



Desenvolvimento Backend

Aula 15 - Preparando o ambiente de produ  o



Material Did tico do Instituto Metr pole Digital - IMD

Termo de uso

Os materiais did ticos aqui disponibilizados est o licenciados atrav s de Creative Commons **Atribui  o-SemDeriva  es-SemDerivados CC BY-NC-ND**. Voc  possui a permiss o para realizar o download e compartilhar, desde que atribua os cr ditos do autor. N o poder  alter -los e nem utiliza-los para fins comerciais.

Atribui  o-SemDeriva  es-SemDerivados

CC BY-NC-ND



<https://creativecommons.org/licenses/by-nc-nd/4.0/>

Apresentação

Nesta aula iremos colocar nossa aplicação em produção para que os usuários possam utilizá-la.

Objetivos

- Conhecer como preparar um projeto para ser colocado em produção
- Conhecer como fazer *deploy* no *Heroku*
- Conhecer como criar banco de dados no *Heroku*

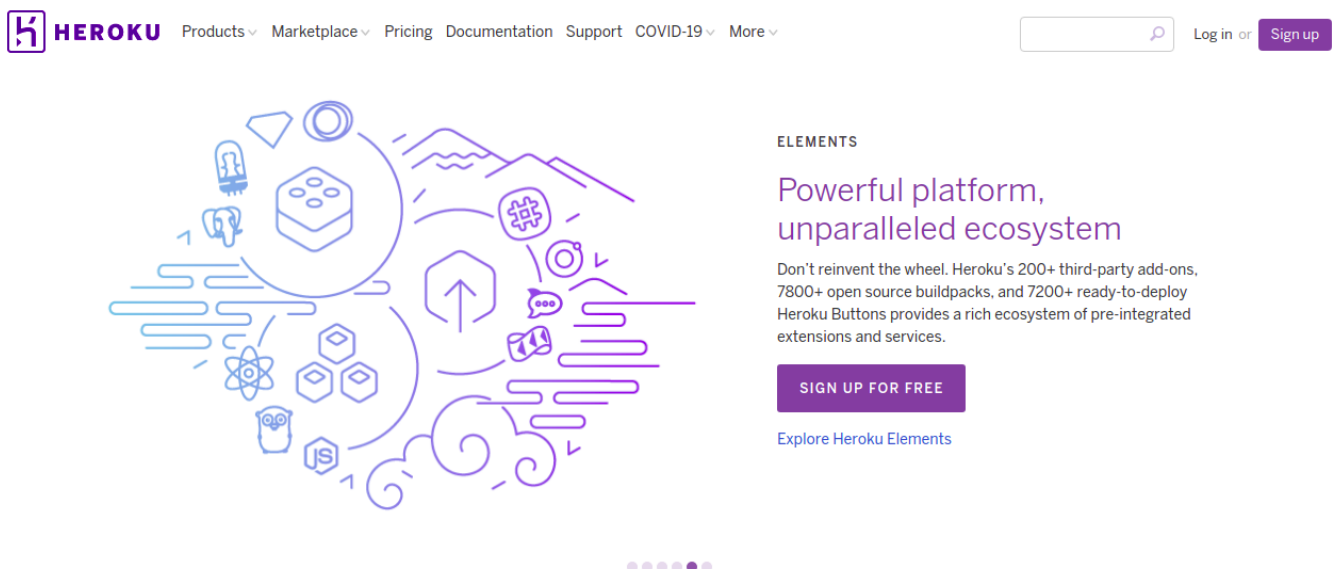
Plataformas de cloud

Link do video da aula: <https://youtu.be/Y6M-yXMrHqU>

Existem várias formas de colocar uma aplicação no ar, sendo ela em computação em nuvem ou em servidores locais. Dentre essas formas, nesta aula, iremos abordar apenas uma delas, que é a computação em nuvem.

Escolhendo o fornecedor

Acompanhe a aula e veja algumas opções de fornecedores. No entanto, para esse projeto, iremos utilizar o [Heroku](#), que é um *Cloud Application Platform*, tornando seu uso mais simples e não necessita de nenhum recurso financeiro.



HEROKU Products Marketplace Pricing Documentation Support COVID-19 More

Log in or Sign up

ELEMENTS

Powerful platform, unparalleled ecosystem

Don't reinvent the wheel. Heroku's 200+ third-party add-ons, 7800+ open source buildpacks, and 7200+ ready-to-deploy Heroku Buttons provides a rich ecosystem of pre-integrated extensions and services.

SIGN UP FOR FREE

Explore Heroku Elements

Preparando o sistema para produção

Link do video da aula: https://youtu.be/qyDMdA9Q_Lc

Antes de começar a utilizar o *Heroku*, é preciso realizar algumas adequações no sistema.

Instalando o *Helmet*

No site do [express](#) existe uma página explicando boas práticas para colocar um sistema em produção, se preocupando para que a aplicação não tenha vulnerabilidades de segurança.

Para auxiliar na questão de segurança, iremos instalar a biblioteca *Helmet* através do comando abaixo:

```
npm install --save helmet
```

Em seguida, no seu código, cadastre o *Helmet* como um *middleware express* utilizando as linhas de código abaixo:

```
const helmet = require('helmet');  
app.use(helmet());
```

Ajustando a porta

Até agora a porta escutada é a '8080', no entanto, é uma boa prática não deixá-la totalmente fixa. Então, para mudar isso será feito o uso de uma variável de ambiente. Sendo assim, altere seu código e, após a alteração, ele deve ficar como o código abaixo:

```
const PORT = process.env.PORT || 8080  
  
app.listen(PORT, () => {  
  logger.info(`Iniciando no ambiente ${process.env.NODE_ENV}`)  
  logger.info(`Servidor pronto na porta ${PORT}`)  
})
```

Ajustando os *scripts*

Nos *scripts* foram definidos o ambiente de desenvolvimento em *start* e de produção em *production*. Entretanto, as plataformas buscam o *script* de *start*, então é mais

usual deixá-lo com o ambiente de produção. Além disso, a cada atualização é necessário aplicar a migração, então coloque-a junto ao *script* de *start*. Ao final, seus scripts ficarão dessa forma:

```
"scripts": {  
  "dev": "cross-env NODE_ENV=development nodemon -e yaml,js,json -  
-exec node src/index.js",  
  "start": "npx sequelize-cli db:migrate && cross-env  
NODE_ENV=production node src/index.js",  
  "migrate-dev": "cross-env NODE_ENV=development npx sequelize-cli  
db:migrate",  
  "migrate-prod": "cross-env NODE_ENV=production npx sequelize-cli  
db:migrate"  
},
```

Ajustando o modelo do *post*

Ao inserir um post no blog o usuário vai adicionar um texto, que está sendo capturado como uma *string*. Todavia, o tipo *string* restringe o tamanho do texto e isso pode atrapalhar uma publicação.

O *javascript* contorna isso utilizando o tipo *TEXT* para capturar dados de texto. Então, no modelo de *post* altere o texto para o tipo *TEXT* como se vê abaixo:

```
Post.init({  
  titulo: DataTypes.STRING,  
  texto: DataTypes.TEXT,  
  userId: DataTypes.INTEGER,  
  foto: DataTypes.STRING  
},
```

Agora, na migração é preciso alterar a criação da tabela, no campo texto, também para o tipo *TEXT*, como abaixo:

```
texto: {  
  type: Sequelize.TEXT  
},
```

Ajustando as sementes (*seeders*)

Quando criamos as sementes, não havíamos visto ainda como criptografar a senha, mas agora, sabendo como fazer isso, não faz mais sentido usar a senha não criptografada. Então, gere uma senha criptografada e altere no arquivo de

sementes.

Veja o arquivo de sementes abaixo:

```
'use strict';

module.exports = {
  up: async (queryInterface, Sequelize) => {
    await queryInterface.bulkInsert('Usuarios', [{
      email: "root@gmail.com",
      senha:
"$2b$10$DYRZPVWXS1nQjeo6qkk0Y0bHiWkCivdiM5dI.WMlTU3ujppbGCmt0",
      createdAt: new Date(),
      updatedAt: new Date()
    }])
  },

  down: async (queryInterface, Sequelize) => {
    await queryInterface.bulkDelete('Usuarios', {email:
"root@gmail.com"}, {});
  }
};
```

Deploy no Heroku

Link do video da aula: <https://youtu.be/2B51WU9wYbU>

Agora que o projeto foi preparado, podemos fazer o deploy no [Heroku](#). Caso você não tenha cadastro ainda, faça-o e efetue o *login*.

Criando uma aplicação

Após feito o *login*, o usuário será direcionado até a pasta das aplicações e verá o botão de 'New+'. Deve ser criada uma aplicação e o caminho é: "New+ -> Create New App" e, após isso, deve-se preencher as informações solicitadas.

Feito isso, surgirá uma tela com as instruções e a primeira delas é instalar o [Heroku CLI](#) conforme o sistema operacional.

Ubuntu 16+:

```
sudo snap install --classic heroku
```

macOS:

```
brew tap heroku/brew && brew install heroku
```

Windows:

Acesse o link e faça o download do executável [aqui](#), em seguida, realize a instalação.

Subindo a aplicação

Após instalar o *Heroku CLI*, crie um novo repositório *Git*.

```
cd meu-projeto /  
git init  
heroku git: remote -a dev-beck-end
```

Nota: É preciso instalar o git antes de usá-lo. Portanto, caso não o tenha em sua máquina, faça a instalação através do link: .

Agora é hora de implantar a aplicação. Mas antes disso, queremos informar que alguns arquivos não devem ser enviados. Então, antes de fazer a implantação, crie um arquivo `'.gitignore'` que vai conter o que deve ser ignorado no envio.

