

AVALIAÇÃO FINAL

<https://forms.gle/kPisCHs3Ed4nEYq26>

QUESTÃO 01 - DOCKER

1. Clone o projeto:

<https://github.com/evandrocustodio/springboot-backend-avaliacao>

2. Edite o arquivo `src/main/resources/application.properties` e altere o valor da propriedade `aluno.nome` para o seu nome completo;

3. Gere a imagem:

`<login-aluno-dockerhub>/avaliação:backend`

4. Faça o push da imagem gerada para o seu repositório dockerhub.

5. Gere uma tag da imagem criada para:

`<login-aluno-dockerhub>/avaliação:backend-tag`

6. Faça o push da tag gerada para o seu repositório dockerhub.

7. Deixe as imagens públicas.

QUESTÃO 02 - DOCKER

1. Clone o projeto:

<https://github.com/evandrocustodio/react-frontend-avaliacao>

2. Gere a imagem:

<login-aluno-dockerhub>/avaliação:frontend

3. Faça o push da imagem gerada para o seu repositório dockerhub.

4. Gere uma tag da imagem criada para:

<login-aluno-dockerhub>/avaliação:frontend-tag

5. Faça o push da tag gerada para o seu repositório dockerhub.

6. Deixe as imagens públicas.

QUESTÃO 03 - DOCKER

1. Instancie os containers:

```
docker run -d --rm --name postgres \
    -e POSTGRES_USER=postgres \
    -e POSTGRES_PASSWORD=avaliacao \
    -e POSTGRES_DB=avaliacao \
    -e POSTGRES_HOST_AUTH_METHOD=trust
postgres:15.2
```

```
docker run -d --rm --name backend \
    --link postgres:postgres \
    -e DB_HOST_NAME=postgres \
    -e DB_PORT=5432 \
    -e DB_NAME=avaliacao \
    -e DB_PASSWORD=avaliacao \
    -p 8080:8080
<login-aluno-dockerhub>/avaliacao:backend
```

```
docker run -d --rm --name frontend \
    --link backend:backend \
    -p 3000:80 \
    <login-aluno-dockerhub>/avaliacao:frontend
```

2. Print a tela com o resultado do comando: **docker ps -a**

3. Acesse o backend e capture a tela dos seguintes serviços:

- `http://localhost:8080/info`
- `http://localhost:8080/despesas/`

4. Acesse o frontend e capture a tela:

- `http://localhost:3000/`

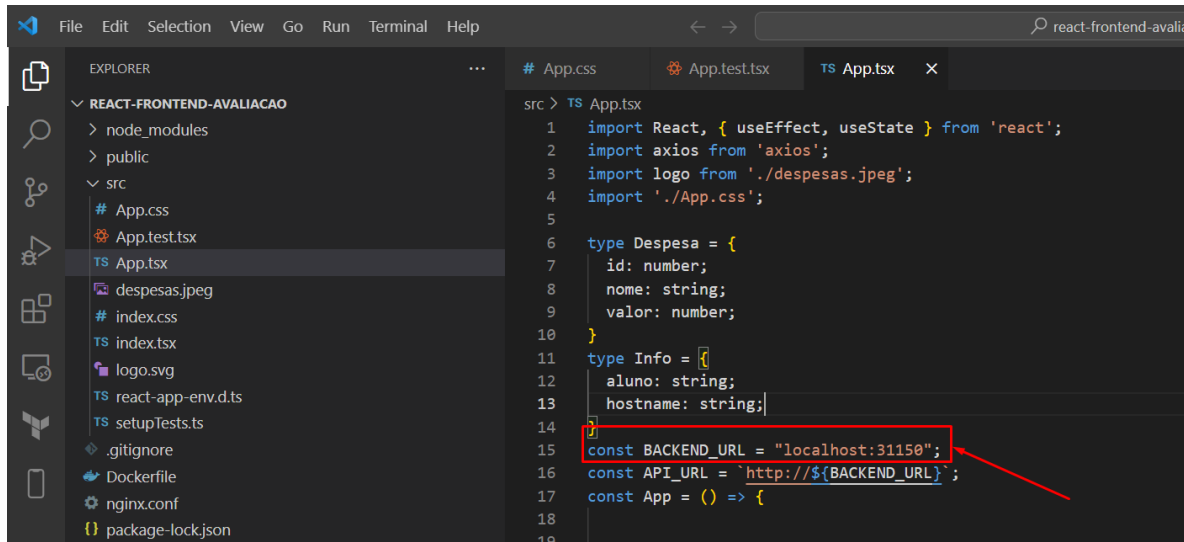
QUESTÃO 04 – DOCKER-COMPOSE e DOCKER SWARM

1. Construa um serviço **na versão 3** para os 3 containers da questão 03.
2. Lembre que o FRONTEND deve ter **2 instâncias**.
3. O FRONTEND não poderá ser instanciado antes do backend.
4. O BACKEND não poderá ser instanciado antes do postgres.
5. Instancie o serviço.
6. Print a tela com o resultado do comando: **docker ps -a**
7. Acesse o backend e capture a tela dos seguintes serviços:
 - `http://localhost:8080/info`
 - `http://localhost:8080/despesas/`
8. Acesse o frontend e capture a tela:
 - <http://localhost:3000/>

QUESTÃO 05 – KUBERNETES - PODS

OBSERVAÇÃO IMPORTANTE:

Abra o arquivo App.tsx e altere a linha 15 e gere novamente a imagem do frontend.



```
src > TS App.tsx
1  import React, { useEffect, useState } from 'react';
2  import axios from 'axios';
3  import logo from './despesas.jpeg';
4  import './App.css';
5
6  type Despesa = {
7    id: number;
8    nome: string;
9    valor: number;
10 }
11 type Info = {
12   aluno: string;
13   hostname: string;
14 }
15 const BACKEND_URL = "localhost:31150";
16 const API_URL = `http://${BACKEND_URL}`;
17 const App = () => {
18
19
```

1. Crie os arquivos YAML para a criação dos **PODS** para os 3 containers da questão 3.
2. Lembre de colocar as tags:
app: postgres, **app: backend** e **app: frontend** nos respectivos PODS.
3. Instancie os 3 **PODS**
4. Capture a tela com o resultado do comando:
kubectl get pods
5. Remova os 3 pods.

QUESTÃO 06 – KUBERNETES - DEPLOYMENTS

1. Crie os arquivos YAML para a criação dos **DEPLOYMENTS** para os 3 containers da questão 3.
2. Lembre de definir 3 instâncias para o FRONTEND
3. Instancie os 3 **DEPLOYMENTS**
4. Capture a tela com o resultado do comando:
kubect1 get all

QUESTÃO 07 – KUBERNETES - SERVICES

1. Crie os arquivos YAML para a criação dos **SERVICES** para os 3 containers da questão 3.
2. Lembre-se que as portas de origem do Backend, frontend e Postgres são respectivamente: 8080, 3000 e 5432.
3. Lembre-se que o backend (Porta 31150) e o Frontend (Porta 32150) devem ser do tipo NodePort e o postgres deve ser do tipo ClusterIP
4. Instancie os 3 SERVIÇOS
5. Capture a tela com o resultado do comando:
kubect1 get svc