Sistema de Logística de Produtos de um Mercado.

Raphael Wilson, Victor Couto

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

raphaelwilsonsilva17@gmail.com, vcneto76@gmail.com

Abstract. *In this project for a market's product logistics system, our goal is to build a Java code for querying a PostgreSQL database that performs the primary CRUD (Create, Read, Update, and Delete) functions of a database. Additionally, it displays options to showcase all registered products in the database and the functionality to display all products with a value greater than one hundred Brazilian reais.*

Resumo. *Neste projeto de sistema de logística de produtos de um mercado visamos construir um código em Java de consulta em banco de dados PostgreSQL que realiza as principais funções de CRUD(Create, Read, Update e Delete) de uma base de dados. Além disso, exibe as opções de mostrar todos os produtos cadastrados no banco de dados e a função de exibir todos os produtos com valor maior do que cem reais*

1. Objetivos do Sistema

O sistema desenvolvido tem o objetivo de facilitar o gerenciamento de produtos de um mercado, possibilitando a fácil criação, acesso, consulta, alteração e remoção dos dados de uma base de dados criada pelo usuário, economizando o tempo do usuário ao lidar com preços, códigos e nomes de produtos.

2. Requisitos do Sistema

Cadastrar produto: Permite que o usuário cadastre um novo produto no banco de dados.

Consultar produto: Exibe uma lista onde gera as seguintes opções: 1- Consultar por ID, 2- Consultar por número, 3- Consultar por nome, 4- Consultar por valor, 5- Produtos com valor acima de R$100, 6- Todos os produtos.

Atualizar produto: Exibe uma lista onde gera as seguintes opções: 1- Atualizar por número, 2- Atualizar por nome, 3- Atualizar por valor.

Deletar produto: Exibe uma lista onde gera as seguintes opções: 1- Excluir por ID, 2- Excluir por número, 3- Excluir por nome, 4- Excluir por valor, 5- Excluir tudo.

Sair: Fecha o programa.

3. Casos de Utilização

O usuário inicia o programa e é verificado se ele já utilizou o programa anteriormente, caso não tenha utilizado será criado uma nova tabela "Produto". Em seguida, será exibido um menu onde possibilita o usuário selecionar uma das opções:

1°) Registrar um Produto: Primeiramente é solicitado o número do produto, depois o nome e por último o valor. Caso o usuário tenha digitado corretamente, o produto será cadastrado no banco de dados e estará disponível para consulta e/ou manipulação.

2°) Consultar um Produto: Ao selecionar a opção de consultar um produto, o usuário será levado a uma tela que exibe as seguintes opções: consultar por ID, consultar por número, consultar por nome, consultar por valor, listar produtos com valor acima de R$100, listar todos os produtos. Nas opções de consulta por ID, número, nome ou valor será solicitado ao usuário que digite o atributo do produto que deseja consultar.

3°) Atualizar um Produto: Ao selecionar a opção de atualizar um produto, o usuário será levado a uma tela que exibe as seguintes opções: atualizar número, atualizar nome ou atualizar valor. Em qualquer uma das opções será solicitado ao usuário, que digite o valor do atributo que deseja alterar, em seguida será solicitado um novo valor que substituirá o antigo.

4°) Deletar um Produto: Ao selecionar a opção de excluir um produto, o usuário será levado a uma tela que exibe as seguintes opções: excluir por ID, excluir por número, excluir por nome, excluir por valor e excluir tudo. Em seguida será solicitado o valor do campo do produto a ser excluído, caso o usuário opte por excluir tudo, toda a tabela será excluída.

5°) Sair: O programa será encerrado.

Em qualquer uma das opções, com exceção da opção sair, ao término da operação o menu será executado novamente.

4. Implementação

4.1. Classe Main

A classe Main, que é onde serão executados os principais métodos que o usuário irá experienciar, instacia seis métodos e uma variável do tipo Scanner. Sendo os métodos: cadastro, registrarProduto, consultarProduto, atualizarProduto, excluirProduto e o método main, onde o código será executado.

1°) Método cadastro: Apresenta ao usuário um menu de escolha entre as diferentes operações realizáveis pelo programa, em seguida cria uma variável que permite receber a entrada do usuário por meio da classe scanner. Em seguida, utiliza um switch na escolha do usuário, chamando o método escolhido de acordo com a opção escolhida do menu, além disso, cria a opção default que exibe uma mensagem de alerta ao usuário caso ele escolha uma opção inválida.

