedidos";
}

// Visualiza detalhes de um pedido específico
@GetMapping("/pedidos/{id}")
public String verPedido(@PathVariable Long id, Model model) {
    Pedido pedido = pedidoService.buscarPorId(id);
    model.addAttribute("pedido", pedido);
    return "admin/ver-pedido";
}

// Atualiza o status de um pedido e adiciona mensagem flash de
sucesso/erro
@GetMapping("/pedidos/{id}/status")
public String atualizarStatus(@PathVariable Long id,
                             @RequestParam StatusPedido status,
                             RedirectAttributes redirectAttributes) {
    try {
        System.out.println("=== ATUALIZANDO STATUS DO PEDIDO ===");
        System.out.println("Pedido ID: " + id);
        System.out.println("Novo Status: " + status);

        pedidoService.atualizarStatus(id, status);

        redirectAttributes.addFlashAttribute("sucesso", "Status do pedido
#" + id + " atualizado para " + status + "!");
    } catch (Exception e) {
        System.err.println("ERRO ao atualizar status: " + e.getMessage());
        e.printStackTrace();
        redirectAttributes.addFlashAttribute("erro", "Erro ao atualizar
status do pedido: " + e.getMessage());
    }
    return "redirect:/admin/pedidos";
}

// Exclui um pedido e informa resultado via flash attributes
@GetMapping("/pedidos/excluir/{id}")
public String excluirPedido(@PathVariable Long id, RedirectAttributes
redirectAttributes) {
    try {
        pedidoService.excluirPedido(id);
        redirectAttributes.addFlashAttribute("sucesso", "Pedido excluído
com sucesso!");
    } catch (Exception e) {
        redirectAttributes.addFlashAttribute("erro", "Erro ao excluir

```

```

pedido: " + e.getMessage());
    }
    return "redirect:/admin/pedidos";
}

// ===== DASHBOARD =====

// Página do dashboard que agrega métricas (total produtos, pedidos,
faturamento, etc.)
@GetMapping("/dashboard")
public String dashboard(Model model) {
    long totalProdutos = produtoService.contarProdutos();
    List<Pedido> todosPedidos = pedidoService.listarTodosPedidos();
    long totalPedidos = todosPedidos.size();
    long pedidosPendentes =
pedidoService.listarPedidosPorStatus(StatusPedido.PENDENTE).size();

    System.out.println("Total produtos: " + totalProdutos);
    System.out.println("Total pedidos: " + totalPedidos);
    System.out.println("Pedidos pendentes: " + pedidosPendentes);

    BigDecimal faturamentoTotal = calcularFaturamentoTotal(todosPedidos);
    BigDecimal faturamentoMes = calcularFaturamentoMes(todosPedidos);

    List<Pedido> pedidosRecentes = todosPedidos.stream()
        .limit(10)
        .collect(Collectors.toList());

    model.addAttribute("totalProdutos", totalProdutos);
    model.addAttribute("totalPedidos", totalPedidos);
    model.addAttribute("pedidosPendentes", pedidosPendentes);
    model.addAttribute("faturamentoTotal", faturamentoTotal);
    model.addAttribute("faturamentoMes", faturamentoMes);
    model.addAttribute("pedidosRecentes", pedidosRecentes);
    model.addAttribute("pedidosHoje", calcularPedidosHoje(todosPedidos));
    model.addAttribute("produtosVendidosHoje",
calcularProdutosVendidosHoje(todosPedidos));

    return "admin/dashboard";
}

// ===== MÉTODOS AUXILIARES =====

// Soma total de todos os pedidos
private BigDecimal calcularFaturamentoTotal(List<Pedido> pedidos) {
    return pedidos.stream()
        .map(Pedido::getTotal)
        .reduce(BigDecimal.ZERO, BigDecimal::add);
}

// Soma do faturamento do mês corrente
private BigDecimal calcularFaturamentoMes(List<Pedido> pedidos) {
    LocalDate inicioMes = LocalDate.now().withDayOfMonth(1);
    return pedidos.stream()
        .filter(pedido -> pedido.getDataCriacao() != null &&
!pedido.getDataCriacao().toLocalDate().isBefore(inicioMes))
        .map(Pedido::getTotal)
        .reduce(BigDecimal.ZERO, BigDecimal::add);
}

```

```

    }

    // Quantidade de pedidos do dia atual
    private long calcularPedidosHoje(List<Pedido> pedidos) {
        LocalDate hoje = LocalDate.now();
        return pedidos.stream()
            .filter(pedido -> pedido.getDataCriacao() != null &&
                pedido.getDataCriacao().toLocalDate().equals(hoje))
            .count();
    }

    // Soma de itens vendidos hoje (quantidades) a partir dos pedidos do dia
    private long calcularProdutosVendidosHoje(List<Pedido> pedidos) {
        LocalDate hoje = LocalDate.now();
        return pedidos.stream()
            .filter(pedido -> pedido.getDataCriacao() != null &&
                pedido.getDataCriacao().toLocalDate().equals(hoje))
            .flatMap(pedido -> pedido.getItens().stream())
            .mapToInt(ItemPedido::getQuantidade)
            .sum();
    }
}

```

Classe ApiAuthController

```

package com.senac.cafeteria.controller;

import com.senac.cafeteria.dtos.AuthRequest;
import com.senac.cafeteria.dtos.AuthResponse;
import com.senac.cafeteria.security.JwtUtil;
import com.senac.cafeteria.services.MyUserDetailsService;

import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpSession;

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.web.context.HttpSessionSecurityContextRepository;

import org.springframework.web.bind.annotation.*;

import java.util.Map;

/*
 * Controller responsável pelos endpoints de autenticação da API.
 * Expõe endpoints para obter token (login) e para criar sessão do Spring a

```

partir de um token JWT.

```
*/
@RestController
@RequestMapping("/api/auth")
public class ApiAuthController {

    // Manager usado para autenticação tradicional (username/password)
    private final AuthenticationManager authenticationManager;
    // Utilitário JWT para gerar e validar tokens
    private final JwtUtil jwtUtil;
    // Serviço que carrega detalhes do usuário (implementa UserDetailsService)
    private final MyUserDetailsService userDetailsService;

    // Construtor com injeção de dependências
    public ApiAuthController(AuthenticationManager authenticationManager,
                             JwtUtil jwtUtil,
                             MyUserDetailsService userDetailsService) {
        this.authenticationManager = authenticationManager;
        this.jwtUtil = jwtUtil;
        this.userDetailsService = userDetailsService;
    }

    /**
     * Endpoint de login:
     * - Recebe AuthRequest (username, password).
     * - Usa AuthenticationManager para autenticar as credenciais.
     * - Se autenticado, gera e retorna um JWT (AuthResponse).
     * - Em caso de credenciais inválidas retorna 401 com mensagem de erro.
     */
    @PostMapping("/login")
    public ResponseEntity<?> login(@RequestBody AuthRequest body) {
        try {
            Authentication auth = authenticationManager.authenticate(
                new
                UsernamePasswordAuthenticationToken(body.getUsername(), body.getPassword()));
            String token = jwtUtil.generateToken(body.getUsername());
            return ResponseEntity.ok(new AuthResponse(token));
        } catch (BadCredentialsException ex) {
            // Retorna 401 quando as credenciais não batem
            return ResponseEntity.status(401).body(Map.of("error",
"Credenciais inválidas"));
        }
    }

    /**
     * Cria uma sessão HTTP do Spring a partir de um token JWT válido.
     * Útil quando uma interface cliente (ex: SPA) obtém um token e quer abrir
uma sessão com JSESSIONID.
     *
     * Fluxo:
     * - Recebe JSON com { "token": "..." }.
     * - Valida o token com JwtUtil.
     * - Extrai username e carrega UserDetails.
     * - Cria uma Authentication e popula o SecurityContextHolder.
     * - Persiste o SecurityContext na sessão HTTP (gera cookie JSESSIONID).
     */
    @PostMapping("/session")
    public ResponseEntity<?> createSession(@RequestBody Map<String, String>
body, HttpServletRequest request) {
```

```

        String token = body.get("token");
        if (token == null || !jwtUtil.validateToken(token)) {
            // Token ausente ou inválido -> 401
            return ResponseEntity.status(401).body(Map.of("error", "token
inválido"));
        }

        String username = jwtUtil.extractUsername(token);
        UserDetails userDetails;
        try {
            // Carrega os detalhes do usuário (username -> UserDetails)
            userDetails = userDetailsService.loadUserByUsername(username);
        } catch (Exception e) {
            // Usuário não encontrado -> 401
            return ResponseEntity.status(401).body(Map.of("error", "usuário
não encontrado"));
        }

        // Cria Authentication com as authorities do usuário e popula o
SecurityContext
        UsernamePasswordAuthenticationToken auth =
            new UsernamePasswordAuthenticationToken(userDetails, null,
userDetails.getAuthorities());
        SecurityContextHolder.getContext().setAuthentication(auth);

        // Persiste o SecurityContext na sessão HTTP para que o Spring
mantenha a autenticação
        HttpSession session = request.getSession(true); // cria sessão se não
existir

        session.setAttribute(HttpSessionSecurityContextRepository.SPRING_SECURITY_CONT
EXT_KEY,
            SecurityContextHolder.getContext());

        // Retorna OK indicando que a sessão foi criada com sucesso
        return ResponseEntity.ok(Map.of("ok", true));
    }
}

```

Classe CadastroController

```

package com.senac.cafeteria.controller;

import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.senac.cafeteria.models.Usuario;
import com.senac.cafeteria.models.enums.Role;
import com.senac.cafeteria.repositories.UsuarioRepository;

@Controller
public class CadastroController {

    private final UsuarioRepository usuarioRepository;
}

```

```

        private final PasswordEncoder passwordEncoder;

        public CadastroController(UsuarioRepository usuarioRepository,
        PasswordEncoder passwordEncoder) {
            this.usuarioRepository = usuarioRepository;
            this.passwordEncoder = passwordEncoder;
        }

        @PostMapping("/cadastro")
        public String processCadastro(@RequestParam String nome,
                                     @RequestParam String email,
                                     @RequestParam String senha,
                                     @RequestParam String endereco,
                                     RedirectAttributes redirectAttributes) {

            // Verificar se o email já existe
            if (usuarioRepository.findByEmail(email).isPresent()) {
                redirectAttributes.addFlashAttribute("erroCadastro", "Este email
já está cadastrado");
                return "redirect:/";
            }

            // Criar novo usuário
            Usuario usuario = new Usuario();
            usuario.setNome(nome);
            usuario.setEmail(email);
            usuario.setSenha(passwordEncoder.encode(senha));
            usuario.setEndereco(endereco);
            usuario.setRole(Role.CLIENTE);

            usuarioRepository.save(usuario);

            redirectAttributes.addFlashAttribute("sucessoCadastro", "Cadastro
realizado com sucesso! Faça login para continuar.");
            return "redirect:/";
        }
    }
}

```

Classe CarrinhoController

```

package com.senac.cafeteria.controller;

import com.senac.cafeteria.models.Usuario;
import com.senac.cafeteria.services.CarrinhoService;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import java.util.Base64;

@Controller
@RequestMapping("/carrinho")
@RequiredArgsConstructor
public class CarrinhoController {

```

```

private final CarrinhoService carrinhoService;

@GetMapping
public String verCarrinho(@AuthenticationPrincipal Usuario usuario, Model
model) {
    var itensCarrinho = carrinhoService.getCarrinho(usuario.getId());

    // Converter imagens para Base64
    itensCarrinho.keySet().forEach(produto -> {
        if (produto.getImagem() != null) {
            String base64Image =
Base64.getEncoder().encodeToString(produto.getImagem());
            produto.setImagemBase64(base64Image);
        }
    });

    model.addAttribute("itensCarrinho", itensCarrinho);
    model.addAttribute("total",
carrinhoService.calcularTotal(usuario.getId()));
    model.addAttribute("quantidadeItens",
carrinhoService.getQuantidadeItens(usuario.getId()));

    return "carrinho/carrinho";
}

@PostMapping("/adicionar/{produtoId}")
public String adicionarAoCarrinho(@AuthenticationPrincipal Usuario
usuario,
                                @PathVariable Long produtoId,
                                @RequestParam(defaultValue = "1") Integer
quantidade,
                                RedirectAttributes redirectAttributes) {
    try {
        carrinhoService.adicionarAoCarrinho(usuario.getId(), produtoId,
quantidade);
        redirectAttributes.addFlashAttribute("sucesso", "Produto
adicionado ao carrinho!");
    } catch (Exception e) {
        redirectAttributes.addFlashAttribute("erro", "Erro ao adicionar
produto: " + e.getMessage());
    }
    return "redirect:/menu";
}

@PostMapping("/remover/{produtoId}")
public String removerDoCarrinho(@AuthenticationPrincipal Usuario usuario,
                                @PathVariable Long produtoId,
                                RedirectAttributes redirectAttributes) {
    try {
        carrinhoService.removerDoCarrinho(usuario.getId(), produtoId);
        redirectAttributes.addFlashAttribute("sucesso", "Produto removido
do carrinho!");
    } catch (Exception e) {
        redirectAttributes.addFlashAttribute("erro", "Erro ao remover
produto: " + e.getMessage());
    }
    return "redirect:/carrinho";
}

```

```

    @PostMapping("/atualizar/{produtoId}")
    public String atualizarQuantidade(@AuthenticationPrincipal Usuario
usuario,
                                     @PathVariable Long produtoId,
                                     @RequestParam Integer quantidade,
                                     RedirectAttributes redirectAttributes) {
        try {
            carrinhoService.atualizarQuantidade(usuario.getId(), produtoId,
quantidade);
            redirectAttributes.addFlashAttribute("sucesso", "Quantidade
atualizada!");
        } catch (Exception e) {
            redirectAttributes.addFlashAttribute("erro", "Erro ao atualizar
quantidade: " + e.getMessage());
        }
        return "redirect:/carrinho";
    }

    @PostMapping("/finalizar")
    public String finalizarPedido(@AuthenticationPrincipal Usuario usuario,
RedirectAttributes redirectAttributes) {
        try {
            var pedido = carrinhoService.finalizarPedido(usuario);
            redirectAttributes.addFlashAttribute("sucesso",
                "Pedido #" + pedido.getId() + " realizado com sucesso!");
            return "redirect:/pedidos";
        } catch (Exception e) {
            redirectAttributes.addFlashAttribute("erro", "Erro ao finalizar
pedido: " + e.getMessage());
            return "redirect:/carrinho";
        }
    }

    @PostMapping("/limpar")
    public String limparCarrinho(@AuthenticationPrincipal Usuario usuario,
RedirectAttributes redirectAttributes) {
        try {
            carrinhoService.limparCarrinho(usuario.getId());
            redirectAttributes.addFlashAttribute("sucesso", "Carrinho
limpo!");
        } catch (Exception e) {
            redirectAttributes.addFlashAttribute("erro", "Erro ao limpar
carrinho: " + e.getMessage());
        }
        return "redirect:/carrinho";
    }
}

```

Classe HomeController

```
package com.senac.cafeteria.controller;

import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

import com.senac.cafeteria.models.Usuario;
import com.senac.cafeteria.models.enums.Role; // ← IMPORT CORRETO para Role

@Controller
public class HomeController {

    @GetMapping("/")
    public String home() {
        return "auth/login"; // ← Agora aponta para templates/auth/login.html
    }

    @GetMapping("/login")
    public String login() {
        return "auth/login"; // ← Mesmo caminho
    }

