Guia Explicativo - Preparação do Ambiente para o Projeto MinhaGaragem

# 1. Criar Estrutura de Pastas do Projeto

Organizar as pastas desde o início ajuda a manter o projeto limpo, separado por funções e fácil de entender.

Criamos três principais pastas dentro da pasta do projeto:

- apps/api → Aqui ficará o código da API (servidor com Python)

- apps/web → Aqui será desenvolvida a aplicação Web (com Next.js)

- apps/mobile → Aqui será desenvolvido o aplicativo para celular (com React Native)

- infra/docker → Pasta reservada para configurações futuras e arquivos relacionados a infraestrutura.

Essa estrutura é chamada de 'monorepo' porque todas as partes do projeto ficam dentro de uma única pasta raiz.

# 2. Instalar Docker Desktop

O Docker permite criar ambientes isolados chamados 'containers'.

Esses containers são como pequenas máquinas virtuais que rodam o código sem depender do sistema operacional do computador.

Isso garante que o sistema funcionará da mesma forma em qualquer máquina.

Instalamos o Docker Desktop para podermos criar e rodar containers da API, banco de dados, painel administrativo e serviço de arquivos.

# 3. Criar o Dockerfile da API Python

O Dockerfile é um arquivo de receita que diz ao Docker como construir a imagem do sistema da API.

Nele, usamos uma imagem base do Python, instalamos bibliotecas necessárias, copiamos os arquivos do projeto e definimos o comando para iniciar o servidor.

Isso garante que todos desenvolvedores ou servidores que rodarem a aplicação terão o mesmo ambiente de execução.

# 4. Criar o arquivo requirements.txt

O requirements.txt é o arquivo onde listamos todas as bibliotecas que o Python precisa instalar para o sistema funcionar.

Colocamos ali o FastAPI (framework da API), Strawberry (GraphQL), SQLAlchemy (acesso ao banco de dados), e outras bibliotecas úteis.

# 5. Criar o docker-compose.yml

Esse arquivo é o maestro que orquestra todos os serviços do projeto.

Nele definimos:

- A API Python (chamada 'api')

- O banco de dados PostgreSQL ('db')

- O painel gráfico para visualizar o banco, chamado pgAdmin ('pgadmin')

- Um sistema para armazenar arquivos chamado MinIO ('minio'), que simula serviços como AWS S3

O docker-compose garante que tudo seja iniciado com um único comando e que os serviços possam conversar entre si.

# 6. Subir os serviços com Docker Compose

Com o comando `docker-compose up --build`, o Docker vai:

- Criar os containers

- Rodar os serviços

- Abrir as portas para acessar o sistema localmente

Assim você já pode acessar:

- A API Python via http://localhost:8000

- O painel do banco de dados via http://localhost:5050

- O painel de arquivos via http://localhost:9001