Sistemas de Compra/Venda/Cadastro de veículos em Java e SQL

Hugo P.B. Cunha, Brenno P. Dos Santos

NITERÓI I - Universidade Estácio de Sá (UNESA)
24020-340 – Niterói – RJ – Brasil

abesimoggg@gmail.com , brennoparaizo14@gmail.com

Abstract: *In this article, the PostgreSQL database is used to store and retrieve information about vehicles. The project's objective is to create a car management system. The application of the system aims to optimize inventory processes, product management, and improve operational efficiency. The main development stages will be discussed, highlighting the implemented features and the benefits obtained from the application of the system.*

Resumo: *Neste artigo, utiliza o banco de dados PostgreSQL para armazenar e recuperar informações sobre os veículos. O objetivo do projeto é criar um sistema de gerenciamento de automóveis. A aplicação do sistema busca otimizar processos de estoque, gerenciamento de produtos e melhorar a eficiência operacional. Serão abordadas as principais etapas do desenvolvimento, destacando as funcionalidades implementadas e os benefícios obtidos com a aplicação do sistema.*

1. Introdução ao Sistema

O nome "CarMart" foi escolhido levando em consideração sua relevância e adequação ao contexto de um local onde é possível encontrar uma variedade de carros para compra. Além disso, o sistema permite que os usuários cadastrem e anunciem seus veículos para venda. Essa escolha de nome foi inspirada na conhecida plataforma online de compra e venda "OLX", reconhecida por sua plataforma de negociação de diversos produtos.

2. Objetivos do Sistema

O CarMart foi desenvolvido para solucionar a problemática do gerenciamento de uma loja de carros, simplificando as operações de venda e administração. O sistema busca melhorar a eficiência e a experiência dos usuários, oferecendo uma plataforma intuitiva para realizar transações e gerenciar informações de veículos de forma mais eficaz.

3. Requisitos do Sistema

O sistema CarMart oferece funcionalidades de CRUD, permitindo criar com o método '**cadastrarVeiculo'**, consultar no método '**visualizaListaCarros'**, atualizar no método '**editarVeiculo'** e excluir no método '**removerVeiculo'**, no banco de dados. Também possui validação de entrada de nomes, utilizando uma lista pré-autorizada. Isso garante

um gerenciamento eficiente e seguro das informações na loja de carros, além de facilitar o gerenciamento de uma loja de carros. Ele otimiza o processo de compra e venda na loja, proporcionando uma experiência interativa e intuitiva aos usuários.

4. Casos de Utilização

O CarMart será utilizado para os usuários visualizarem os carros disponíveis, comprar veículos para efetuar o cadastro de veículos à venda, permitindo a edição das informações dos veículos e sua remoção do sistema. Essas funcionalidades simplificam o gerenciamento e a compra/venda de carros na loja, proporcionando uma experiência agradável aos usuários.

5. Implementações: Principais classes e métodos

Neste código, são utilizados classes e métodos para interagir com o banco de dados e realizar as operações necessárias. Abaixo são explicados tais implementações presentes no código:

5.1. Classes

Classe "**TrabalhoJava**": Esta é a classe principal que contém o método main, é o ponto de entrada do programa e onde a execução começa. O método main estabelece a conexão com o banco de dados, cria a tabela "automoveis" se ela não existir e exibe um menu para o usuário interagir com o sistema.

Classe "**Connection**": Possibilita a conexão com o banco de dados. É obtida através do método getConnection da classe DriverManager. No código, a conexão é estabelecida com o banco de dados PostgreSQL usando a URL, nome de usuário e senha fornecidos.

Classe "**Statement**": É uma interface usada para executar instruções SQL estáticas e retornar os resultados produzidos. No código, é utilizado para executar as instruções SQL de criação de tabela, inserção de dados, exclusão de dados e atualização de dados.

Classe "**PreparedStatement**": É uma subinterface de Statement que representa uma instrução SQL precompilada. É usada para executar instruções SQL parametrizadas. No código, é utilizada para realizar consultas parametrizadas, como selecionar um veículo específico para compra ou edição.

Classe "**ResultSet**": Representa um conjunto de resultados de uma consulta SQL. Permite navegar pelos registros retornados pela consulta e obter os valores das colunas. No código, é utilizado para recuperar os resultados das consultas realizadas, como a lista de carros disponíveis ou um veículo selecionado.