    @GetMapping("/redirecionarPorRole")
    public String redirecionarPorRole(@AuthenticationPrincipal Usuario
usuuario) {
        if (usuuario != null) {
            if (usuuario.getRole() == Role.FUNCIONARIO) {
                return "redirect:/admin/dashboard"; // ← REDIRECIONA PARA
DASHBOARD
            } else if (usuuario.getRole() == Role.CLIENTE) {
                return "redirect:/menu";
            }
        }
        return "redirect:/login";
    }
}
```

Classe MenuController

```
package com.senac.cafeteria.controller;

import com.senac.cafeteria.services.ProdutoService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import java.util.Base64;

@Controller
public class MenuController {

    @Autowired
    private ProdutoService produtoService;
```

```

@GetMapping("/menu")
public String menu(Model model) {
    var produtos = produtoService.listarTodos();

    // Converter imagens para Base64
    for (var produto : produtos) {
        if (produto.getImagem() != null) {
            String base64Image =
Base64.getEncoder().encodeToString(produto.getImagem());
            produto.setImagemBase64(base64Image);
        }
    }

    model.addAttribute("produtos", produtos);
    return "public/menu";
}
}

```

Classe PedidoController

```

package com.senac.cafeteria.controller;

import com.senac.cafeteria.models.Usuario;
import com.senac.cafeteria.services.PedidoService;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

/*
 * Controller para endpoints relacionados aos pedidos do cliente.
 * Exibe a lista de pedidos do usuário autenticado e detalhes de um pedido.
 */
@Controller
@RequestMapping("/pedidos")
@RequiredArgsConstructor
public class PedidoController {

    // Serviço que fornece operações sobre pedidos (listar, buscar por id)
    private final PedidoService pedidoService;

    // Lista os pedidos do usuário autenticado e adiciona ao model para a view
    @GetMapping
    public String meusPedidos(@AuthenticationPrincipal Usuario usuario, Model
model) {
        var pedidos = pedidoService.listarPedidosPorUsuario(usuario);
        model.addAttribute("pedidos", pedidos);
        return "cliente/pedidos";
    }

    // Exibe detalhes de um pedido verificando autorização (dono do pedido ou
funcionário)
    @GetMapping("/{id}")
    public String detalhesPedido(@AuthenticationPrincipal Usuario usuario,
        @PathVariable Long id,
        Model model) {

```

```

        var pedido = pedidoService.buscarPorId(id);

        // Verificar se o pedido pertence ao usuário ou se o usuário é
funcionário
        if (!pedido.getUsuario().getId().equals(usuario.getId()) &&
            !usuario.getRole().name().equals("FUNCIONARIO")) {
            throw new RuntimeException("Acesso negado");
        }

        model.addAttribute("pedido", pedido);
        return "cliente/detalhes-pedido";
    }
}

```

2.3.2 Models

Classe ItemPedido

```

package com.senac.cafeteria.models;

import jakarta.persistence.*;
import java.math.BigDecimal;

/**
 * Classe que representa a RELAÇÃO N:N entre Pedido e Produto.
 *
 * Um Pedido pode conter vários Produtos.
 * Um Produto pode estar em vários Pedidos.
 *
 * Essa relação N:N é implementada de forma explícita por meio
 * desta entidade intermediária: ITEM_PEDIDO.
 *
 * Cada ItemPedido liga um Produto a um Pedido e armazena
 * informações adicionais, como quantidade e preço unitário.
 */
@Entity
@Table(name = "item_pedido")
public class ItemPedido {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    /**
     * Muitos itens pertencem a um mesmo Pedido.
     * (Lado N da relação com Pedido)
     */
    @ManyToOne
    @JoinColumn(name = "pedido_id", nullable = false)
    private Pedido pedido;

    /**
     * Muitos itens podem referenciar o mesmo Produto.
     * (Lado N da relação com Produto)
     */
    @ManyToOne
    @JoinColumn(name = "produto_id", nullable = false)
    private Produto produto;
}

```

```

private Integer quantidade;
private BigDecimal precoUnitario;

// Construtores
public ItemPedido() {}

public ItemPedido(Produto produto, Integer quantidade) {
    this.produto = produto;
    this.quantidade = quantidade;
    this.precoUnitario = produto.getPreco();
}

// Getters e Setters
public Long getId() { return id; }
public void setId(Long id) { this.id = id; }

public Pedido getPedido() { return pedido; }
public void setPedido(Pedido pedido) { this.pedido = pedido; }

public Produto getProduto() { return produto; }
public void setProduto(Produto produto) { this.produto = produto; }

public Integer getQuantidade() { return quantidade; }
public void setQuantidade(Integer quantidade) { this.quantidade =
quantidade; }

public BigDecimal getPrecoUnitario() { return precoUnitario; }
public void setPrecoUnitario(BigDecimal precoUnitario) {
this.precoUnitario = precoUnitario; }

public BigDecimal getSubtotal() {
    return precoUnitario.multiply(BigDecimal.valueOf(quantidade));
}
}

```

Classe Pedido

```

package com.senac.cafeteria.models;

import jakarta.persistence.*;
import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

import com.senac.cafeteria.models.enums.StatusPedido;

@Entity
@Table(name = "pedido")
public class Pedido {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne

```

```

@JoinColumn(name = "usuario_id")
private Usuario usuario;

private LocalDateTime dataCriacao;
private BigDecimal total;

@Enumerated(EnumType.STRING)
private StatusPedido status;

// Relação funcional (1:N) com ItemPedido
@OneToMany(mappedBy = "pedido", cascade = CascadeType.ALL, orphanRemoval =
true)
private List<ItemPedido> itens = new ArrayList<>();

/**
 * Relação ManyToMany simbólica (para requisito acadêmico)
 * Não interfere na lógica funcional de ItemPedido.
 */
@ManyToMany
@JoinTable(
    name = "pedido_produto_relacionado",
    joinColumns = @JoinColumn(name = "pedido_id"),
    inverseJoinColumns = @JoinColumn(name = "produto_id")
)
private List<Produto> produtosRelacionados = new ArrayList<>();

// Construtores
public Pedido() {
    this.dataCriacao = LocalDateTime.now();
    this.total = BigDecimal.ZERO;
    this.status = StatusPedido.PENDENTE;
}

// Getters e Setters
public Long getId() { return id; }
public void setId(Long id) { this.id = id; }

public Usuario getUsuario() { return usuario; }
public void setUsuario(Usuario usuario) { this.usuario = usuario; }

public LocalDateTime getDataCriacao() { return dataCriacao; }
public void setDataCriacao(LocalDateTime dataCriacao) { this.dataCriacao =
dataCriacao; }

public BigDecimal getTotal() { return total; }
public void setTotal(BigDecimal total) { this.total = total; }

public StatusPedido getStatus() { return status; }
public void setStatus(StatusPedido status) { this.status = status; }

public List<ItemPedido> getItens() { return itens; }
public void setItens(List<ItemPedido> itens) { this.itens = itens; }

public List<Produto> getProdutosRelacionados() { return
produtosRelacionados; }
public void setProdutosRelacionados(List<Produto> produtosRelacionados) {
this.produtosRelacionados = produtosRelacionados; }

// Métodos auxiliares reais

```

```

    public void adicionarItem(ItemPedido item) {
        item.setPedido(this);
        this.itens.add(item);
        calcularTotal();
    }

    public void removerItem(ItemPedido item) {
        this.itens.remove(item);
        calcularTotal();
    }

    public void calcularTotal() {
        this.total = itens.stream()
            .map(ItemPedido::getSubtotal)
            .reduce(BigDecimal.ZERO, BigDecimal::add);
    }
}

```

Classe Produto

```

package com.senac.cafeteria.models;

import com.fasterxml.jackson.annotation.JsonIgnore;
import jakarta.persistence.*;
import lombok.Data;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;

@Entity
@Table(name = "produto")
@Data
public class Produto {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nome;
    private String descricao;
    private BigDecimal preco;