2°) Método registrarProduto: Cria um objeto da classe InserindoDados, em seguida, pede ao usuário que digite o número do produto a ser cadastrado e cria uma variável do tipo inteiro utilizando o scanner para receber o dado. Após isso, é solicitado o nome do produto e o armazena em uma variável do tipo string usando o scanner. Posteriormente, pede ao usuário que digite o valor do produto e o armazena em uma variável do tipo double também utilizando o scanner. Logo depois, utiliza o método inserir da classe InserindoDados no objeto criado anteriormente, utilizando como parâmetros as variáveis previamente preenchidas pelo usuário.

3°) Método consultarProduto: Cria um objeto da classe ConsultandoDados, em seguida, pede ao usuário que digite o método de consulta escolhido e cria uma variável do tipo

inteiro utilizando o scanner para receber o dado. Em seguida, utiliza um switch na escolha do usuário, caso ele escolha as opções de consultar por ID, por número, por nome ou por valor, serão exibidas mensagens que pedem ao usuário que digite o valor do atributo do produto escolhido para que sejam apresentadas todas as informações do produto, em seguida, uma variável do tipo do atributo escolhido será criada e armazenará a entrada do usuário para que seja passado como parâmetro no método consultar da classe ConsultandoDados, que será chamado logo em seguida. Caso ele escolha a opção de listar produtos com valor acima de R$100,00 ou a opção de listar todos os produtos apenas será chamado o método consultar e o parâmetro será null. Por fim, cria a opção default que exibe uma mensagem de alerta ao usuário caso ele escolha uma opção inválida.

4°) Método atualizarProduto: Cria um objeto da classe AtualizandoDados, em seguida, pede ao usuário que digite o método de atualização escolhido e cria uma variável do tipo inteiro utilizando o scanner para receber o dado. Em seguida, utiliza um switch na escolha do usuário e exibe, de acordo com a opção escolhida, uma mensagem que solicita ao usuário que digite o valor do atributo a ser atualizado e o armazena em uma variável do tipo do atributo escolhido. Após isso, solicita-se ao usuário que digite o novo valor do atributo que deseja atualizar e armazena o valor digitado em uma variável do tipo do atributo escolhido. Logo depois, utiliza o método atualizar da classe AtualizandoDados no objeto criado anteriormente, utilizando como parâmetros as variáveis previamente preenchidas pelo usuário. Por fim, cria a opção default que exibe uma mensagem de alerta ao usuário caso ele escolha uma opção inválida.

5°) Método excluirProduto: Cria um objeto da classe ExcluindoDados, em seguida, pede ao usuário que digite o método de exclusão escolhido e cria uma variável do tipo inteiro utilizando o scanner para receber o dado. Em seguida, utiliza um switch na escolha do usuário e exibe, de acordo com a opção escolhida, uma mensagem que solicita ao usuário que digite o valor do atributo a ser excluído e o armazena em uma variável do tipo do atributo escolhido. Caso a opção escolhida seja a opção de excluir tudo, toda a tabela do banco de dados será excluída. Logo depois, utiliza o método excluir da classe ExcluindoDados no objeto criado anteriormente, utilizando como parâmetros as variáveis previamente preenchidas pelo usuário, caso a opção escolhida seja a opção de excluir tudo, o parâmetro do método será null. Por fim, cria a opção default que exibe uma mensagem de alerta ao usuário caso ele escolha uma opção inválida.

6°) Método main: Cria objetos das classes ConectandoDriver, ConectandoBD e CriandoTabela, em seguida, utiliza o método conectarDriver da classe ConectandoDriver, conectarBD da classe ConectandoBD e criarTabela da classe CriandoTabela. Por fim, executa o método cadastro.

4.2. Classe ConectandoDriver

A classe ConectandoDriver cria o método conectarDriver, que, para tratar erros, utiliza o bloco try e catch. Dentro do try é exibida uma mensagem que informa ao usuário que a conexão está sendo realizada, se não houver erros, indica posteriormente que o driver foi carregado. Caso algum erro ocorra, o bloco catch irá indicar que houve uma falha no carregamento do driver.