5.2. Métodos

Além dessas classes, existem diversos métodos presentes no código que realizam diversas operações, como:

visualizarListaCarros: Recupera os dados de todos os carros disponíveis no banco de dados e os exibe na tela.

comprarVeiculo: Permite ao usuário selecionar e comprar um veículo, removendo-o do banco de dados.

cadastrarVeiculo: Solicita ao usuário os dados de um novo veículo e o cadastra no banco de dados.

editarVeiculo: Permite ao usuário selecionar e editar os dados de um veículo existente no banco de dados.

visualizarListaVeiculos: Recupera os dados de todos os carros disponíveis no banco de dados, ordenados por ano de forma decrescente, e os exibe na tela.

removerVeiculo: Permite ao usuário selecionar e remover um veículo do banco de dados.

Esses são os principais elementos presentes no código fornecido. Cada classe e método desempenha um papel específico no gerenciamento e manipulação dos dados do banco de dados.

5.3. Banco de dados

É feito de uma forma que utiliza um banco de dados PostgreSQL para armazenar informações sobre automóveis. O banco de dados é acessado por meio de conexões JDBC (Java Data-base Connectivity), que permitem a comunicação entre a aplicação Java e o banco de dados. A estrutura do banco de dados é definida através de uma tabela chamada "automoveis". Essa tabela possui as seguintes colunas:

"**id**": Um identificador único para cada automóvel, com tipo de dados **SERIAL**, que é um tipo numérico auto incremental no PostgreSQL.

"**id_estoque**": Um identificador para controle do estoque de automóveis. É um número inteiro.

"**marca**": Uma coluna para armazenar a marca do automóvel, com tipo de dados **VARCHAR(100)**, que permite até 100 caracteres.

"**modelo**": uma coluna para armazenar o modelo do automóvel, também com tipo VARCHAR(100).

"**ano**": uma coluna para armazenar o ano de fabricação do automóvel, com tipo INT (inteiro).

"**valor**": uma coluna para armazenar o valor do automóvel, com tipo NUMERIC(10, 2), que representa um número decimal com até 10 dígitos, sendo 2 deles reservados para a parte decimal.

No código, são realizadas operações de criação da tabela "automoveis" caso ela ainda não exista, inserção de registros na tabela para representar automóveis, leitura dos registros da tabela para exibição na tela, compra de um automóvel (removendo-o da tabela), cadastro de um novo automóvel na tabela, visualização da lista de carros em ordem decrescente por ano e edição de um automóvel existente na tabela. Essas operações são realizadas utilizando comandos SQL (Structured Query Language)

executados através de objetos Statement e PreparedStatement do JDBC, que permitem enviar consultas e atualizações para o banco de dados.

5.4. Elementos

No código, foram escolhidos os seguintes elementos relacionados ao banco de dados PostgreSQL:

PostgreSQL: É um sistema gerenciador de banco de dados relacional conhecido por sua confiabilidade e recursos avançados.

JDBC: API do Java que permite a interação com diferentes bancos de dados relacionais.

SQL: Linguagem padrão para manipular bancos de dados relacionais.

PreparedStatement: Classe do JDBC que cria consultas parametrizadas para garantir segurança e eficiência.

Essas escolhas visam garantir a integridade, segurança e eficiência na manipulação dos dados do banco de dados PostgreSQL, aproveitando a compatibilidade entre o Java e o PostgreSQL fornecida pelo JDBC.

6. Conclusão

No nosso código, é possível identificar diversas funcionalidades como a criação de tabelas de banco de dados para representar os atributos dos carros, como marca, modelo, ano, preço e disponibilidade.

Através do código, é possível executar operações de CRUD (Create, Read, Update, Delete), isso significa que é possível adicionar novos registros de carros, recuperar informações existentes, atualizar os detalhes de um carro específico e remover carros do sistema.

Agradecemos pela atenção! Estamos felizes com nosso código, conseguimos implementar tudo que planejávamos. Esperamos ter mais experiências desafiadoras como essa e nos sentimos gratificados por concluí-la com sucesso.

Obrigado pela sua ajuda e orientação. Guardaremos essa experiência como uma lembrança inspiradora. Desejamos o melhor e estamos ansiosos por novos desafios!