**Documentação Completa: Sistema de Gestão de Estética Automotiva**

**Curso:** Técnico em Desenvolvimento de Sistemas  
**Instituição:** Senai

**Objetivo**

Desenvolver um sistema ERP para gestão de serviços de estética automotiva, com módulos para:

Agendamentos

Clientes

Funcionários

Produtos

Vendas

Dashboard analítico

**2. Pré-requisitos**

**Java 21:** JDK instalado e configurado.

**Maven:** Para gerenciamento de dependências.

**MySQL:** Banco de dados com Workbench.

**IDE:** VS Code ou IntelliJ.

**3. Configuração do Projeto**

**3.1. Estrutura do Projeto**

src/  
├── main/  
│ ├── java/  
│ │ └── com/erp/estetica/  
│ │ ├── controller/  
│ │ ├── model/  
│ │ ├── repository/  
│ │ └── EsteticaAutomotivaApplication.java  
│ └── resources/  
│ ├── static/  
│ │ ├── img/  
│ │ ├── css/  
│ │ └── js/  
│ └── templates/  
│ ├── fragments/  
│ ├── clientes/  
│ ├── agendamentos/  
│ └── ...

**3.2. Dependências (pom.xml)**

<dependencies>  
 <!-- Spring Boot -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
 </dependency>  
 <!-- MySQL -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <!-- DevTools -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
</dependencies>

**3.3. Configuração do Banco (application.yml)**

spring:  
 datasource:  
 url: jdbc:mysql://localhost:3306/estetica\_auto?useSSL=false&serverTimezone=UTC  
 username: root  
 password: senai123  
 driver-class-name: com.mysql.cj.jdbc.Driver  
 jpa:  
 hibernate:  
 ddl-auto: update  
 show-sql: true

**4. Entidades (Model)**

**4.1. Cliente**

@Entity  
@Table(name = "clientes")  
public class Cliente {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Integer id;  
 private String nome;  
 private String telefone;  
 // Getters e Setters  
}

**4.2. Agendamento**

@Entity  
@Table(name = "agendamentos")  
public class Agendamento {  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private Integer id;  
 @ManyToOne  
 private Servico servico;  
 private String clienteNome;  
 private LocalDateTime dataHora;  
 // Getters e Setters  
}

(O mesmo para Funcionario, Produto, Venda, Servico)

**5. Repositórios (JPA)**

public interface ClienteRepository extends JpaRepository<Cliente, Integer> {}

**6. Controllers**

**6.1. ClienteController**

@Controller  
@RequestMapping("/clientes")  
public class ClienteController {  
 @Autowired  
 private ClienteRepository clienteRepo;  
  
 @GetMapping  
 public String listar(Model model) {  
 model.addAttribute("clientes", clienteRepo.findAll());  
 return "clientes/lista";  
 }  
}

**6.2. DashboardController**

@Controller  
@RequestMapping("/dashboard")  
public class DashboardController {  
 @Autowired  
 private VendaRepository vendaRepo;  
  
 @GetMapping  
 public String dashboard(Model model) {  
 model.addAttribute("vendasMes", vendaRepo.totalVendasMes());  
 return "dashboard";  
 }  
}

**7.2. Página Sobre (sobre.html)**

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <link rel="stylesheet" th:href="@{/css/style.css}">  
</head>  
<body>  
 <div th:replace="~{fragments/navbar :: navbar}"></div>  
 <main class="container">  
 <h1>Sobre Nós</h1>  
 <p>Fundada em 2019, especializada em serviços automotivos...</p>  
 </main>  
</body>  
</html>

**7.3. Dashboard (dashboard.html)**

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>  
</head>  
<body>  
 <div th:replace="~{fragments/navbar :: navbar}"></div>  
 <canvas id="vendasChart"></canvas>  
 <script th:inline="javascript">  
 new Chart(document.getElementById('vendasChart'), {  
 type: 'bar',  
 data: {  
 labels: ['Jan', 'Fev', 'Mar'],  
 datasets: [{  
 label: 'Vendas',  
 data: [[${vendasMes}]],  
 }]  
 }  
 });  
 </script>  
</body>  
</html>

**8. Funcionalidades Avançadas**

**8.1. JavaScript (Tema Escuro)**

// static/js/theme.js  
document.getElementById('themeToggle').addEventListener('click', () => {  
 document.body.classList.toggle('dark-mode');  
});

**9. Execução e Deploy**

**9.1. Executar Localmente**

mvn spring-boot:run

Acesse: http://localhost:8080

**9.2. Build para Produção**

mvn clean package

Arquivo gerado: target/estetica-auto-0.0.1-SNAPSHOT.jar

**9.3. Acesso Remoto**

Configure o firewall para permitir a porta 8080.

Use o IP da máquina: http://[SEU\_IP]:8080.

**10. Conclusão**

Funcionalidades Implementadas:

CRUD para todos os módulos.

Dashboard com gráficos dinâmicos.

Navbar responsiva.

Páginas institucionais (Sobre, Contato).

Próximos Passos:

Adicionar autenticação com Spring Security.

Implementar relatórios em PDF.

Configurar envio de e-mails para agendamentos.

Documentação Preparada por: [Gesiel Henrique/Lucas Matheus]  
Data: 17/03/2025