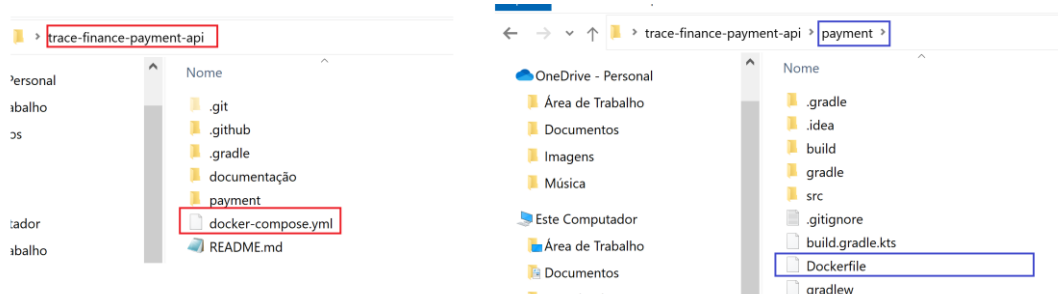




Como a aplicação está containerizada

É importante saber como está montado os contêineres da aplicação. Temos dois arquivos principais no projeto que cuidam dessa responsabilidade, são eles o “Dockerfile” e o docker-compose.yml (repare que o **docker-compose** está na **pasta raiz**, enquanto o **Dockerfile** está dentro da pasta do projeto “./payment”):



Basicamente falando, o **docker-compose** é o arquivo principal aqui. Ele é o arquivo que vai montar as imagens e subir os containers, nele tem instruções que dizem o que subir e como subir.

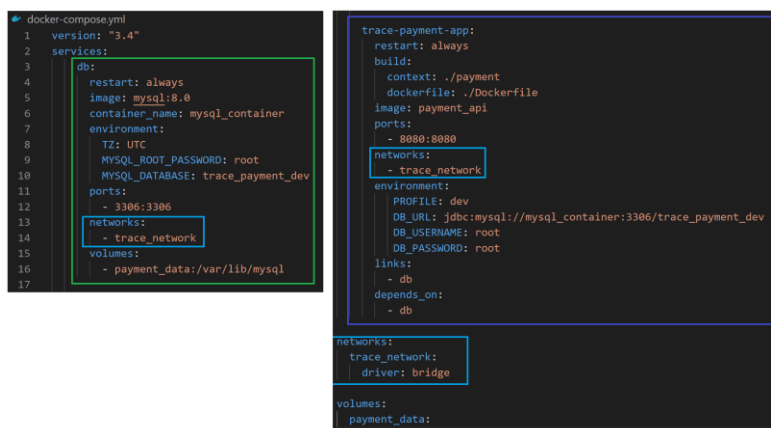
Para saber mais sobre container e docker, com mais detalhes, eu deixo aqui um link de um outro repositório meu, quando comecei a estudar o assunto: <https://github.com/LucasTrevizanbr/Devops/tree/main/Docker>

Os containers da aplicação

Pra que a aplicação funcione, ela precisa de um banco de dados, esse banco de dados vai rodar em um container isolado, no **docker-compose** eu defino coisas como: Qual imagem de **banco de dados usar**, qual suas variáveis de ambiente (*usuario, senha, timezone etc..*), a porta que esse serviço será exposto, o volume e uma “network”

A outra imagem desse compose é a própria aplicação, essa imagem é construída com base em um Dockerfile que eu mesmo configurei. Então para que o compose ache esse Dockerfile é preciso informar a propriedade de build, onde eu passo o caminho desse **Dockerfile**.

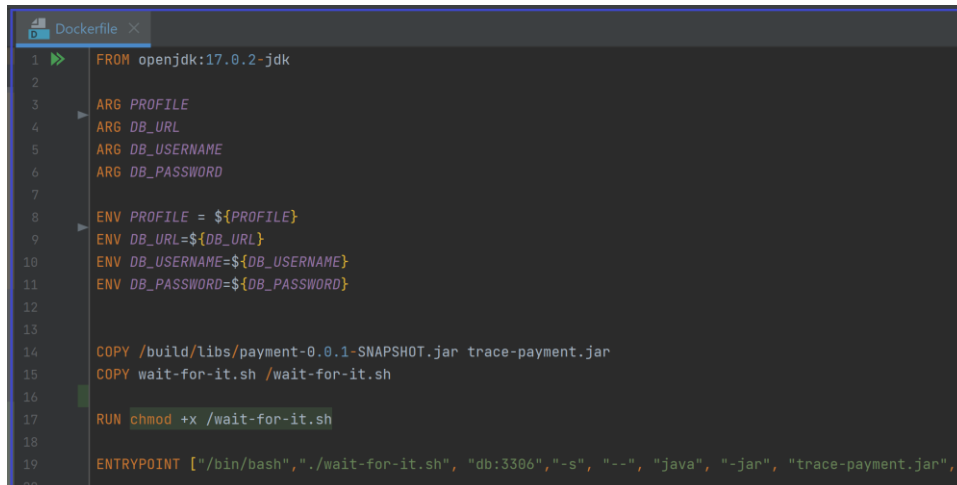
Outro ponto importante é a **network**, essa network definida faz com que os containers se conheçam, **então eu posso referencia-los pelo nome**, por exemplo na variável de ambiente do banco de dados da aplicação, onde eu passo apenas o nome do container do banco de dados e a porta que ele está exposto:



A imagem da aplicação

A imagem da aplicação é apenas uma série de instruções, usar o openjdk 17 como imagem padrão, pegar .jar da aplicação e colocar dentro do container, pegar o wait-for-it.sh e colocar dentro do container. O wait-for-it é uma peça importante aqui, ele garante que minha aplicação só vai rodar, se o meu container de banco de dados subir.

As variáveis de ambiente são definidas na imagem, mas elas só são passadas no compose, então se quiser altera-las, sabe onde ir. O Entrypoint são as instruções que serão executadas assim que o container subir:

A screenshot of a code editor window titled 'Dockerfile'. The code is written in a dark theme and includes line numbers on the left. The Dockerfile instructions are as follows:

```
1 FROM openjdk:17.0.2-jdk
2
3 ARG PROFILE
4 ARG DB_URL
5 ARG DB_USERNAME
6 ARG DB_PASSWORD
7
8 ENV PROFILE = ${PROFILE}
9 ENV DB_URL=${DB_URL}
10 ENV DB_USERNAME=${DB_USERNAME}
11 ENV DB_PASSWORD=${DB_PASSWORD}
12
13
14 COPY /build/libs/payment-0.0.1-SNAPSHOT.jar trace-payment.jar
15 COPY wait-for-it.sh /wait-for-it.sh
16
17 RUN chmod +x /wait-for-it.sh
18
19 ENTRYPOINT ["/bin/bash", "./wait-for-it.sh", "db:3306", "-s", "--", "java", "-jar", "trace-payment.jar",
```

Se quiser mexer no projeto pela ide e alterar ele, pode subir apenas um container com **“docker-compose up xxxx”**, então pode subir apenas o banco de dados. Mas lembre-se de trocar a url de conexão para “localhost”, pois o container roda de forma isolada.