Desenvolvimento de Sistemas de Logística: Abordagens Java e SQL

Gabriela P. Campos, Matheus M. P. Falcao

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

gabi.peeh@outlook.com, mpfalcao30@gmail.com

Abstract. *In this article, we present the development of a logistics system in Java, using the PostgreSQL database. The project's objective is to create an efficient solution for the registration and manipulation of inventory products in a grocery store. The system's application aims to optimize stock processes, product management, and improve operational efficiency. We will be addressing the main stages of development, highlighting the implemented functionalities and the benefits obtained from the application of the logistics system.*

Resumo. *Neste artigo, apresentamos o desenvolvimento de um sistema de logística em Java, utilizando o banco de dados PostgreSQL. O objetivo do projeto é criar uma solução eficiente para o cadastro e manipulação de produtos de estoque em um hortifrúti. 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 de logística.*

1. Introdução ao Sistema

A nomenclatura "FruitSync" foi escolhida levando em consideração a sua relevância e adequação ao contexto de um hortifrúti, onde o sistema será utilizado e aplicado.

2. Objetivos do Sistema

O FruitSync foi projetado para oferecer uma solução eficiente no cadastro e manipulação de produtos de estoque em um hortifrúti. Com a aplicação do FruitSync, é possível otimizar os processos de estoque, simplificar o gerenciamento de produtos e aprimorar a eficiência operacional.

3. Requisitos do Sistema

O sistema incorpora as funcionalidades de CRUD, que englobam a criação, consulta, atualização e exclusão de dados, colunas e linhas do banco de dados. Além disso, inclui uma validação de entrada de nomes, utilizando uma lista pré-selecionada e autorizada de nomes permitidos.

4. Casos de Utilização

O banco de dados do sistema FruitSync apresenta diversos casos de uso, visando atender às necessidades do gerenciamento de produtos de estoque em um hortifrúti. Alguns exemplos de possíveis casos de uso incluem cadastro de produtos, consulta de produtos, atualização de estoque, histórico de movimentações, entre outras possíveis utilizações.

5. Implementações

O sistema contém diversas funções e opções feitas para facilitar o cadastramento e logística de itens, assim como permitindo a conexão com o banco de dados respectivo. A seguir está a explicação breve de todo o código;

5.1. Importe

O código utiliza a API de JDBC para importar classes e interfaces java.sql, que são utilizadas para conectar e manipular o banco de dados existente no PostgreSQL, já o java.util.scanner é utilizada para a leitura de dados do usuário, e java.util.ArrayList e java.util.List são utilizados para criação de lista dinâmicas no código.

5.2. Métodos de conexão base

Table 1. Listagem dos métodos de conexão.

conectarDrive()	Faz a conexão do driver JDBC para o postgreSQL no projeto, necessária antes de estabelecer a conexão com o banco de dados.
conectarBD()	Faz a conexão com o banco de dados postgreSQL usando a URL da conexão, nome de usuário e senha fornecidos.

5.3. Funções de tabela

Table 2. Listagem dos métodos de tabela.

criarTabela**()**	Pede ao usuário o nome da nova tabela, verificando se já existe uma tabela com o mesmo nome inserido. Se a tabela for autorizada, é criada uma lista de strings chamada “colunas” , com as colunas “codigo” para int, “nome” para string e “estoque” para int. Então, ele cria a tabela vazia no banco de dados com as colunas acima e o nome declarado, verificando qualquer erro no processo de criação.

inserirDados()	Pergunta ao usuário em qual tabela deseja inserir os dados, verificando a existência da tabela com o nome inserido. Se for confirmada a existência, é pedido os 3 dados obrigatórios para o usuário: o “codigo” do item que precisa ter 5 dígitos numéricos, o “nome” e a quantidade de “estoque” do item. Os dados são únicos e imutáveis nessa função, não sendo permitido a inserção repetida de um dado do mesmo item. Então ocorre a inserção dos dados dentro da tabela do banco de dados, verificando qualquer erro no processo de inserção.
atualizarTabela()	Pergunta ao usuário em qual tabela deseja inserir os dados, verificando a existência da tabela com o nome inserido. É impresso a lista dos códigos dos itens existentes na tabela para a comodidade do usuário. Se for confirmada a existência do item, é pedido o “codigo” do item que se deseja alterar os dados, verificando a existência de tal item dentro da tabela informada previamente. O usuário é dado 3 opções para alteração dos 3 dados do item com o codigo informado, “codigo”, “nome” e “estoque”, assim como anteriormente, somente autorizando códigos com 5 dígitos numéricos. Se os dados forem digitados corretamente, o codigo faz o update na linha do item com os novos dados, verificando qualquer erro no processo.
removerItem()	Pergunta ao usuário em qual tabela deseja deletar o item, verificando a existência da tabela com o nome inserido. É impresso a lista dos códigos dos itens existentes na tabela para a comodidade do usuário. Se for confirmada a existência do item, será pedida uma confirmação da deleção para o usuário, se escolhido “não” o usuário retornara ao menu de “editar tabela”, se escolhido “sim”, o item será deletado da tabela. Se o codigo rodar corretamente, o codigo faz o update na tabela removendo o item, verificando qualquer erro no processo.