4.3. Classe ConectandoBD

A classe ConectandoBD cria o método conectarBD, que, para tratar erros, utiliza o bloco try e catch. Primeiramente é criada uma variável do tipo string que contém o URL de

conexão com o driver do PostgreSQL, juntamente com o endereço IP do servidor, a porta de conexão e o nome do banco de dados. No cabeçalho do bloco try é estabelecida a conexão com o banco de dados utilizando a URL de conexão, o nome de usuário e a senha. Dentro do bloco try utiliza-se a condicional if para verificar se a conexão retorna um valor nulo, caso retorne, é apresentada uma mensagem que indica que a conexão falhou. Já no bloco catch, verifica-se se houve algum erro do tipo SQLException, caso ocorra algum erro deste tipo será apresentada uma mensagem fornecendo informações sobre o erro ocorrido.

4.4. Classe CriandoTabela

A classe CriandoTabela cria o método criarTabela, que, para tratar erros, utiliza o bloco try e catch. Primeiramente é criada uma variável do tipo string que contém o SQL necessário para criar uma tabela de nome produto com os atribulos id do tipo serial, numeroproduto do tipo inteiro, nomeproduto do tipo caracter variando até 60 caracteres e valorproduto do tipo real. Em seguida, é criada uma variável do tipo string que contém o URL de conexão com o driver do PostgreSQL, juntamente com o endereço IP do servidor, a porta de conexão e o nome do banco de dados. Após isso, é criado um objeto do tipo Statement para executar instruções SQL. No cabeçalho do bloco try é estabelecida a conexão com o banco de dados utilizando a URL de conexão, o nome de usuário e a senha. Dentro do bloco try utiliza-se a condicional if para verificar se a conexão retorna um valor nulo, caso retorne, é apresentada uma mensagem que indica que a conexão falhou. Depois disso, é criado o objeto do tipo DatabaseMetaData que retorna informações sobre o banco de dados, como tabelas existentes. Essa informação é utilizada logo em seguida, no objeto do tipo ResultSet, que irá retornar informações sobre a tabela chamada produto. Uma condicional if é estabelecida para verificar se não existe uma tabela produto, utilizando o objeto do tipo ResultSet criado anteriormente, caso não exista, a condicional será verdadeira e será criada uma tabela utilizando o objeto Statement e o método executeUpdate com a variável com as informações para criar a tabela. Logo após, será fechado objeto Statement e o objeto ResultSet. Já no bloco catch, verifica-se se houve algum erro do tipo SQLException, caso ocorra algum erro deste tipo será apresentada uma mensagem fornecendo informações sobre o erro ocorrido.

4.5. Classe InserindoDados

A classe InserindoDados cria o método inserir, que recebe como parâmetros o número do produto do tipo inteiro, o nome do produto do tipo string e o valor do produto, do tipo double, todos recebidos do usuário. Nele, é criada a variável do tipo string que contém as informações SQL que permitem adicionar à tabela do banco de dados os valores recebidos como parâmetros, a variável do tipo string que contém o URL de conexão e o objeto do tipo Statement que é inicializado com o valor null. Logo após, é aberto um bloco try, que tenta fazer a conexão com o banco de dados, utilizando a string com a URL, o usuário e a senha do utilizador do banco de dados. Dentro do bloco try, utiliza-se a condicional if para verificar se a conexão retorna um valor nulo, caso retorne, é apresentada uma mensagem que indica que a conexão falhou. Caso contrário, será exibida uma mensagem que indica que os dados serão inseridos na tabela e é utilizado o método executeUpdate do objeto statement, passando como parâmetro a string que contém as informações a serem inseridas, o objeto statement e a conexão são encerradas logo em seguida. Já no

bloco catch, verifica-se se houve algum erro do tipo SQLException, caso ocorra algum erro deste tipo será apresentada uma mensagem fornecendo informações sobre o erro ocorrido.

4.6. Classe ConsultandoDados

A classe ConsultandoDados cria o método consultar, que recebe como parâmetros a escolha do usuário e o parâmetro do item consultado, ambos recebidos do usuário. Primeiramente, é criada uma variável do tipo string com o valor de uma string vazia que servirá para ser reatribuída no bloco switch que vem a seguir. Em seguida, é criado o bloco switch que verifica a escolha do usuário e retorna o resultado baseado na escolha do método de consulta do método consultarProduto da classe Main. De acordo com cada escolha é atribuído um valor diferente na variável criada anteriormente, esse valor contém a instrução SQL necessária para fazer a operação solicitada. Também é criada a opção default que exibe uma mensagem de alerta ao usuário caso ele escolha uma opção inválida. Logo após é criada a variável do tipo string que contém o URL de conexão e o objeto do tipo Statement que é inicializado com o valor null. Depois disso é aberto um bloco try, que tenta fazer a conexão com o banco de dados, utilizando a string com a URL, o usuário e a senha do utilizador do banco de dados. Dentro do bloco try, utiliza-se a condicional if para verificar se a conexão retorna um valor nulo, caso retorne, é apresentada uma mensagem que indica que a conexão falhou. Caso contrário, será exibida uma mensagem que indica que os dados serão consultados na tabela e é utilizado o método executeQuery do objeto statement, passando como parâmetro a string que contém as informações a serem consultadas. Posteriormente, inicia-se um loop while que imprimirá todas as informações do produto até que o objeto ResultSet pare de retornar itens na tabela. O objeto statement e a conexão são encerradas logo em seguida. Já no bloco catch, verifica-se se houve algum erro do tipo SQLException, caso ocorra algum erro deste tipo será apresentada uma mensagem fornecendo informações sobre o erro ocorrido.

