

## Gabarito dos exercícios

A seguir, encontram-se sugestões de soluções para os exercícios propostos.

## Exercícios da Aula

Antes de começar os exercícios não se esqueça de fazer o fork dos projetos e cloná-los em sua máquina. [documentação](#) de como fazer um fork.

1. Realize o deploy de uma aplicação Front-end no Heroku utilizando um build-pack.
  - Entre no diretório `heroker-exercise-frontend`, crie sua própria branch e instale as dependências do projeto:

```
$ cd heroker-exercise-frontend
$ git checkout -b nome-da-minha-branch
$ npm install
```

Copiar

- Após concluída a instalação, abra o projeto no navegador e verifique que está tudo funcionando corretamente:

```
$ npm start
```

Copiar

Você deverá ver uma tela com o título "loading". Em breve iremos fazer o deploy do back-end e o verdadeiro conteúdo irá aparecer 😊

- Crie o repositório do seu projeto no Heroku. Não se esqueça de adicionar o buildpack na criação:

```
$ heroku create nome-do-App --buildpack mars/create-react-app
```

Copiar

- Agora basta enviar para o repositório remoto do heroku para que o deploy seja feito:

```
$ git add .
$ git commit -m "Faz deploy da aplicação React no heroku"
$ git push heroku nome-da-sua-branch:master
```

Copiar

2. Realize o deploy de uma API Node no Heroku utilizando Docker.

- Entre no diretório `heroker-exercise-backend`, crie sua própria branch e instale as dependências do projeto:

```
$ cd heroker-exercise-backend
$ git checkout -b nome-da-minha-branch
$ npm install
```

Copiar

Crie o arquivo `Dockerfile` na raiz do projeto para definirmos como será nossa imagem Docker.

Atente-se para o fato de que nossa aplicação não possui um arquivo `index.js`.

No diretório `src/api/` temos dois arquivos, `app.js` e `server.js` (como nosso listen está no `server.js`, usaremos ele para iniciar nossa aplicação):

- Crie o arquivo `Dockerfile` na raiz do projeto para definirmos como será nossa imagem Docker. Atente-se para o fato de que nossa aplicação não p

```
//Arquivo Dockerfile
FROM node:alpine

WORKDIR /app

COPY package.json .

RUN npm install

COPY . .

CMD ["node", "src/api/server.js"]
```

- Crie o arquivo `heroku.yml` na raiz do projeto:

```
//Arquivo heroku.yml
build:
  docker:
    web: Dockerfile
run:
  web: node src/api/server.js
```

- Crie um app dentro do Heroku:

```
$ heroku create nome-do-App
```

- Altere a stack de trabalho para que utilize a imagem de um container:

```
$ heroku stack:set container
```

- Altere a constante PORT no arquivo `server.js` adicionando o `dotenv`:

```
//Arquivo server.js

const app = require('./app');
require('dotenv/config');

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => console.log(`Rodando na porta ${PORT}`));
```

- Adicione as alterações feitas para o git, faça os commits e realize um push para o remote do heroku:

```
$ git add .
$ git commit -m "Faz deploy da aplicação Node no heroku"
$ git push heroku nome-da-sua-branch:master
```

### 3. Crie um banco de dados no SUPABASE

- Acesse sua conta no SUPABASE e comece um novo projeto;

4. Configure as variáveis de ambiente do Back-end. O arquivo de configuração do sequelize está no caminho `/src/sequelize/config/config.js`

- Crie um arquivo `.env` na raiz do projeto e declare as variáveis de ambientes a serem utilizadas no arquivo de configuração do sequelize:

Copiar

```
//Arquivo .env
//# EXEMPLO
PASSWORD_POSTGRES=13244
HOST=db.minhaurldconexao.supabase.co
DATABASE=meubanco
DB_USERNAME=userhost
DB_PORT=6543
```

```
//Arquivo config.js
require('dotenv/config');

const { HOST, PASSWORD_POSTGRES, DATABASE, DB_USERNAME, DB_PORT } = process.env;

module.exports = {
  "development": {
    "username": DB_USERNAME,
    "password": PASSWORD_POSTGRES,
    "database": DATABASE,
    "host": HOST,
    "port": DB_PORT,
    "dialect": "postgres"
  },
```

```
  "test": {
    "username": DB_USERNAME,
    "password": PASSWORD_POSTGRES,
    "database": DATABASE,
    "host": HOST,
    "port": DB_PORT,
    "dialect": "postgres"
  },
  "production": {
    "username": DB_USERNAME,
    "password": PASSWORD_POSTGRES,
    "database": DATABASE,
    "host": HOST,
    "port": DB_PORT,
    "dialect": "postgres"
  }
}
```

- Declare as variáveis de ambiente na aplicação back-end do heroku. Pode ser feito via client do heroku ou via CLI:

Copiar

```
$ heroku config:set NOME_DA_VARIÁVEL=VALOR_DA_VARIÁVEL
```

5. Conecte sua aplicação back-end ao banco de dados.

- Execute no terminal:

Copiar

```
$ npx run utils/testSequelizeConnection.js
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
1:node
fernando@inspiron-5558:~/Documentos/Trybe/SummerTrybe/test-Heroku/Exercicios-aula/herocker-exercise-backend$ npx run utils/testSeque
lizeConnection.js
npx: installed 5 in 1.808s
Watching /home/fernando/Documentos/Trybe/SummerTrybe/test-Heroku/Exercicios-aula/herocker-exercise-backend and all sub-directories n
ot excluded by your .gitignore. Will not monitor dotfiles.
Found & ignored ./node_modules ; is listed in .gitignore

Starting: utils/testSequelizeConnection.js
5432
Executing (default): SELECT 1+1 AS result
Conexão foi estabelecida com sucesso.
```

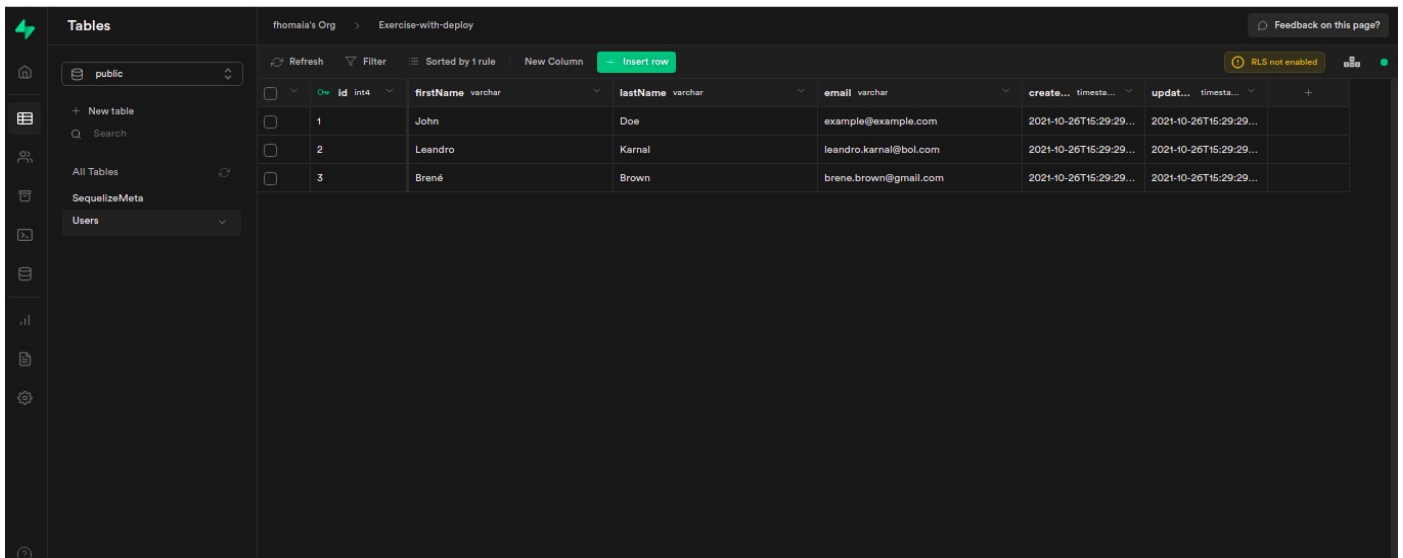
Tela de  
conexão com  
sucesso  
mostrada no  
terminal

