Catalog Api

Algumas instruções  
*Nesse documento irei deixar algumas configurações que foram realizadas.*

Irei deixar algumas configurações para docker e a execução de outros scripts.

# Comunicando-se com o container do Mongo

Esses passos são para executar o container do mongo do qual irá ser acessado por uma aplicação que está rodando localmente, sem ter um container para o app.

1. Baixar e executar a imagem.
2. docker run -d -p 27017:27017 --name catalog-mongo mongo
3. Aplicação fora do container acessar o Mongo
4. Só apontar o webconfig para o mongo.

{

"DatabaseSettings": {

"ConnectionString": "mongodb://localhost:27017",

"DatabaseName": "ProductDb",

"CollectionName": "Products"

},

# Configuração do DockerFile

O dockerfile pode ser criado mediante a ferramenta de suporte do visual studio e a imagem e o container pode ser criado executando Docker.

Também é possível criar a imagem:

docker build -t apicatalogo:1.0 -f Catalog.Api/Dockerfile .

Para criar o container:

docker container run -it --rm -p 3000:80 --name apicatalogocontainer apicatalogo:1.0

Lembrando que vai ser o docker-compose que irá gerenciar o docker file.

# Configuração o Docker Compose

O docker compose pode ser criado mediante a ferramenta do visual studio.

version: '3.4'

# ~~> DEFININDO OS VOLUMES

volumes:

mongo\_data:

# ~~> DEFININDO CONTAINER DO MONGO

services:

catalogdb:

container\_name: catalogdb

restart: always

image: mongo

ports:

- "27017:27017"

volumes:

- mongo\_data:/data/db

# ~~> DEFININDO CONTAINER DA CATALOG.API

catalog.api:

container\_name: catalog.api

environment:

- ASPNETCORE\_ENVIRONMENT=Development

image: catalogapi

build:

context: .

dockerfile: Catalog.Api/Dockerfile

depends\_on:

- catalogdb

ports:

- "8000:80"

O comando cria os dois containers, mongo e api catalog.

docker-compose -f docker-compose.yml up -d