**Curso de Node.JS**

**Bluuweb**

<https://www.youtube.com/watch?v=mG4U9t5nWG8&list=PLPl81lqbj-4IEnmCXEJeEXPepr8gWtsl6>

<https://github.com/bluuweb/curso-node-youtube/tree/crud-mongo>

Documentação utilizada:

<https://bluuweb.github.io/node/>

# Aula 01 - Fundamentos

### O que é Node.js?

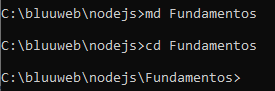
* Node.js é um ambiente em tempo de execução multiplataforma, de código aberto para a camada do servidor (mas não limitado a ele) baseado na linguagem de programação JavaScript.
* Ao contrário da maioria do código JavaScript, ele não é executado em um navegador, mas sim no servidor.
* É baseado no motor V8 do Google.
* Permite que você crie aplicativos que rodam no servidor.
* Geralmente você encontrará muitas informações sobre node.js vs php (para aplicativos da web)

## Primeiros passos

- Instale o Node e o Visual Studio Code.



- Crie uma pasta chamada "Fundamentos"





- Dentro dessa pasta adicione um arquivo chamado app.js:



**app.js**

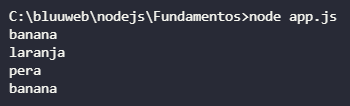
const frutas = ['banana', 'laranja', 'pera', 'banana'];

frutas.forEach(fruta => {

console.log(fruta);

})

node app.js



**app.js**

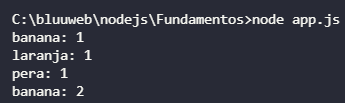
const frutas = ['banana', 'laranja', 'pera', 'banana'];

frutas.forEach(fruta => {

console.count(fruta);

})

node app.js

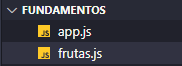


# Aula 02 - Módulos - Exports e Require

Node.js tem um sistema de módulo integrado. Um arquivo Node.js pode importar a funcionalidade exposta por outros arquivos Node.js.

## Exportando

- Adicione um arquivo chamado frutas.js:



**frutas.js**

const frutas = ['banana', 'laranja', 'pera', 'banana'];

module.exports = frutas;

## Importando

Quando você quiser importar algo, use:

**app.js**

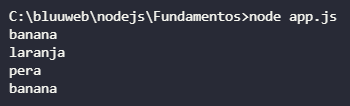
const frutas = require('./frutas');

frutas.forEach(fruta => {

console.log(fruta);

})

node app.js



## Exportando mais de uma constante

**frutas.js**

const frutas = ['banana', 'laranja', 'pera', 'banana'];

const valor = 1000;

module.exports = {

frutas: frutas,

valor

};

**app.js**

const {frutas, valor} = require('./frutas');

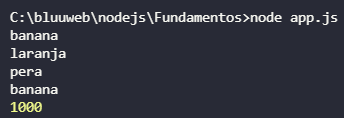
frutas.forEach(fruta => {

console.log(fruta);

})

console.log(valor);

node app.js

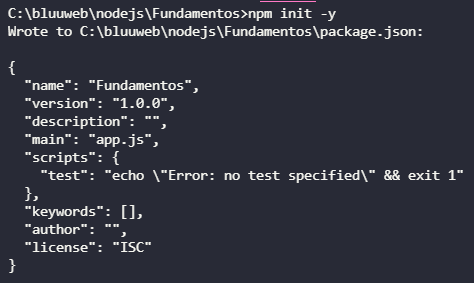


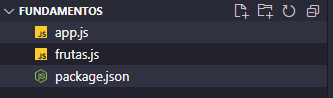
# Aula 03 - package.json e npm

## Gerando package.json

npm init -y

Vamos criar um arquivo que terá informações sobre o nosso projeto, o mais relevante no momento serão suas dependências e scripts





**package.json**

{

"name": "Fundamentos",

"version": "1.0.0",

"description": "",

"main": "app.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

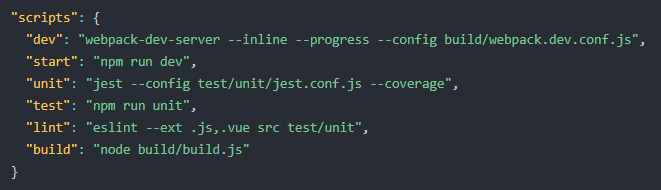
"keywords": [],

"author": "",

"license": "ISC"

}

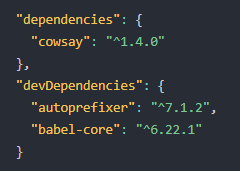
### Exemplo de scripts



Esses scripts são aplicativos de linha de comando. Você pode executá-los chamando npm run XXXX, onde "XXXX" é o nome do comando. Exemplo: npm run dev

Você pode usar o nome que quiser para um comando e os scripts podem fazer literalmente tudo o que você quiser.

### Dependências e devDependencies



* Aqui está a lista de pacotes instalados por meio do npm.
* devDependencies refere-se a pacotes que não são necessários para a produção, eles devem ser instalados apenas em uma máquina de desenvolvimento, eles não são necessários para executar o código na produção.

## NPM (Node Package Manager)

npm é o gerenciador de pacotes padrão para Node.js.

### Atualizando pacotes

npm update

Atualizando pacotes: o npm pesquisará todos os pacotes por uma versão mais recente que satisfaça suas restrições de versão.

### Onde o npm instala os pacotes?

Ao instalar um pacote com npm, você pode realizar 2 tipos de instalação:

* Local: npm install <package-name> (a pasta node\_modules é criada)
* Global: npm install -g <package-name>

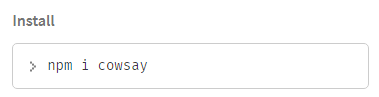
Geralmente utilizamos instalação global para os CLI, por exemplo o cli de vue.js, outra opção são para os pacotes frequentes como nodemon.

### Exemplo de instalação de um pacote (cowsay)

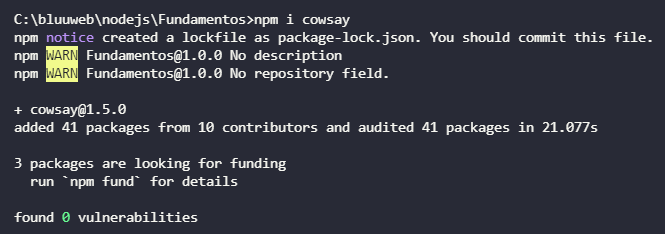
<https://www.npmjs.com/package/cowsay>

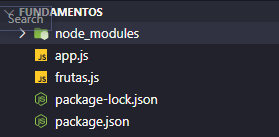


### Instalando o pacote cowsay de forma local



npm i cowsay





**package.json**

{

"name": "Fundamentos",

"version": "1.0.0",

"description": "",

"main": "app.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [],

"author": "",

"license": "ISC",

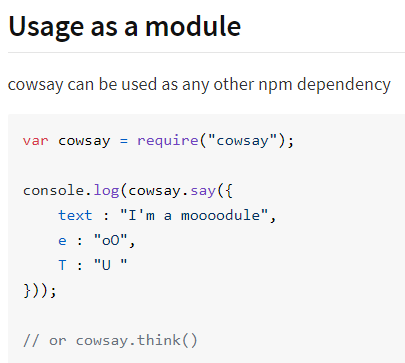
"dependencies": {

"cowsay": "^1.5.0"

}

}

- Uso:



**app.js**

const cowsay = require("cowsay");

console.log(cowsay.say({

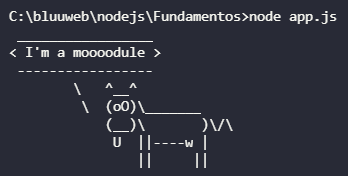
text : "I'm a moooodule",

e : "oO",

T : "U "

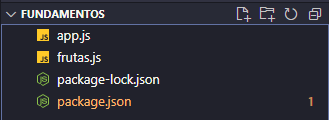
}));

node app.js



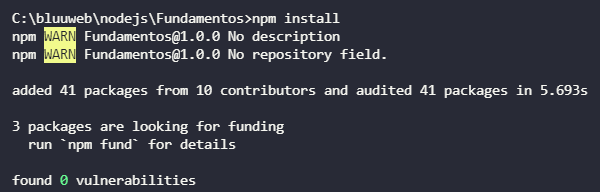
A pasta node\_modules nunca deve ser enviada ao servidor.

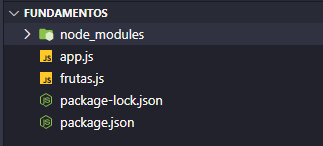
- Apague a pasta node\_modules.



- Para recuperá-la (com todas as dependências listadas no arquivo package.json), basta rodar o comando:

npm install





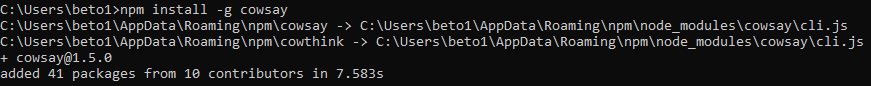
### Instalando o pacote cowsay de forma global

- É instalado no computador e não especificamente no projeto.

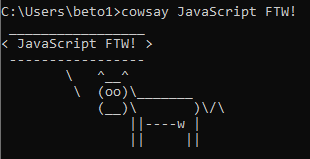


Em qualquer pasta, entre com o comando:

npm install -g cowsay



cowsay JavaScript FTW!



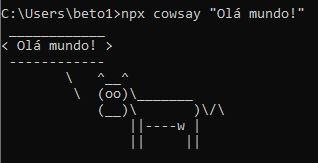
## npx

- Muito usado em React e Vue.JS

- Executa um código sem a necessidade de instalar o pacote de maneira global.

- Por exemplo:

npx cowsay "Olá mundo!"



# Aula 04 - Fundamentos de servidor HTTP

Servidor HTTP?

Hypertext Transfer Protocol: O Hypertext Transfer Protocol é o protocolo de comunicação que permite a transferência de informações na World Wide Web.

* Troca de informações entre cliente e servidor.
* O servidor espera por alguma solicitação HTTP do cliente e fornece uma resposta.

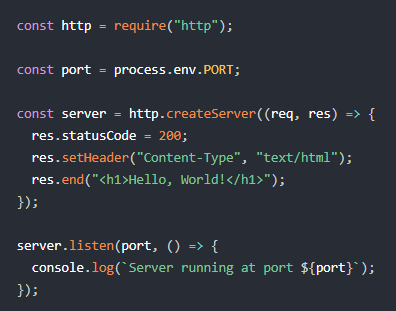
Quando visitamos um site, fazemos uma solicitação HTTP GET e o servidor nos retorna, por exemplo, um index.html com o site.

Vamos configurar essas respostas em nosso "servidor web ou servidor http" com node.js

## Como começo com o Node.js depois de instalá-lo?

O Node.js vem com alguns módulos que não precisam de instalação, um deles é o http.

Depois de instalar o Node.js, vamos construir nosso primeiro servidor web. Crie um arquivo chamado app.js contendo o seguinte conteúdo:



Agora, execute o seu servidor web usando node app.js. Visite <http://localhost:3000> e você verá uma mensagem dizendo "Hello World".

Ctrl + C para encerrar o servidor

### Verbos ou métodos HTTP

O HTTP define um conjunto de métodos de solicitação para indicar a ação a ser executada para um determinado recurso.

Os mais populares são: GET, POST, PUT, DELETE.

**app.js**

const http = require('http');

const hostname = '127.0.0.1';

const port = 3000;

const server = http.createServer((req, res) => {

res.statusCode = 200;

res.setHeader('Content-Type', 'text/plain');

res.end('Hello World');

});

server.listen(port, hostname, () => {

console.log(`Server running at http://${hostname}:${port}/`);

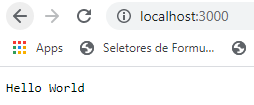
});

node app.js



- No browser:

localhost:3000



Faça a seguinte alteração na mensagem, e em seguida a salve:

**app.js**

const http = require('http');

const hostname = '127.0.0.1';

const port = 3000;

const server = http.createServer((req, res) => {

res.statusCode = 200;

res.setHeader('Content-Type', 'text/plain');

res.end('Bem-vindo ao Node.JS!');

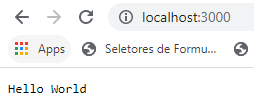
});

server.listen(port, hostname, () => {

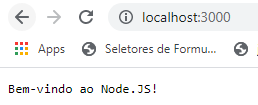
console.log(`Server running at http://${hostname}:${port}/`);

});

- Recarregue o browser e perceba que a alteração não foi refletida:



- Para que isso ocorra é necessário, interromper o servidor e em seguida reiniciá-lo:

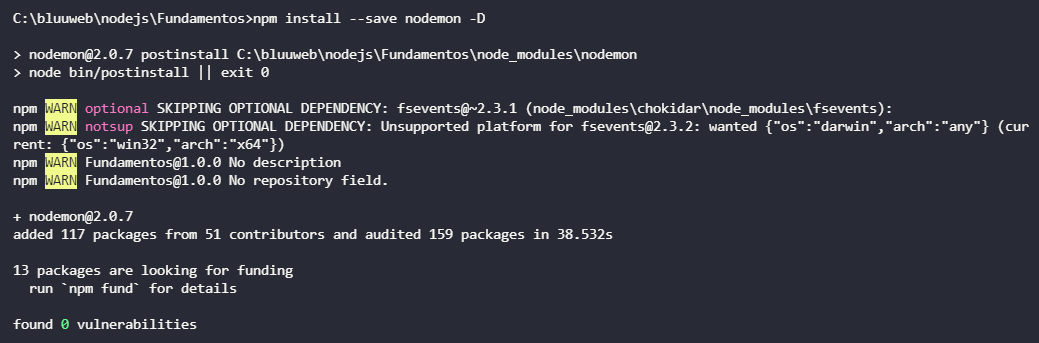


- Para evitar que tenhamos que fazer isso cada vez que for feita uma alteração no projeto devemos instalar e utilizar um pacote chamado nodemon

Essa ferramenta é instalada por meio do npm e nos ajuda a ouvir as alterações em nossa configuração node.js e reiniciar o servidor automaticamente.

## Instalando o pacote nodemon

npm install --save nodemon -D



Obs: -D informa que o nodemon será utilizado apenas em ambiente de desenvolvimento.

**package.json**

{

"name": "Fundamentos",

"version": "1.0.0",

"description": "",

"main": "app.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"cowsay": "^1.5.0"

},

"devDependencies": {

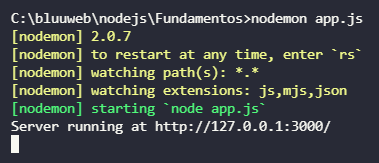
"nodemon": "^2.0.7"

}

}

- Agora, ao invés de utilizarmos node app.js, usamos nodemon app.js

nodemon app.js



- Faça uma alteração na mensagem:

**app.js**

const http = require('http');

const hostname = '127.0.0.1';

const port = 3000;

const server = http.createServer((req, res) => {

res.statusCode = 200;

res.setHeader('Content-Type', 'text/plain');

res.end('Seja bem-vindo ao Node.JS!');

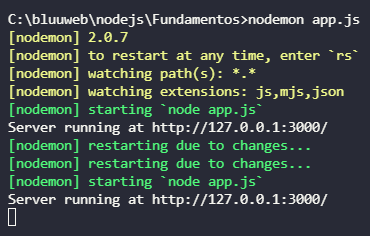
});

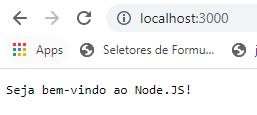
server.listen(port, hostname, () => {

console.log(`Server running at http://${hostname}:${port}/`);

});

- Depois de gravar o arquivo, a alteração será refletida no browser ao recarregarmos:



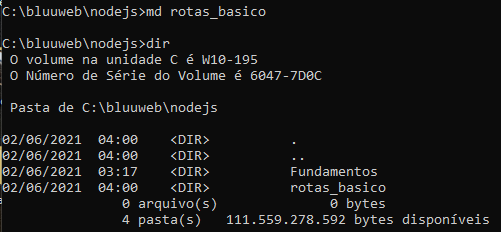


# Aula 05 - Introdução ao Express

Express é uma estrutura de aplicativo da web Node.js mínima e flexível que fornece um conjunto robusto de recursos para aplicativos da web e móveis (Framework).

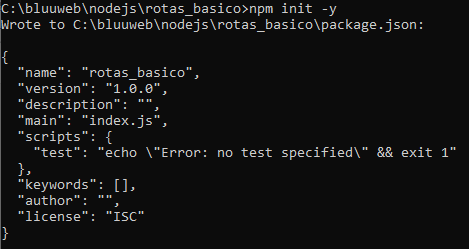
Com milhares de métodos de middleware e utilitários HTTP à sua disposição, construir uma API robusta é rápido e fácil.

- Em C:\bluuweb\nodejs crie uma pasta chamada "rotas\_basico"



## Gerando package.json

npm init -y

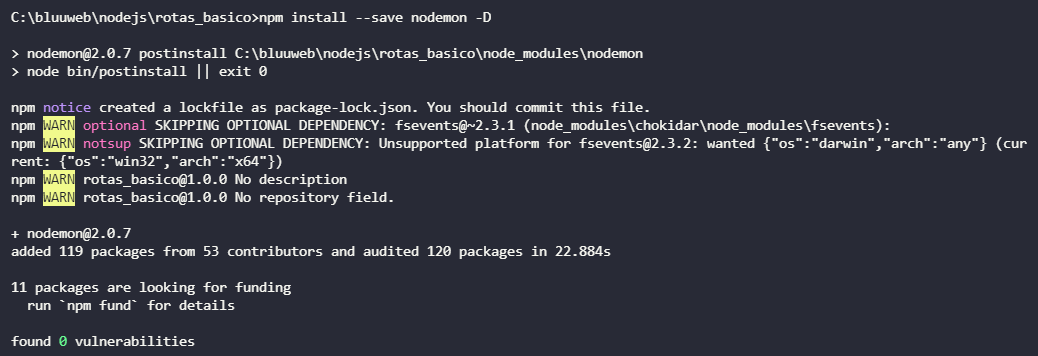


code .



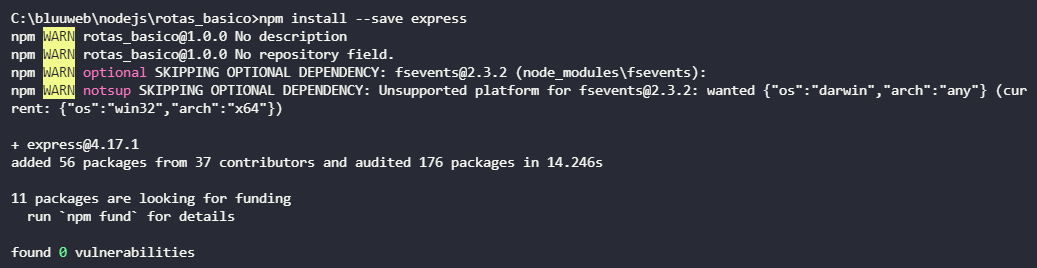
## Instalando o nodemon

npm install --save nodemon -D

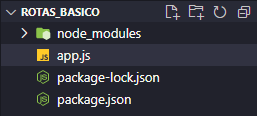


## Instalando o Express

npm install --save express



- Na pasta do projeto adicione um arquivo chamado app.js:



**app.js**

const express = require('express');

const app = express();

const port = 3000;

app.get('/', (req, res) => {

res.send('Hello World!');

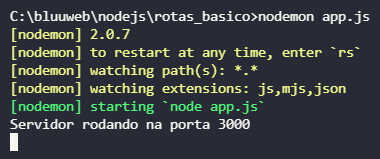
});

app.listen(port, () =>{

console.log('Servidor rodando na porta ' + port);

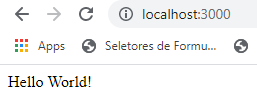
});

nodemon app.js



- No browser:

localhost:3000



**app.js**

const express = require('express');

const app = express();

const port = 3000;

// Rotas

app.get('/', (req, res) => {

res.send('Hello World!');

});

app.get('/services', (req, res) => {

res.send('Página de serviços!');

});

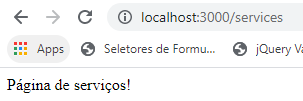
app.listen(port, () =>{

console.log('Servidor rodando na porta ' + port);

});

- No browser:

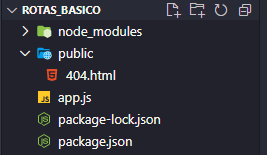
localhost:3000/services



## Arquivos estáticos

Para servir arquivos estáticos, como imagens, arquivos CSS e arquivos JavaScript, use a função de middleware integrada express.static do Express.

- Na pasta inicial do projeto adicione uma subpasta chamada "public" e dentro dela adicione um arquivo chamado "404.html":



### Configurando um middleware

Em palavras simples, middleware é uma ação executada antes de nossa função de rota.

**app.js**

const express = require('express');

const app = express();

const port = 3000;

// Rotas

app.get('/', (req, res) => {

res.send('Hello World!');

});

app.get('/services', (req, res) => {

res.send('Página de serviços!');

});

app.use(express.static(\_\_dirname + "/public"));

app.listen(port, () =>{

console.log('Servidor rodando na porta ' + port);

});

\_\_dirname é o caminho de acordo com a máquina onde o código é executado:

**public\404.html**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Erro 404</title>

</head>

<body>

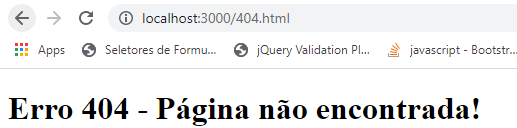
<h1>Erro 404 - Página não encontrada!</h1>

</body>

</html>

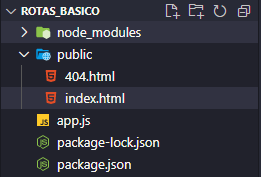
- No browser:

http://localhost:3000/404.html



### Configurando página inicial como um arquivo estático

- Dentro da pasta public adicione um arquivo chamado index.html:



**app.js**

const express = require('express');

const app = express();

const port = 3000;

app.use(express.static(\_\_dirname + "/public"));

// Rotas

app.get('/', (req, res) => {

res.send('Hello World!');

});

app.get('/services', (req, res) => {

res.send('Página de serviços!');

});

app.listen(port, () =>{

console.log('Servidor rodando na porta ' + port);

});

**public\index.html**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Home</title>

</head>

<body>

<h1>Página inicial</h1>

</body>

</html>

- No browser:

localhost:3000



## Tratando rota não encontrada

**app.js**

const express = require('express');

const app = express();

const port = 3000;

app.use(express.static(\_\_dirname + "/public"));

// Rotas

app.get('/', (req, res) => {

res.send('Hello World!');

});

app.get('/services', (req, res) => {

res.send('Página de serviços!');

});

// Rota não encontrada

app.use((req, res, next) => {

res.status(404).sendFile(\_\_dirname + "/public/404.html");

});

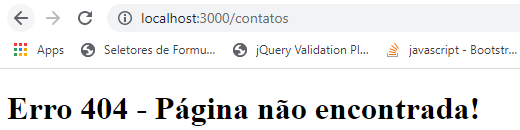
app.listen(port, () =>{

console.log('Servidor rodando na porta ' + port);

});

- No browser:

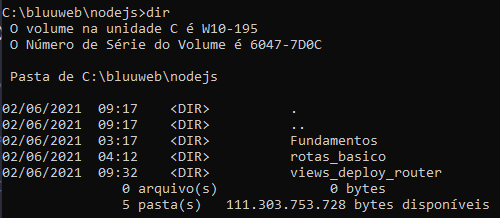
http://localhost:3000/contatos



# Aula 06 - Template Engines (EJS) com Express

- Em C:\bluuweb\nodejs crie uma pasta chamada "views\_deploy\_router"

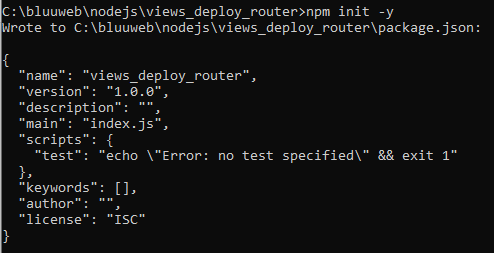






## Gerando package.json

npm init -y

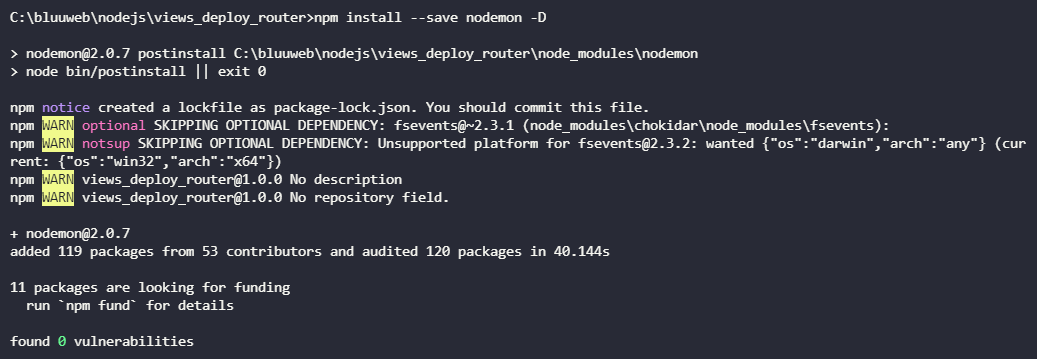


code .



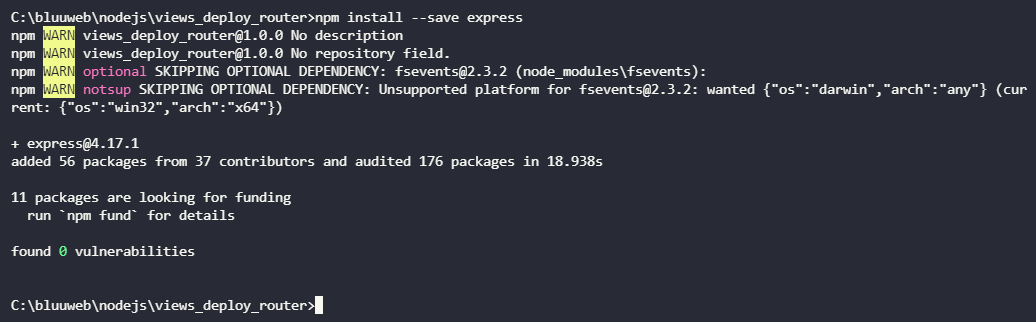
## Instalando o nodemon

npm install --save nodemon -D



## Instalando o Express

npm install --save express



- Na pasta do projeto adicione um arquivo chamado app.js:



**app.js**

const express = require('express');

const app = express();

const port = 3000;

app.get('/', (req, res) => {

res.send('Hello World!');

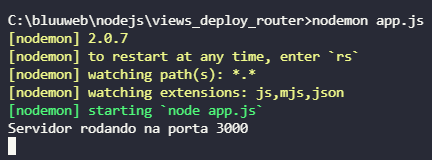
});

app.listen(port, () =>{

console.log('Servidor rodando na porta ' + port);

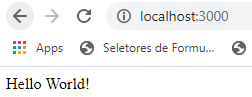
});

nodemon app.js



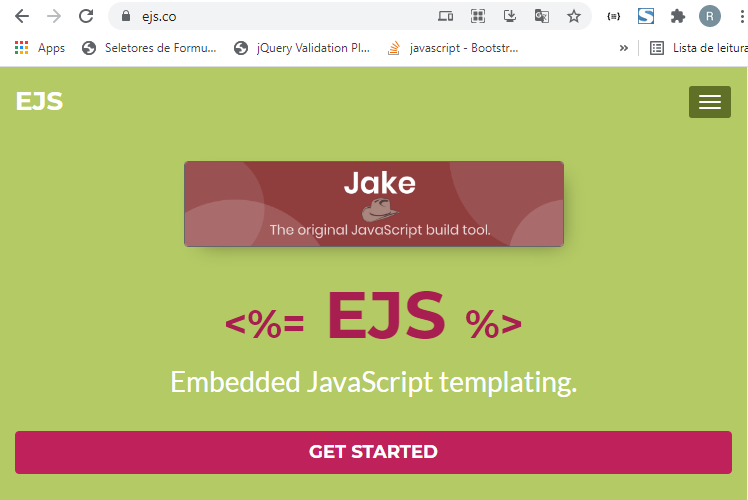
- No browser:

localhost:3000

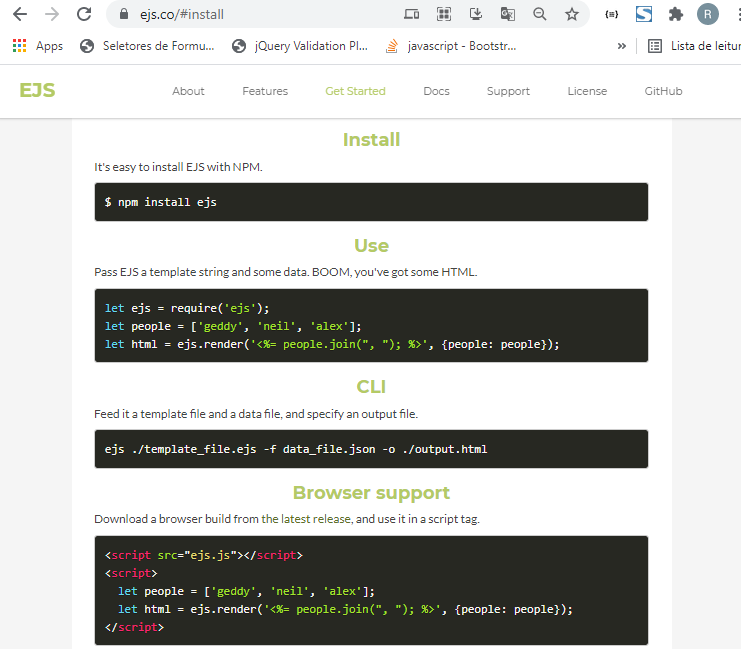


## ejs

<https://ejs.co/>

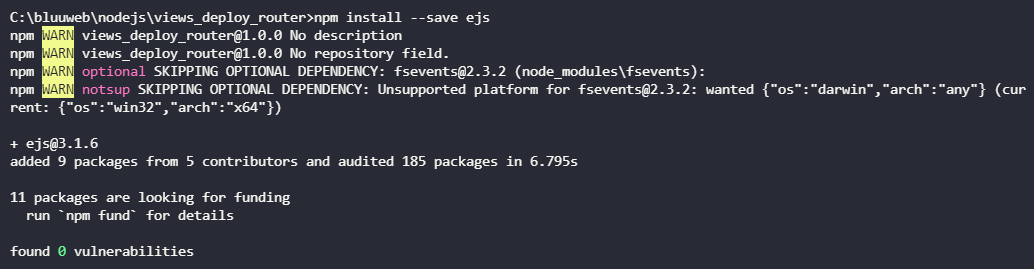


<https://ejs.co/#install>

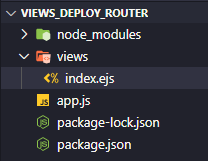


## Instalando EJS

npm install --save ejs



- Na pasta do projeto, crie uma subpasta chamada "views". Dentro dela adicione um arquivo chamado "index.ejs"



**views\index.ejs**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title><%= titulo %></title>

</head>

<body>

<h1><%= conteudo %></h1>

</body>

</html>

**app.js**

const express = require('express');

const app = express();

const port = 3000;

// motor de templates

app.set('view engine', 'ejs');

app.set('views', \_\_dirname + '/views');

app.use(express.static(\_\_dirname + "/public"));

// Rotas

app.get('/', (req, res) => {

res.render("index", {titulo: "Template EJS", conteudo: "Título dinâmico"});

});

app.get('/services', (req, res) => {

res.send('Página de serviços!');

});

app.get('/contact', (req, res) => {

res.send('Página de contato!');

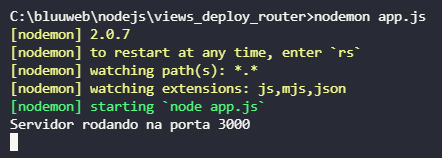
});

app.listen(port, () =>{

console.log('Servidor rodando na porta ' + port);

});

nodemon app.js

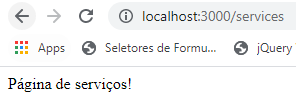


- No browser:

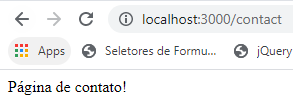
http://localhost:3000/



http://localhost:3000/services

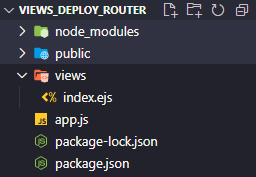


http://localhost:3000/contact

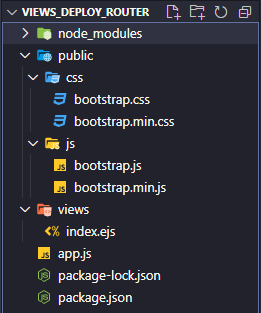


# Aula 07 - Navbar Responsive com Bootstrap 4

- Dentro da pasta do projeto, adicione uma semipasta chamada "public":

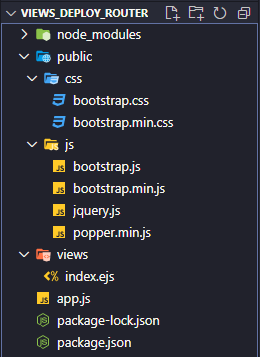


- Baixe o bootstrap e instale as pastas css e js dentro da pasta public. Mantenha apenas os arquivos exibidos na figura abaixo, os demais apague.



- Baixe o jquery 3.2.1.js e o adicione na pasta js.

- Baixe o arquivo popper.min.js e também adicione na pasta js.

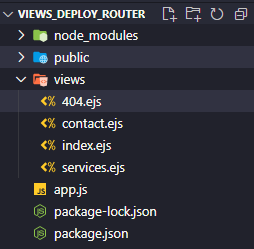


- Dentro da pasta views, adicione os seguintes arquivos:

services.ejs

contact.ejs

404.ejs



**app.js**

const express = require('express');

const app = express();

const port = 3000;

// motor de templates

app.set('view engine', 'ejs');

app.set('views', \_\_dirname + '/views');

app.use(express.static(\_\_dirname + "/public"));

// Rotas

app.get('/', (req, res) => {

// res.send('Hello World!');

res.render("index", {bodyTitle: "Página inicial", bodyContent: "Seja bem-vindo ao nosso site!"});

});

app.get('/services', (req, res) => {

// res.send('Página de serviços!');

res.render("services", {bodyTitle: "Página de Serviços", bodyContent: "Oferecemos os melhores serviços aos nossos clientes!"});

});

app.get('/contact', (req, res) => {

// res.send('Página de contato!');

res.render("contact", {bodyTitle: "Página de Contato", bodyContent: "Entre em contato conosco!"});

});

// Rota não encontrada

app.use((req, res, next) => {

// res.status(404).sendFile(\_\_dirname + "/public/404.html");

res.status(404).render("404", {bodyTitle: "Página não encontrada!", bodyContent: "Essa página não existe!"});

});

app.listen(port, () =>{

console.log('Servidor rodando na porta ' + port);

});

**views\index.ejs**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>

<%= bodyTitle %>

</title>

</head>

<body>

<h1>

<%= bodyContent %>

</h1>

</body>

</html>

**views\services.ejs**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>

<%= bodyTitle %>

</title>

</head>

<body>

<h1>

<%= bodyContent %>

</h1>

</body>

</html>

**views\contact.ejs**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>

<%= bodyTitle %>

</title>

</head>

<body>

<h1>

<%= bodyContent %>

</h1>

</body>

</html>

**views\404.ejs**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>

<%= bodyTitle %>

</title>

</head>

<body>

<h1>

<%= bodyContent %>

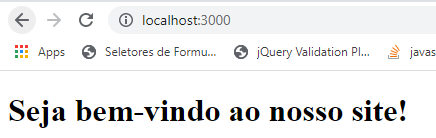
</h1>

</body>

</html>

- No browser:

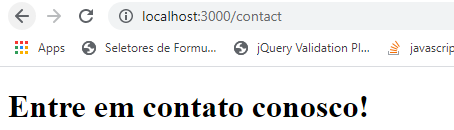
localhost:3000



localhost:3000/services



localhost:3000/contact



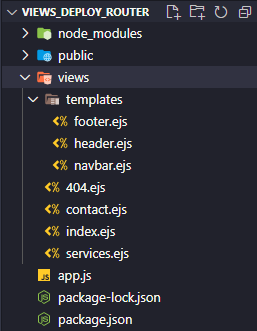
localhost:3000/contato



## Usando templates

- Dentro da pasta views adicione uma pasta chamada "templates" e dentro dela adicione os arquivos:

* header.ejs
* footer.ejs
* navbar.ejs



### Templates

**views\templates\header.ejs**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>

<%= bodyTitle %>

</title>

<link rel="stylesheet" href="/css/bootstrap.css">

</head>

<body>

<%- include("./navbar.ejs") %>

**views\templates\footer.ejs**

<div class="text-white bg-dark text-center p-3 mb-2 mt-1" role="alert">

<a style="color: yellow" href="">Orion3</a> - By: Roberto Pinheiro -

Copyright

&copy;

<?php echo date('Y'); ?> - Todos os direitos reservados.

</div>

<script src="/js/jquery.js"></script>

<script src="/js/popper.min.js"></script>

<script src="/js/bootstrap.min.js"></script>

</body>

</html>

**views\templates\navbar.ejs**

<nav class="navbar navbar-expand-md navbar-dark bg-dark">

<a class="navbar-brand" href="/">Orion3</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"

aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="/services">Serviços</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/contact">Contato</a>

</li>

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown"

aria-haspopup="true" aria-expanded="false">

Downloads

</a>

<div class="dropdown-menu" aria-labelledby="navbarDropdown">

<a class="dropdown-item" href="#">Download 1</a>

<a class="dropdown-item" href="#">Download 2</a>

<div class="dropdown-divider"></div>

<a class="dropdown-item" href="#">Outros downloads</a>

</div>

</li>

</ul>

<form class="form-inline my-2 my-lg-0">

<input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">

<button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>

</form>

</div>

</nav>

- Altere as views:

## Views

**views\index.ejs**

<%- include("templates/header", {title: "Home" }) %>

<div class="container mb-4 mt-2">

<h1 class="text-primary"><%= bodyTitle %></h1>

<h2><%= bodyContent %></h2>

</div>

<%- include("templates/footer") %>

**views\services.ejs**

<%- include("templates/header", {title: "Serviços"}) %>

<div class="container mb-4 mt-2">

<h1 class="text-success"><%= bodyTitle %></h1>

<h2><%= bodyContent %></h2>

</div>

<%- include("templates/footer") %>

**views\contact.ejs**

<%- include("templates/header", {title: "Contato" }) %>

<div class="container mb-4 mt-2">

<h1 class="text-info"><%= bodyTitle %></h1>

<h2><%= bodyContent %></h2>

</div>

<%- include("templates/footer") %>

**views\404.ejs**

<%- include("templates/header", {title: "Erro 404" }) %>

<div class="container mb-4 mt-2">

<h1 class="text-danger"><%= bodyTitle %></h1>

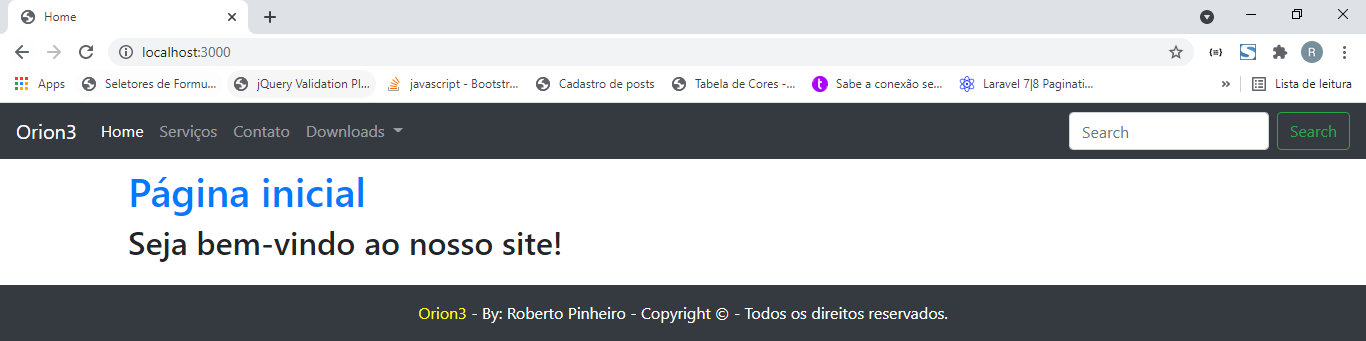
<h2><%= bodyContent %></h2>

</div>

<%- include("templates/footer") %>

- No browser:

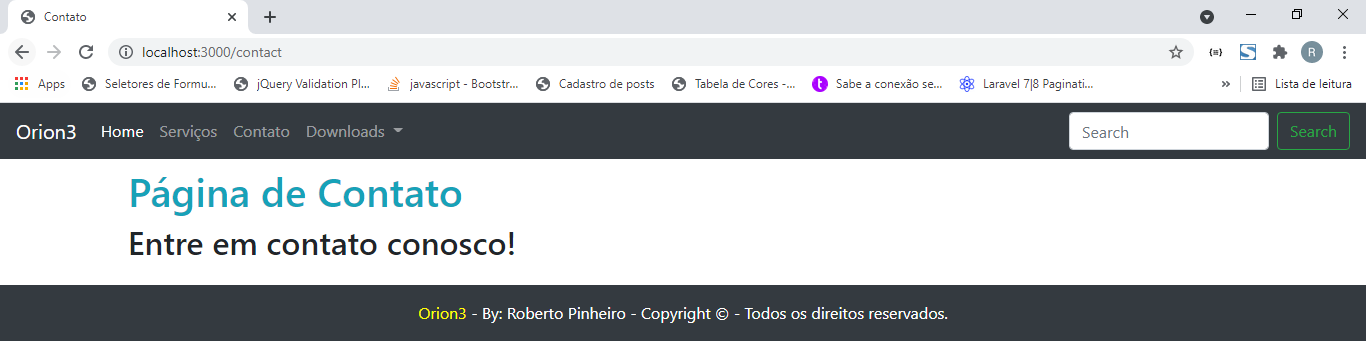
localhost:3000



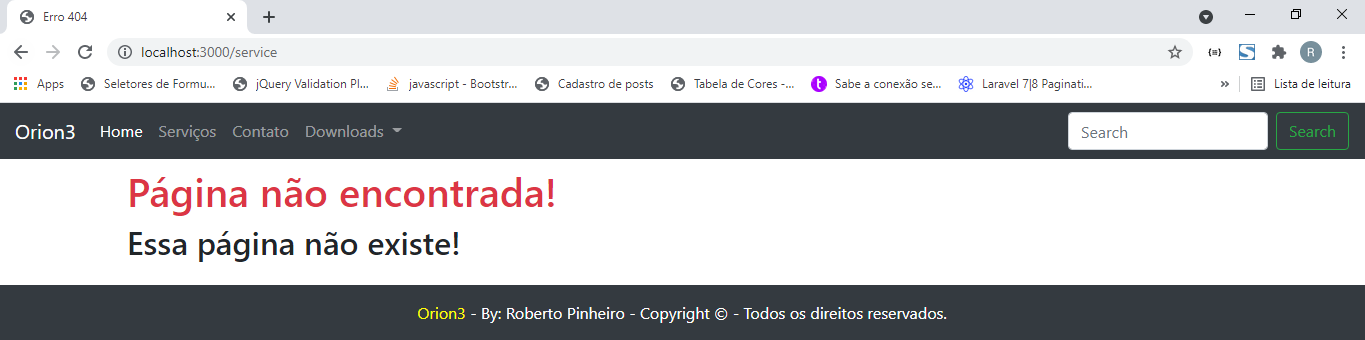
localhost:3000/services

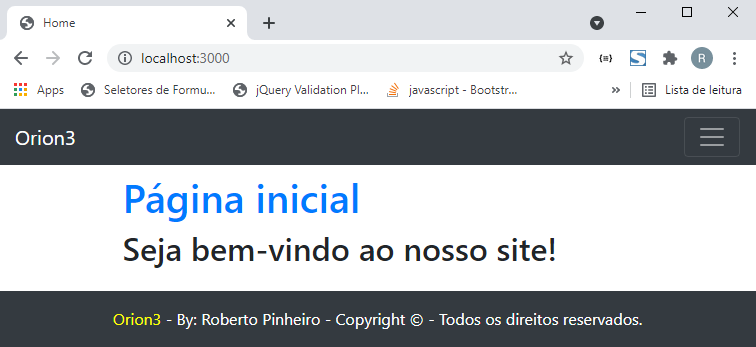


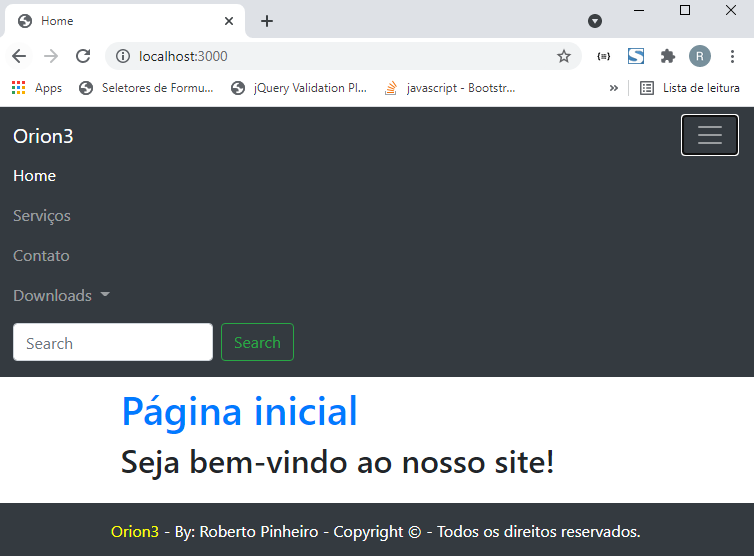
localhost:3000/contact

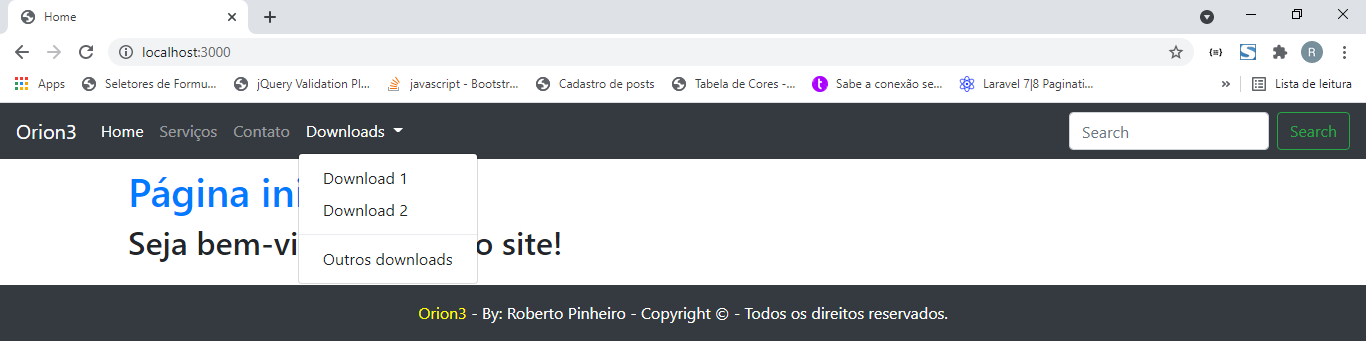


localhost:3000/service









# Aula 08 - Deploy com Heroku CLI

- Abra uma conta no Heroku.

- Instale Heroku CLI.

## Configurações

**package.json**

{

"name": "views\_deploy\_router",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "node app"

},

"keywords": [],

"author": "",

"license": "ISC",

"devDependencies": {

"nodemon": "^2.0.7"

},

"dependencies": {

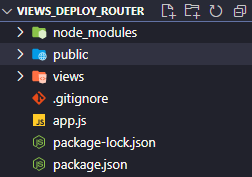
"ejs": "^3.1.6",

"express": "^4.17.1"

}

}

- Na pasta raiz do projeto, adicione o arquivo .gitignore:



**.gitignore**

node\_modules

- Altere o arquivo app.js:

**app.js**

const express = require('express');

const app = express();

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

// motor de templates

app.set('view engine', 'ejs');

app.set('views', \_\_dirname + '/views');

app.use(express.static(\_\_dirname + "/public"));

// Rotas

app.get('/', (req, res) => {

// res.send('Hello World!');

res.render("index", {bodyTitle: "Página inicial", bodyContent: "Seja bem-vindo ao nosso site!"});

});

app.get('/services', (req, res) => {

// res.send('Página de serviços!');

res.render("services", {bodyTitle: "Página de Serviços", bodyContent: "Oferecemos os melhores serviços aos nossos clientes!"});

});

app.get('/contact', (req, res) => {

// res.send('Página de contato!');

res.render("contact", {bodyTitle: "Página de Contato", bodyContent: "Entre em contato conosco!"});

});

// Rota não encontrada

app.use((req, res, next) => {

// res.status(404).sendFile(\_\_dirname + "/public/404.html");

res.status(404).render("404", {bodyTitle: "Página não encontrada!", bodyContent: "Essa página não existe!"});

});

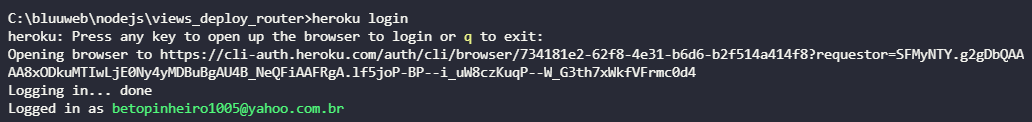
app.listen(port, () =>{

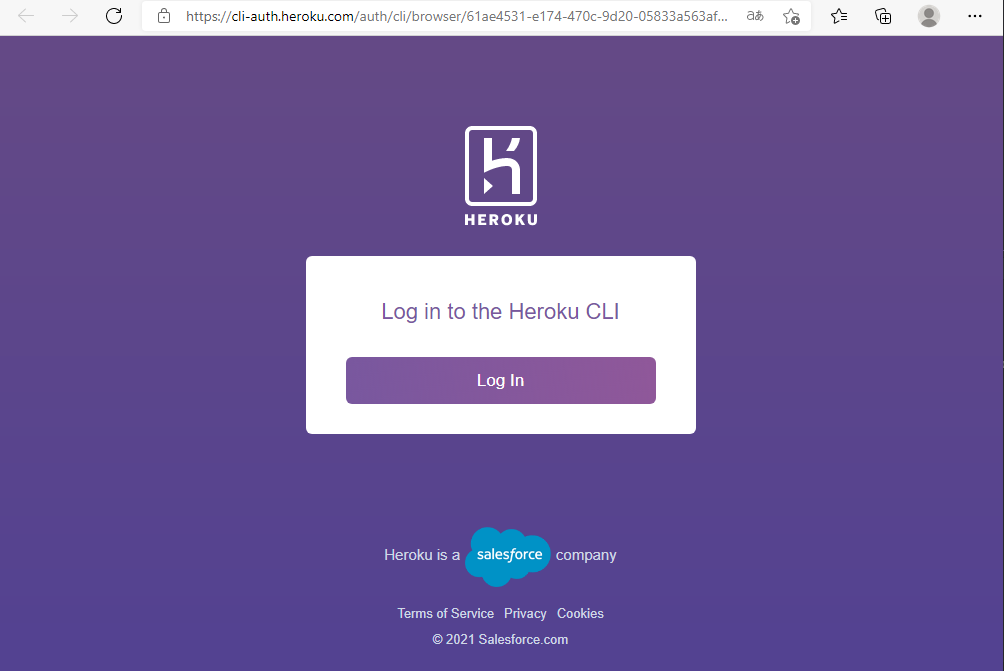
console.log('Servidor rodando na porta ' + port);

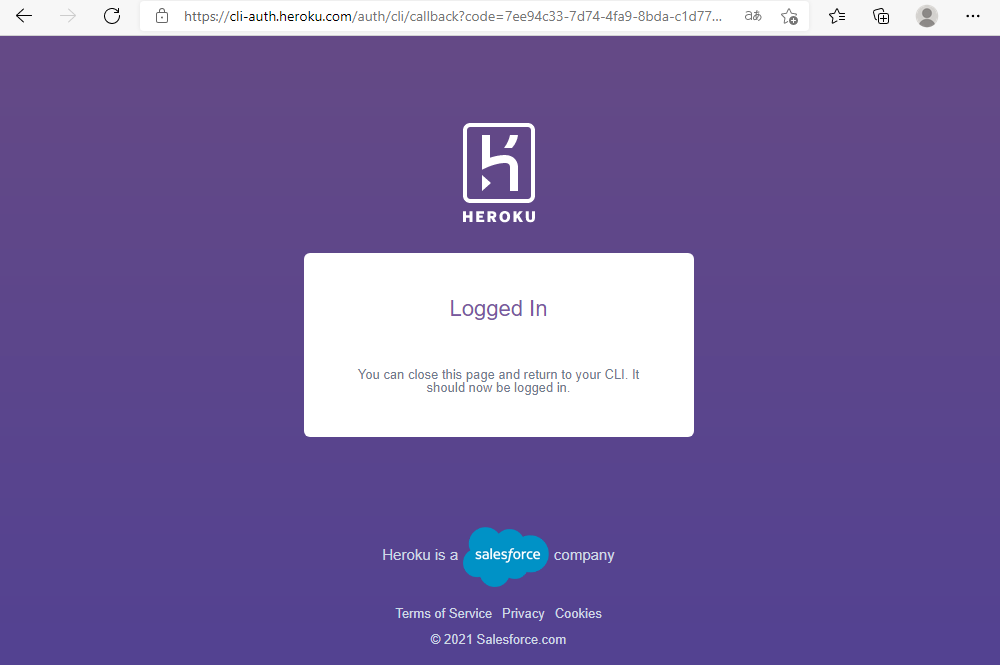
});

## Deploy

heroku login

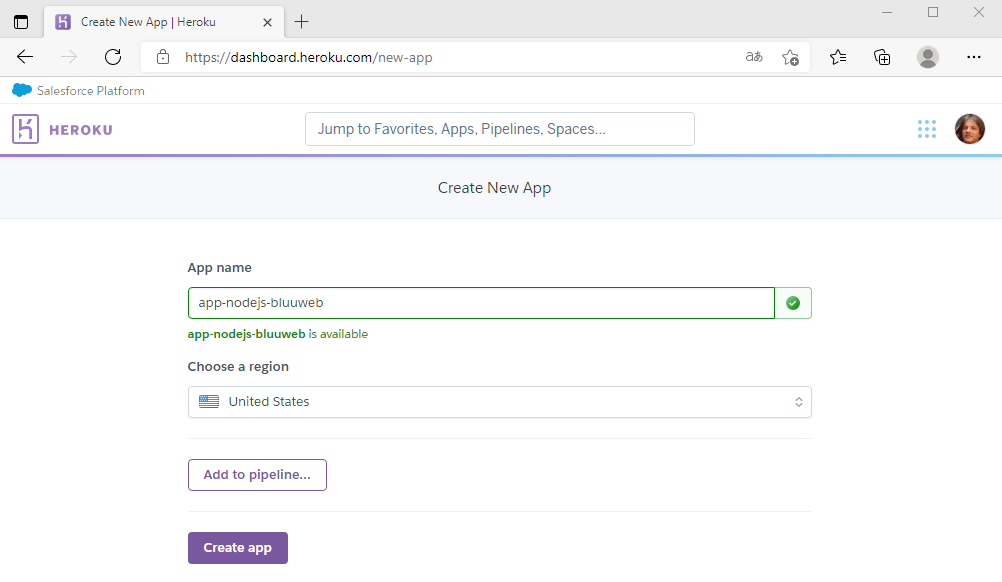


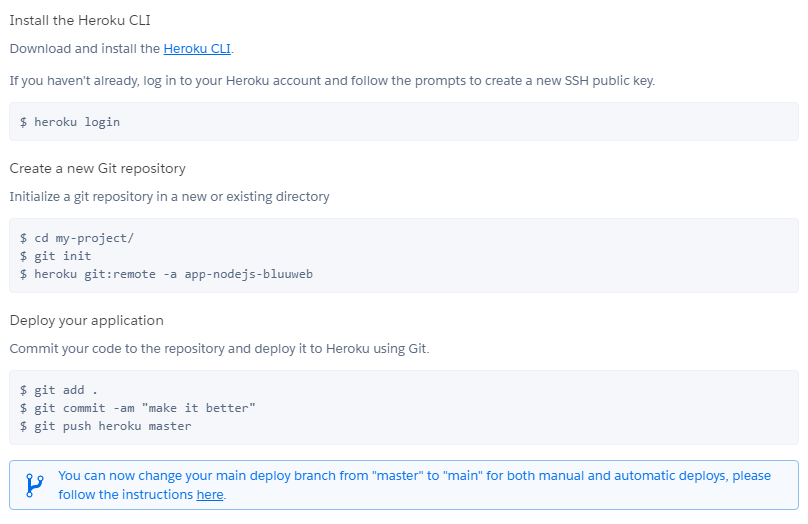




- Crie uma nova aplicação no Heroku:

app-nodejs-bluuweb





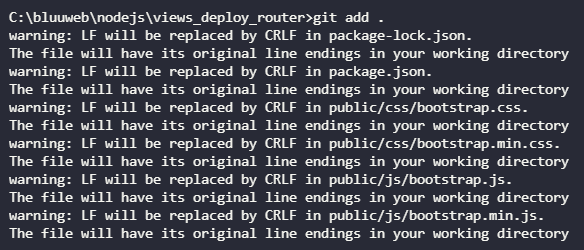
git init



heroku git:remote -a app-nodejs-bluuweb



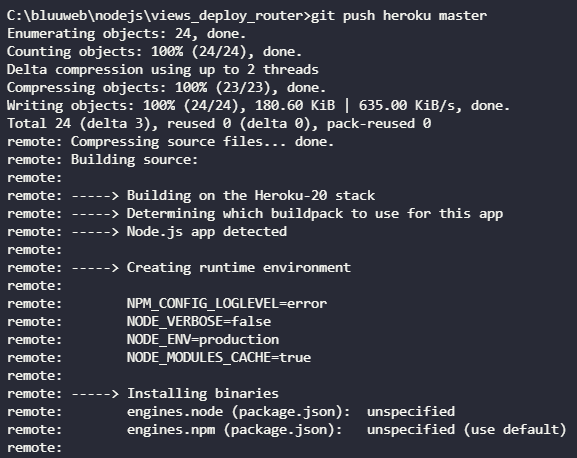
git add .



git commit -am "First commit"



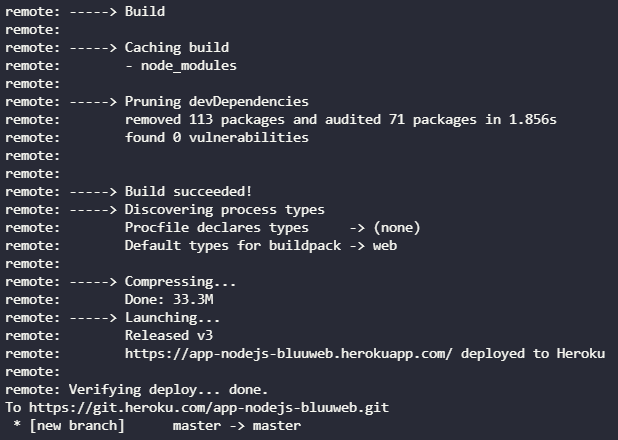
git push heroku master



.

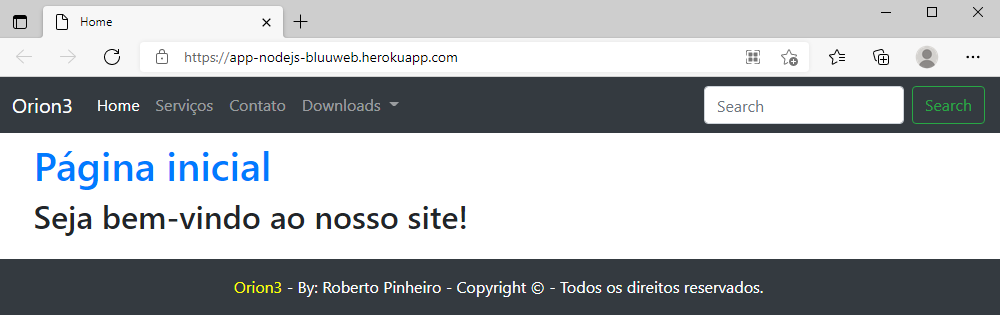
.