    /**
     * Armazenamento binário da imagem no banco.
     * Mantido para não quebrar o sistema de imagens.
     */
    @Lob
    private byte[] imagem;

    /**
     * Campo transitório usado para expor a imagem em Base64 na API.
     * Não é persistido no banco.
     */
    @Transient
    private String imagemBase64;
}

```

```

/**
 * Relação funcional (1:N) com ItemPedido – mantém a lógica do pedido.
 */
@JsonIgnore
@OneToMany(mappedBy = "produto", fetch = FetchType.LAZY)
private List<ItemPedido> itensPedido = new ArrayList<>();

/**
 * Relação ManyToMany simbólica com Pedido (requisito acadêmico).
 * Mapeado por "produtosRelacionados" no lado Pedido.
 * Não utilizada na lógica de cálculo do pedido.
 */
@JsonIgnore
@ManyToMany(mappedBy = "produtosRelacionados", fetch = FetchType.LAZY)
private List<Pedido> pedidosRelacionados = new ArrayList<>();
}

```

Classe Usuario

```

package com.senac.cafeteria.models;

import com.fasterxml.jackson.annotation.JsonIgnore;
import jakarta.persistence.*;
import java.util.Collection;
import java.util.List;
import lombok.Data;
import com.senac.cafeteria.models.enums.Role;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

@Entity
@Data
@Table(name = "usuario")
public class Usuario implements UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String nome;

    @Column(unique = true, nullable = false)
    private String email;

    @JsonIgnore
    private String senha;

    private String endereco;

    @Enumerated(EnumType.STRING)
    private Role role;

    @OneToMany(mappedBy = "usuario")
    private List<Pedido> pedidos;
}

```

```
// UserDetails methods

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    return List.of(new SimpleGrantedAuthority(role.getAuthority()));
}

@Override
public String getPassword() {
    return senha;
}

@Override
public String getUsername() {
    return email;
}

@Override
public boolean isAccountNonExpired() { return true; }

@Override
public boolean isAccountNonLocked() { return true; }

@Override
public boolean isCredentialsNonExpired() { return true; }

@Override
public boolean isEnabled() { return true; }
}
```

2.3.3 Enums

Enum Role

```
package com.senac.cafeteria.models.enums;

public enum Role {
    CLIENTE("ROLE_CLIENTE"),
    FUNCIONARIO("ROLE_FUNCIONARIO");

    private final String authority;

    Role(String authority) {
        this.authority = authority;
    }

    public String getAuthority() {
        return authority; // ← Este método é ESSENCIAL
    }
}
```

Enum StatusPedido

```
package com.senac.cafeteria.models.enums;

public enum StatusPedido {
```

```
    PENDENTE,  
    CONFIRMADO,  
    PREPARANDO,  
    PRONTO,  
    ENTREGUE,  
    CANCELADO  
}
```

Enum TipoEntrega

```
package com.senac.cafeteria.models.enums;  
  
public enum TipoEntrega {  
    RETIRADA_LOJA("Retirada na loja"),  
    ENTREGA("Entrega em domicílio");  
  
    private final String descricao;  
  
    TipoEntrega(String descricao) {  
        this.descricao = descricao;  
    }  
  
    public String getDescricao() {  
        return descricao;  
    }  
}
```

2.3.4 Repositories

Interface ItemPedidoRepository

```
package com.senac.cafeteria.repositories;  
  
import com.senac.cafeteria.models.ItemPedido;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface ItemPedidoRepository extends JpaRepository<ItemPedido, Long>  
{  
}
```

Interface PedidoRepository

```
package com.senac.cafeteria.repositories;  
  
import com.senac.cafeteria.models.Pedido;  
import com.senac.cafeteria.models.Usuario;  
import com.senac.cafeteria.models.enums.StatusPedido;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import java.util.List;
```

```

public interface PedidoRepository extends JpaRepository<Pedido, Long> {
    List<Pedido> findByUsuarioOrderByDataCriacaoDesc(Usuario usuario);
    List<Pedido> findByStatusOrderByDataCriacaoDesc(StatusPedido status); //
    // Mudar de Asc para Desc
    List<Pedido> findAllByOrderByDataCriacaoDesc();
}

```

Interface ProdutoRepository

```

package com.senac.cafeteria.repositories;

import com.senac.cafeteria.models.Produto;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.util.List;
import java.util.Optional;

public interface ProdutoRepository extends JpaRepository<Produto, Long> {

    // Buscar produtos por nome (ignorando maiúsculas/minúsculas)
    List<Produto> findByNameContainingIgnoreCase(String nome);

    // Buscar produto por nome exato
    Optional<Produto> findByName(String nome);

    // Buscar produtos ordenados por preço (crescente)
    List<Produto> findAllByOrderByPrecoAsc();

    // Buscar produtos ordenados por preço (decrescente)
    List<Produto> findAllByOrderByPrecoDesc();

    // Buscar produtos com preço maior que
    List<Produto> findByPrecoGreaterThan(Double preco);

    // Buscar produtos com preço entre valores
    List<Produto> findByPrecoBetween(Double precoMin, Double precoMax);

    // Query personalizada com JPQL
    @Query("SELECT p FROM Produto p WHERE p.descricao LIKE %:termo%")
    List<Produto> buscarPorDescricao(@Param("termo") String termo);
}

```

Interface UsuarioRepository

```

package com.senac.cafeteria.repositories;

import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;

import com.senac.cafeteria.models.Usuario;

public interface UsuarioRepository extends JpaRepository<Usuario, Long> {

```

```
Optional<Usuario> findByEmail(String email);  
}
```

2.3.5 Services

Classe CarrinhoService

```
package com.senac.cafeteria.services;  
  
import com.senac.cafeteria.models.*;  
import com.senac.cafeteria.models.enums.StatusPedido;  
import com.senac.cafeteria.repositories.PedidoRepository;  
import lombok.RequiredArgsConstructor;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.math.BigDecimal;  
import java.util.HashMap;  
import java.util.Map;  
  
@Service  
@RequiredArgsConstructor  
@Transactional  
public class CarrinhoService {  
  
    private final PedidoRepository pedidoRepository;  
    private final ProdutoService produtoService;  
  
    // Carrinho em memória (simplificado - em produção use Redis ou sessão)  
    private final Map<Long, Map<Long, Integer>> carrinhos = new HashMap<>();  
  
    public void adicionarAoCarrinho(Long usuarioId, Long produtoId, Integer  
quantidade) {  
        carrinhos.putIfAbsent(usuarioId, new HashMap<>());  
        Map<Long, Integer> carrinhoUsuario = carrinhos.get(usuarioId);  
  
        carrinhoUsuario.merge(produtoId, quantidade, Integer::sum);  
    }  
  
    public void removerDoCarrinho(Long usuarioId, Long produtoId) {  
        if (carrinhos.containsKey(usuarioId)) {  
            carrinhos.get(usuarioId).remove(produtoId);  
        }  
    }  
  
    public void atualizarQuantidade(Long usuarioId, Long produtoId, Integer  
quantidade) {  
        if (carrinhos.containsKey(usuarioId) && quantidade > 0) {  
            carrinhos.get(usuarioId).put(produtoId, quantidade);  
        }  
    }  
  
    public Map<Produto, Integer> getCarrinho(Long usuarioId) {  
        Map<Produto, Integer> carrinhoComProdutos = new HashMap<>();  
  
        if (carrinhos.containsKey(usuarioId)) {  
            Map<Long, Integer> carrinhoUsuario = carrinhos.get(usuarioId);
```

```

        for (Map.Entry<Long, Integer> entry : carrinhoUsuario.entrySet())
        {
            Produto produto = produtoService.buscarPorId(entry.getKey());
            carrinhoComProdutos.put(produto, entry.getValue());
        }

        return carrinhoComProdutos;
    }

    public BigDecimal calcularTotal(Long usuarioId) {
        Map<Produto, Integer> carrinho = getCarrinho(usuarioId);
        return carrinho.entrySet().stream()
            .map(entry ->
                entry.getKey().getPreco().multiply(BigDecimal.valueOf(entry.getValue())))
            .reduce(BigDecimal.ZERO, BigDecimal::add);
    }

    public void limparCarrinho(Long usuarioId) {
        carrinhos.remove(usuarioId);
    }

    public Integer getQuantidadeItens(Long usuarioId) {
        if (carrinhos.containsKey(usuarioId)) {
            return carrinhos.get(usuarioId).values().stream()
                .mapToInt(Integer::intValue)
                .sum();
        }
        return 0;
    }

    @Transactional
    public Pedido finalizarPedido(Usuario usuario) {
        Map<Produto, Integer> itensCarrinho = getCarrinho(usuario.getId());

        if (itensCarrinho.isEmpty()) {
            throw new RuntimeException("Carrinho vazio");
        }

        Pedido pedido = new Pedido();
        pedido.setUsuario(usuario);
        pedido.setStatus(StatusPedido.PENDENTE);

        for (Map.Entry<Produto, Integer> entry : itensCarrinho.entrySet()) {
            ItemPedido item = new ItemPedido(entry.getKey(),
                entry.getValue());
            pedido.adicionarItem(item);
        }

        pedido.calcularTotal();
        Pedido pedidoSalvo = pedidoRepository.save(pedido);
        limparCarrinho(usuario.getId());

        return pedidoSalvo;
    }
}

```

Classe CustomUserDetailsService

```
package com.senac.cafeteria.services;

import com.senac.cafeteria.models.Usuario;
import com.senac.cafeteria.repositories.UsuarioRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor
public class CustomUserDetailsService implements UserDetailsService {

    private final UsuarioRepository usuarioRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws
    UsernameNotFoundException {
        Usuario usuario = usuarioRepository.findByEmail(username)
            .orElseThrow(() -> new UsernameNotFoundException("Usuário não
    encontrado: " + username));

        return usuario; // ← Retorna a entidade Usuario que agora implementa
    UserDetails
    }
}
```

Classe PedidoService

```
package com.senac.cafeteria.services;

import com.senac.cafeteria.models.Pedido;
import com.senac.cafeteria.models.Usuario;
import com.senac.cafeteria.models.enums.StatusPedido;
import com.senac.cafeteria.repositories.PedidoRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
@RequiredArgsConstructor
public class PedidoService {

    private final PedidoRepository pedidoRepository;

    public List<Pedido> listarPedidosPorUsuario(Usuario usuario) {
        return pedidoRepository.findByUsuarioOrderByDataCriacaoDesc(usuario);
    }
}
```

```

    public List<Pedido> listarTodosPedidos() {
        return pedidoRepository.findAllByOrderByDataCriacaoDesc();
    }

    public List<Pedido> listarPedidosPorStatus(StatusPedido status) {
        return pedidoRepository.findByStatusOrderByDataCriacaoDesc(status);
    }

    public Pedido buscarPorId(Long id) {
        return pedidoRepository.findById(id)
            .orElseThrow(() -> new RuntimeException("Pedido não
encontrado"));
    }

    public void atualizarStatus(Long id, StatusPedido status) {
        Pedido pedido = buscarPorId(id);
        pedido.setStatus(status);
        pedidoRepository.save(pedido);
    }

    public void excluirPedido(Long id) {
        Pedido pedido = buscarPorId(id);
        pedidoRepository.delete(pedido);
    }
}

```

Classe ProdutoService

```

package com.senac.cafeteria.services;

import java.io.IOException;
import java.util.List;
import java.util.Optional;

import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import com.senac.cafeteria.models.Produto;
import com.senac.cafeteria.repositories.ProdutoRepository;

import lombok.RequiredArgsConstructor;

@Service
@RequiredArgsConstructor
public class ProdutoService {
    private final ProdutoRepository produtoRepository;

    public Produto salvarProduto(Produto produto, MultipartFile imagem) throws
IOException {
        if(imagem != null && !imagem.isEmpty()) {
            produto.setImagem(imagem.getBytes());
        }
        return produtoRepository.save(produto);
    }

    public void excluirProduto(Long id) {
        produtoRepository.deleteById(id);
    }
}

```

```

    }

    // Método para listar todos os produtos
    public List<Produto> listarTodos() {
        return produtoRepository.findAll();
    }

    // Método para buscar produto por ID
    public Produto buscarPorId(Long id) {
        Optional<Produto> produto = produtoRepository.findById(id);
        return produto.orElseThrow(() -> new RuntimeException("Produto não encontrado com ID: " + id));
    }

    // Método para atualizar produto
    public Produto atualizarProduto(Long id, Produto produtoAtualizado,
        MultipartFile imagem) throws IOException {
        Produto produtoExistente = buscarPorId(id);

        produtoExistente.setNome(produtoAtualizado.getNome());
        produtoExistente.setDescricao(produtoAtualizado.getDescricao());
        produtoExistente.setPreco(produtoAtualizado.getPreco());

        if (imagem != null && !imagem.isEmpty()) {
            produtoExistente.setImagem(imagem.getBytes());
        }

        return produtoRepository.save(produtoExistente);
    }

    // Método para buscar produtos por nome (opcional)
    public List<Produto> buscarPorNome(String nome) {
        return produtoRepository.findByNomeContainingIgnoreCase(nome);
    }

    // Método para verificar se produto existe
    public boolean existeProduto(Long id) {
        return produtoRepository.existsById(id);
    }

    // Método para contar total de produtos
    public long contarProdutos() {
        return produtoRepository.count();
    }
}

```

2.3.6 Config

Classe DataInitializer

```
package com.senac.cafeteria.config;

import jakarta.annotation.PostConstruct;

import org.springframework.context.annotation.Configuration;
import org.springframework.security.crypto.password.PasswordEncoder;

import com.senac.cafeteria.models.Produto;
import com.senac.cafeteria.models.Usuario;
import com.senac.cafeteria.models.enums.Role;
import com.senac.cafeteria.repositories.ProdutoRepository;
import com.senac.cafeteria.repositories.UsuarioRepository;

import lombok.RequiredArgsConstructor;

import java.math.BigDecimal;
import java.util.Arrays;
import java.util.List;

@Configuration
@RequiredArgsConstructor
public class DataInitializer {

    private final UsuarioRepository usuarioRepository;
    private final ProdutoRepository produtoRepository;
    private final PasswordEncoder passwordEncoder;

    @PostConstruct
    public void init() {
        criarUsuarios();
        criarProdutosIniciais();
    }

    private void criarUsuarios() {
        // Criar usuário funcionário se não existir
        if (usuarioRepository.findByEmail("funcionario@cafe.com").isEmpty()) {
            Usuario funcionario = new Usuario();
            funcionario.setNome("Funcionário Admin");
            funcionario.setEmail("funcionario@cafe.com");
            funcionario.setSenha(passwordEncoder.encode("123456"));
            funcionario.setRole(Role.FUNCIONARIO);
            usuarioRepository.save(funcionario);
            System.out.println("Usuário funcionário criado:
funcionario@cafe.com / 123456");
        }

        // Criar usuário cliente de teste
        if (usuarioRepository.findByEmail("cliente@teste.com").isEmpty()) {
            Usuario cliente = new Usuario();
            cliente.setNome("Cliente Teste");
            cliente.setEmail("cliente@teste.com");
            cliente.setSenha(passwordEncoder.encode("123456"));
            cliente.setRole(Role.CLIENTE);
            usuarioRepository.save(cliente);
            System.out.println("Usuário cliente criado: cliente@teste.com /
```

