**Curso de Node.JS**

**Marcos Rivas (vidamrr)**

Vídeos:

<https://www.youtube.com/watch?v=APbb-z8A3Zc&list=PLvRPaExkZHFn_EGzNOXsWqtAAGl8ZCbUo>

Repositório Github:

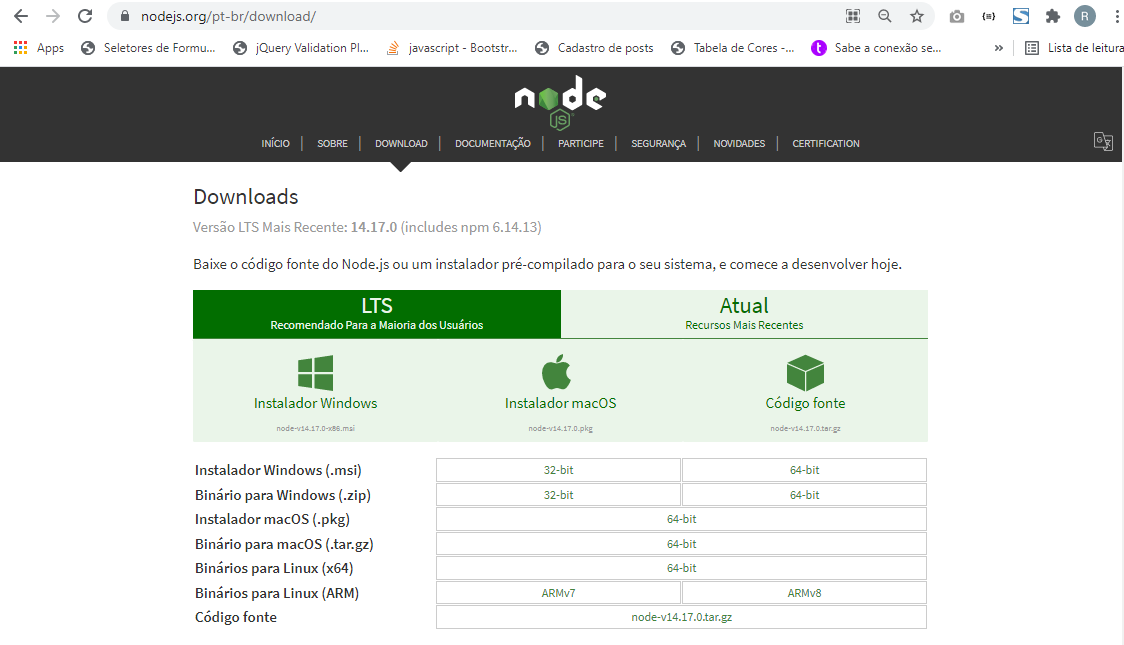
<https://github.com/marcosrivasr/Curso-de-NodeJS>

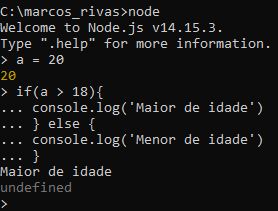
Resumo do curso feito por Roberto Pinheiro

# Aula 01 - Introdução

- Baixe e instale o Node

https://nodejs.org/pt-br/download/

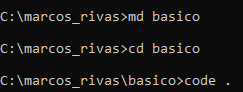




md basico

cd basico

code .



**01-instalacao\app.js**

process.stdout.write("Olá\n");

var nome = "Marcos Rivas";

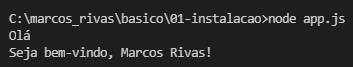
function saudar(){

console.log("Seja bem-vindo, " + nome + "!");

};

saudar();

node app.js



# Aula 02 - Exercícios básicos

node -v



**02-primeiros\_passos\basicos.js**

var nome = 'Roberto'

var path = require('path')

console.log(nome)

var a = 5;

var b = 10;

soma = a + b;

produto = a \* b;

console.log('Soma: ' + soma);

console.log(`Produto: ${produto}`);

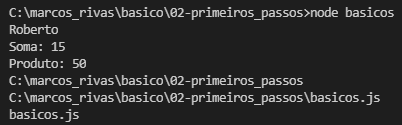
// Variáveis globais

console.log(\_\_dirname);

console.log(\_\_filename);

console.log(path.basename(\_\_filename));

node basicos.js

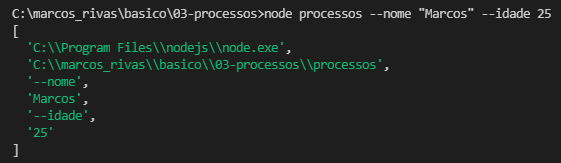


# Aula 03 - Processos

**03-processos\processos.js**

console.log(process.argv);

node processos --nome "Marcos" --idade 25



**03-processos\processos.js**

function param(p){

var index = process.argv.indexOf(p);

return process.argv[index + 1];

};

var nome = param('--nome');

var idade = param('--idade');

console.log(`Teu nome é ${nome}. Você tem ${idade} anos de idade.`);

node processos.js --nome "Marcos" --idade 25



# Aula 04 - Ler e escrever no console

## Processos stdout e stdin

**04-stdin\_stdout\app.js**

process.stdout.write("Olá mundo!\n")

node app.js



**04-stdin\_stdout\app.js**

// Processos stdout e stdin

var perguntas = ['Qual é o seu nome? ',

'Qual é a sua idade? ',

'Qual é a sua linguagem de programação favorita? '];

var respostas = [];

function pergunta(i){

process.stdout.write(perguntas[i]);

};

process.stdin.on('data', function(data){

respostas.push(data.toString().trim());

if(respostas.length < perguntas.length){

pergunta(respostas.length);

} else {

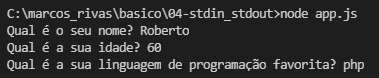
process.exit();

}

});

pergunta(0);

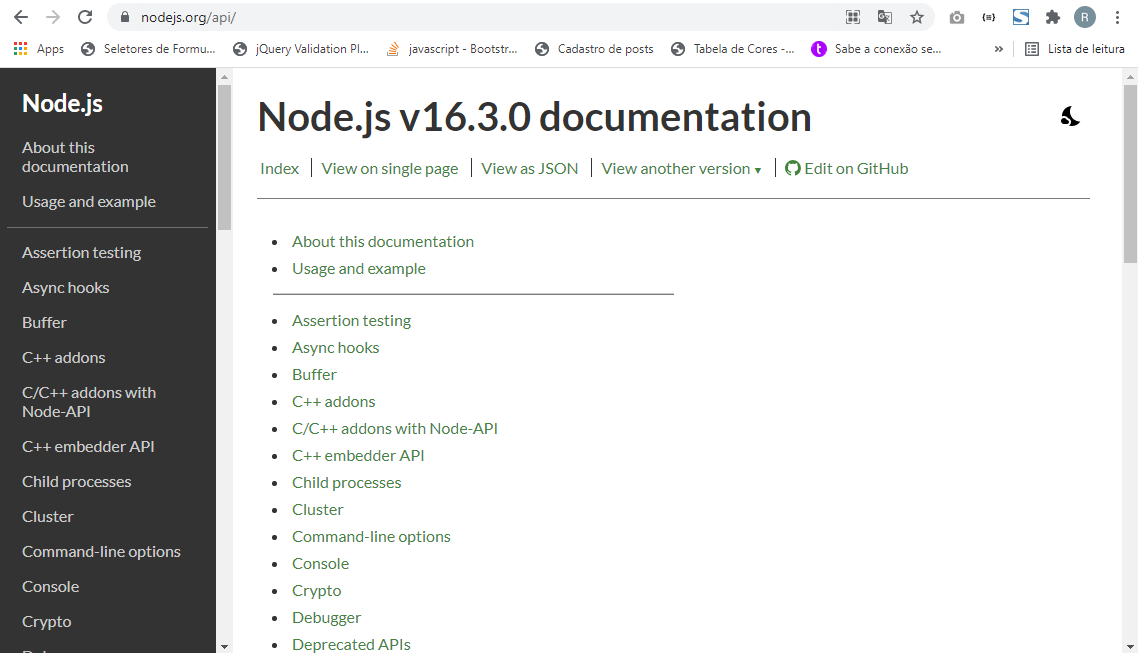
node app.js



# Aula 05 - Como funcionam os módulos

## Documentação do Node.JS

https://nodejs.org/api/



## Módulo path

**05-modulos\path.js**

// modulo path

var path = require('path');

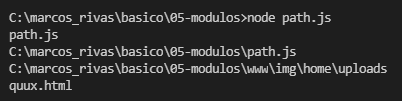
console.log(path.basename(\_\_filename));

console.log(\_\_filename);

console.log(path.join(\_\_dirname, 'www', 'img', 'home', 'uploads'));

console.log(path.basename('/foo/bar/baz/asdf/quux.html')); // Returns: 'quux.html'

node path.js



## Módulo util

**05-modulos\util.js**

// modulo util

var util = require('util');

util.log("Hello world!");

var nome = "Marcos";

var idade = 25;

var texto = util.format("Olá %s! Sua idade é %d anos", nome, idade);

console.log(texto);

node util.js



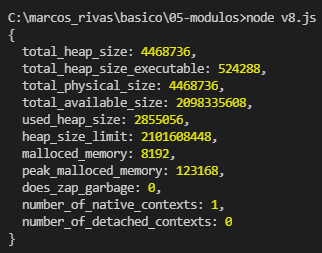
## Módulo v8

**05-modulos\v8.js**

var v8 = require('v8');

console.log(v8.getHeapStatistics());

node v8.js



# Aula 06 - Uso do módulo Readline

**06-readline\readline1.js**

var readline = require('readline');

var rl = readline.createInterface(process.stdin, process.stdout);

rl.question('Qual é o seu nome? ', (resposta) => {

console.log(`Olá, ${resposta}!`);

process.exit();

});

node readline1.js



**06-readline\readline2.js**

var readline = require('readline');

var util = require('util');

var rl = readline.createInterface(process.stdin, process.stdout);

var pessoa = {

nome: '',

comentarios: []

};

rl.question('Qual é o seu nome? ', (resposta) => {

pessoa.nome = resposta;

rl.setPrompt('Deixe um comentário: ');

rl.prompt();

});

rl.on('line', (input) =>{

if(input.trim() === 'sair'){

var message = util.format("Seu nome é %s. Seus comentários: %j", pessoa.nome, pessoa.comentarios);

console.log(message);

process.exit();

}

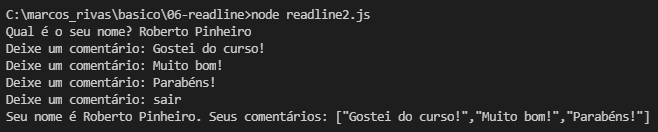
pessoa.comentarios.push(input.trim());

rl.setPrompt('Deixe um comentário: ');

rl.prompt();

});

node readline2.js



# Aula 07 - Como exportar módulos

**07-eventos\libs\pessoa.js**

const eventEmitter = require('events').EventEmitter;

const util = require('util');

var Pessoa = function(nome){

this.nome = nome;

}

util.inherits(Pessoa, eventEmitter);

module.exports = Pessoa;

**07-eventos\evento.js**

const pessoa = require('./libs/pessoa');

let pablo = new pessoa('Pablo');

pablo.on('fala', (message) => {

console.log(`${pablo.nome}: ${message}`);

});

pablo.emit('fala', 'Hoje será um grande dia!');

node evento.js



# Aula 08 - Manipulação de eventos

**08-manipulando\_eventos\evento1.js**

const events = require('events');

const emitter = new events.EventEmitter();

emitter.on('eventoCustom', (message, status) => {

console.log(`${status}: ${message}`);

});

emitter.emit('eventoCustom', 'Mensagem carregada com sucesso!', 200);

node evento1.js



**08-manipulando\_eventos\evento2.js**

const eventEmitter = require('events').EventEmitter

var Pessoa = function (nome){

this.nome = nome;

};

let pessoa = new Pessoa('Bob');

console.log(`Me chamo ${pessoa.nome}`);

node evento2.js



**08-manipulando\_eventos\evento3.js**

const eventEmitter = require('events').EventEmitter

const util = require('util');

var Pessoa = function (nome){

this.nome = nome;

};

util.inherits(Pessoa, eventEmitter);

let pessoa = new Pessoa('Bob');

// console.log(`Me chamo ${pessoa.nome}`);

pessoa.on('falar', (message) => {

console.log(`${pessoa.nome}: ${message}`);

});

pessoa.emit('falar', 'Hoje é um grande dia!');

node evento3.js



# Aula 09 - Executar processos em NodeJS

**09-executar\_processos\processo1.js**

const exec = require('child\_process').exec

exec('dir /b', (err, stdout) =>{

if(err){

throw err;

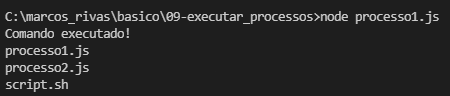
}

console.log('Comando executado!');

console.log(stdout);

});

node processo1.js



**09-executar\_processos\processo2.js**

const exec = require('child\_process').exec

exec('ls', (err, stdout) =>{

if(err){

throw err;

}

console.log('Comando executado!');

console.log(stdout);

});

node processo2.js



**09-executar\_processos\processo3.js**

const exec = require('child\_process').exec

const x = 10;

const y = 15;

exec('sh script.sh ' + x + ' ' + y, (err, stdout) =>{

if(err){

throw err;

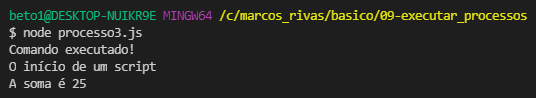
}

console.log('Comando executado!');

console.log(stdout);

});

node processo3.js



# Aula 10 - Ler arquivos com NodeJS

**10-ler\_arquivos\arquivo.txt**

Este é um arquivo de texto plano. Se pode ler, pode abri-lo.

