Relatório docker 05

Criando a network com o comando :

```
PS C:\Users\Eduardo\Desktop\Docker\ex5> docker network create rede_edu
622d6945fd4e268f8224078b0642a3cc139e059896f268f19da55910d2a8767a
```

Inspecionando a network que acabei de criar:

```
PS C:\Users\Eduardo\Desktop\Docker\ex5> docker network inspect rede_edu
[
    {
        "Name": "rede_edu",
        "Id": "622d6945fd4e268f8224078b0642a3cc139e059896f268f19da55910d2a8767a",
        "Created": "2023-05-23T00:42:18.9171847857",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```

Em seguida baixei a imagem do PostgreSQL:

```
PS C:\Users\Eduardo\Desktop\Docker\ex5> docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
9e3ea8720c6d: Already exists
7782b3e1be4b: Pull complete
247ec4ff783a: Pull complete
f7ead6900700: Pull complete
e7afdbe9a191: Extracting [==================================================>]  8.046MB/8.046MB
3ef71fe7cece: Download complete
1459ebb56be5: Download complete
3595124f6861: Verifying Checksum
26ececf90b13: Downloading [==>                                                ]  4.307MB/92.35MB
f4809920222b: Download complete
c3f8fa066d45: Download complete
```

Usei o comando abaixo para rodar um container com a imagem Postgres que acabei de baixar configurando as credenciais:

```
PS C:\Users\Eduardo\Desktop\Docker\ex5> docker run -d `
>>     --name ex5-backend `
>>     --network rede_edu  `
>>     -e HOST=localhost `
>>     -e PORT=8080 `
>>     -e PG_HOST=localhost `
>>     -e PG_USER=postgres `
>>     -e PG_DATABASE=postgres `
>>     -e PG_PASSWORD=811944 `
>>     -e PG_PORT=5432 `
>>     -e JWTSECRET=TypescriptComCafe `
>>     -p 8080:8080 `
>>     ex5
fd5a4daa6127c78f60fc1e9ad760bdd335d4d43f8a0641d2624e19288d911716
```

Para verificar usei o comando docker inspect:

```
956ac6ef928729e4c31fdfb8d90a0b0e761987d93c298ad1ccaa7c90796ed6ec
PS C:\Users\Eduardo\Desktop\Docker\ex5> docker inspect 956ac6ef928729e4c31fdfb8d90a0b0e761987d93c298ad1ccaa7c90796ed6ec
```

Resultado:


```
"Networks": {
    "rede_edu": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": [
            "956ac6ef9287"
        ],
        "NetworkID": "622d6945fd4e268f8224078b0642a3cc139e059896f268f19da55910d2a8767a",
        "EndpointID": "913ffca6a03a1780802190fd813688e147be7c8200e12ac6307caac31c186b1b",
        "Gateway": "172.18.0.1",
        "IPAddress": "172.18.0.2",
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "MacAddress": "02:42:ac:12:00:02",
        "DriverOpts": null
    }
}
```


Conectando ao postgres:

```
PS C:\Users\Eduardo\Desktop\Docker\ex5> docker exec -it postgres psql -U postgres
psql (15.3 (Debian 15.3-1.pgdg110+1))
Type "help" for help.

postgres=#
```

Criando a tabela accounts:

```
postgres=# create table accounts(id serial primary key not null, email varchar(255), name varchar(255), password varchar(60));
CREATE TABLE
postgres=#
```

Inseri um usuário a tabela:

```
postgres=# insert into accounts (email, name, password) values ('user@gmail.com', 'user', 'senha123');
INSERT 0 1
postgres=# select * from accounts;
 id |     email      | name | password
----+----------------+------+----------
  1 | user@gmail.com | user | senha123
(1 row)

postgres=#
```

Criei o dockerfile:

```
# syntax=docker/dockerfile:1

FROM node:18-alpine
WORKDIR /app
COPY . .
RUN ls
RUN yarn install --production
COPY package*.json .
CMD ["node", "dist/serve.js"]
EXPOSE 3000
```

Criando a imagem:

```
PS C:\Users\Eduardo\Desktop\Docker\ex5> docker build -t ex5 .
[+] Building 59.2s (14/14) FINISHED
 => [internal] load build definition from dockerfile
 => => transferring dockerfile: 220B
 => [internal] load .dockerignore
 => => transferring context: 59B
 => resolve image config for docker.io/docker/dockerfile:1
 => [auth] docker/dockerfile:pull token for registry-1.docker.io
 => CACHED docker-image://docker.io/docker/dockerfile:1@sha256:39b85bbfa7536a5feceb737
 => [internal] load metadata for docker.io/library/node:18-alpine
 => [1/6] FROM docker.io/library/node:18-alpine
```

Verificando a rede:


```
"Containers": {
    "47ca5ee73b69653ce2ad061e2ea670ff73551385db55f6c3d8fbd667ec951a84": {
        "Name": "ex5-backend",
        "EndpointID": "13adfbebcd80b9bcc577552a5155f914d3df42f944483a161c36aae1463d48f3",
        "MacAddress": "02:42:ac:12:00:03",
        "IPv4Address": "172.18.0.3/16",
        "IPv6Address": ""
    },
    "956ac6ef928729e4c31fdfb8d90a0b0e761987d93c298ad1ccaa7c90796ed6ec": {
        "Name": "postgres",
        "EndpointID": "913ffca6a03a1780802190fd813688e147be7c8200e12ac6307caac31c186b1b",
        "MacAddress": "02:42:ac:12:00:02",
        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""
    }
},
```


Verificando o se o .env está no container do backend:

```
PS C:\Users\Eduardo\Desktop\Docker\ex5> docker exec -it b972bb5bbe30 ls -a /app
.                   .env.example        dockerfile          package.json        tsconfig.json
..                  DataBase_commands.sql  node_modules     public              yarn.lock
.dockerignore       dist                package-lock.json   src
PS C:\Users\Eduardo\Desktop\Docker\ex5>
```

Fazendo o login no pelo postman: