Relatório atividade 4 da trilha de docker

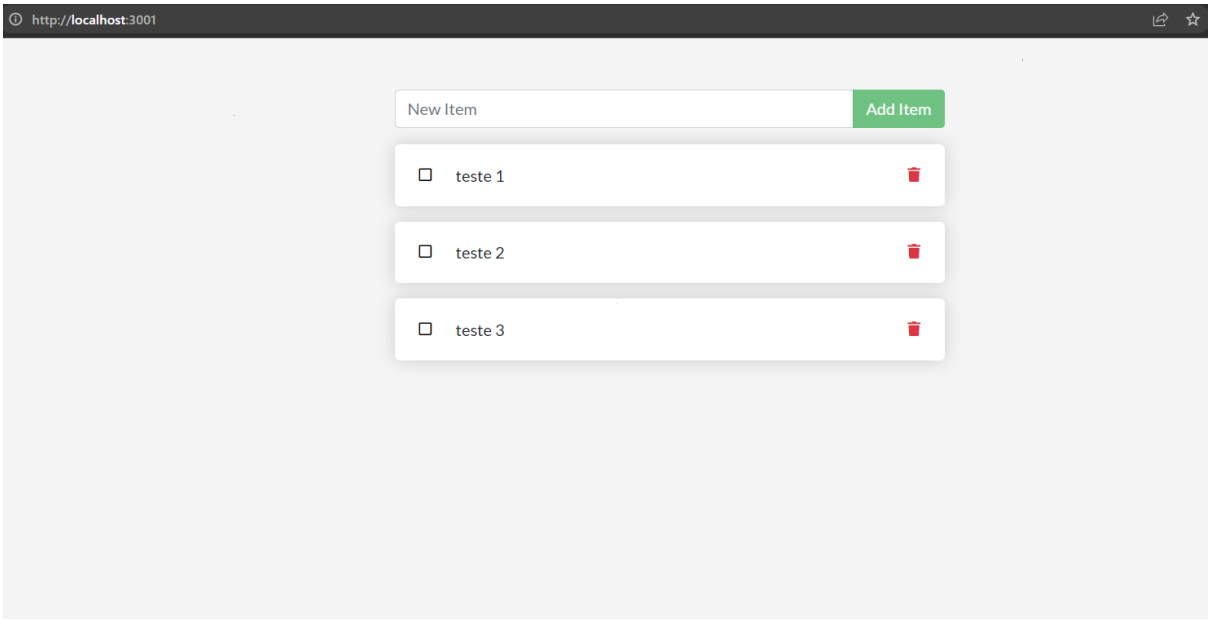
- **Bind mounts:**
    - Primeiramente criei o volume que será usando pelo container:

```
PS C:\Users\Eduardo\Desktop\Docker\getting-started> docker volume create todo-db
todo-db
```

    - Depois de criado o volume crei o container e coloquei para rodar com o comando abaixo:

```
PS C:\Users\Eduardo\Desktop\Docker\getting-started> docker run -dp 3001:3000 --mount type=volume,src=todo-db,target=/etc/todos getting-started
2dad9aedbf61a10351b6ff15b9d430d9f2a38a35a401ba2ff24dc2cc8867af8f
PS C:\Users\Eduardo\Desktop\Docker\getting-started>
```

    - Verifiquei o localhost na porta em que configurei para o container rodar:



    - Cadastrei alguns itens no app, e depois de parar o container e verifiquei se os dados tinham persistido, e sim, eles tinham.

```
PS C:\Users\Eduardo\Desktop\Docker\getting-started> docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED         STATUS         PORTS                    NAMES
[
    {
        "CreatedAt": "2023-05-21T01:55:37Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/todo-db/_data",
        "Name": "todo-db",
        "Options": null,
        "Scope": "local"
    }
]
PS C:\Users\Eduardo\Desktop\Docker\getting-started>
```

- **Volumes:**
    - Criei o volume que será usado no container para persistir os dados com o comando abaixo:

```
PS C:\Users\Eduardo\Desktop\Docker\getting-started_02\app> docker run -it --mount type=bind,src="$(pwd)",target=/src ubuntu bash
```

