

# **Sistema de Cadastro de Automóveis AutoPrime**

Lucas Diniz de Araujo Pereira

Ciências da Computação - Universidade Estácio de Sá (UNESA)

CEP: 24020340 - Niterói - RJ – Brasil

Programação Orientada a Objetos em Java

[202203897411@alunos.estacio.br](mailto:202203897411@alunos.estacio.br)

**Abstract.** This article will explain and demonstrate the operation of a Project in Java language implemented with PostgreSQL Database, with the function of registering cars of a fictitious company “AutoPrime”.

**Resumo.** Este artigo irá explicar e demonstrar o funcionamento de um Projeto em linguagem Java implementado com Banco de Dados PostgreSQL, com a função de realizar o Cadastro de carros de uma empresa fictícia chamada “AutoPrime”.

## **1. Nome do sistema**

Cadastro de Automóveis AutoPrime.

## **2. Objetivo do sistema**

O sistema tem como meta facilitar a manipulação de informações dos dados contidos na empresa “AutoPrime”.

## **3. Requisitos do sistema / o que o sistema irá fazer**

Este é um projeto de linguagem Java com SQL que faz o cadastro de automóveis, onde o mesmo consegue criar, consultar ou excluir dados que estão dentro do banco de dados de carros da empresa. Ao iniciar o programa, será apresentado ao usuário um menu no console com as 4 operações de CRUD do Banco de Dados para serem escolhidas. Além disso, o sistema apresenta outro diferencial de mostrar os carros que foram fabricados após o ano de 2010, se for de preferência do usuário.

#### **4. Caso de uso do sistema**

Esse sistema pode ser usado por exemplo, para um possível administrador de uma empresa que venda veículos multimarcas e deseja registrar os carros que o mesmo possui, de forma simples e eficiente.

#### **5. Explicação do que foi implementado**

##### **5.1 SistemaCadastro.java**

Certifique se o JDBC está instalado na máquina e ativado na IDE correspondente.

Primeiramente é criada a classe principal do programa, denominada de “SistemaCadastro.java”. Na linha 4, define uma variável estática chamada `driverJDBC` do tipo *String*. Essa variável armazena o valor “org.postgresql.Driver”, que é o nome da classe do driver JDBC para o PostgreSQL. Logo após isso, o programa tenta carregar o driver para começar o funcionamento através dos blocos *try* e *catch*. Depois do driver ter carregado com sucesso, será imprimido na tela o menu inicial do programa, exigindo que o usuário digite alguma das opções apresentadas.

##### **5.2 CriarTabela.java**

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

```

No início da classe java “CriarTabela” são declaradas as classes necessárias para trabalhar com o JDBC, que é uma API para conectar e interagir com bancos de dados em Java. Na linha 8, é declarada uma variável chamada “SQLcriarTabela” do tipo *String* que contém a instrução SQL para criar uma tabela chamada "automovel" no banco de dados. A tabela possui quatro colunas: "modelo", "marca", "ano" e "cor", com seus respectivos tipos de dados e tamanhos. Depois declara uma variável chamada driver do tipo String que armazena a URL de conexão com o banco de dados PostgreSQL. Nesse caso, a URL indica que estamos conectando ao banco "SistemaAutomoveis" localmente na máquina ("localhost").

```

try (Connection conn = DriverManager.getConnection(url:driver, user: "postgres", password: "1234")){
    if(conn!=null){
        System.out.println(x: "Connected to the database!");
    }else{
        System.out.println(x: "Failed to make connection!");
    }
    System.out.println(x: "Criando dados da tabela, aguarde...");
    st = conn.createStatement();
    st.executeUpdate(sql:SQLcriarTabela);
    System.out.println(x: "Tabela criada c/ sucesso!");
    st.close();
    conn.close();
} catch (SQLException e){
    System.err.format(format: "SQL State: %s\n%s", args: e.getSQLState(), args: e.getMessage());
}

```

Esse trecho verifica se o sistema consegue se conectar com sucesso ao banco de dados nos blocos de try e catch. Caso se conecte, ele irá dar a instrução pro banco de dados criar a tabela de acordo com as especificações pré definidas. Caso ocorra falha ao se conectar ao banco, é imprimido na tela o estado do SQL, através das funções “e.getSQLState()” e “e.getMessage()”, que servem para capturar e tratar qualquer exceção do tipo SQLException que possa ocorrer durante a execução do bloco try.

### 5.3 AtualizarDados.java

Nesta classe que será permitido a manipulação dos dados. O usuário terá que preencher os campos pedidos pelo console, como Modelo, marca, ano e cor.

