Documentação do Projeto de Marketplace

Visão Geral

Este projeto de marketplace é uma aplicação básica e demonstrativa, focada em funcionalidade e simplicidade, sem práticas avançadas de segurança. Algumas decisões foram tomadas para otimizar o tempo e reduzir a complexidade: por exemplo, as imagens dos produtos não são armazenadas no back-end, sendo exibidas apenas no front-end como placeholders.

A aplicação é conteinerizada com Docker, o que facilita o gerenciamento do ambiente, tanto em desenvolvimento quanto em produção. O back-end foi desenvolvido com NestJS e usa PostgreSQL como banco de dados, ambos configurados para rodar em containers orquestrados com Docker Compose.

Estrutura de Conteinerização com Docker

- Dockerfile: Define o container do back-end, incluindo a instalação de dependências, build do projeto e configuração da porta 3000 para a API.
- docker-compose.yml: Arquivo de orquestração que conecta o back-end e o banco de dados.
  - Backend: Configura a porta 3000 e variáveis básicas como JWT e credenciais do banco.
  - Database: Usa a imagem do PostgreSQL, mapeando a porta 5432 e configurando as credenciais.

Comandos Básicos

- Iniciar containers: docker-compose up --build -d
- Parar containers: docker-compose down
- Ver logs:

- Backend: docker logs marketplace_backend

- Database: docker logs marketplace_database

- Acessar o banco: docker exec -it marketplace_database psql -U postgres -d marketplace

Planejamento e Estrutura da API com NestJS

A API está organizada em três partes principais: Modules, Controllers e Services. Essa arquitetura facilita a expansão e a manutenção.

- Modules (Módulos): Cada módulo organiza uma funcionalidade específica do sistema. Neste projeto, temos módulos para usuários, anúncios e transações. Cada módulo é independente, agrupando tudo o que cada área precisa para funcionar.
- Controllers: Os Controllers recebem e tratam as requisições que chegam à API, direcionando cada uma para o lugar certo. Por exemplo, o AdController cuida das rotas de anúncios, como listar, criar, atualizar ou excluir.
- Services: Os Services contêm a lógica de negócios principal, com as operações de cada módulo, como salvar um novo anúncio ou processar uma transação. Esse isolamento facilita a manutenção, pois cada serviço lida diretamente com o que cada módulo precisa.

A organização em módulos, controllers e services torna a API modular e prática. Lembrando que esta versão é uma demonstração, então não possui configurações de segurança avançadas ou validações mais robustas.

Comunicação HTTP no Front-end com Flutter

Para o front-end, usamos o Flutter e o pacote http, que facilita a comunicação com a API. A troca de informações é direta e prática, enviando e recebendo dados para permitir ao usuário interagir com o sistema.

- Configuração das Requisições HTTP: O pacote http permite fazer as operações necessárias (como GET, POST, etc.) para logar o usuário, listar anúncios, entre outros. As requisições são feitas em JSON, o que facilita o envio e o recebimento de dados.
- Feedback para o Usuário: Dependendo da resposta da API, o app mostra uma mensagem ao usuário sobre o status da operação. Por exemplo, após criar um anúncio, aparece uma notificação de confirmação.
- Exemplo Simples: Para criar um anúncio, o app envia título, descrição e preço, e a API responde se deu certo ou se houve algum problema. Mensagens de erro ou sucesso são exibidas usando um SnackBar.

Essa estrutura usando http é funcional e atende bem ao propósito do projeto. Em uma aplicação completa, teríamos controles extras de segurança e validações mais robustas.

Endpoints Utilizados no Front-end

Autenticação e Registro

- POST /auth/login: Realiza o login do usuário. Envia credenciais e recebe um token JWT para autenticação.
- POST /auth/register: Registra um novo usuário no sistema, criando a conta com dados básicos.

Anúncios

- GET /ads/valid: Lista todos os anúncios disponíveis na tela inicial para o usuário navegar.
- GET /ads/user-ads: Mostra os anúncios criados pelo usuário logado, exibidos na área "Meus Anúncios".
- POST /ads: Cria um novo anúncio com título, descrição e preço, retornando uma confirmação de sucesso.

- PATCH /ads/:id: Atualiza os detalhes de um anúncio específico do usuário, como preço e status.
- DELETE /ads/:id: Exclui um anúncio do usuário e confirma a remoção com uma atualização da interface.

Transações

- POST /transactions: Realiza a compra de um anúncio, processando a transação.
- GET /transactions/purchased: Lista as transações que o usuário realizou, mostrando as compras finalizadas.

Essa lista cobre os principais endpoints que o app utiliza para realizar o login, criar anúncios, visualizar, atualizar, deletar e processar transações de compra.

Esta documentação cobre a estrutura da API e como o front-end se comunica com ela, focando em funcionalidade e simplicidade. Toda a aplicação foi feita para ser simples e direta, atendendo ao propósito demonstrativo de um marketplace básico.