removerTabela()	Pergunta ao usuário qual tabela deseja deletar, verificando a existência da tabela com o nome inserido. Se for confirmada a existência da tabela, será pedida uma confirmação da deleção para o usuário, se escolhido “não” o usuário retornara ao menu de “editar tabela”, se escolhido “sim”, a tabela será deletada. Se o codigo rodar corretamente, o codigo faz o update no banco de dados removendo a tabela do sistema, verificando qualquer erro no processo.
imprimirTabela()	Pergunta ao usuário qual tabela deseja deletar, verificando a existência da tabela com o nome inserido. Se for confirmada a existência da tabela, é verificado se a tabela está vazia ou preenchida. Se a tabela estiver preenchida, com um ou mais itens, eles são impressos no terminal na ordem de inserção previa do usuário, verificando qualquer erro no processo.
imprimirCrescente()	Pergunta ao usuário qual tabela deseja deletar, verificando a existência da tabela com o nome inserido. Se for confirmada a existência da tabela, é verificado se a tabela está vazia ou preenchida. Se a tabela estiver preenchida, com um ou mais itens, utilizando a clausula “ASC” a tabela é impressa em ordem crescente com base no numero de estoque dos itens. verificando qualquer erro no processo.
imprimirDecrescente()	Pergunta ao usuário qual tabela deseja deletar, verificando a existência da tabela com o nome inserido. Se for confirmada a existência da tabela, é verificado se a tabela está vazia ou preenchida. Se a tabela estiver preenchida, com um ou mais itens, utilizando a clausula “DESC” a tabela é impressa em ordem decrescente com base no número de estoque dos itens. verificando qualquer erro no processo.
listarTabelas()	Obtém os nomes das tabelas existentes no banco de dados, imprimindo-as por ordem de criação do usuário.

O método main executa todos os métodos, provendo o usuário com uma lista de opções dependendo de suas necessidades. Essa lista tem as opções:

Criar tabela, onde chama listarTabelas() para conveniência do usuário, e provem as opções criarTabela() e inserirDados(), assim como uma opção de retorno para o menu principal.

Consultar tabela, onde chama listarTabelas() para conveniência do usuário, e provem as opções imprimirTabela(), imprimirCrescente() e imprimirDecrescente(), assim como uma opção de retorno para o menu principal.

Editar tabela, onde chama listarTabelas() para conveniência do usuário, e provem as opções atualizarTabela(), removerItem e removerTabela(), assim como uma opção de retorno para o menu principal.

O código confirma todas as opções, verificando se o input é valido, e encerrando conectarBD e conectarDrive se o usuário decidir sair do programa, efetivamente fechando todas as conexões e terminando todos os processos.

5.4. Banco de dados

O Banco de Dados permite o input do “codigo”, “nome” e “estoque” listados em colunas, utilizando o codigo como identificador do item, e determinando sua nomenclatura e quantidade de estoque, provendo uma maneira rápida e eficaz para controle de itens e estoque para a empresa.

6. Conclusão

Após concluir esse trabalho, adquirimos conhecimentos em diversas aplicações e métodos utilizados no PostgreSQL e NetBeans, além de aprofundarmos nosso domínio em Java e programação orientada a objetos.

Com o objetivo de facilitar a gestão de itens e estoques para empresas, o nosso bando de dados faz um bom trabalho e proporciona o usuário com diversas opções, possibilitando um maior controle sobre as adições e mudanças no bando de dados.

Estamos extremamente satisfeitos com os resultados alcançados e nos sentimos orgulhosos do nosso trabalho. No entanto, para projetos futuros, caso tenhamos a oportunidade, gostaríamos de implementar uma interface interativa, além de adicionar recursos extras, como uma tela de carregamento e uma tela de autenticação de usuários, visando proporcionar maior comodidade aos usuários.

Em última análise, estamos felizes com o resultado alcançado e somos gratos pela oportunidade de realizar um trabalho tão interessante e desafiador como este..