.

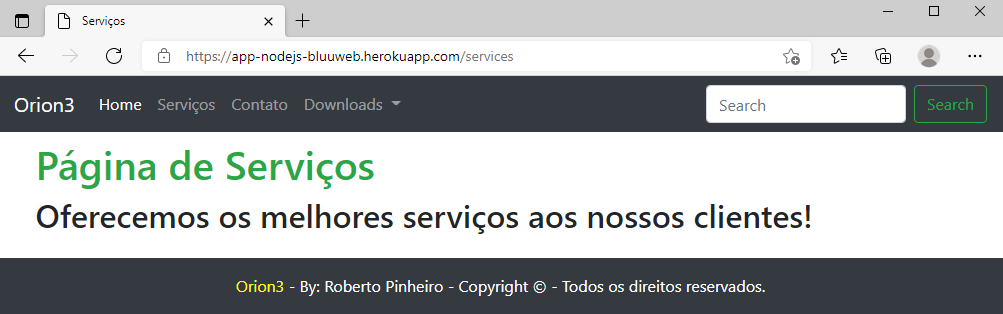


- No browser:

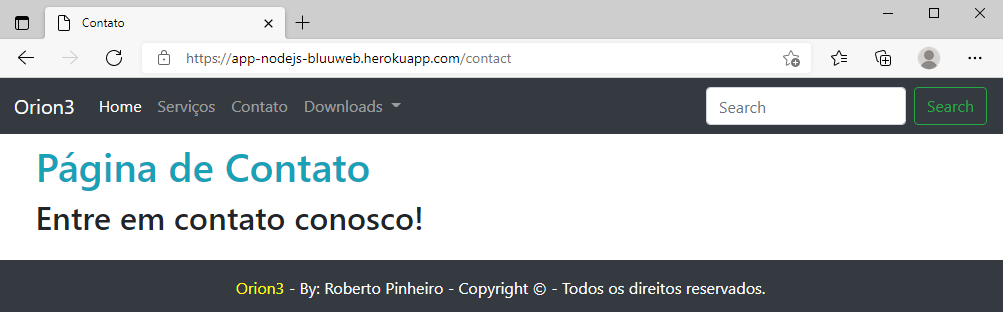
https://app-nodejs-bluuweb.herokuapp.com/



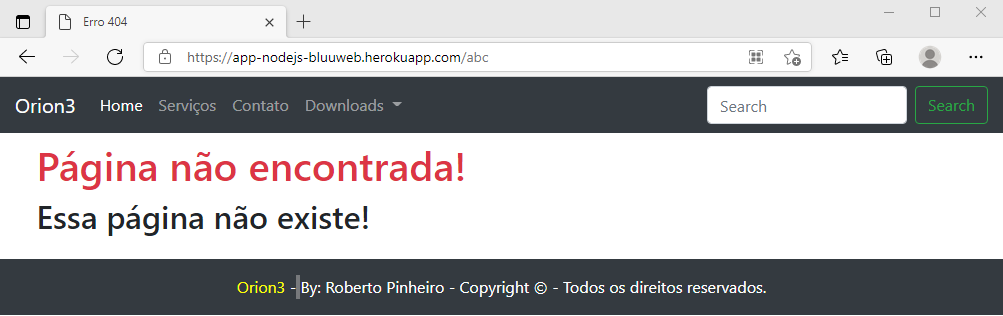
https://app-nodejs-bluuweb.herokuapp.com/services



https://app-nodejs-bluuweb.herokuapp.com/contact



https://app-nodejs-bluuweb.herokuapp.com/abc



# Aula 09 - Router Express

**package.json**

{

"name": "views\_deploy\_router",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"start": "node app",

"dev": "nodemon app"

},

"keywords": [],

"author": "",

"license": "ISC",

"devDependencies": {

"nodemon": "^2.0.7"

},

"dependencies": {

"ejs": "^3.1.6",

"express": "^4.17.1"

}

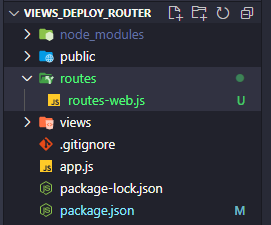
}

npm run dev



## Rotas

- Na pasta do projeto adicione uma subpasta chamada "routes" e dentro dela adicione um arquivo chamado "routes-web.js":



routes\routes-web.js

const express = require('express');

const router = express.Router();

// Rotas

router.get('/', (req, res) => {

res.render("index", {bodyTitle: "Página inicial", bodyContent: "Seja bem-vindo ao nosso site!"});

});

router.get('/services', (req, res) => {

res.render("services", {bodyTitle: "Página de Serviços", bodyContent: "Oferecemos os melhores serviços aos nossos clientes!"});

});

router.get('/contact', (req, res) => {

res.render("contact", {bodyTitle: "Página de Contato", bodyContent: "Entre em contato conosco!"});

});

module.exports = router;

- Altere o arquivo "app-js":

**app.js**

const express = require('express');

const app = express();

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

// motor de templates

app.set('view engine', 'ejs');

app.set('views', \_\_dirname + '/views');

app.use(express.static(\_\_dirname + "/public"));

// Rotas Web

app.use('/', require('./routes/routes-web'));

// Rota não encontrada

app.use((req, res, next) => {

// res.status(404).sendFile(\_\_dirname + "/public/404.html");

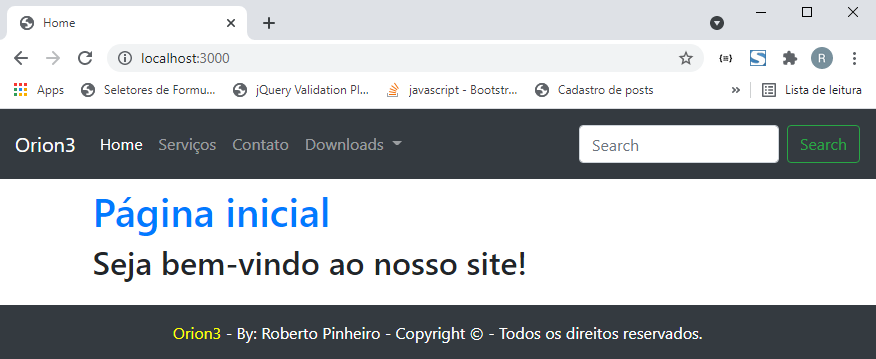
res.status(404).render("404", {bodyTitle: "Página não encontrada!", bodyContent: "Essa página não existe!"});

});

app.listen(port, () =>{

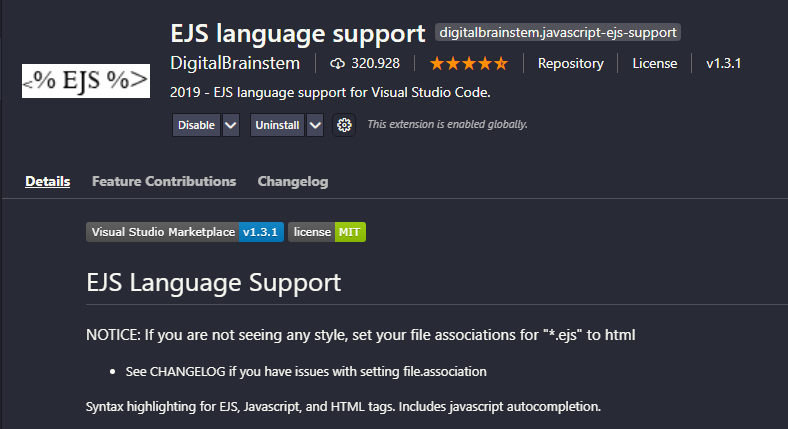
console.log('Servidor rodando na porta ' + port);

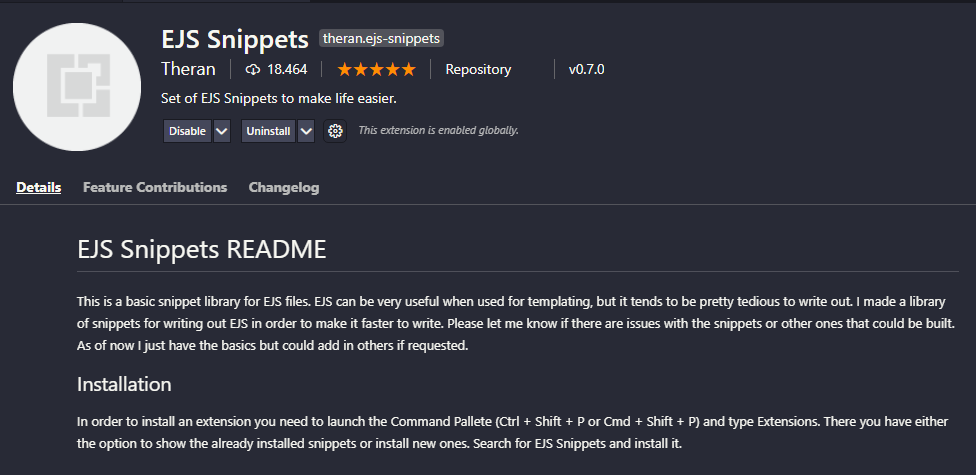
});



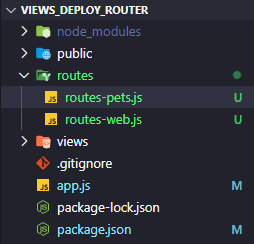
- Instale duas extensões no Visual Studio Code:

* EJS language support
* EJS Snippets

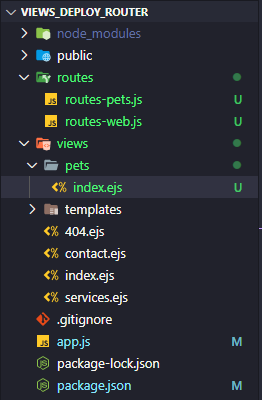




- Dentro da pasta "routes" e adicione um arquivo chamado "routes-pets.js":



- Dentro da pasta views adicione a subpasta pets e dentro dela o arquivo index.ejs:



**routes\routes-pets.js**

const express = require('express');

const router = express.Router();

// Rotas

router.get('/', (req, res) => {

res.render("pets/index", {

arrayPets: [

{id: 1, nome: "Brisa", especie: "gato", sexo: 'F'},

{id: 2, nome: "Fred", especie: "gato", sexo: 'M'},

{id: 3, nome: "Teo", especie: "gato", sexo: 'M'},

{id: 4, nome: "Benjamin", especie: "gato", sexo: 'M'},

{id: 5, nome: "Tiquinho", especie: "gato", sexo: 'M'},

{id: 6, nome: "Siloeh", especie: "cachorro", sexo: 'F'},

{id: 7, nome: "Valentim", especie: "cachorro", sexo: 'M'},

{id: 8, nome: "Roque", especie: "cachorro", sexo: 'M'}

]

});

});

module.exports = router;

**app.js**

const express = require('express');

const app = express();

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

// motor de templates

app.set('view engine', 'ejs');

app.set('views', \_\_dirname + '/views');

app.use(express.static(\_\_dirname + "/public"));

// Rotas Web

app.use('/', require('./routes/rotasWeb'));

app.use('/pets', require('./routes/routes-pets'));

// Rota não encontrada

app.use((req, res, next) => {

// res.status(404).sendFile(\_\_dirname + "/public/404.html");

res.status(404).render("404", {bodyTitle: "Página não encontrada!", bodyContent: "Essa página não existe!"});

});

app.listen(port, () =>{

console.log('Servidor rodando na porta ' + port);

});

**views\pets\index.ejs**

<%- include("../templates/header", {bodyTitle: "Home" }) %>

<div class="container mb-4 mt-2">

<table class="table">

<thead>

<tr>

<th scope="col">Id</th>

<th scope="col">Nome</th>

<th scope="col">Espécie</th>

<th scope="col">Sexo</th>

<th scope="col">Ações</th>

</tr>

</thead>

<tbody>

<% arrayPets.forEach(pet => { %>

<tr>

<th scope="row"><%= pet.id %> </th>

<td><%= pet.nome %> </td>

<td><%= pet.especie %> </td>

<td><%= pet.sexo %> </td>

<td></td>

</tr>

<% }) %>

</tbody>

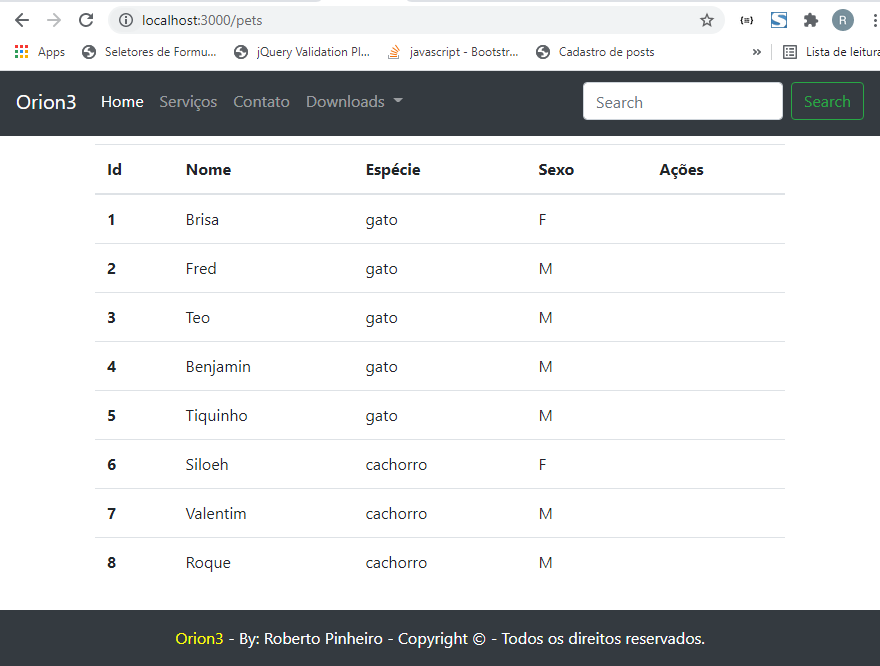
</table>

</div>

<%- include("../templates/footer") %>

- No browser:

http://localhost:3000/pets



# Aula 10 - Conexão a MongoDB

- Em C:\bluuweb\nodejs crie uma pasta chamada "mongodb"



cd mongodb

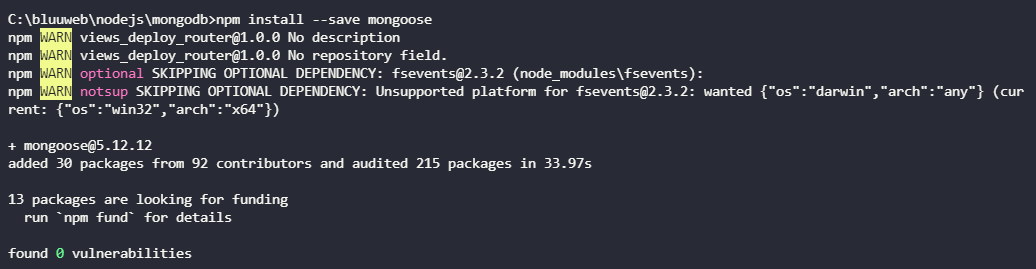


- Copie todo o conteúdo da pasta C:\bluuweb\nodejs\views\_deploy\_router para dentro da pasta C:\bluuweb\nodejs\mongodb

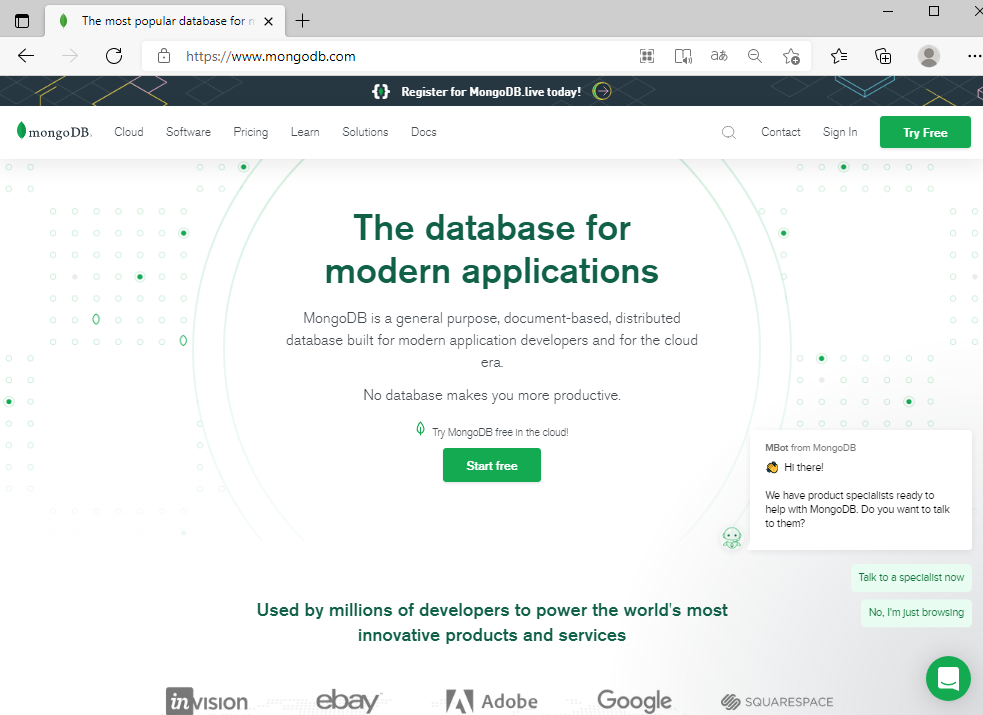
code .

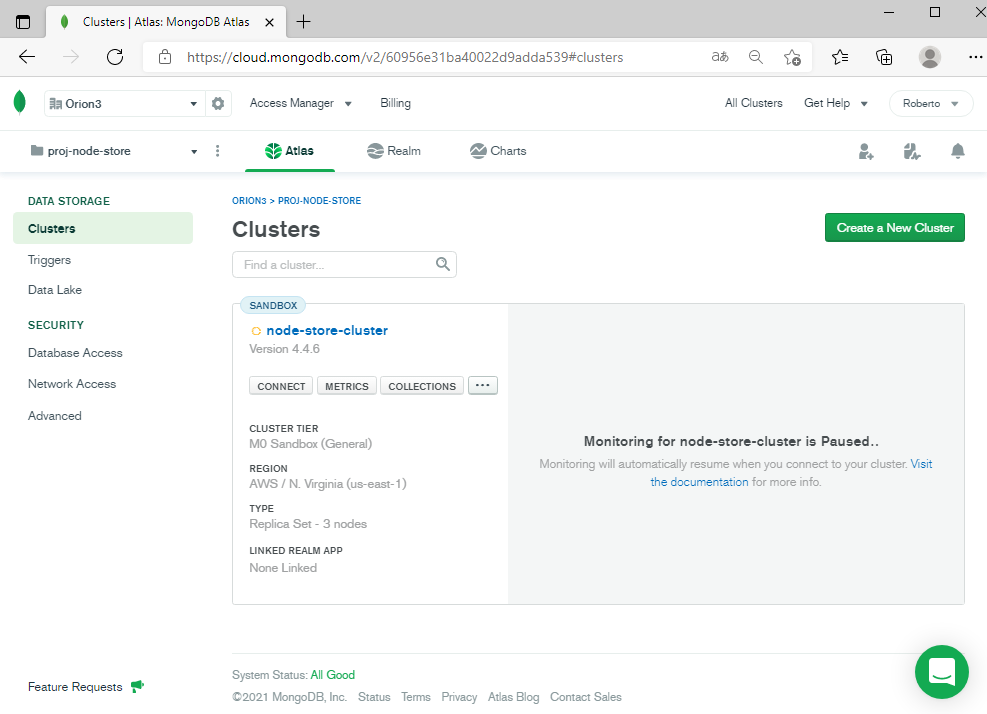
## Instalando Mongoose

npm install --save mongoose

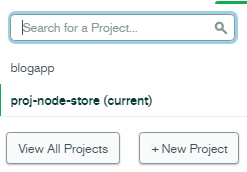


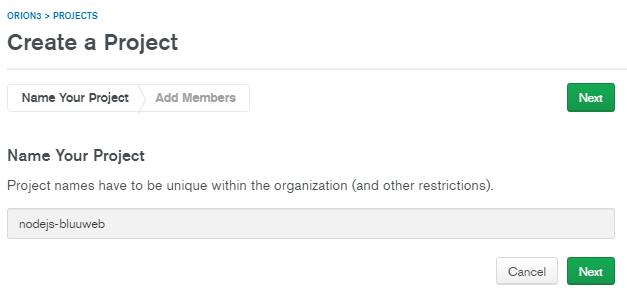
<https://www.mongodb.com/>

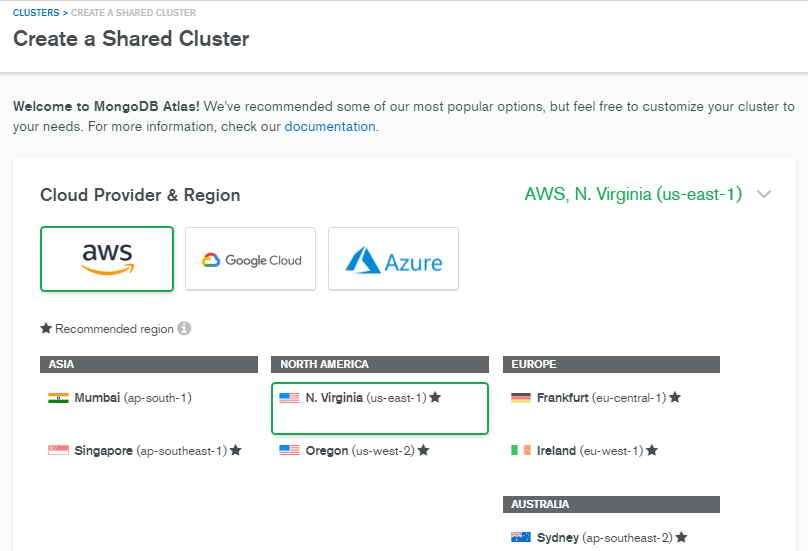


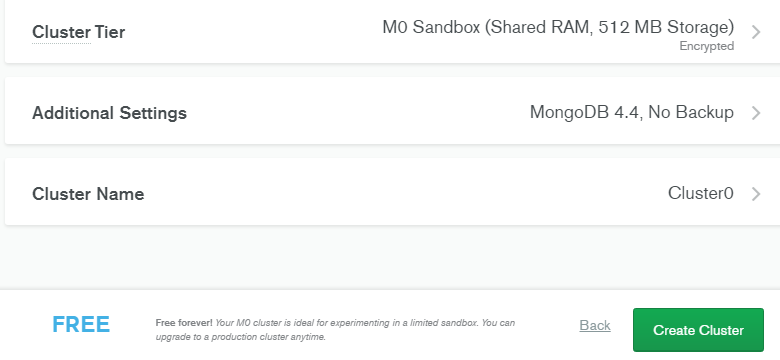


- Crie um projeto chamado "nodejs-bluuweb":

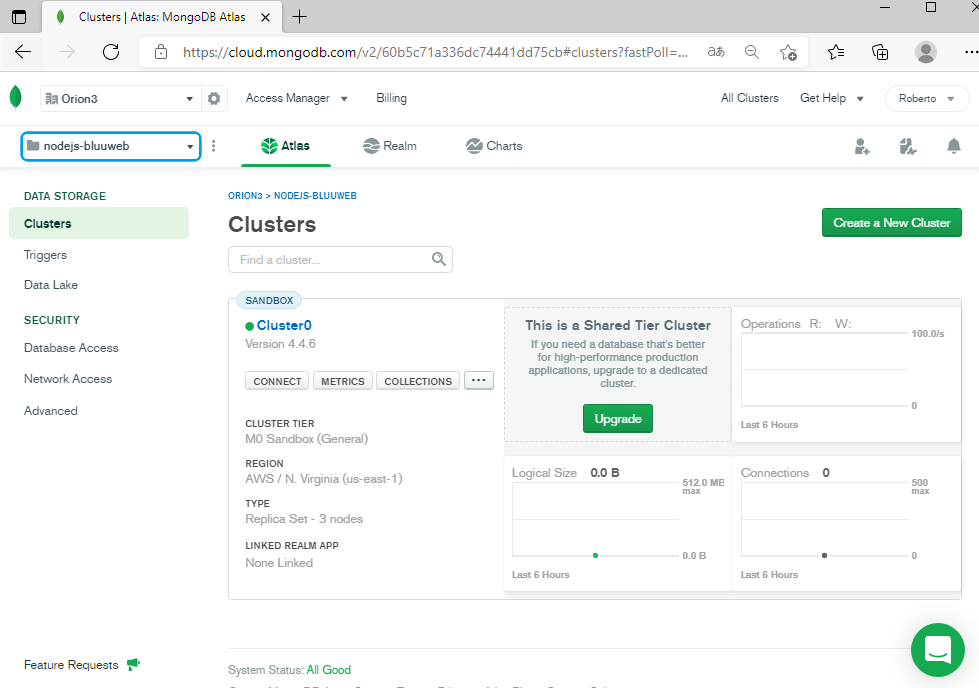




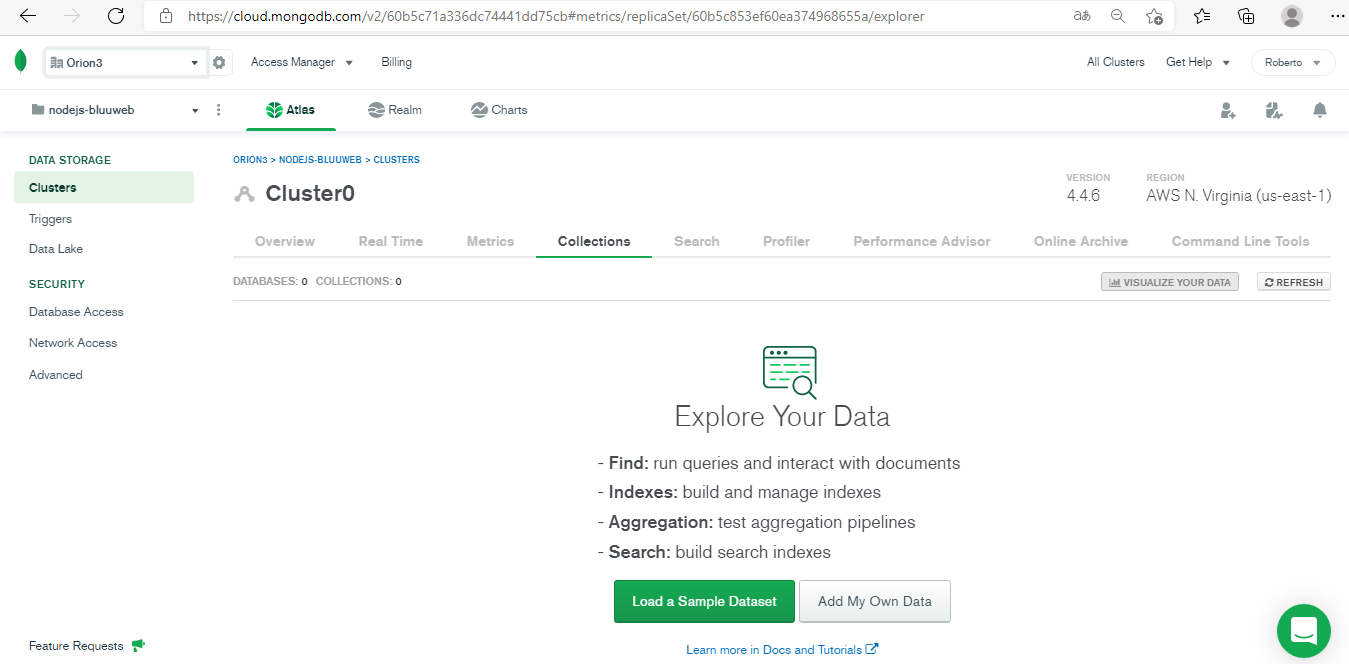




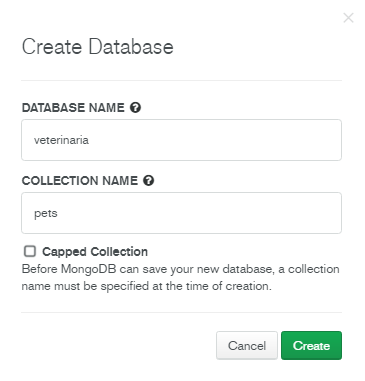
- Clique no botão "Create Cluster" (FREE) e aguarde a criação (o processo leva alguns minutos):



- Clique no botão "COLLECTIONS":



- Clique no botão "Add My Own Data":

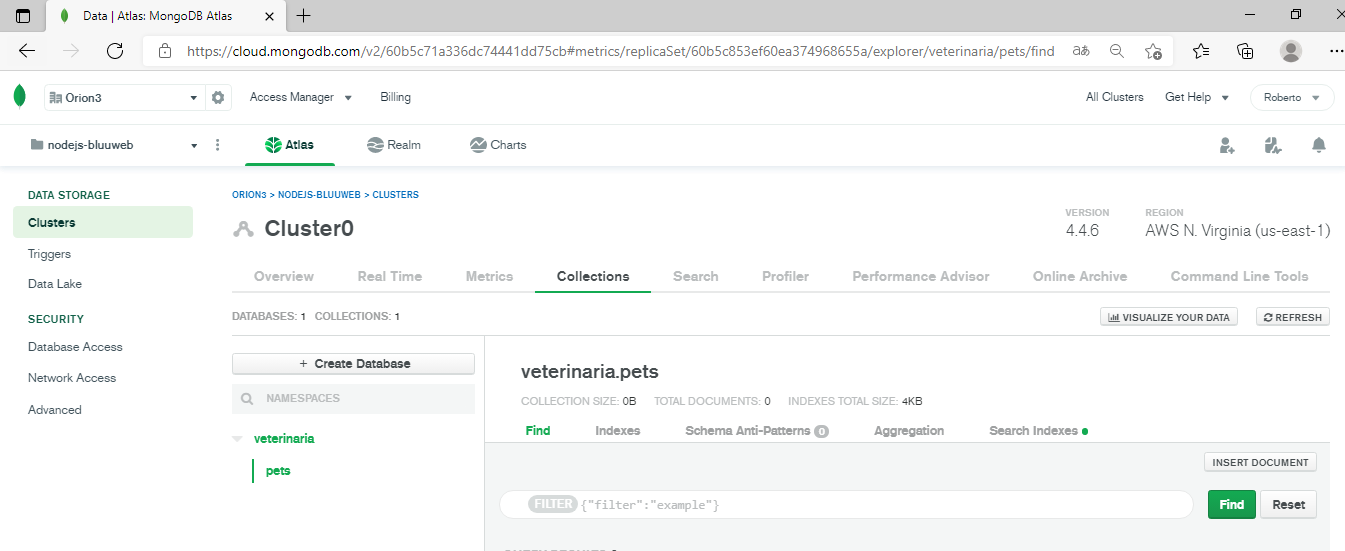


- Entre com:

DATABASE NAME: veterinaria

COLLECTION NAME: pets

- Clique no botão "Create"



- Clique no botão "INSERT DOCUMENT" e insira o primeiro pet (campos: nome, especie, sexo com os respectivos valores).

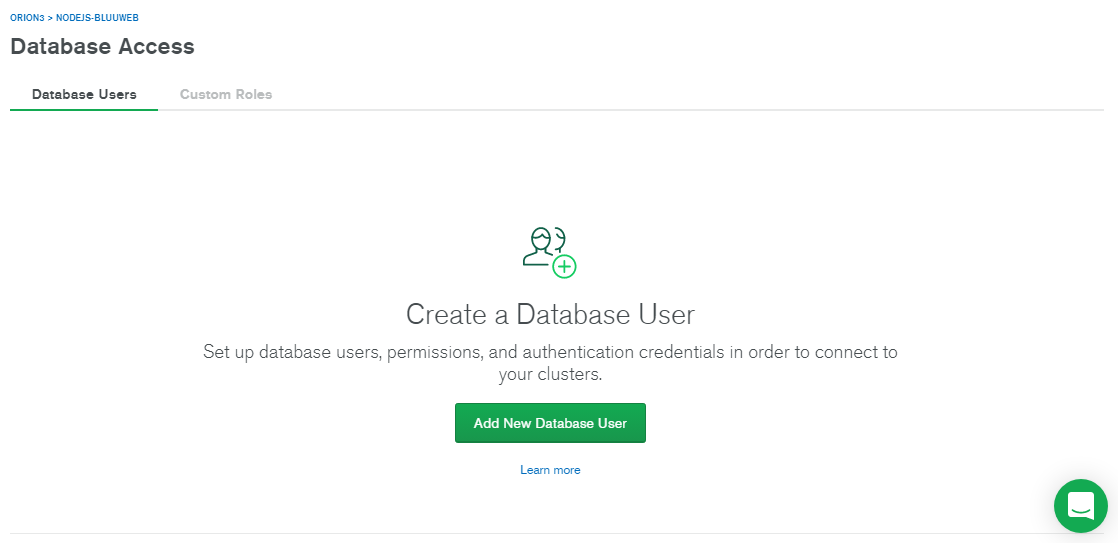


- Repita a operação para todos os demais pets.

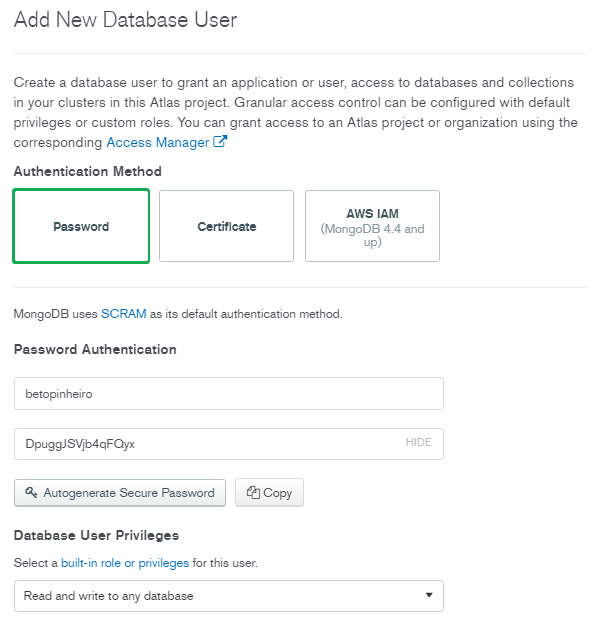




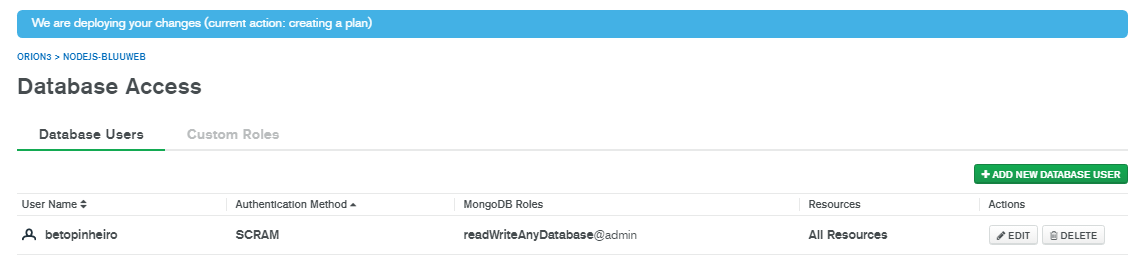
- Em "SECURITY" clique em "Database access"



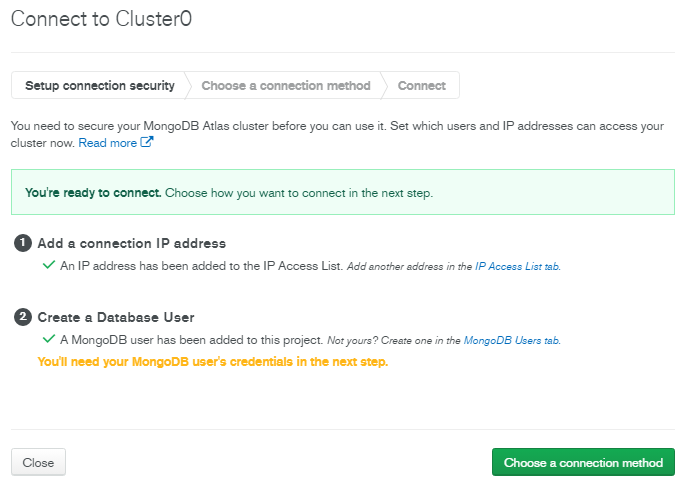
- Clique no botão "Add New Database User" e entre com o nome do usuário e a senha:



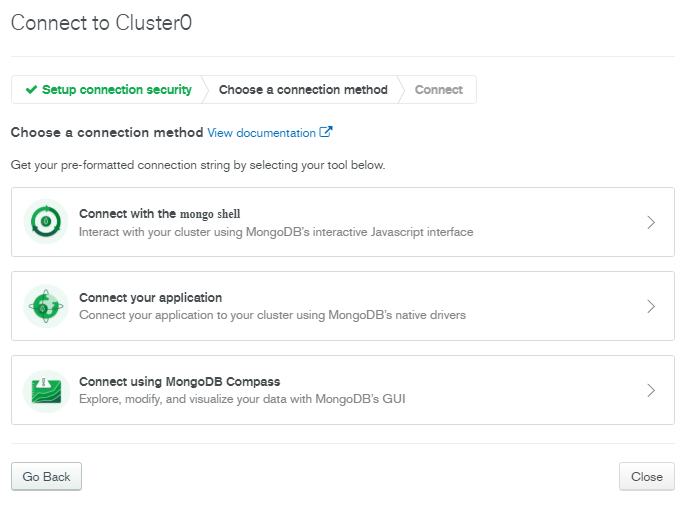
- Clique no botão "Add User"



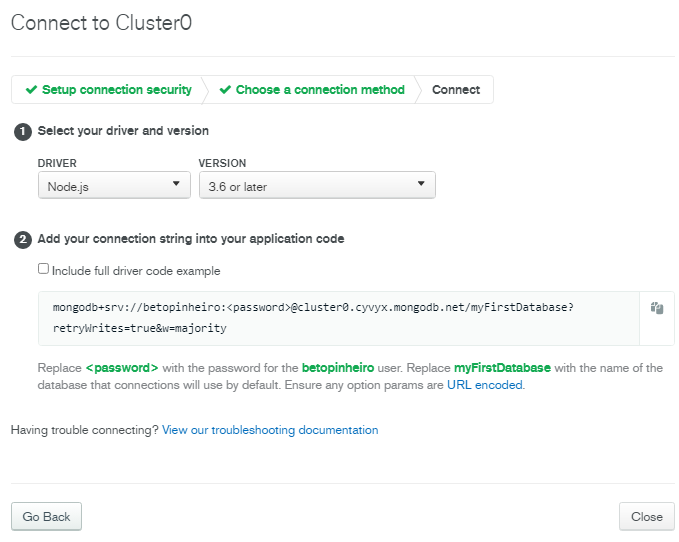
- Em Cluster0, clique no botão "CONNECT":



- Clique no botão "Choose a connection method":



- Clique em "Connect your application"



- Copie a string de conexão, substituindo <passoword> por DpuggJSVjb4qFQyx e myFirstDatabase por veterinaria.

mongodb+srv://betopinheiro:DpuggJSVjb4qFQyx@cluster0.cyvyx.mongodb.net/veterinaria?retryWrites=true&w=majority

- Essa é a string a ser utilizada na configuração de conexão ao BD.