```

123456");
    }
}

private void criarProdutosIniciais() {
    if (produtoRepository.count() == 0) {
        List<Produto> produtos = Arrays.asList(
            criarProduto("Café Expresso", "Café forte e aromático,
preparado na hora", new BigDecimal("5.90")),
            criarProduto("Cappuccino", "Café com leite vaporizado e espuma
cremosa", new BigDecimal("8.50")),
            criarProduto("Latte", "Café com leite vaporizado e uma suave
camada de espuma", new BigDecimal("9.00")),
            criarProduto("Mocha", "Café com chocolate, leite vaporizado e
chantilly", new BigDecimal("12.00")),

            criarProduto("Suco de Laranja Natural", "Suco fresco de
laranja, feito na hora", new BigDecimal("8.00")),
            criarProduto("Sanduíche Natural", "Pão integral com peito de
peru e queijo branco", new BigDecimal("15.00")),
            criarProduto("Bolo de Chocolate", "Fatia de bolo de chocolate
com cobertura", new BigDecimal("9.90")),
            criarProduto("Croissant", "Croissant folhado e crocante,
perfeito para acompanhar", new BigDecimal("6.00"))
        );

        produtoRepository.saveAll(produtos);
        System.out.println("10 produtos iniciais criados com sucesso!");
    } else {
        System.out.println("Produtos já existem no banco. Nenhum produto
inicial criado.");
    }
}

private Produto criarProduto(String nome, String descricao, BigDecimal
preco) {
    Produto produto = new Produto();
    produto.setNome(nome);
    produto.setDescricao(descricao);
    produto.setPreco(preco);
    // As imagens ficarão nulas inicialmente - você pode adicionar depois
    pelo admin
    return produto;
}
}

```

Classe SecurityConfig

```

package com.senac.cafeteria.config;

import com.senac.cafeteria.security.JwtAuthenticationFilter;
import com.senac.cafeteria.services.MyUserDetailsService;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.annotation.Order;
import org.springframework.security.authentication.AuthenticationManager;
import

```

```

org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.configuration.Au
thenticationConfiguration;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.UsernamePasswordAuthentication
Filter;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.authentication.AuthenticationProvider;

/*
 * Classe de configuração de segurança da aplicação.
 * Aqui são definidos os beans e as duas SecurityFilterChain (API e Web).
 */
@Configuration
public class SecurityConfig {

    // Filtro JWT injetado para verificar tokens nas requisições
    private final JwtAuthenticationFilter jwtAuthenticationFilter;

    // Serviço que carrega detalhes do usuário (implementação do
    UserDetailsService)
    private final MyUserDetailsService userDetailsService;

    // Construtor com injeção de dependências
    public SecurityConfig(JwtAuthenticationFilter jwtAuthenticationFilter,
    MyUserDetailsService userDetailsService) {
        this.jwtAuthenticationFilter = jwtAuthenticationFilter; // inicializa
o filtro JWT
        this.userDetailsService = userDetailsService; // inicializa o serviço
de usuários
    }

    /*
     * Bean que fornece um AuthenticationProvider baseado em DAO (usuário +
    senha).
     * O DaoAuthenticationProvider usa o MyUserDetailsService e um
    PasswordEncoder.
     */
    @Bean
    public AuthenticationProvider daoAuthenticationProvider() {
        DaoAuthenticationProvider provider = new DaoAuthenticationProvider();
    // provedor padrão que busca usuários via UserDetailsService
        provider.setUserDetailsService(userDetailsService); // configura o
serviço de detalhes do usuário
        provider.setPasswordEncoder(passwordEncoder()); // configura o encoder
de senhas (BCrypt)
        return provider; // retorna o provedor configurado como bean
    }

    // Rotas do Swagger/OpenAPI que devem permanecer públicas (acesso sem
    autenticação)
    private static final String[] SWAGGER_MATCHERS = new String[] {
        "/v3/api-docs/**",

```

```

        "/swagger-ui/**",
        "/swagger-ui.html",
        "/swagger-ui/index.html",
        "/swagger-resources/**",
        "/webjars/**"
    };

    /*
     * SecurityFilterChain para a API que utiliza JWT – prioridade alta (Order
1).
     * Esta chain trata rotas que começam com /api/** e utiliza autenticação
stateless.
     */
    @Bean
    @Order(1)
    public SecurityFilterChain apiFilterChain(HttpSecurity http) throws
Exception {
        http
            // Aplica esta chain apenas para caminhos que combinam com
/api/**
            .securityMatcher("/api/**")
            // Desabilita CSRF para a API (quando se usa JWT geralmente se
desabilita)
            .csrf().disable()
            // Autorizações específicas para endpoints da API
            .authorizeHttpRequests(auth -> auth
                // Permite acesso livre a endpoints de auth
(login/registro)
                .requestMatchers("/api/auth/**").permitAll()
                // Permite acesso livre aos endpoints do
Swagger/OpenAPI
                .requestMatchers(SWAGGER_MATCHERS).permitAll()
                // Demais requisições da API exigem autenticação
                .anyRequest().authenticated()
            )
            // Configura gerenciamento de sessão para stateless (sem
sessão no servidor)
            .sessionManagement(sess ->
sess.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
            // Adiciona o AuthenticationProvider configurado anteriormente
(DAO + encoder)
            .authenticationProvider(daoAuthenticationProvider())
            // Adiciona o filtro JWT antes do filtro padrão de
autenticação por username/senha
            .addFilterBefore(jwtAuthenticationFilter,
UsernamePasswordAuthenticationFilter.class);

        // Constrói e retorna a SecurityFilterChain para a API
        return http.build();
    }

    /*
     * SecurityFilterChain para a parte web da aplicação (form login /
Thymeleaf).
     * Menor prioridade (Order 2) – usada para rotas não começando com /api/.
     */
    @Bean
    @Order(2)
    public SecurityFilterChain webFilterChain(HttpSecurity http) throws

```

```

Exception {
    http
        // Regras de autorização para recursos estáticos, páginas
        públicas e áreas restritas
        .authorizeHttpRequests(authz -> authz
            // Permite recursos estáticos (css, js, imagens,
webjars, favicon)
            .requestMatchers("/css/**", "/js/**", "/images/**",
"/images_homapage/**", "/webjars/**", "/favicon.ico").permitAll()
            // Permite acesso ao Swagger UI
            .requestMatchers(SWAGGER_MATCHERS).permitAll()
            // Páginas públicas (home, menu, cadastro, login,
about)
            .requestMatchers("/", "/menu", "/cadastro", "/login",
"/about").permitAll()
            // Rotas do carrinho e perfil restritas a clientes
            (ROLE_CLIENTE)
            .requestMatchers("/carrinho/**",
"/perfil").hasAuthority("ROLE_CLIENTE")
            // Rotas administrativas restritas a funcionários
            (ROLE_FUNCIONARIO)
            .requestMatchers("/admin/**").hasAuthority("ROLE_FUNCIONARIO")
            // Demais requisições exigem autenticação
            .anyRequest().authenticated()
        )
        // Configuração do formulário de login
        .formLogin(form -> form
            .loginPage("/login") // página customizada de login
            .loginProcessingUrl("/login") // endpoint que processa
o POST do login
            .defaultSuccessUrl("/redirecionarPorRole", true) //
redireciona após sucesso baseado na role
            .failureUrl("/login?error=true") // página em caso de
falha no login
            .permitAll() // permite acesso ao formulário de login
sem autenticação
        )
        // Configuração do logout
        .logout(logout -> logout
            .logoutUrl("/logout") // endpoint de logout
            .logoutSuccessUrl("/?logout=true") // redireciona após
logout
            .invalidateHttpSession(true) // invalida a sessão HTTP
            .deleteCookies("JSESSIONID") // remove cookie de
sessão
            .permitAll() // permite acesso ao logout sem
autenticação
        )
        // Gerenciamento de sessão para a aplicação web (proteção
contra fixation)
        .sessionManagement(session -> session
            .sessionFixation().migrateSession() // migra sessão
para evitar session fixation
        )
        // Usa o mesmo AuthenticationProvider configurado (DAO +
encoder)
        .authenticationProvider(daoAuthenticationProvider())
        // Desabilita CSRF para simplificar (adequar conforme

```

```

necessidade)
        .csrf(csrf -> csrf.disable());

        // Comentário: se desejar aceitar também Authorization: Bearer <token>
nas requisições web,
        // pode-se adicionar o jwtAuthenticationFilter na cadeia com
addFilterBefore.

