**Documentação Técnica - Testes no Backend**

## **1. Erro: "No database selected"**

### **Contexto do Teste:**

* **Cenário:**  
   Ao acessar o módulo de **Clientes** ou outros módulos que executam consultas no banco de dados, a aplicação retorna um erro de conexão.
* **Mensagem de Erro:**

pgsql

CopiarEditar

JDBC exception executing SQL [select c1\_0.id,c1\_0.nome,c1\_0.telefone from clientes c1\_0] [No database selected] [n/a]

### **Análise:**

* O erro indica que o MySQL não sabe qual banco de dados utilizar, pois o nome do banco não foi especificado na URL de conexão.

### **Solução Aplicada:**

* **Atualização do arquivo application.properties:**
  + Incluímos o nome do banco de dados na URL de conexão.
  + Exemplo:

properties

CopiarEditar

spring.datasource.url=jdbc:mysql://localhost:3306/estetica\_auto?allowPublicKeyRetrieval=true&useSSL=false&serverTimezone=UTC  
spring.datasource.username=root  
spring.datasource.password=

* **Resultado:**  
   Após essa alteração, a conexão com o MySQL foi estabelecida corretamente e as consultas passaram a funcionar.

## **2. Erro: Dependência do MySQL ausente no pom.xml**

### **Contexto do Teste:**

* **Cenário:**  
   Ao iniciar a aplicação, o Spring Boot não conseguia conectar-se ao banco de dados MySQL, pois a dependência do driver JDBC não estava adicionada ao pom.xml.
* **Mensagem de Erro:**

pgsql

CopiarEditar

Cannot load JDBC driver class 'com.mysql.cj.jdbc.Driver'

### **Análise:**

* O erro ocorre porque o projeto não encontrava o driver necessário para se conectar ao MySQL.
* A ausência da dependência mysql-connector-java no pom.xml impedia a comunicação entre a aplicação e o banco de dados.

### **Solução Aplicada:**

* **Adição da dependência do MySQL no pom.xml**

xml

CopiarEditar

<dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <scope>runtime</scope>  
</dependency>

* **Resultado:**  
   Após a inclusão dessa dependência e a atualização do Maven, a aplicação conseguiu reconhecer o driver e conectar-se corretamente ao banco de dados.

## **3. Erro: Problema na Formatação de Data em Template Thymeleaf**

### **Contexto do Teste:**

* **Cenário:**  
   Na tela de **Agendamentos**, ao tentar exibir a data e hora formatadas utilizando o atributo:

html

CopiarEditar

<td th:text="${#dates.format(agendamento.dataHora, 'dd/MM/yyyy HH:mm')}"></td>

a aplicação retornava uma exceção durante o parsing do template.

* **Mensagem de Erro:**

vbnet

CopiarEditar

Exception evaluating SpringEL expression: "#dates.format(agendamento.dataHora, 'dd/MM/yyyy HH:mm')"   
...  
Caused by: org.springframework.expression.spel.SpelEvaluationException: EL1004E: Method call: Method format(java.time.LocalDateTime,java.lang.String) cannot be found on type org.thymeleaf.expression.Dates

### **Análise:**

* A expressão #dates não possui o método format para objetos do tipo java.time.LocalDateTime.
* O Thymeleaf (versões recentes) disponibiliza o utilitário #temporals para manipulação e formatação de objetos da API java.time.

### **Solução Aplicada:**

* **Alteração do template:**
  + Substituímos a expressão #dates.format(...) por #temporals.format(...).
  + Exemplo corrigido:

html

CopiarEditar

<td th:text="${#temporals.format(agendamento.dataHora, 'dd/MM/yyyy HH:mm')}"></td>

* **Resultado:**  
   Com a alteração, o template foi processado corretamente e as datas passaram a ser exibidas no formato desejado.

## **Conclusão**

Esses testes e as respectivas correções garantiram que:

* A conexão com o banco de dados seja estabelecida corretamente, evitando erros de "No database selected".
* O driver MySQL seja reconhecido pelo Spring Boot, permitindo a conexão com o banco.
* Os templates Thymeleaf sejam renderizados corretamente, com a formatação de datas ajustada para trabalhar com objetos LocalDateTime.