**app.js**

const express = require('express');

const app = express();

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

const mongoose = require('mongoose');

// Conexão a base de dados

const user = "betopinheiro";

const password = "DpuggJSVjb4qFQyx";

const dbname = "veterinaria";

const uri = `mongodb+srv://${user}:${password}@cluster0.cyvyx.mongodb.net/${dbname}?retryWrites=true&w=majority`;

mongoose.connect(uri, {useNewUrlParser: true, useUnifiedTopology: true})

.then(() => {

console.log('Base de dados conectada!');

}).catch((error) => {

console.log(error);

});

// motor de templates

app.set('view engine', 'ejs');

app.set('views', \_\_dirname + '/views');

app.use(express.static(\_\_dirname + "/public"));

// Rotas Web

app.use('/', require('./routes/routes-web'));

app.use('/pets', require('./routes/routes-pets'));

// Rota não encontrada

app.use((req, res, next) => {

// res.status(404).sendFile(\_\_dirname + "/public/404.html");

res.status(404).render("404", {bodyTitle: "Página não encontrada!", bodyContent: "Essa página não existe!"});

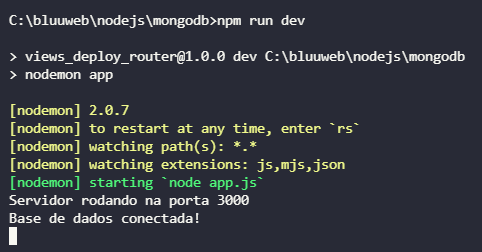
});

app.listen(port, () =>{

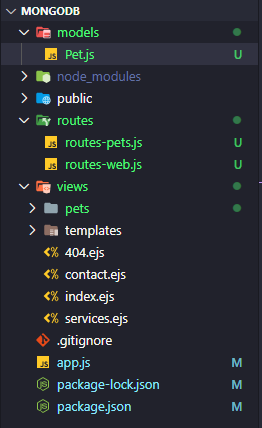
console.log('Servidor rodando na porta ' + port);

});

npm run dev



- Na pasta do projeto crie uma subpasta chamada "models" e dentro dela adicione um arquivo chamado Pet.js:



**models\Pet.js**

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const petSchema = new Schema({

nome: String,

especie: String,

sexo: String

});

// Criar model

const Pet = mongoose.model('Pet', petSchema);

module.exports = Pet;

**routes\routes-pets.js**

const express = require('express');

const router = express.Router();

const Pet = require('../models/Pet');

// Rotas

router.get('/', async (req, res) => {

try {

const arrayPetsDB = await Pet.find();

console.log(arrayPetsDB);

} catch (error) {

console.log(error);

}

res.render("pets/index", {

arrayPets: [

{id: 1, nome: "Brisa", especie: "gato", sexo: 'F'},

{id: 2, nome: "Fred", especie: "gato", sexo: 'M'},

{id: 3, nome: "Teo", especie: "gato", sexo: 'M'},

{id: 4, nome: "Benjamin", especie: "gato", sexo: 'M'},

{id: 5, nome: "Tiquinho", especie: "gato", sexo: 'M'},

{id: 6, nome: "Siloeh", especie: "cachorro", sexo: 'F'},

{id: 7, nome: "Valentim", especie: "cachorro", sexo: 'M'},

{id: 8, nome: "Roque", especie: "cachorro", sexo: 'M'}

]

});

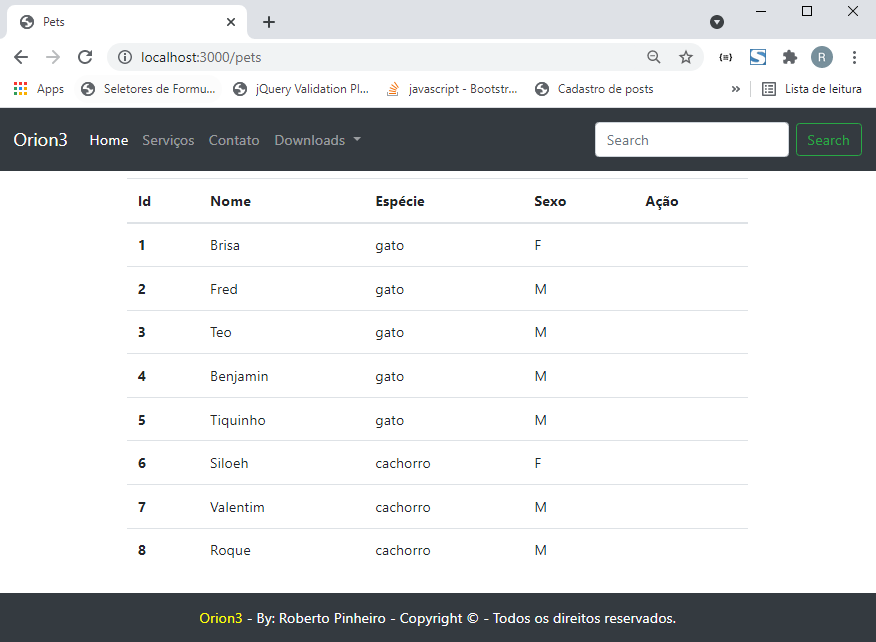
});

module.exports = router;

npm run dev

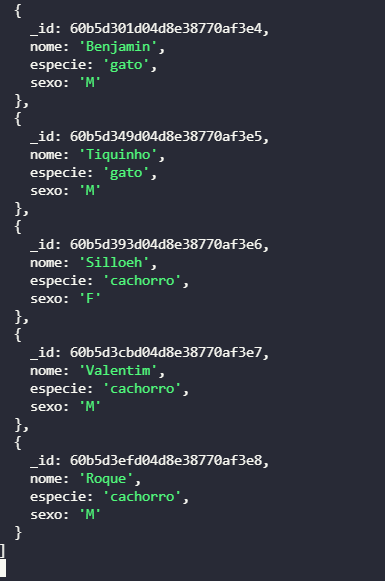


- Recarregue o browser:



- Observe no terminal:





**routes\routes-pets.js**

const express = require('express');

const router = express.Router();

const Pet = require('../models/Pet');

// Rotas

router.get('/', async (req, res) => {

try {

const arrayPetsDB = await Pet.find();

console.log(arrayPetsDB);

res.render("pets/index", {arrayPets: arrayPetsDB});

} catch (error) {

console.log(error);

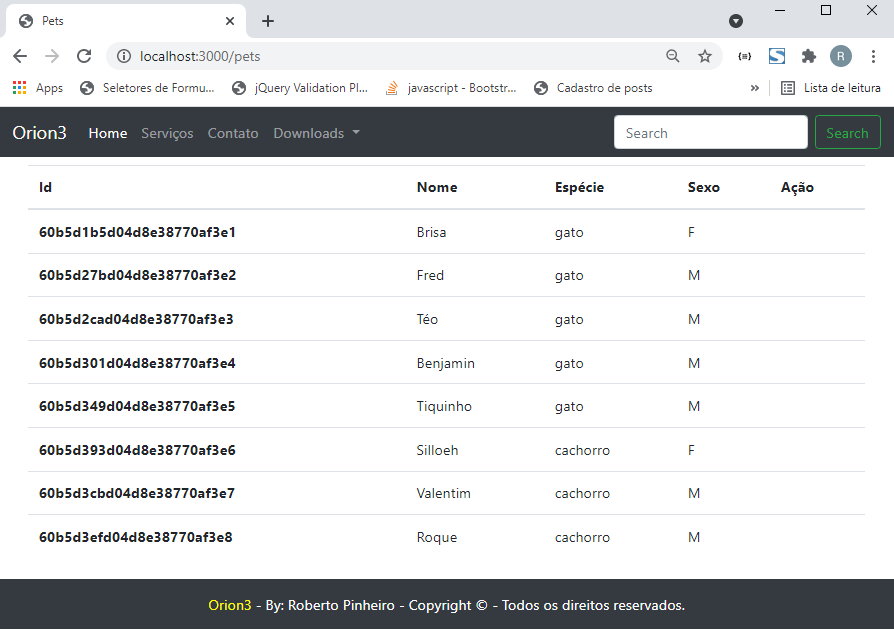
}

});

module.exports = router;

- No browser:

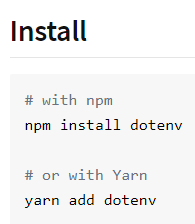
http://localhost:3000/pets



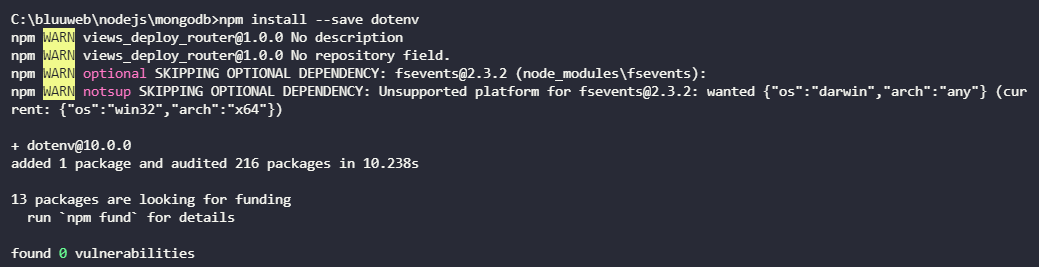
# Aula 11 - Variáveis de ambiente - dotenv (.env)

## Instalando dotenv

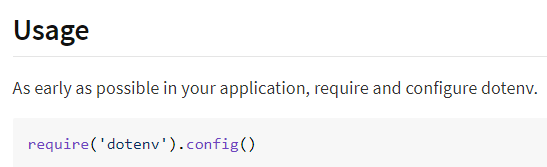
<https://www.npmjs.com/package/dotenv>



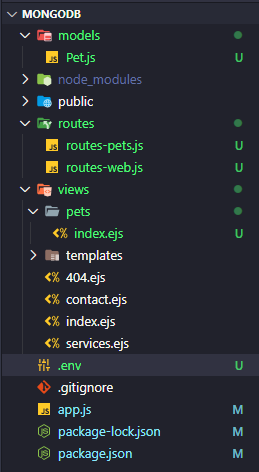
npm install --save dotenv



### Uso



- Na pasta do projeto, adicione um arquivo chamado .env:



**.env**

PORT=3002

USER=betopinheiro

PASSWORD=DpuggJSVjb4qFQyx

DBNAME=veterinaria

**app.js**

const express = require('express');

const app = express();

const mongoose = require('mongoose');

require('dotenv').config();

// Conexão a base de dados

const uri = `mongodb+srv://${process.env.USER}:${process.env.PASSWORD}@cluster0.cyvyx.mongodb.net/${process.env.DBNAME}?retryWrites=true&w=majority`;

mongoose.connect(uri, {useNewUrlParser: true, useUnifiedTopology: true})

.then(() => {

console.log('Base de dados conectada!');

}).catch((error) => {

console.log(error);

});

// motor de templates

app.set('view engine', 'ejs');

app.set('views', \_\_dirname + '/views');

app.use(express.static(\_\_dirname + "/public"));

// Rotas Web

app.use('/', require('./routes/rotasWeb'));

app.use('/pets', require('./routes/routes-pets'));

// Rota não encontrada

app.use((req, res, next) => {

// res.status(404).sendFile(\_\_dirname + "/public/404.html");

res.status(404).render("404", {bodyTitle: "Página não encontrada!", bodyContent: "Essa página não existe!"});

});

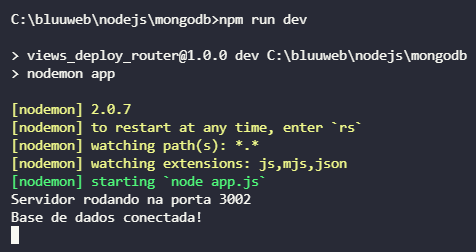
const port = process.env.PORT || 3000;

app.listen(port, () =>{

console.log('Servidor rodando na porta ' + port);

});

npm run dev



**.gitignore**

node\_modules

.env

## Configurando as variáveis de ambiente no Heroku

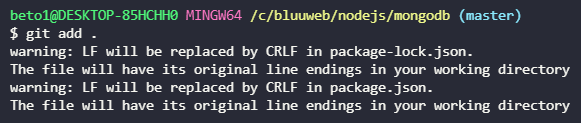
<https://dashboard.heroku.com/apps/app-nodejs-bluuweb/settings>



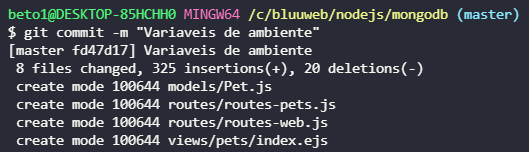
## Deploy - Atualizando

heroku login

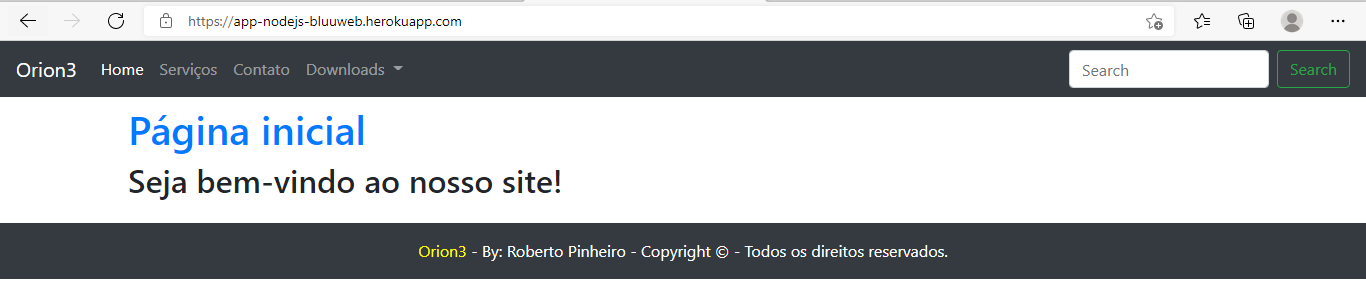
git add .



git commit -m "Variaveis de ambiente"

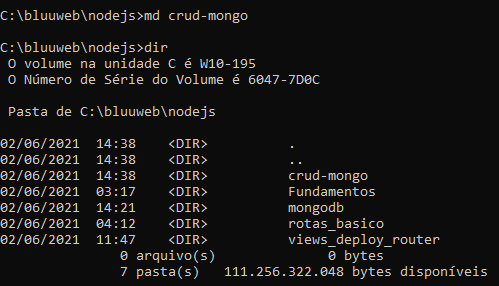


git push heroku master

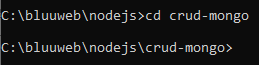


# Aula 12 - CRUD - Criar documentos em MongoDB

- Em C:\bluuweb\nodejs crie uma pasta chamada "crud-mongo"



cd mongodb

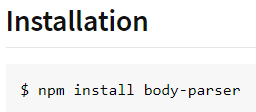


- Copie todo o conteúdo da pasta C:\bluuweb\nodejs\mongodb para dentro da pasta C:\bluuweb\nodejs\crud-mongo

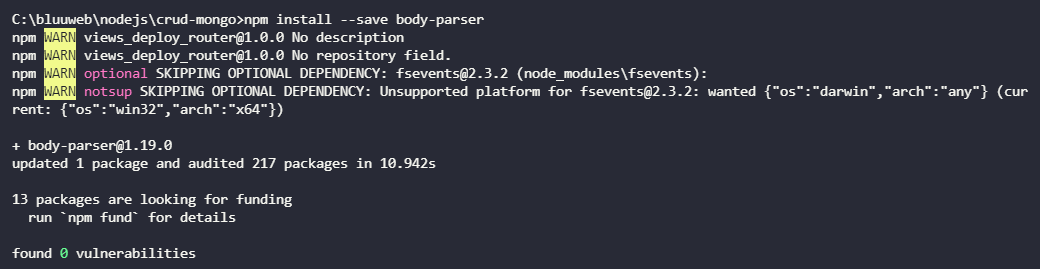
code .

## Instalando body-parser

<https://www.npmjs.com/package/body-parser>



npm install --save body-parser



### Uso

const bodyParser = require('body-parser')

### Configurações

**app.js**

const express = require('express');

const app = express();

const mongoose = require('mongoose');

require('dotenv').config();

const bodyParser = require('body-parser')

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

// parse application/x-www-form-urlencoded

app.use(bodyParser.urlencoded({ extended: false }))

// parse application/json

app.use(bodyParser.json())

// Conexão a base de dados

const uri = `mongodb+srv://${process.env.USER}:${process.env.PASSWORD}@cluster0.cyvyx.mongodb.net/${process.env.DBNAME}?retryWrites=true&w=majority`;

mongoose.connect(uri, {useNewUrlParser: true, useUnifiedTopology: true})

.then(() => {

console.log('Base de dados conectada!');

}).catch((error) => {

console.log(error);

});

// motor de templates

app.set('view engine', 'ejs');

app.set('views', \_\_dirname + '/views');

app.use(express.static(\_\_dirname + "/public"));

// Rotas Web

app.use('/', require('./routes/routes-web'));

app.use('/pets', require('./routes/routes-pets'));

// Rota não encontrada

app.use((req, res, next) => {

// res.status(404).sendFile(\_\_dirname + "/public/404.html");

res.status(404).render("404", {bodyTitle: "Página não encontrada!", bodyContent: "Essa página não existe!"});

});

app.listen(port, () =>{

console.log('Servidor rodando na porta ' + port);

});

**routes\routes-pets.js**

const express = require('express');

const router = express.Router();

const Pet = require('../models/Pet');

// Rotas

router.get('/', async (req, res) => {

try {

const arrayPetsDB = await Pet.find();

console.log(arrayPetsDB);

res.render("pets", {arrayPets: arrayPetsDB});

} catch (error) {

console.log(error);

}

});

router.get('/create', (req, res) => {

res.render("pets/create");

});

router.post('/', async (req, res) => {

const body = req.body

try {

//const petDB = new Pet(body);

//await petDB.save();

await Pet.create(body);

res.redirect('/pets');

// console.log("Pet cadastrado: ", petDB);

} catch (error) {

console.log(error);

}

});

module.exports = router;

**views\pets\create.ejs**

<%- include("../templates/header", {bodyTitle: "Cadastrar um pet" }) %>

<div class="container mb-4 mt-2">

<div class="row">

<div class="col-md-2"></div>

<div class="col-md-8">

<h1 class="text-center">Cadastrar Pet</h1>

<form action="/pets" method="POST">

<label for="nome">Nome:</label>

<input type="text" name="nome" placeholder="Digite o nome do Pet" class="form-control my-2"><br />

<label for="especie">Espécie:</label>

<select name="especie" class="form-control">

<option value="cachorro">Cachorro</option>

<option value="gato">Gato</option>

<option value="coelho">Coelho</option>

<option value="hamster">Hamster</option>

</select><br />

<label for="sexo">Sexo:</label>

<select name="sexo" class="form-control">

<option value="M">Macho</option>

<option value="F">Fêmea</option>

</select><br /><br />

<button class="btn btn-success btn-block" type="submit">

Cadastrar

</button>

</form>

</div>>

<div class="col-md-2"></div>

</div>

</div>

<%- include("../templates/footer") %>

**views\templates\navbar.ejs**

<nav class="navbar navbar-expand-md navbar-dark bg-dark">

<a class="navbar-brand" href="/">Orion3</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"

aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav mr-auto">

<li class="nav-item active">

<a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>

</li>

<li class="nav-item">

<a class="nav-link" href="/services">Serviços</a>

</li>

<li class="nav-item">

<a class="nav-link" href="/contact">Contato</a>

</li>

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown"

aria-haspopup="true" aria-expanded="false">

Pets

</a>

<div class="dropdown-menu" aria-labelledby="navbarDropdown">

<a class="dropdown-item" href="/pets">Listar</a>

<a class="dropdown-item" href="/pets/create">Cadastrar</a>

</div>

</li>

</ul>

<form class="form-inline my-2 my-lg-0">

<input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">

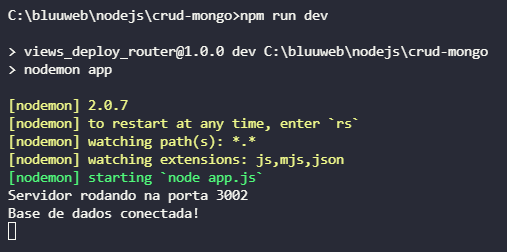
<button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>

