Documentação Técnica: AppRestauranteDJ

Versão 0.01 – 20240421-h11m01

William Silva
Analista Desenvolvedor

1. Descrição do Projeto

AppRestauranteDJ é uma aplicação de demonstração desenvolvida com Spring Boot. Ele tem como objetivo fornecer uma estrutura básica para um sistema de gerenciamento de restaurantes, utilizando tecnologias modernas como Spring Boot, JPA, Hibernate, PostgreSQL e outros componentes. O projeto é configurado para ambientes de desenvolvimento e produção, com foco em escalabilidade e modularidade.

2. Configuração do Projeto

2.1 Maven POM

O arquivo POM do projeto usa o Spring Boot como projeto pai, configurado para a versão 2.6.10. As dependências básicas incluem:

- Spring Boot Starter para JPA e Web
- Hibernate Core para gerenciamento de banco de dados
- PostgreSQL como banco de dados
- Spring Cloud Gateway para integração com outros serviços
- JUnit e Mockito para testes unitários

Além disso, o arquivo POM possui configurações de perfis para diferentes ambientes (desenvolvimento e produção).

2.2 Configuração do Build

O projeto usa o Spring Boot Maven Plugin para simplificar o processo de build e execução. Com esse plugin, é possível empacotar a aplicação como um arquivo JAR auto-executável.

2.3 Perfis de Configuração

Existem perfis de configuração para ambientes de desenvolvimento e produção, permitindo ajustes específicos para cada ambiente, como URLs de banco de dados, configurações de segurança, e outras propriedades do Spring Boot.

3. Funcionalidades Principais

AppRestauranteDJ oferece as seguintes funcionalidades básicas:

- **Gerenciamento de Pedidos**: Criação, edição e visualização de pedidos de clientes.
- **Gestão de Produtos**: Adição, remoção e edição de itens do cardápio.
- **Relatórios**: Geração de relatórios para acompanhamento de vendas e desempenho do restaurante.

4. Instruções de Uso

4.1 Para Desenvolvedores

Para configurar o ambiente de desenvolvimento, siga estes passos:

1. **Requisitos**: Certifique-se de ter o JDK 11 ou mais recente, Maven, e um ambiente de desenvolvimento integrado (IDE) como IntelliJ ou Eclipse.
2. **Clone o Repositório**: Use Git para clonar o projeto para o seu ambiente local.
3. **Build do Projeto**: No diretório do projeto, execute `mvn clean install` para baixar as dependências e compilar o código-fonte.
4. **Executar o Projeto**: Use `mvn spring-boot:run` para iniciar a aplicação localmente. O aplicativo deve estar disponível em `http://localhost:8080`.

4.2 Para Usuários Finais

Como usuário final, você pode interagir com a aplicação para realizar as seguintes ações:

1. **Fazer Pedidos**: Acesse a interface do usuário para criar um novo pedido. Insira os itens desejados e finalize o pedido.
2. **Visualizar Cardápio**: Navegue pelo cardápio para ver as opções de comida e bebida disponíveis.
3. **Ver Relatórios**: Para administradores, é possível gerar relatórios para analisar o desempenho do restaurante.

5. Problemas Conhecidos e Soluções

- **Erro na Conexão com Banco de Dados**: Certifique-se de que o PostgreSQL está configurado corretamente e em execução. Verifique a configuração do banco de dados no arquivo `application.properties`.
- **Dependências Ausentes**: Se houver problemas de dependências, execute `mvn clean install` para baixar novamente as dependências.
- **Erros durante Testes**: Verifique se o escopo das dependências para teste está configurado corretamente no arquivo POM.

6. Script para Criação e Geração da Base de Dados no PostgreSQL

sqlCopy code

```
CREATE
CREATE TABLE
PRIMARY VARCHAR 100 NOT NULL
DECIMAL 10 2 NOT NULL VARCHAR 50 NOT NULL
CREATE TABLE PRIMARY VARCHAR 50
NOT NULL VARCHAR 100 NOT NULL CREATE
TABLE PRIMARY INT NOT NULL
TIMESTAMP DEFAULT CURRENT_TIMESTAMP DECIMAL 10 2 NOT NULL
VARCHAR 50 NOT NULL FOREIGN REFERENCES
CREATE TABLE PRIMARY
INT NOT NULL INT NOT NULL INT NOT NULL
DECIMAL 10 2 NOT NULL FOREIGN REFERENCES
FOREIGN REFERENCES
```

