Projeto: Wolsen Venda de Produtos

Descrição do Projeto

A aplicação será uma API de gerenciamento de vendas focada na retenção de impostos sobre as vendas realizadas. A API permitirá o cadastro de produtos, registro e consulta de vendas, e cálculo de impostos retidos. O objetivo é proporcionar uma gestão eficiente das vendas e impostos, garantindo a conformidade fiscal e facilitando a análise de desempenho de vendas.

A aplicação será desenvolvida utilizando Spring Boot e Gradle, com persistência de dados em um banco de dados MySQL. A arquitetura clean será adotada para garantir a separação de responsabilidades, facilitando a manutenção e a extensibilidade do sistema.

Estrutura do Projeto (Arquitetura Clean)

Camadas:

1. **Entities**: Contém os modelos de negócios.
2. **Use Cases**: Contém a lógica específica de cada caso de uso.
3. **Interface Adapters**: Contém os adaptadores para a interface do usuário e para a interface de dados.
4. **Frameworks & Drivers**: Contém implementações específicas de frameworks (ex.: Spring Boot, JPA).

Principais Pacotes do Projeto:

- **br.com.wolsen.vendas.domain.entities:** Contém as entidades de domínio.
- **br.com.wolsen.vendas.domain.usecases:** Contém os casos de uso.
- **br.com.wolsen.vendas.application.services:** Contém os serviços utilizados pelos casos de uso.
- **br.com.wolsen.vendas.infra.controllers:** Contém os controladores.
- **br.com.wolsen.vendas.infra.repositories:** Contém os adaptadores de repositório.

Histórias de Usuário

História de Usuário 1: Cadastro de Produtos

Título: Cadastro de novos produtos

Descrição: Como um administrador de vendas, eu quero cadastrar novos produtos no sistema para que possam ser vendidos.

Critérios de Aceitação:

1. Deve haver um endpoint POST /produtos que aceite dados do produto (nome, descrição, preço, código fiscal).
2. As informações devem ser validadas antes de serem salvas no banco de dados.
3. Um produto deve ter um ID único gerado pelo sistema.
4. O produto deve ser salvo na base de dados MySQL.
5. Deve retornar uma mensagem de sucesso com o ID do produto.

Tarefas Técnicas:

1. Criar a entidade Produto em br.com.wolsen.vendas.domain.entities.
2. Implementar o caso de uso CadastrarProduto em br.com.wolsen.vendas.domain.usecases.
3. Implementar o controlador ProdutoController em br.com.wolsen.vendas.infra.controllers para o endpoint.
4. Configurar o repositório JPA ProdutoRepository em br.com.wolsen.vendas.infra.repositories
5. Adicionar validações com javax.validation.constraints.
6. Escrever testes unitários para o caso de uso e o controlador.
7. Documentar o endpoint com Swagger.

Tela de Cadastro de Produtos

Componentes:

- **Formulário de Cadastro de Produtos**
    - **Campos:**
        - Nome
        - Descrição
        - Preço
        - Código Fiscal
    - **Botão:**
        - "Cadastrar Produto"

História de Usuário 2: Consulta de Produtos

Título: Consulta de produtos cadastrados

Descrição: Como um administrador de vendas, eu quero consultar produtos cadastrados no sistema para verificar suas informações.

Critérios de Aceitação:

1. Deve haver um endpoint GET /produtos que retorne uma lista de todos os produtos cadastrados.
2. Deve ser possível filtrar produtos por nome e código fiscal.
3. A lista de produtos deve ser retornada em formato JSON.

Tarefas Técnicas:

1. Implementar o caso de uso ConsultarProdutos em br.com.wolsen.vendas.domain.usecases.
2. Criar métodos no controlador ProdutoController para o endpoint GET.
3. Utilizar Streams em Java para aplicar filtros.
4. Escrever testes unitários para o caso de uso e o controlador.
5. Documentar o endpoint com Swagger.