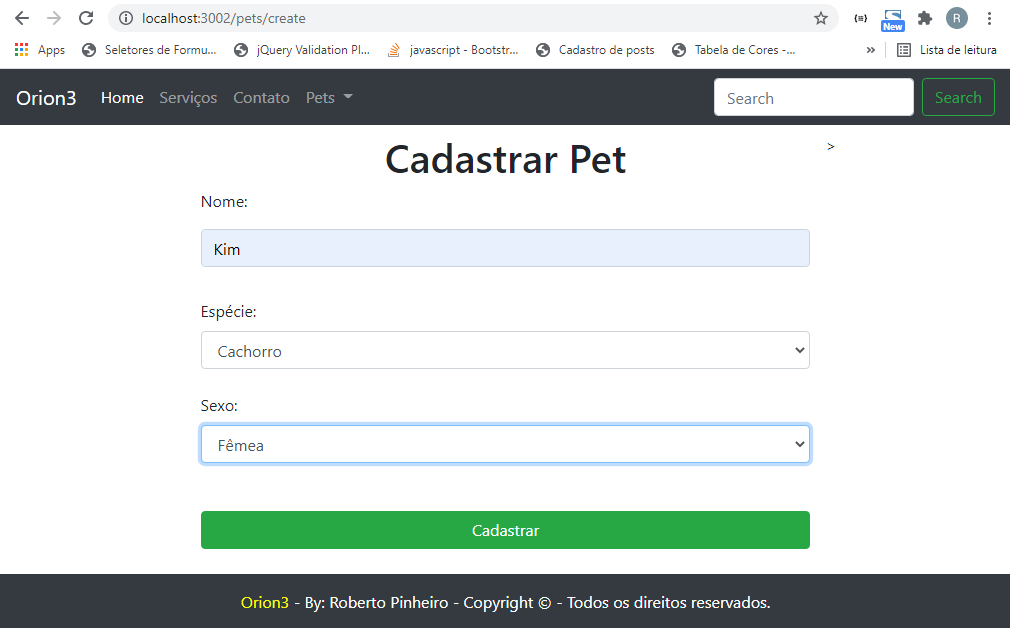
</form>

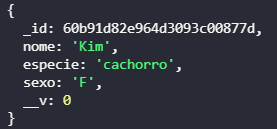
</div>

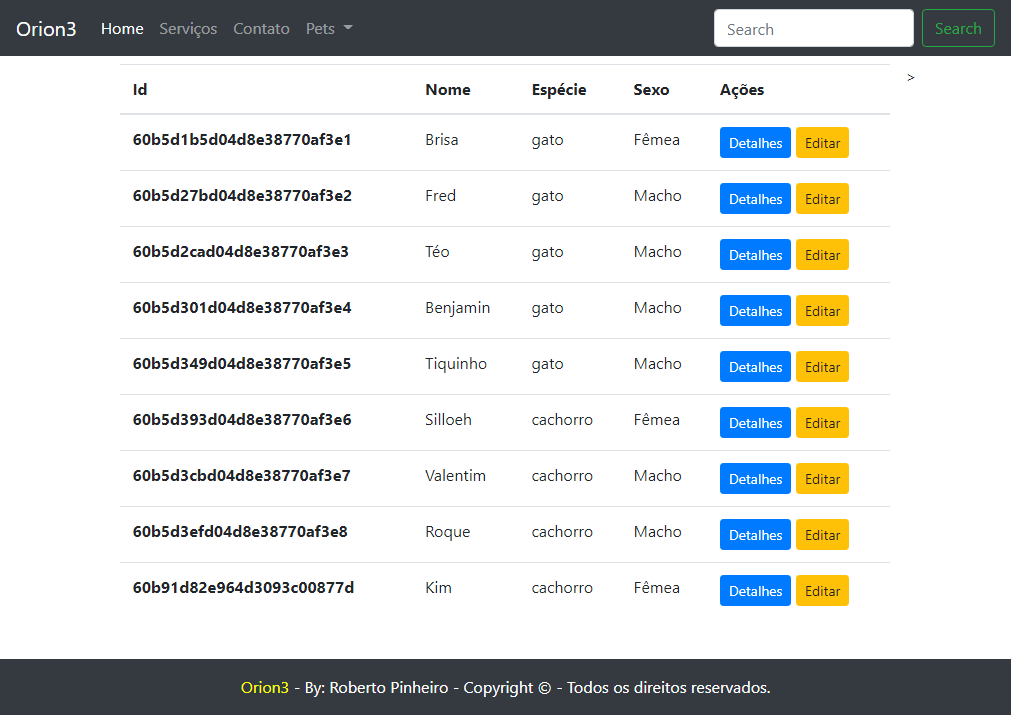
</nav>

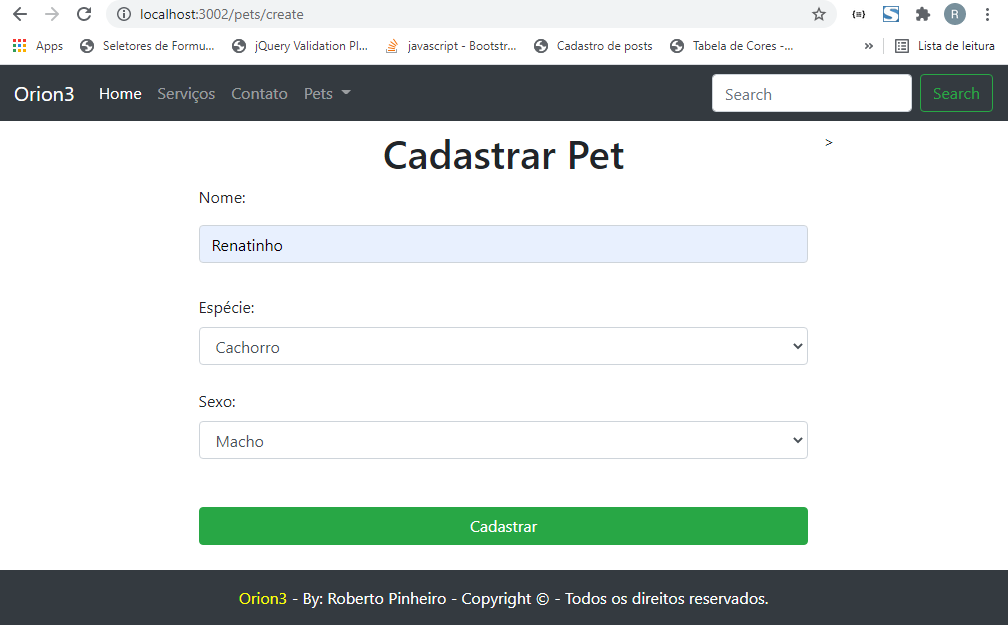
npm run dev

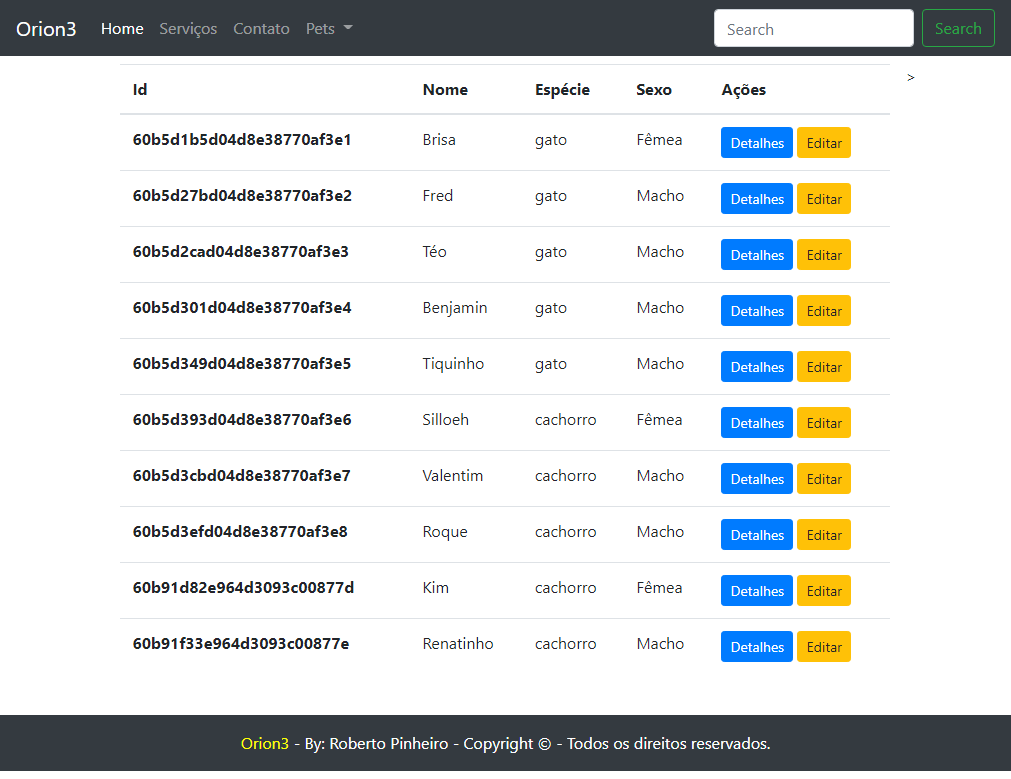


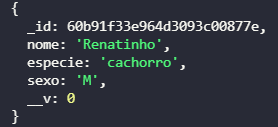












# Aula 13 - CRUD - Exibir detalhes de um documento em MongoDB

**routes\routes-pets.js**

const express = require('express');

const router = express.Router();

const Pet = require('../models/Pet');

// Rotas

router.get('/', async (req, res) => {

try {

const arrayPetsDB = await Pet.find();

console.log(arrayPetsDB);

res.render("pets/index", {arrayPets: arrayPetsDB});

} catch (error) {

console.log(error);

}

});

router.get('/create', (req, res) => {

res.render("pets/create");

});

router.post('/', async (req, res) => {

const body = req.body

try {

//const petDB = new Pet(body);

//await petDB.save();

await Pet.create(body);

res.redirect('/pets');

// console.log("Pet cadastrado: ", petDB);

} catch (error) {

console.log(error);

}

});

router.get('/:id', async (req, res) => {

const id = req.params.id

try {

const petDB = await Pet.findOne({ \_id: id });

console.log(petDB);

res.render('pets/show', {

pet: petDB,

error: false

});

} catch (error) {

console.log(error);

res.render('pets/show', {

error: true,

message: 'Não encontrado o id selecionado!'

});

}

});

module.exports = router;

**views\show.ejs**

<%- include("../templates/header", {bodyTitle: "Detalhes do pet" }) %>

<div class="container mb-4 mt-2">

<h1 class="text-center">Detalhes do pet</h1>

<% if (error) { %>

<h3 class="text-danger text-center mb-5">

<%= message %>

</h3>

<a href="/pets" class="btn btn-success btn-block">Listar Pets</a>

<% } %>

<% if (!error) { %>

<table class="table">

<thead>

<tr>

<th scope="col">Id</th>

<th scope="col">Nome</th>

<th scope="col">Espécie</th>

<th scope="col">Sexo</th>

<th scope="col">Ação</th>

</tr>

</thead>

<tbody>

<tr>

<th scope="row">

<%= pet.id %>

</th>

<td>

<%= pet.nome %>

</td>

<td>

<%= pet.especie %>

</td>

<td>

<% if (pet.sexo==='M' ) { %>

Macho

<% } %>

<% if (pet.sexo==='F' ) { %>

Fêmea

<% } %>

</td>

<td>

<button class="btn btn-danger btn-sm btn-block" id="btnEliminar" data-id="<%= pet.id %>">

Excluir

</button>

</td>

</tr>

</tbody>

</table>

<% } %>

</div>

<%- include("../templates/footer") %>

**views\pets\index.ejs**

<%- include("../templates/header", {bodyTitle: "Home" }) %>

<div class="container mb-4 mt-2">

<div class="row">

<div class="col-md-1"></div>

<div class="col-md-10">

<table class="table">

<thead>

<tr>

<th scope="col">Id</th>

<th scope="col">Nome</th>

<th scope="col">Espécie</th>

<th scope="col">Sexo</th>

<th scope="col">Ações</th>

</tr>

</thead>

<tbody>

<% arrayPets.forEach(pet=> { %>

<tr>

<th scope="row">

<%= pet.id %>

</th>

<td>

<%= pet.nome %>

</td>

<td>

<%= pet.especie %>

</td>

<td>

<% if (pet.sexo === 'M') { %>

Macho

<% } %>

<% if (pet.sexo==='F' ) { %>

Fêmea

<% } %>

</td>

<td><a class="btn btn-primary btn-sm" href="/pets/<%= pet.id %>">Detalhes</a>

<a class="btn btn-warning btn-sm" href="#">Editar</a>

</td>

</tr>

<% }) %>

</tbody>

</table>

</div>>

<div class="col-md-1"></div>

</div>

</div>

<%- include("../templates/footer") %>

**views\pets\show.ejs**

<%- include("../templates/header", {bodyTitle: "Detalhes do pet" }) %>

<div class="container mb-4 mt-2">

<h1 class="text-center">Detalhes do pet</h1>

<% if (error) { %>

<h3 class="text-danger text-center mb-5">

<%= message %>

</h3>

<a href="/pets" class="btn btn-success btn-block">Listar Pets</a>

<% } %>

<% if (!error) { %>

<table class="table">

<thead>

<tr>

<th scope="col">Id</th>

<th scope="col">Nome</th>

<th scope="col">Espécie</th>

<th scope="col">Sexo</th>

<th scope="col">Ação</th>

</tr>

</thead>

<tbody>

<tr>

<th scope="row">

<%= pet.id %>

</th>

<td>

<%= pet.nome %>

</td>

<td>

<%= pet.especie %>

</td>

<td>

<% if (pet.sexo==='M' ) { %>

Macho

<% } %>

<% if (pet.sexo==='F' ) { %>

Fêmea

<% } %>

</td>

<td>

<button class="btn btn-danger btn-sm btn-block" id="btnEliminar" data-id="<%= pet.id %>">

Excluir

</button>

</td>

</tr>

</tbody>

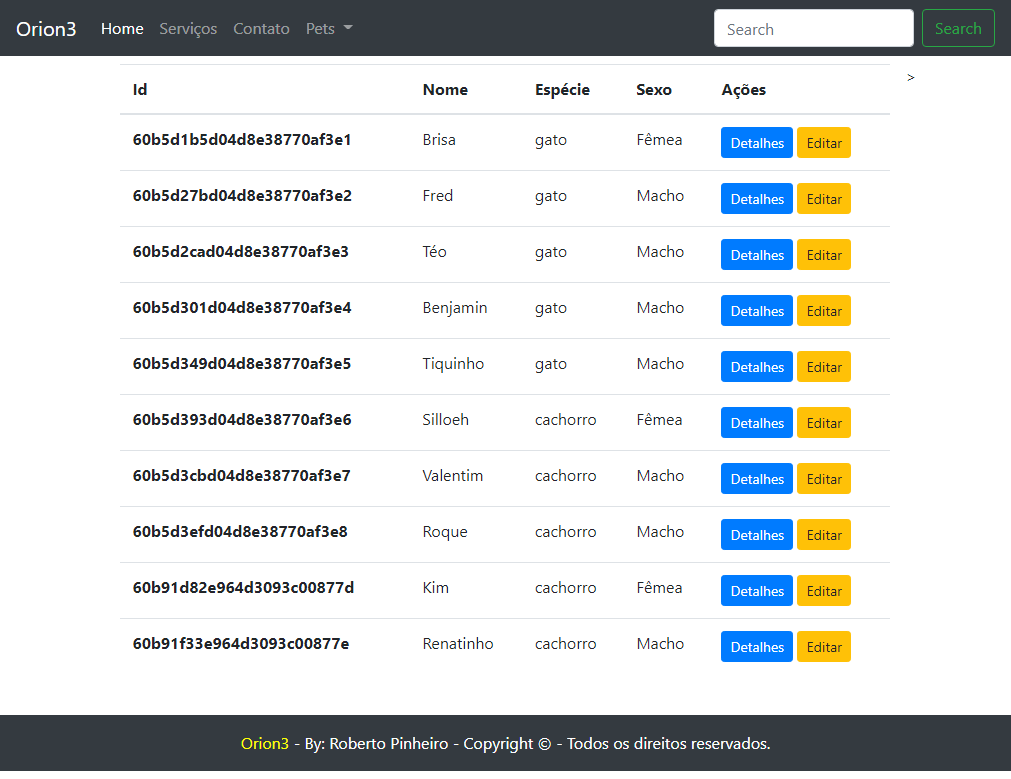
</table>

<% } %>

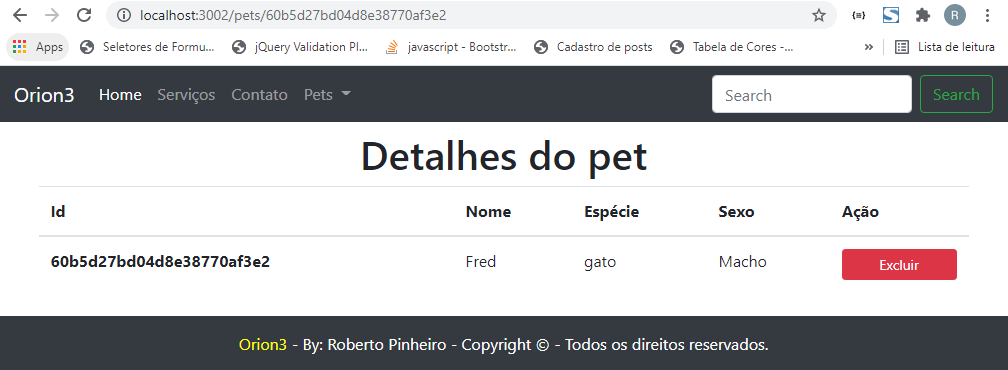
</div>

<%- include("../templates/footer") %>

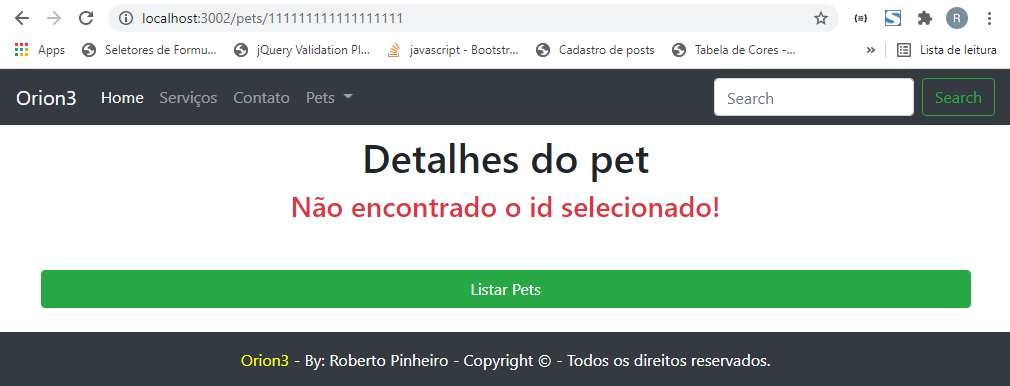
http://localhost:3002/pets



http://localhost:3002/pets/60b5d27bd04d8e38770af3e2



http://localhost:3002/pets/111111111111111111



# Aula 14 - CRUD - Eliminar documentos em MongoDB

**routes\routes-pets.js**

const express = require('express');

const router = express.Router();

const Pet = require('../models/Pet');

router.get('/', async (req, res) => {

try {

const arrayPetsDB = await Pet.find();

console.log(arrayPetsDB);

res.render("pets", {arrayPets: arrayPetsDB});

} catch (error) {

console.log(error);

}

});

router.get('/create', (req, res) => {

res.render("create");

});

router.post('/', async (req, res) => {

const body = req.body

try {

//const petDB = new Pet(body);

//await petDB.save();

await Pet.create(body);

res.redirect('/pets');

// console.log("Pet cadastrado: ", petDB);

} catch (error) {

console.log(error);

}

});

router.get('/:id', async (req, res) => {

const id = req.params.id

try {

const petDB = await Pet.findOne({ \_id: id });

console.log(petDB);

res.render('show', {

pet: petDB,

error: false

});

} catch (error) {

console.log(error);

res.render('show', {

error: true,

message: 'Não encontrado o id selecionado!'

});

}

});

router.delete('/:id', async (req, res) => {

const id = req.params.id;

console.log('id desde backend', id);

try {

const petDB = await Pet.findByIdAndDelete({ \_id: id });

console.log(petDB);

if(!petDB){

res.json({

status: false,

message: "Erro ao excluir o documento!"

});

} else {

res.json({

status: true,

message: "Eliminado com sucesso!"

