04

Pegando dados de um banco

Transcrição

Para praticar o que vimos sobre redes no Docker, vamos criar uma pequena aplicação que se conectará ao banco de dados, utilizando tudo o que vimos no vídeo anterior.

O que vamos fazer é utilizar a aplicação **alura-books**, que irá pegar os dados de um banco de dados de livros e exibi-los em uma página web. É uma aplicação feita em Node.js e o banco de dados é o MondoDB.

Pegando dados de um banco em um outro container

Então, primeiramente vamos baixar essas duas imagens, a imagem **douglasq/alura-books** na versão **cap05** e a imagem **mongo**:

```
docker pull douglasq/alura-books:cap05
docker pull mongo
```

Na imagem **douglasq/alura-books**, não há muito mistério. Ela possui o arquivo **server.js**, que carrega algumas dependências e módulos que são instalados no momento em que rodamos a imagem. Esse arquivo carrega também as configurações do banco, que diz onde o banco de dados estará *hosteado*, no caso o seu *host* será **meu-mongo**, e o *database*, com nome de **alura-books**. Então, quando formos rodar o *container* de MongoDB, seu nome deverá ser **meu-mongo**. Além disso, o arquivo realiza a conexão com o banco, configura a porta que será utilizada (**3000**) e levanta o servidor .

No **Dockerfile** da imagem, também não há mistério, é basicamente o que vimos no vídeo anterior. Por fim, temos as **rotas**, que são duas: a rota `/` , que carrega os livros e os exibe na página, e a rota `/seed` , que salva os livros no banco de dados.

> *Caso queira, você pode baixar [aqui (https://s3.amazonaws.com/caelum-online-public/646-docker/05/projetos/alura-docker-cap05.zip)](https://s3.amazonaws.com/caelum-online-public/646-docker/05/projetos/alura-docker-cap05.zip) o código da versão **cap05** da imagem **alura-books**.*

Visto isso, já podemos subir a imagem:

```
docker run -d -p 8080:3000 douglasq/alura-books:cap05
```

Ao acessar a página **http://localhost:8080/ (http://localhost:8080/)**, nenhum livro nos é exibido, pois além de não termos levantado o banco de dados, nós não salvamos nenhum dado nele. Então, vamos excluir esse *container* e subir o *container* do MongoDB, lembrando que o seu nome deve ser **meu-mongo**, e vamos colocá-lo na rede que criamos no vídeo anterior:

```
docker run -d --name meu-mongo --network minha-rede mongo
```

Com o banco de dados rodando, podemos subir a aplicação do mesmo jeito que fizemos anteriormente, mas **não podemos nos esquecer que ele deve estar na mesma rede do banco de dados**, logo vamos configurar isso também:

```
docker run --network minha-rede -d -p 8080:3000 douglasq/alura-books:cap05
```

Agora, acessamos a página **http://localhost:8080/seed/ (http://localhost:8080/seed/)** para salvar os livros no banco de dados. Após isso, acessamos a página **http://localhost:8080/ (http://localhost:8080/)** e vemos os dados livros são extraídos do banco e são exibidos na página. Para provar isso, podemos parar a execução do **meu-mongo** e atualizar a página, veremos que nenhum livro mais será exibido.

Então, esse foi um exemplo para praticar a comunicação entre *containers*, sempre lembrando que devemos colocá-los na mesma rede. Na próxima aula, veremos um jeito de orquestrar melhor diversos *containers* e automatizar esse processo de levantá-los e configurá-los, ao invés de fazer tudo na mão.