UNIVERSIDADE SALVADOR

ANÁLISE E DESENVOLVIMENTO E SISTEMAS

Bruno do Nascimento dos Santos – 12723210249

Elane Cristina da Fonseca Souza – 1272223165

Gabriel do Carmo Adriano Maia - 12722133049

Luís Igor Nobre Dos Santos – 1272228572

Victor Matheus dos Santos Pimenta – 12722133074

RELATÓRIO A3

Salvador

2023

Introdução

O sistema foi desenvolvido com o objetivo de fornecer uma solução completa e integrada para empresas que buscam otimizar seus negócios. Nosso objetivo é um gerenciamento eficaz de estoque e vendas, implementando a funcionalidade CRUD para clientes e estoque, permitindo registrar e manter informações importantes, bem como processar pedidos de compra com eficiência.

Os relatórios estatísticos são uma parte importante deste sistema, fornecendo informações valiosas sobre a tomada de decisões estratégicas. Analisar seus produtos mais vendidos, analisar o histórico de compras por cliente, analisar o uso médio, identificar produtos que estão em falta etc. são algumas das ferramentas que podem ajudá-lo a entender a saúde do seu negócio.

Utilizamos a linguagem Java, com a framework Springboot e na solução implementamos uma API para captar as vendas da loja.

O banco de dados utilizado foi o relacional PostgreeSQL.

Na criação do banco de dados usamos Migrations, que é um recurso para gerenciar as mudanças no BD, permitindo criar, alterar ou excluir tabelas ou colunas.

Implementação das funcionalidades:

Gerenciamento de Cliente:

CRUD Cliente: Implementar métodos para criar, ler, atualizar e deletar informações de clientes.

Gerenciamento de Vendas**:**

CRUD Estoque: Funções para gerenciar o estoque, incluindo adicionar, atualizar e remover produtos.

Receber Pedido de Compra: Processar pedidos de compra, atualizando o estoque e registrando a transação.

Geração de Relatórios Estatísticos:

Relatório de Produtos Mais Vendidos: Analisar as vendas para identificar e apresentar os produtos mais vendidos.

Relatório de Produto por Cliente: Mostrar os produtos comprados por cliente.

Relatório de Consumo Médio do Cliente: Calcular e exibir o consumo médio de produtos por cliente.

Relatório de Produto com Baixo Estoque: Identificar e listar os produtos com estoque abaixo de um limite definido.

Arquitetura Geral:

A aplicação pode seguir uma estrutura de camadas:

Lógica de Negócio: Responsável por processar as operações de gerenciamento de clientes, vendas e geração de relatórios.

Acesso a Dados: Realiza a comunicação com o banco de dados para persistência e recuperação de informações.

O arquivo do projeto se encontra no repositório, no formato .zip com o nome "Teste1.zip".

Banco de Dados Relacional (PostgreSQL):

Tabelas: Criar tabelas para Clientes, Produtos, Vendas etc.

Relacionamentos: Estabelecer relacionamentos entre tabelas conforme necessário.

Consultas SQL: Desenvolver consultas para atender às necessidades dos relatórios e funcionalidades.

Conexão:

spring.jpa.database=POSTGRESQL

spring.datasource.platform=postgres

spring.datasource.url=jdbc:postgresql://localhost:5432/produto

spring.datasource.username=postgres

spring.datasource.password=1234

spring,jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true

Rotas API:

Nossa classe de vendas está responsável pelos pedidos e nossa classe de estoque pelos produtos. Em todos os métodos DELETE é necessário passar o id do produto na URL do postman assim como no GET RELATORIOPRODUTOPORCLIENTE.

Segue rotas:

Cliente:

Post - http://localhost:8080/cliente

Get - http://localhost:8080/cliente

Put - http://localhost:8080/cliente

Delete - http://localhost:8080/cliente/

Estoque:

Post - http://localhost:8080/estoque

Get - http://localhost:8080/estoque

Put - http://localhost:8080/estoque

Delete - http://localhost:8080/estoque/

Get:(relatorioProdutoMenorEstoque)-http://localhost:8080/estoque/produto-menor-quantidade

Vendas:

Get - http://localhost:8080/vendas

Post - http://localhost:8080/vendas

Get (RelatorioProdutoMaisVendido) - http://localhost:8080/vendas/produto-mais-vendido

Get (RelatorioProdutoPorCliente) - http://localhost:8080/vendas/produto-por-cliente/

Get (RelatorioConsumoMedioDoCliente) - http://localhost:8080/vendas/media-consumo-por-cliente

Migrations:

(Depois de criar o banco de dados através do migrations, o arquivo de inicialização foi perdido, como não houve alteração em nenhuma funcionalidade e não teve impacto negativo no projeto, além de termos certas dificuldades em reimplementá-lo optamos por não o refazer).

No banco de dados o migrations é um conjunto de scripts e/ou arquivos que ajudam a gerenciar e controlar as alterações na estrutura do banco de dados ao longo do tempo, de uma forma controlada e versionada.

Os scripts são utilizados para realizar mudanças na estrutura do banco de dados, como criar tabelas, modificar colunas existentes, adicionar índices, entre outras alterações necessárias para evoluir o esquema do banco.

Script de Inicialização do Projeto:

Passo 1: Obter o código-fonte da aplicação:

Se você ainda não tem o código-fonte da aplicação, certifique-se de tê-lo disponível. Pode ser um arquivo ZIP ou um repositório Git.

Passo 2: Descompactar o arquivo ZIP (se necessário) ou clonar o repositório Git.

Passo 3: Importar o projeto no IDE (por exemplo, IntelliJ IDEA ou Eclipse)

Abra sua IDE e importe o projeto. Se você estiver usando o IntelliJ IDEA, escolha a opção "Open" e selecione o diretório do projeto. Se estiver usando o Eclipse, escolha a opção "Import" e selecione "Existing Maven Projects".

Passo 4: Configurar o ambiente de desenvolvimento

Certifique-se de que as versões do Java e do Maven utilizadas no projeto estão instaladas na máquina. Isso pode ser feito configurando as variáveis de ambiente JAVA_HOME e M2_HOME (ou MAVEN_HOME). Certifique-se também de que o Maven está no seu PATH.

Passo 5: Configurar as propriedades da aplicação (se necessário)

Edite o arquivo application.properties ou application.yml para configurar qualquer propriedade específica da máquina, como configurações de banco de dados, portas, etc. Este arquivo geralmente está localizado no diretório src/main/resources.

Passo 6: Executar a aplicação

Na sua IDE, localize a classe principal (geralmente uma classe com o método main). Essa classe geralmente está anotada com @SpringBootApplication. Execute essa classe como uma aplicação Java.

Passo 7: Verificar se a aplicação está em execução

Abra um navegador e acesse http://localhost:8080 (ou a porta que foi configurada na aplicação). Se tudo estiver configurado corretamente, você verá a aplicação em execução.

Lembre-se de que, em um ambiente de produção, você pode precisar ajustar a configuração da aplicação, como as propriedades do banco de dados, configurações de segurança etc.

Considerações Finais:

Testes: Desenvolver testes unitários e de integração para garantir o funcionamento correto das funcionalidades.

Documentação: Criar documentação detalhada de cada componente e sua interação.

Este é um esboço geral de como um projeto poderia ser estruturado para atender aos requisitos fornecidos. Cada componente exigirá implementação detalhada e pode ser dividido em várias partes para melhorar a modularidade e a escalabilidade do sistema.

Utilizamos as seguintes versões:

Spring Boot versão 3.2, Java Versão 21

O relatório de consumo médio por cliente retorna o valor total de cada venda pela quantidade de vendas feitas.