7. Script para Criação do Banco de Dados no H2

sqlCopy code

```
CREATE NOT EXISTS

CREATE TABLE INT PRIMARY
VARCHAR 100 NOT NULL DECIMAL 10 2 NOT NULL
VARCHAR 50 NOT NULL CREATE TABLE
INT PRIMARY VARCHAR 50 NOT NULL
VARCHAR 100 NOT NULL CREATE TABLE
INT PRIMARY INT NOT NULL TIMESTAMP
DEFAULT CURRENT_TIMESTAMP DECIMAL 10 2 NOT NULL
VARCHAR 50 NOT NULL FOREIGN REFERENCES
CREATE TABLE INT
PRIMARY INT NOT NULL INT NOT NULL
INT NOT NULL DECIMAL 10 2 NOT NULL FOREIGN
REFERENCES FOREIGN REFERENCES
```

8. Dados Usados nos Testes

8.1 Dados para Produtos

jsonCopy code

```
    "nome" "Hambúrguer" "preco" 10.50 "categoria" "Prato Principal"       "nome" "Coca-Cola" "preco" 5.00 "categoria" "Bebida"       "nome" "Batata Frita" "preco" 7.00 "categoria" "Entrada" "nome" "Sorvete" "preco" 8.50 "categoria" "Sobremesa"
```

8.2 Dados para Usuários

jsonCopy code

```
    "username" "joao123" "password" "senha123"       "username" "maria456" "password" "senha456"
```

8.3 Dados para Pedidos

jsonCopy code

```
    "usuario_id" 1 "valor_total" 25.50 "status" "Pendente"       "usuario_id" 2 "valor_total" 13.50 "status" "Entregue"
```

8.4 Dados para Itens do Pedido

jsonCopy code

```
    "pedido_id" 1 "produto_id" 1 "quantidade" 2 "preco_unitario" 10.50       "pedido_id" 1 "produto_id" 2 "quantidade" 1 "preco_unitario" 5.00       "pedido_id" 2 "produto_id" 3 "quantidade" 1 "preco_unitario" 7.00       "pedido_id" 2 "produto_id" 4 "quantidade" 1 "preco_unitario" 8.50
```

Estrutura do Projeto

1. **Pacote Principal**: `com.kipho.AppRestauranteDJ`
    - Contém a classe principal do aplicativo Spring Boot (`DemoApplication.java`), onde a execução começa.
    - Inclui a configuração do Swagger (`SwaggerConfig.java`), que define a documentação da API.
2. **Pacotes Secundários**:
    - **Business**: Contém classes relacionadas à lógica de negócios, como `ItemPedido`.
    - **Configuration**: Define configurações personalizadas para o Spring Boot. Por exemplo, `ApiGatewayConfiguration` e `RouteLocatorBuilder` podem conter configurações do Gateway.
    - **Controllers**: Define os controladores REST para as entidades principais do aplicativo, como `PedidoController` e `ProdutoController`.
    - **Models**: Define as entidades do aplicativo, como `Pedido` e `Produto`.

- **Repository**: Define os repositórios para acessar o banco de dados, como `PedidoRepository` e `ProdutoRepository`.
- **Services**: Contém a lógica de serviço do aplicativo, como `PedidoService` e `ProdutoService`.

3. **Recursos**:
   - **Templates**: Contém arquivos de configuração e outros recursos, como `application.yml` (configurações do Spring Boot) e `logback.xml` (configuração do log).
   - **Static**: Provavelmente contém recursos estáticos como CSS ou JavaScript.
4. **Testes**: Localizados em `src/test/java`, incluem testes para diferentes componentes do aplicativo, como `DemoApplicationTests`, `PedidoControllerTest`, e outros testes de unidade e integração.
5. **Configurações Gerais**:
   - **POM**: O arquivo `pom.xml` é a configuração do Maven para o projeto. Ele define dependências e plugins para construção e execução.
   - **Maven Wrapper**: Os arquivos `mvnw` e `mvnw.cmd` são scripts para executar o Maven sem instalação prévia.
6. **Diretórios Adicionais**:
   - **Target**: O diretório onde o Maven gera os arquivos de build.
   - **Documentação**: O arquivo `HELP.md` pode conter informações úteis para desenvolvedores.