Tela de Consulta de Produtos

Componentes:

- **Lista de Produtos**
  - **Tabela com colunas:**
    - Nome
    - Descrição
    - Preço
    - Código Fiscal
- **Filtros de Pesquisa**
  - **Campos:**
    - Nome
    - Código Fiscal
  - **Botão:**
    - "Pesquisar"

História de Usuário 3: Registro de Vendas

Título: Registro de novas vendas

Descrição: Como um vendedor, eu quero registrar vendas de produtos para controlar as transações e calcular a retenção de impostos.

Critérios de Aceitação:

1. Deve haver um endpoint POST /vendas que aceite dados da venda (ID do produto, quantidade, data da venda).

2. As vendas devem ser validadas e associadas ao produto correspondente.
3. O valor total da venda e o imposto retido devem ser calculados e armazenados.
4. A venda deve ser salva na base de dados MySQL.
5. Deve retornar uma mensagem de sucesso com o ID da venda.

Tarefas Técnicas:

1. Criar a entidade Venda em br.com.wolsen.vendas.domain.entities.
2. Implementar o caso de uso RegistrarVenda em br.com.wolsen.vendas.domain.usecases.
3. Implementar o controlador VendaController em br.com.wolsen.vendas.infra.controllers para o endpoint.
4. Configurar o repositório JPA VendaRepository em br.com.wolsen.vendas.infra.repositories.
5. Implementar a lógica de cálculo de impostos no caso de uso.
6. Escrever testes unitários para o caso de uso e o controlador.
7. Documentar o endpoint com Swagger.

Tela de Registro de Vendas

Componentes:

- **Formulário de Registro de Vendas**
    - **Campos:**
        - ID do Produto
        - Quantidade
        - Data da Venda
    - **Botão:**
        - "Registrar Venda"

História de Usuário 4: Consulta de Vendas

Título: Consulta de vendas registradas

Descrição: Como um administrador de vendas, eu quero consultar as vendas registradas no sistema para analisar o desempenho de vendas e os impostos retidos.

Critérios de Aceitação:

1. Deve haver um endpoint GET /vendas que retorne uma lista de todas as vendas registradas.
2. Deve ser possível filtrar vendas por produto e data.
3. A lista de vendas deve ser retornada em formato JSON.

Tarefas Técnicas:

1. Implementar o caso de uso ConsultarVendas em br.com.wolsen.vendas.domain.usecases.
2. Criar métodos no controlador VendaController para o endpoint GET.
3. Utilizar Streams em Java para aplicar filtros.
4. Escrever testes unitários para o caso de uso e o controlador.
5. Documentar o endpoint com Swagger.

Tela de Consulta de Vendas

Componentes:

- **Lista de Vendas**
  - **Tabela com colunas:**
    - ID do Produto
    - Quantidade
    - Data da Venda
    - Valor Total
- **Filtros de Pesquisa**
  - **Campos:**
    - ID do Produto
    - Data da Venda
  - **Botão:**
    - "Pesquisar"

História de Usuário 5: Implantação e Documentação

Título: Documentação e implantação da aplicação

Descrição: Como um desenvolvedor, eu quero documentar a API e implantar a aplicação em um contêiner Docker para facilitar a integração e o uso.

Critérios de Aceitação:

1. A API deve ser documentada utilizando Swagger.
2. Deve haver um arquivo **Dockerfile** para construir a imagem da aplicação.
3. Deve haver um arquivo **docker-compose.yml** para gerenciar os contêineres.
4. A aplicação deve ser executada corretamente dentro do contêiner Docker.

Tarefas Técnicas:

1. Configurar Swagger para documentar os endpoints.

2. Criar um **Dockerfile** para a aplicação Spring Boot.
3. Criar um **docker-compose.yml** para orquestrar a aplicação e outros serviços necessários (ex.: banco de dados).
4. Testar a aplicação dentro do contêiner Docker.
5. Atualizar a documentação do projeto para incluir instruções de implantação.

Exemplo de build.gradle

```
plugins {
    id 'org.springframework.boot' version '3.1.0'
    id 'io.spring.dependency-management' version '1.1.0'
    id 'java'
}

group = 'com.seuprojeto'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-
jpa'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'io.springfox:springfox-boot-starter:3.0.0'
    runtimeOnly 'mysql:mysql-connector-java'
    testImplementation 'org.springframework.boot:spring-boot-starter-
test'
}

test {
    useJUnitPlatform()
}
```