});

}

} catch (error) {

console.log(error);

}

});

module.exports = router;

**views\show.ejs**

<%- include("../templates/header", {bodyTitle: "Detalhes do pet" }) %>

<div class="container mb-4 mt-2">

<h1 class="text-center">Detalhes do pet</h1>

<% if (error) { %>

<h3 class="text-danger text-center mb-5">

<%= message %>

</h3>

<a href="/pets" class="btn btn-success btn-block">Listar Pets</a>

<% } %>

<% if (!error) { %>

<table class="table">

<thead>

<tr>

<th scope="col">Id</th>

<th scope="col">Nome</th>

<th scope="col">Espécie</th>

<th scope="col">Sexo</th>

<th scope="col">Ação</th>

</tr>

</thead>

<tbody>

<tr>

<th scope="row">

<%= pet.id %>

</th>

<td>

<%= pet.nome %>

</td>

<td>

<%= pet.especie %>

</td>

<td>

<% if (pet.sexo==='M' ) { %>

Macho

<% } %>

<% if (pet.sexo==='F' ) { %>

Fêmea

<% } %>

</td>

<td>

<button class="btn btn-danger btn-sm btn-block" id="btnEliminar" data-id="<%= pet.id %>">

Excluir

</button>

</td>

</tr>

</tbody>

</table>

<% } %>

</div>

<%- include("../templates/footer") %>

<script>

const btnEliminar = document.querySelector('.btn-danger')

btnEliminar.addEventListener('click', async () => {

console.log(btnEliminar.dataset.id)

try {

const data = await fetch(`/pets/${btnEliminar.dataset.id}`, {

method: 'delete'

})

const res = await data.json()

console.log(res)

if (res.status) {

window.location.href = '/pets'

} else {

console.log(res)

}

} catch (error) {

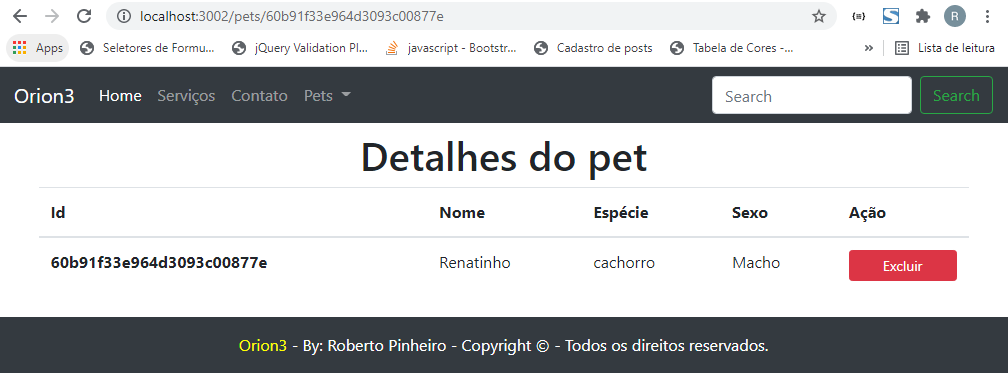
console.log(error)

}

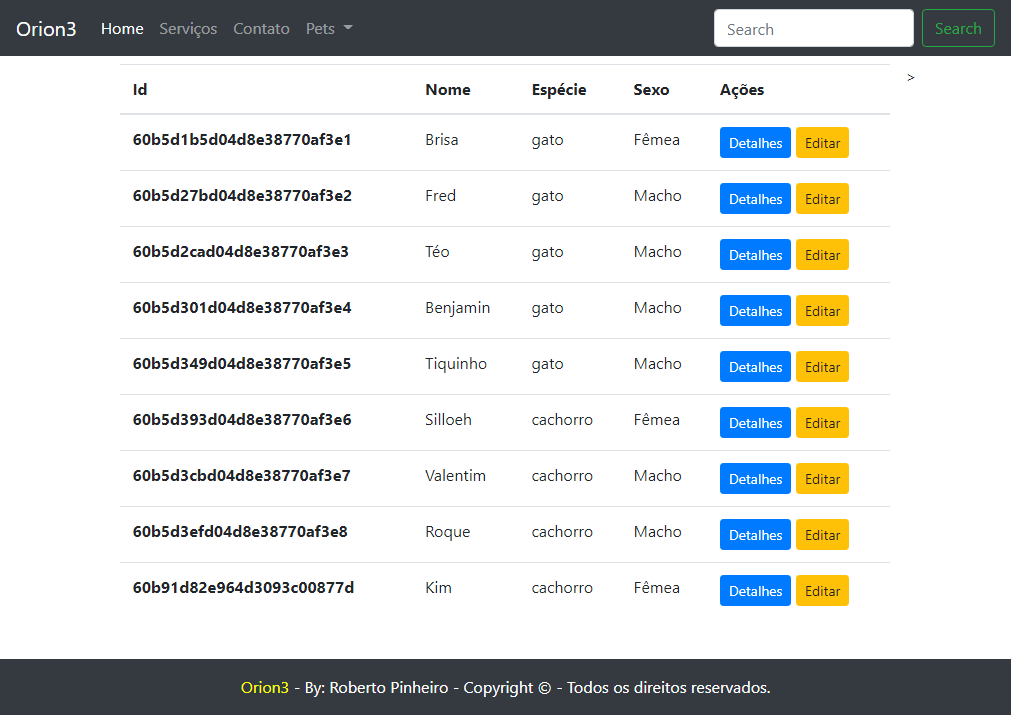
})

</script>

http://localhost:3002/pets/60b91f33e964d3093c00877e

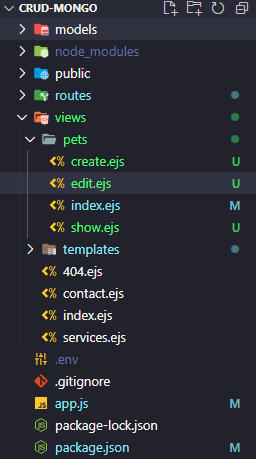


- Clique no botão "Excluir"



# Aula 15 - CRUD - Editar documento MongoDB e Deploy a Heroku

- Na pasta views\pets adicione um arquivo chamado edit.ejs:



**routes\routes-pets.js**

const express = require('express');

const router = express.Router();

const Pet = require('../models/Pet');

// Rotas

router.get('/', async (req, res) => {

try {

const arrayPetsDB = await Pet.find();

console.log(arrayPetsDB);

res.render("pets/index", {arrayPets: arrayPetsDB});

} catch (error) {

console.log(error);

}

});

router.get('/create', (req, res) => {

res.render("pets/create");

});

router.post('/', async (req, res) => {

const body = req.body

try {

//const petDB = new Pet(body);

//await petDB.save();

await Pet.create(body);

res.redirect('/pets');

// console.log("Pet cadastrado: ", petDB);

} catch (error) {

console.log(error);

}

});

router.get('/:id', async (req, res) => {

const id = req.params.id

try {

const petDB = await Pet.findOne({ \_id: id });

console.log(petDB);

res.render('pets/show', {

pet: petDB,

error: false

});

} catch (error) {

console.log(error);

res.render('pets/show', {

error: true,

message: 'Não encontrado o id selecionado!'

});

}

});

router.delete('/:id', async (req, res) => {

const id = req.params.id;

console.log('id desde backend', id);

try {

const petDB = await Pet.findByIdAndDelete({ \_id: id });

console.log(petDB);

if(!petDB){

res.json({

status: false,

message: "Erro ao excluir o documento!"

});

} else {

res.json({

status: true,

message: "Eliminado com sucesso!"

});

}

} catch (error) {

console.log(error);

}

});

router.get('/edit/:id', async (req, res) => {

const id = req.params.id

try {

const petDB = await Pet.findOne({ \_id: id });

console.log(petDB);

res.render('pets/edit', {

pet: petDB,

error: false

});

} catch (error) {

console.log(error);

res.render('pets/edit', {

error: true,

message: 'Não encontrado o id selecionado!'

});

}

});

router.put('/:id', async (req, res) => {

const id = req.params.id

const body = req.body

try {

const petDB = await Pet.findByIdAndUpdate(id, body, { useFindAndModify: false })

console.log(petDB)

res.json({

estado: true,

mensaje: 'Editado'

})

} catch (error) {

console.log(error)

res.json({

estado: false,

mensaje: 'Fallamos!!'

})

}

})

module.exports = router;

**views\pets\edit.ejs**

<%- include("../templates/header", {bodyTitle: "Detalhes do pet" }) %>

<div class="container mb-4 mt-2">

<div class="row">

<div class="col-md-2"></div>

<div class="col-md-8">

<h1 class="text-center mb-3">Atualização de dados do pet</h1>

<% if (error) { %>

<h3 class="text-danger text-center mb-5">

<%= message %>

</h3>

<a href="/pets" class="btn btn-success btn-block">Listar Pets</a>

<% } %>

<% if (!error) { %>

<form id="formularioEditar" data-id="<%= pet.id %>">

<label for="nome">Nome</label>

<input type="text" class="form-control my-2" name="nome" value="<%= pet.nome %>" id="nomeInput"><br />

<label for="especie">Espécie:</label>

<select name="especie" id="especieSelect" class="form-control">

<% if (pet.especie==='cachorro' ) { %>

<option value="cachorro" selected="selected">Cachorro</option>

<% } else {%>

<option value="cachorro">Cachorro</option>

<% } %>

<% if (pet.especie==='gato' ) { %>

<option value="gato" selected="selected">Gato</option>

<% } else {%>

<option value="gato">Gato</option>

<% } %>

<% if (pet.especie==='coelho' ) { %>

<option value="coelho" selected="selected">Coelho</option>

<% } else {%>

<option value="coelho">Coelho</option>

<% } %>

<% if (pet.especie==='hamster' ) { %>

<option value="hamster" selected="selected">Hamster</option>

<% } else {%>

<option value="hamster">Hamster</option>

<% } %>

</select><br /><br />

<label for="sexo">Sexo:</label>

<select name="sexo" id="sexoSelect" class="form-control">

<% if (pet.sexo==='F' ) { %>

<option value="F" selected="selected">Fêmea</option>

<% } else {%>

<option value="F">Fêmea</option>

<% } %>

<% if (pet.sexo==='M' ) { %>

<option value="M" selected="selected">Macho</option>

<% } else {%>

<option value="M">Macho</option>

<% } %>

</select><br /><br />

<button class="btn btn-warning btn-block type=" submit">

Gravar

</button>

</form>

<% } %>

</div>

</div>

<div class="col-md-2"></div>

</div>

<%- include("../templates/footer") %>

<script>

const formularioEditar = document.querySelector('#formularioEditar')

formularioEditar.addEventListener('submit', async (e) => {

e.preventDefault()

const nome = formularioEditar.elements['nome'].value

const especie = document.querySelector('#especieSelect').value

const sexo = document.querySelector('#sexoSelect').value

const id = formularioEditar.dataset.id

try {

const data = await fetch(`/pets/${id}`, {

method: 'put',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify({ nome, especie, sexo })

})

const res = await data.json()

if (res.estado) {

window.location.href = '/pets'

} else {

console.log(res)

}

} catch (error) {

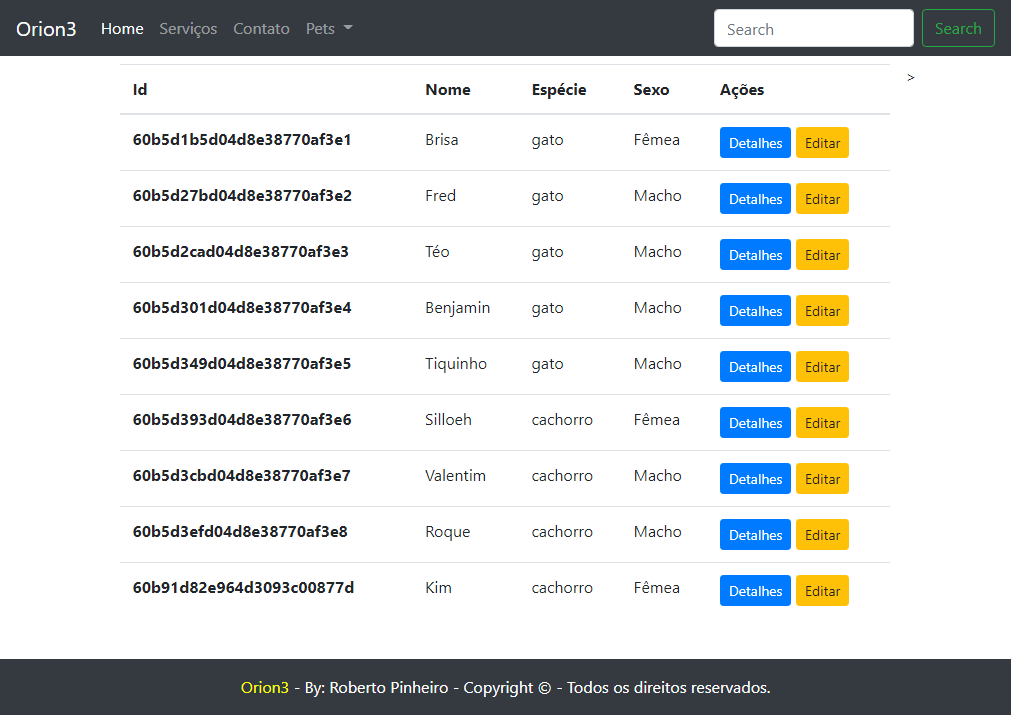
console.log(error)

}

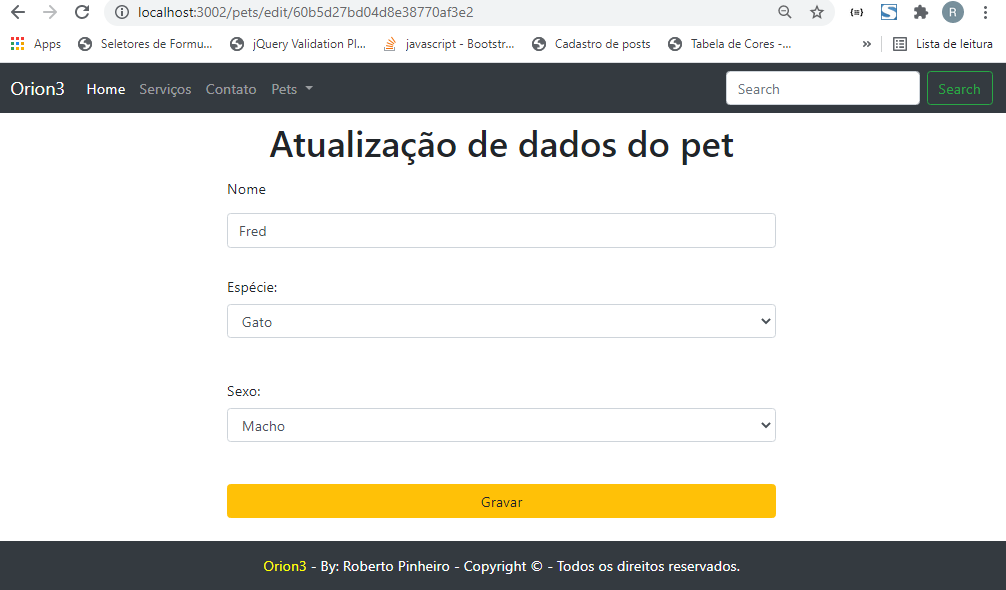
})

</script>

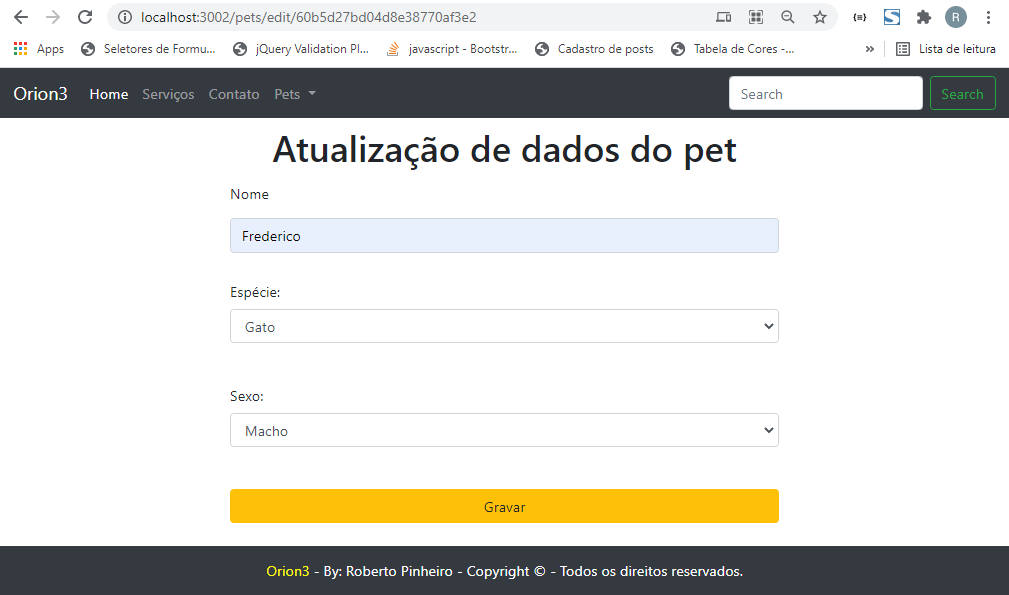
http://localhost:3002/pets



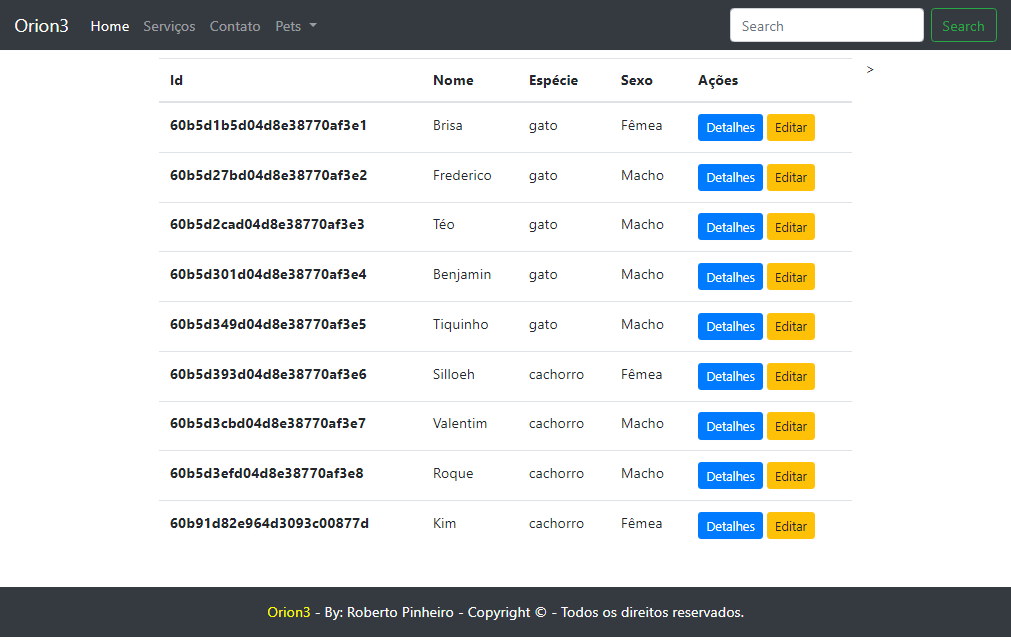
http://localhost:3002/pets/edit/60b5d27bd04d8e38770af3e2



- Altere o nome de "Fred" para "Frederico":

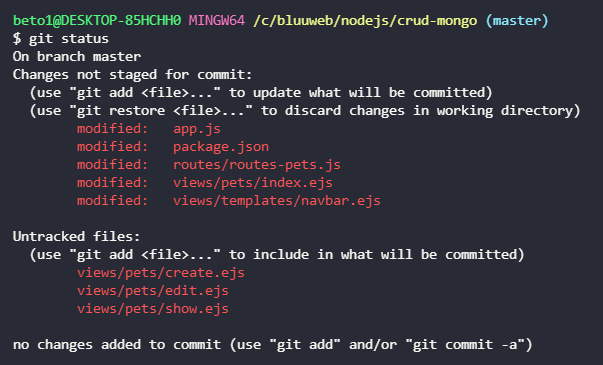


- Clique no botão "Gravar":

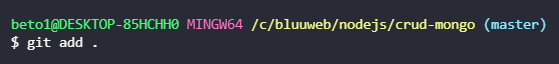


## Deploy

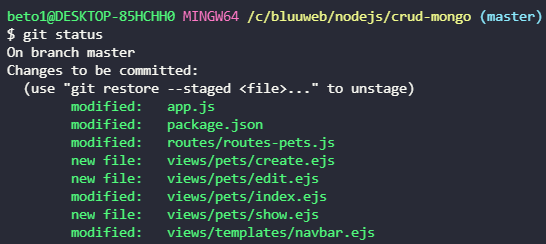
git status



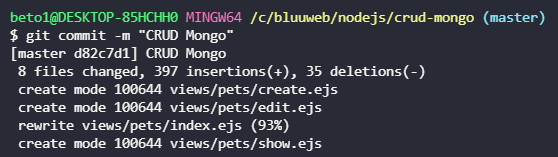
git add .



git status



git commit -m "CRUD Mongo"



git push heroku master