6. Popule o banco de dados com informações de pessoas usuárias ou qualquer tipo de dados que viram uma tabela:

- Instale o cliente do sequelize e execute as migrations e as seeds:

Copiar

```
$ npm install -g sequelize-cli
$ npx sequelize db:migrate
$ npx sequelize-cli db:seed:all
```



Tables

thomalia's Org > Exercise-with-deploy

Feedback on this page?

Refresh Filter Sorted by 1 rule New Column Insert row

public

+ New table

Search

All Tables

SequelizeMeta

Users

	id int4	firstName varchar	lastName varchar	email varchar	create... timesta...	updat... timesta...	+
<input type="checkbox"/>	1	John	Doe	example@example.com	2021-10-26T15:29:29...	2021-10-26T15:29:29...	
<input type="checkbox"/>	2	Leandro	Karnal	leandro.karnal@bol.com	2021-10-26T15:29:29...	2021-10-26T15:29:29...	
<input type="checkbox"/>	3	Brené	Brown	brene.brown@gmail.com	2021-10-26T15:29:29...	2021-10-26T15:29:29...	

RLS not enabled

## Inserção com sucesso no banco

7. Consuma sua API pela aplicação front-end e renderize todas as informações retornadas da API.

- Instale o pacote `cors` em sua aplicação back-end:

Copiar

```
$ npm install cors
```

Copiar

- Faça a importação da biblioteca Cors no arquivo app.js da sua aplicação back-end e declare seu uso pela api:

```
//Arquivo app.js
//As partes destacadas correspondem às alterações no arquivo
//const express = require('express');
//const rescue = require('express-rescue');
const cors = require('cors');

//const UserController = require('../controllers/user');
//const errorMiddleware = require('../middlewares/error');

//const app = express();

app.use(cors());

//app.get('/users', rescue(UserController));

//app.use(errorMiddleware);

//module.exports = app;
```

Em seguida, altere o arquivo `app.js` da sua aplicação front-end para que os dados recebidos venham de sua aplicação back-end, que por sua vez realiza a consulta no banco de dados do Supabase.

```
// Arquivo app.js
// As partes destacadas correspondem às alterações no arquivo
//import './App.css';

//import React, { useEffect, useState } from 'react';

const API_ENDPOINT = 'URL_API_BACKEND_HEROKU/users';

//function App() {
//  const [data, setData] = useState([]);
//  const [loading, setLoading] = useState(true);

//  useEffect(() => {
//    fetch(API_ENDPOINT)
//      .then((res) => res.json())
//      .then((parsedData) => { setData(parsedData); setLoading(false); });
//  }, []);

//  if (loading) return <h1>loading....</h1>;
```

- Faça o deploy do front com as atualizações:

```
$ git add .
$ git commit -m "Conecta aplicação React à Api do Supabase"
$ git push heroku nome-da-sua-branch:master
```

#### 8. Crie Actions do Github de lint para Front-end e Back-end;

Você deve seguir o mesmo passo-a-passo em ambos os repositórios:

- Crie uma pasta `.github` na raiz do seu projeto;
- Dentro desta pasta, crie uma nova pasta chamada `workflows` e então crie o arquivo `main.yml` ;
- Configure o arquivo `main.yml` criado:

```
on: [push, pull_request]
```

```
jobs:
```

```
  eslint:
```

```
    runs-on: ubuntu-latest
```

```
    steps:
```

```
      - uses: actions/checkout@v1
```

```
      - uses: stefanoeb/eslint-action@1.0.2
```

- Suba as alterações para seu repositório remoto do gitHub:

```
$ git add .
```

```
$ git commit -m "Conecta aplicação React à Api do Supabase"
```

```
$ git push origin nome-da-sua-branch
```