Veja abaixo o `'.gitignore'` do projeto:

```
logs  
node_modules  
.env  
.post.http  
.usuario.http
```

Agora faça commit do seu código no repositório e implante-o no Heroku usando o Git.

```
git add.  
git commit -am "make it better"  
git push heroku master
```

Criando banco de dados no Heroku

Link do video da aula: <https://youtu.be/iUHtMdBMaGY>

Como não foi configurado no *Heroku*, a comunicação com o banco de dados não aconteceu após o *deploy*. Então, agora iremos criar um banco de dados no servidor.

Criando banco de dados *Postgres*

Nas opções disponibilizadas pelo *Heroku*, a que vamos utilizar será o *Postgres*, pois embora não a tenhamos usado antes, isso não vai gerar problema, porque estamos utilizando o *sequelize*, que dá suporte também ao *Postgres*.

Seguindo a aula, será feita a criação do banco no *Heroku* e a aplicação terá um banco de dados associado. No mais, é preciso agora associar um ao outro, pois a aplicação ainda não sabe onde está esse banco.

Associando a aplicação ao banco

Já na área do banco no *Heroku*, acesse a opção de *settings* e é possível ter acesso às credenciais do banco de dados. Dentre essas opções, temos a *URI* que condensa todas as informações.

O *sequelize* consegue ler essas informações de uma variável de ambiente, mas para isso, é preciso alterar o ambiente de produção nas configurações do banco de dados.

Acompanhando a aula e realizando as modificações, a configuração do ambiente de produção deve ficar como abaixo:

```
"production": {
  "use_env_variable": "DATABASE_URL",
  "username": "root",
  "password": "changeme",
  "database": "dbend",
  "host": "127.0.0.1",
  "dialect": "postgres",
  "dialectOptions": {
    "ssl": {
      "require": true,
      "rejectUnauthorized": false
    }
  }
}
```

Para poder utilizar o dialeto do *postgres*, é necessário fazer a instalação através do comando abaixo:

```
npm install --save pg pg-hstore
```

Feitas as alterações, é preciso enviá-las ao *Heroku*. Antes, porém, é importante configurar a variável de ambiente lá no *Heroku*.

Então, seguindo a aula, configure essa variável no servidor.

Enviando as alterações

Para enviar as alterações feitas, é só seguir o mesmo processo de *deploy*. Seguem abaixo os comandos:

```
git add.  
git commit -am "ajusta configurações de acesso ao banco"  
git push heroku master
```

Testando o sistema em produção

Link do video da aula: <https://youtu.be/3vbrTh7Fudo>

Para finalizar, iremos fazer alguns ajustes nos arquivos do projeto e testar a aplicação adicionando alguns posts ao blog.

Ajustes no projeto

Até o momento nossa documentação só possui um servidor local, pois quando foi criada não tínhamos ainda outro endereço. Então o primeiro ajuste será adicionar uma nova url na documentação para o servidor em produção.

```
- url: https://dev-backend-imd.herokuapp.com/api  
  description: Servidor de produção
```

Nota: a url deve ser a do seu deploy, essa é apenas um exemplo

Após isso, o próximo ajuste será no `'.gitignore'`, adicionando o banco de desenvolvimento aos arquivos que devem ser ignorados no envio.

Veja abaixo o `'.gitignore'`:

```
logs  
node_modules  
.env
```



```
.post.http  
.usuario.http  
dev.sqlite
```

Feitas as alterações, podemos enviar o projeto, seguindo os mesmos passos já realizados anteriormente.

Adicionando *post* ao *blog*

Para adicionar novos posts ao blog, será utilizado o *Insominia*. Então, inicialmente precisamos criar um usuário no servidor utilizando a semente (*seed*) do projeto. Para executá-la, rode o comando abaixo no terminal.

```
heroku run npx sequelize-cli db:seed:all
```

Após isso, já teremos um usuário criado e poderemos fazer o login.

Com o auxílio do *Insominia*, acesse a rota de login, entrando com o e-mail e senha criado na semente. Com isso, será retornado um *token* que pode deve ser colocado no *Bearer* das requisições seguintes.

Então, prepare uma requisição de adicionar *post* no *Insominia* com todos os campos necessário e utilizando o *token*.

Agora o blog já deve conter os *posts* adicionados por você.

Mensagem final

Link do video da aula: <https://youtu.be/HzNBO4gDGlo>

Parabéns, você chegou ao final do curso de Desenvolvimento *Back-end*! Nesse curso nós passamos por uma infinidade de conteúdos, de conceitos: autenticação, como criar uma *API*, como colocar no ar etc.

Espero que esse conteúdo tenha sido útil na sua formação e desejo muito sucesso. Continue estudando, pois o mundo do desenvolvimento tem muito a nos ensinar.

Fico por aqui, até a próxima.

Resumo

Nesta aula, vimos como fazer o *deploy* de nossa *API*, colocando-a em produção e realizando os testes para verificar seu funcionamento. Ademais, vimos algumas

opções de servidores, focando especificamente no *Heroku* para utilizar suas funcionalidades.