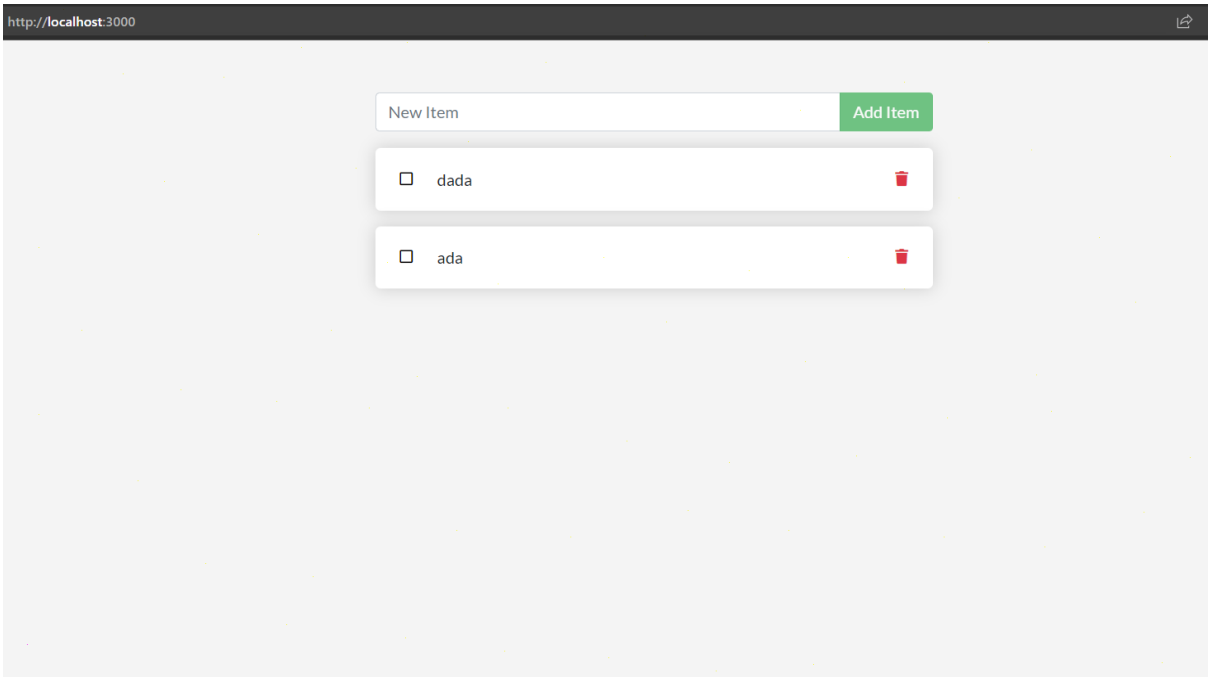
    - Criei o container que usa o volume criado:

```
PS C:\Users\Eduardo\Desktop\Docker\getting-started_02\app> docker run -dp 3000:3000 `
>>     -w /app --mount "type=bind,src=$pwd,target=/app" `
>>     node:18-alpine `
>>     sh -c "yarn install && yarn run dev"
Unable to find image 'node:18-alpine' locally
18-alpine: Pulling from library/node
f56be85fc22e: Already exists
931b0e865bc2: Already exists
60542df8b663: Already exists
062e26bc2446: Already exists
Digest: sha256:1ccc70acda680aa4ba47f53e7c40b2d4d6892de74817128e0662d32647dd7f4d
Status: Downloaded newer image for node:18-alpine
75d2a35412e9527510e86d578c84b9ed4fc5c075a20de6a51a89a2271533c70a
PS C:\Users\Eduardo\Desktop\Docker\getting-started_02\app>
```

    - Verifiquei os logs:

```
PS C:\Users\Eduardo\Desktop\Docker\getting-started_02\app> docker logs -f 75d2a35412e9527510e86d578c84b9ed4fc5c075a20de6a51a89a2271533c70a
yarn install v1.22.19
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
Done in 95.84s.
yarn run v1.22.19
$ nodemon src/index.js
[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node src/index.js`
Using sqlite database at /etc/todos/todo.db
Listening on port 3000
```

    - Resultado:



    - Os dados persistiram com ao stop e o start do container.

- **tmpfs:**
    - Usando o wsl 2, coloquei o seguinte comando no terminal:

```
eduardo@DESKTOP-UTCHH73:/mnt/c/Users/Eduardo/Desktop/Docker/getting-started_01$ docker run -d \
  -it \
  --name tmptest \
  --mount type=tmpfs,destination=/app \
  nginx:latest
dcff986718f07cbb381f23a09e3207fc058a971db5196fcabbd6ec863d153bb4
```

    - Dei inspect no container:

```
eduardo@DESKTOP-UTCHH73:/mnt/c/Users/Eduardo/Desktop/Docker/getting-started_01$ docker inspect tmptest --format ''
```