        // Constrói e retorna a SecurityFilterChain para a web
return http.build();
    }

    /*
     * Bean que fornece o PasswordEncoder usado para armazenar e verificar
senhas.
     * BCrypt é seguro e recomendado por padrão.
     */
    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder(); // instancia o encoder BCrypt
    }

    /*
     * Bean que expõe o AuthenticationManager a partir da
AuthenticationConfiguration.
     * Útil para realizar autenticação manual (por exemplo, ao gerar token no
endpoint /api/auth).
     */
    @Bean
    public AuthenticationManager
authenticationManager(AuthenticationConfiguration authConfig) throws Exception
    {
        return authConfig.getAuthenticationManager(); // retorna o
AuthenticationManager configurado pelo Spring
    }
}

```

2.3.7 dtos

Classe AuthRequest

```

package com.senac.cafeteria.dtos;

public class AuthRequest {
    private String username; // email
    private String password;

    public AuthRequest() {}
    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }
    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }
}

```

Classe AuthResponse

```
package com.senac.cafeteria.dtos;

public class AuthResponse {
    private String token;
    public AuthResponse() {}
    public AuthResponse(String token) { this.token = token; }
    public String getToken() { return token; }
    public void setToken(String token) { this.token = token; }
}
```

2.3.8 Security

Classe JwtAuthenticationFilter

```
package com.senac.cafeteria.security;

import com.senac.cafeteria.services.MyUserDetailsService;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;
import java.io.IOException;

// Componente Spring que representa um filtro aplicado a cada requisição (uma vez por requisição)
@Component
public class JwtAuthenticationFilter extends OncePerRequestFilter {

    // Utilitário para operações com JWT (validação, extração de username, etc.)
    private final JwtUtil jwtUtil;
    // Serviço que carrega os detalhes do usuário (implementação de UserDetailsService)
    private final MyUserDetailsService userDetailsService;

    // Construtor com injeção de dependências do utilitário JWT e do serviço de usuários
    public JwtAuthenticationFilter(JwtUtil jwtUtil, MyUserDetailsService userDetailsService) {
        this.jwtUtil = jwtUtil;
        this.userDetailsService = userDetailsService;
    }

    /*
     * Método principal do filtro que é executado para cada requisição HTTP.
     * - Extrai o header Authorization
     * - Se houver um token Bearer válido, valida e obtém o username do token
     */
}
```

```

    * - Carrega UserDetails e cria uma Authentication para popular o
SecurityContext
    * - Continua a cadeia de filtros chamando filterChain.doFilter(...)
    */
    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain filterChain) throws
ServletException, IOException {

        // Lê o header Authorization da requisição
        String authHeader = request.getHeader("Authorization");
        String token = null;
        String username = null;

        // Verifica se o header começa com "Bearer " e extrai o token
        if (authHeader != null && authHeader.startsWith("Bearer ")) {
            token = authHeader.substring(7); // remove "Bearer " do começo
            // Valida o token e extrai o username se válido
            if (jwtUtil.validateToken(token)) {
                username = jwtUtil.extractUsername(token);
            }
        }

        // Se obtivemos um username e ainda não há autenticação no contexto de
segurança
        if (username != null &&
SecurityContextHolder.getContext().getAuthentication() == null) {
            // Carrega os detalhes do usuário a partir do serviço
            UserDetails userDetails =
userDetailsService.loadUserByUsername(username);
            // Cria um token de autenticação com as authorities do usuário
            UsernamePasswordAuthenticationToken authToken =
                new UsernamePasswordAuthenticationToken(userDetails, null,
userDetails.getAuthorities());
            // Define a autenticação no contexto de segurança do Spring
            SecurityContextHolder.getContext().setAuthentication(authToken);
        }

        // Continua a cadeia de filtros (essencial para que a requisição
prossiga)
        filterChain.doFilter(request, response);
    }

    /*
    * Método que permite pular o filtro para caminhos específicos.
    * Aqui está configurado para não filtrar endpoints de autenticação da
API.
    */
    @Override
    protected boolean shouldNotFilter(HttpServletRequest request) {
        // Obtém o caminho da requisição (ex: /api/auth/login)
        String path = request.getServletPath();
        // Retorna true para não aplicar o filtro quando o caminho começar com
/api/auth
        return path.startsWith("/api/auth");
    }
}

```

Classe JwtUtil

```
package com.senac.cafeteria.security;

import io.jsonwebtoken.*;
import io.jsonwebtoken.security.Keys;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

import java.security.Key;
import java.util.Date;

/*
 * Componente utilitário para geração, validação e extração de informações de
 * JWT.
 * Centraliza a lógica relacionada aos tokens JWT usados pela aplicação.
 */
@Component
public class JwtUtil {

    // Chave HMAC usada para assinar/validar tokens
    private final Key key;
    // Tempo de expiração em milissegundos configurado via
    application.properties
    private final long expirationMillis;

    /*
     * Construtor injeta a secret e a expiração a partir das propriedades da
     * aplicação.
     * A secret é convertida em Key compatível com a biblioteca jjwt.
     */
    public JwtUtil(@Value("${jwt.secret}") String secret,
                   @Value("${jwt.expiration}") long expirationMillis) {
        this.key = Keys.hmacShaKeyFor(secret.getBytes()); // cria a Key a
        partir do secret
        this.expirationMillis = expirationMillis; // guarda o tempo de
        expiração
    }

    /*
     * Gera um token JWT com o username como subject, issuedAt e expiration.
     * O token é assinado com a Key e o algoritmo HS256.
     */
    public String generateToken(String username) {
        Date now = new Date();
        Date expiry = new Date(now.getTime() + expirationMillis);

        return Jwts.builder()
            .setSubject(username)           // identifica o usuário no
            subject
            .setIssuedAt(now)                // data de emissão
            .setExpiration(expiry)          // data de expiração
            .signWith(key, SignatureAlgorithm.HS256) // assinatura do
            token
            .compact();                     // retorna o token compactado
    }

    /*
     * Valida o token: tenta fazer o parse com a chave e captura exceções de

```

```

validação.
    * Retorna true quando o token é válido e não expirou.
    */
    public boolean validateToken(String token) {
        try {

Jwts.parserBuilder().setSigningKey(key).build().parseClaimsJws(token);
            return true;
        } catch (JwtException | IllegalArgumentException ex) {
            // Qualquer problema ao parsear/validar o token indica token
            inválido
            return false;
        }
    }

    /**
     * Extrai o username (subject) do token JWT já validado.
     */
    public String extractUsername(String token) {
        Claims claims = Jwts.parserBuilder().setSigningKey(key).build()
            .parseClaimsJws(token).getBody();
        return claims.getSubject();
    }
}

```

2.4 Design Pattern Adotado

O projeto Cafeteria foi desenvolvido seguindo o padrão de arquitetura MVC (Model–View–Controller), que tem como principal objetivo manter o código organizado, modular e de fácil manutenção. Esse padrão divide o sistema em três camadas principais: Model, View e Controller, cada uma com responsabilidades bem definidas.

A camada Model representa os dados e as regras de negócio da aplicação, sendo composta por classes como Produto, Pedido e Usuario. Essa camada é responsável por armazenar, manipular e fornecer as informações necessárias para o funcionamento do sistema.

A camada View corresponde à interface gráfica que o usuário interage. No projeto, ela é formada pelas páginas localizadas na pasta templates/public, desenvolvidas com Thymeleaf, que exibem de forma visual as informações enviadas pelos controladores.

A camada Controller é responsável por intermediar a comunicação entre o Model e a View, recebendo as requisições dos usuários, processando os dados e retornando as páginas adequadas

Um exemplo claro dessa camada é a classe MenuController.java:

```
@Controller
public class MenuController {

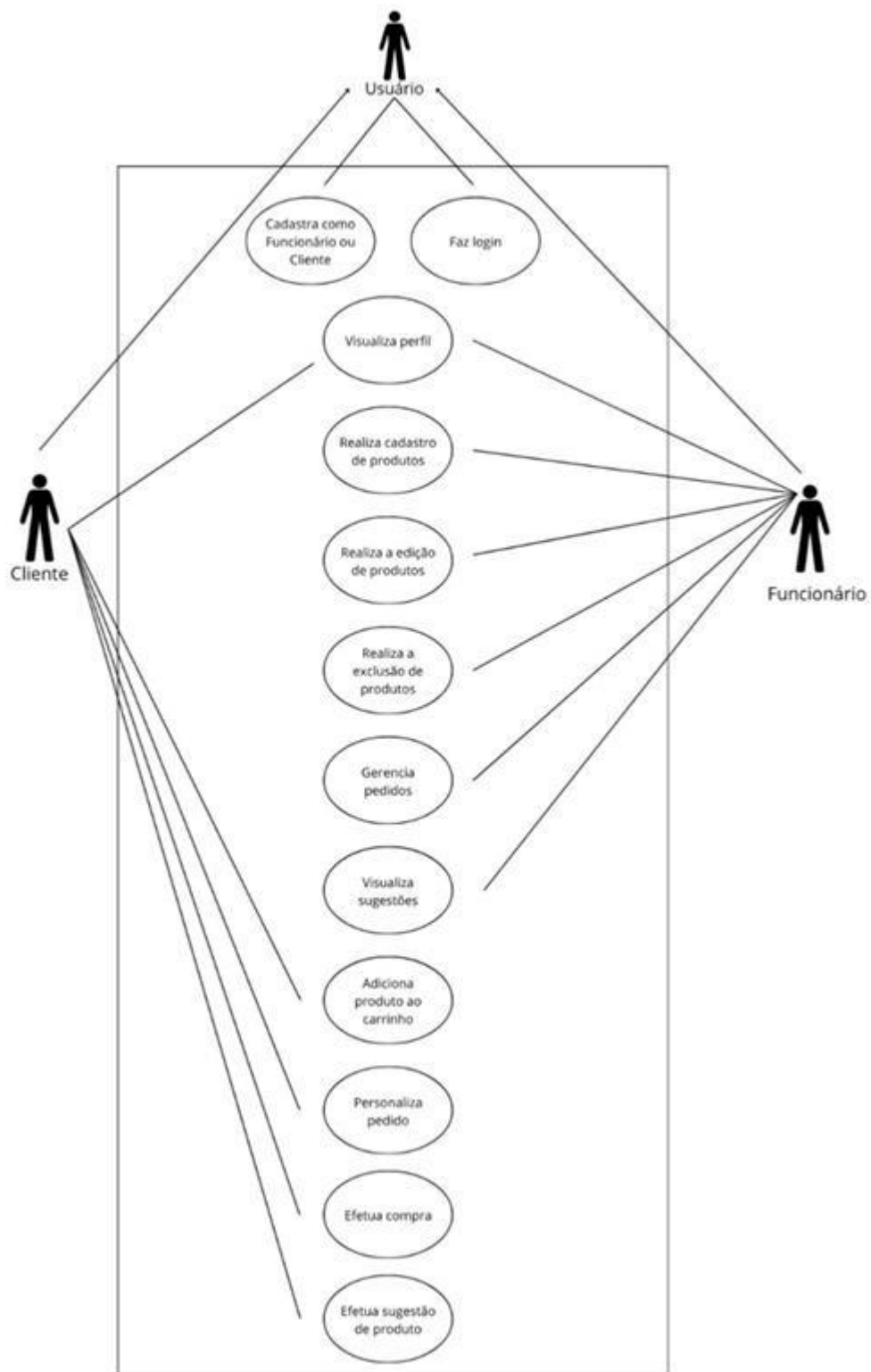
    @Autowired
    private ProdutoService produtoService;

    @GetMapping("/menu")
    public String menu(Model model) {
        var produtos = produtoService.listarTodos();

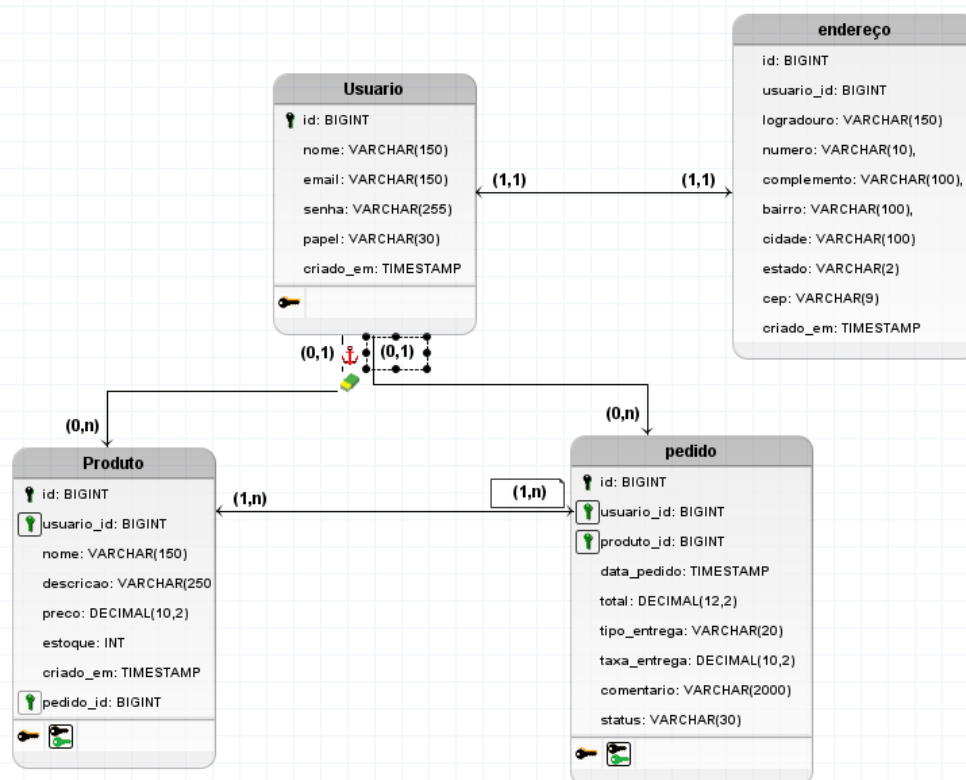
        // Converter imagens para Base64
        for (var produto : produtos) {
            if (produto.getImagem() != null) {
                String base64Image =
                    Base64.getEncoder().encodeToString(produto.getImagem());
                produto.setImagemBase64(base64Image);
            }
        }

        model.addAttribute("produtos", produtos);
        return "public/menu";
    }
}
```

2.5 Diagrama de Casos de Uso



2.6 Modelo Lógico



3. CONCLUSÃO

O desenvolvimento do sistema de e-commerce atendeu ao objetivo principal de criar um ambiente digital eficiente e intuitivo para a compra e venda de produtos.

A implementação das operações CRUD para vendedores e do sistema de carrinho de compras para clientes proporcionou maior praticidade no gerenciamento e aquisição de produtos.

A adição da aba de sugestões demonstrou ser um recurso de grande valor, promovendo uma relação mais próxima entre vendedor e cliente, com base na sugestão de novos produtos.

Com isso, o projeto cumpre seu papel como ferramenta tecnológica voltada para aprimorar o comércio eletrônico e a experiência do usuário.