4.7. Classe AtualizandoDados

A classe AtualizandoDados cria o método atualizar, que recebe como parâmetros a escolha do usuário, o valor a ser atualizado e o novo valor, todos recebidos do usuário. Primeiramente, é criada uma variável do tipo string com o valor de uma string vazia que servirá para ser reatribuída no bloco switch que vem a seguir. Em seguida, é criado o bloco switch que verifica a escolha do usuário e retorna o resultado baseado na escolha do método de atualização do método atualizarProduto da classe Main. De acordo com cada escolha é atribuído um valor diferente na variável criada anteriormente, esse valor contém a instrução SQL necessária para fazer a atualização do valor antigo pelo valor novo. Também é criada a opção default que exibe uma mensagem de alerta ao usuário caso ele escolha uma opção inválida. Logo após, é criada a variável do tipo string que contém o URL de conexão e o objeto do tipo Statement que é inicializado com o valor null. Depois disso é aberto um bloco try, que tenta fazer a conexão com o banco de dados, utilizando a string com a URL, o usuário e a senha do utilizador do banco de dados. Dentro do bloco try, utiliza-se a condicional if para verificar se a conexão retorna um valor nulo, caso retorne, é apresentada uma mensagem que indica que a conexão falhou. Caso contrário, será exibida uma mensagem que indica que os dados serão atualizados na tabela e é utilizado o método executeUpdate do objeto statement, passando como

parâmetro a string que contém as informações a serem atualizadas, o objeto statement e a conexão são encerradas logo em seguida. Já no bloco catch, verifica-se se houve algum erro do tipo SQLException, caso ocorra algum erro deste tipo será apresentada uma mensagem fornecendo informações sobre o erro ocorrido.

4.8. Classe ExcluindoDados

A classe ExcluindoDados cria o método excluir, que recebe como parâmetros a escolha do usuário, e o parâmetro do item a ser excluído da tabela. Primeiramente, é criada uma variável do tipo string com o valor de uma string vazia que servirá para ser reatribuída no bloco switch que vem a seguir. Em seguida, é criado o bloco switch que verifica a escolha do usuário e retorna o resultado baseado na escolha do método de exclusão do método excluirProduto da classe Main. De acordo com cada escolha é atribuído um valor diferente na variável criada anteriormente, esse valor contém a instrução SQL necessária para fazer a exclusão do produto da tabela. Também é criada a opção default que exibe uma mensagem de alerta ao usuário caso ele escolha uma opção inválida. Logo após, é criada a variável do tipo string que contém o URL de conexão e o objeto do tipo Statement que é inicializado com o valor null. Depois disso é aberto um bloco try, que tenta fazer a conexão com o banco de dados, utilizando a string com a URL, o usuário e a senha do utilizador do banco de dados. Dentro do bloco try, utiliza-se a condicional if para verificar se a conexão retorna um valor nulo, caso retorne, é apresentada uma mensagem que indica que a conexão falhou. Caso contrário, será exibida uma mensagem que indica que os dados serão excluídos da tabela e é utilizado o método executeUpdate do objeto statement, passando como parâmetro a string que contém as informações a serem excluídas, o objeto statement e a conexão são encerradas logo em seguida. Já no bloco catch, verifica-se se houve algum erro do tipo SQLException, caso ocorra algum erro deste tipo será apresentada uma mensagem fornecendo informações sobre o erro ocorrido.

5. Conclusão

O sistema desenvolvido é um sistema básico de gerenciamento de um banco de dados, sem uma interface gráfica ou métodos avançados de consulta, mas serve bem ao propósito de facilitar o gerenciamento de pequenos negócios que visam obter um melhor controle sobre seus produtos e preços.