```
PreparedStatement pstmt = conn.prepareStatement(sql:SQLInserirDados);  
pstmt.setString(parameterIndex:1, x: modelo);  
pstmt.setString(parameterIndex:2, x: marca);  
pstmt.setInt(parameterIndex:3, x: ano);  
pstmt.setString(parameterIndex:4, x: cor);  
  
System.out.println(x: "Atualizando dados na tabela...");  
pstmt.executeUpdate();  
System.out.println(x: "Dados Inseridos!");
```

No código apresentado acima, é realizado a atualização dos dados na tabela do banco de dados.

Na linha 31, o método `executeUpdate()` é chamado no objeto `pstmt`, que representa um `PreparedStatement`. Esse método executa a instrução SQL preparada e atualiza os dados no banco de dados.

Em seguida, na linha 32, são definidos os valores dos parâmetros da instrução SQL preparada utilizando os métodos `setString()` e `setInt()` do objeto `pstmt`. Cada valor é associado a um marcador de posição (?) na instrução SQL.

Na linha 38, o programa executa a atualização dos dados chamando o método `executeUpdate()` no objeto `pstmt`. Isso faz com que a instrução SQL seja enviada ao banco de dados e os dados sejam atualizados na tabela.

Por fim, nas linhas 41 e 42, os objetos `pstmt` e `conn` são fechados utilizando o método `close()`, liberando os recursos utilizados. Caso ocorra uma exceção do tipo `SQLException`, a mensagem de erro é exibida nas linhas 43 e 44.

Ao executar o código, já é possível ver os dados inseridos no Banco de Dados PostgreSQL:

Data Output Messages Notifications				
	modelo character varying (60)	marca character varying (60)	ano integer	cor character varying (60)
1	Palio	Fiat	1997	prata

## 5.4 LerDados.java

Esta é uma classe mais simples, feita para consultar as informações existentes no Banco de Dados. Na linha 25, é iniciado um loop while para percorrer os registros do ResultSet. A cada iteração do loop, a função next() é chamada no objeto result para avançar para o próximo registro. Se houver um próximo registro, o loop continua. Caso contrário, o loop é encerrado.

Dentro do loop, os valores de cada coluna do registro atual são recuperados utilizando os métodos getString() e getInt() do objeto result. Os valores são então exibidos na saída padrão, apresentando os detalhes do automóvel. O loop continua até que todos os registros sejam percorridos.

```
Informe a opção desejada:
2
Connected to the database!
Lendo dados da tabela...
-----
Modelo: Palio
Marca: Fiat
Ano: 1997
Cor: prata
```

**Resultado após a compilação do código.**

## 5.5 ApagarDados.java

Essa classe é responsável por remover todos os dados existentes contidos no banco. Na linha 8, é declarada a variável “SQLExcluirDados” em que atribui a instrução de SQL para apagar

completamente as informações contidas dentro da Database. O comando é enviado para o banco de dados na linha 20, com a instrução “st.executeUpdate(SQLExcluirDados)”.

## 5.6 Mostrar2010.java

Por fim, a última classe do programa, tem a função de imprimir na tela apenas os carros fabricados após 2010. É definida na linha 9 a consulta SQL que será executada no banco de dados. A consulta busca todos os registros da tabela "automovel" em que o ano seja maior que 2010.

```
Informe a opção desejada:
5
Connected to the database!
Carros fabricados após 2010...
-----
Modelo: Gol
Marca: Volkswagen
Ano: 2011
Cor: Branco
```

Após a compilação do código.

## 6. Conclusão

Em resumo, este sistema consegue solucionar o problema e oferece todas as funções básicas de armazenamento de dados de forma prática e simples, porém pela falta de uma interface gráfica, pode não ser visualmente receptivo para todos os públicos, fazendo com que possivelmente algumas pessoas ao utilizarem o sistema pela primeira vez sintam-se confusas. No geral, apesar da simplicidade, o sistema de cadastro cumpre com eficiência todas as normas propostas pelo trabalho.