**10-ler\_arquivos\read1.js**

// leitura de forma síncrona

var fs = require('fs');

var files = fs.readdirSync('./');

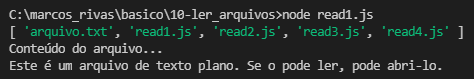
console.log(files);

var arquivo = fs.readFileSync('./arquivo.txt', 'utf8');

console.log('Conteúdo do arquivo...');

console.log(arquivo);

node read1.js



**10-ler\_arquivos\read2.js**

// leitura de forma assíncrona

var fs = require('fs');

fs.readdir('./', (error, files) => {

if(error){

throw(error);

}

console.log(files);

}

fs.readFile('./arquivo.txt', 'utf8', (error, arquivo) => {

if(error){

throw(error);

}

console.log(arquivo);

}

console.log('Conteúdo do arquivo...');

});

node read2.js



# Aula 11 - Escrever arquivos com NodeJS

**11-escrever\_arquivos\write1.js**

// validar, criar e/ou escrever em um arquivo (de forma síncrona)

const fs = require('fs');

const arquivo = 'prova.txt';

const conteudo = 'Este é o conteúdo de um texto';

const textoAdicional = "\nAqui foi adicionada uma nova linha de código!";

escrever();

function escrever(){

if(fs.existsSync(arquivo)){

console.log('O arquivo existe!');

// escrever uma linha no final do arquivo (de forma síncrona)

fs.appendFileSync(arquivo, textoAdicional);

console.log('Foi acrescentada uma nova linha no final do arquivo!');

} else {

console.log('O arquivo não existe!');

// criar e escrever em um arquivo (de forma síncrona)

fs.writeFileSync(arquivo, conteudo);

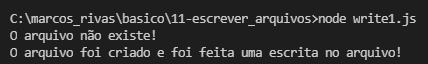
console.log('O arquivo foi criado e foi feita uma escrita no arquivo!');

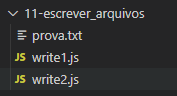
}

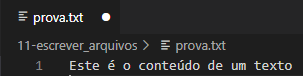
}



node write1.js

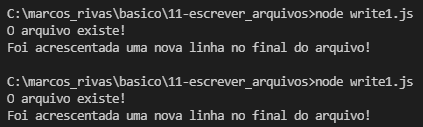


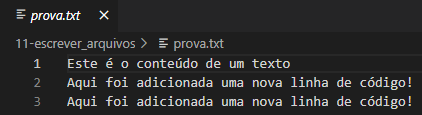




node write1.js

node write1.js





**11-escrever\_arquivos\write2.js**

// validar, criar e/ou escrever em um arquivo (de forma assíncrona)

const fs = require('fs');

const arquivo = 'prova.txt';

const conteudo = 'Este é o conteúdo de um texto';

const textoAdicional = "\nAqui foi adicionada uma nova linha de código!";

escrever();

function escrever(){

fs.access(arquivo, fs.constants.F\_OK, (err) => {

if(err){

console.log('O arquivo não existe!');

// criar e escrever em um arquivo (de forma assíncrona)

fs.writeFile(arquivo, conteudo, (err) =>{

if(err){

throw('Ocorreu um erro ao escrever no arquivo!');

} else {

console.log('O arquivo foi criado e foi feita uma escrita no arquivo!');

}

});

} else {

console.log('O arquivo existe!');

// escrever uma linha no final do arquivo (de forma assíncrona)

fs.appendFile(arquivo, textoAdicional, (err) => {

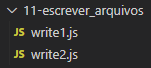
console.log('Foi acrescentada uma nova linha no final do arquivo!');

});

}

})

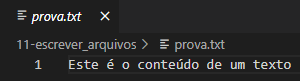
}



node write2.js

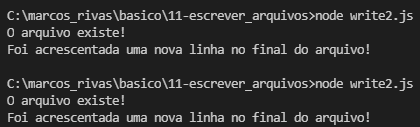






node write2.js

node write2.js





# Aula 12 - Usar diretórios em NodeJS

**12-diretorios\diretorio1.js**

// criar diretório de forma síncrona

const fs = require('fs');

// validar se existe um diretório

if(fs.existsSync('img')){

console.log('O diretório já existe!');

} else {

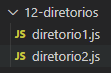
console.log('O diretório não existe!');

// criando o diretório

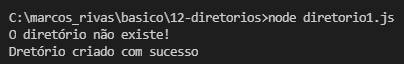
fs.mkdirSync('img');

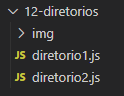
console.log('Dretório criado com sucesso');

}



node diretorio1.js





node diretorio1.js



**12-diretorios\diretorio2.js**

// criando diretório de forma assíncrona

const fs = require('fs');

if (fs.existsSync('css')) {

console.log('O diretório já existe!');

} else {

console.log('O diretório não existe!')

// criando o diretório

fs.mkdir('css', function (err) {

if(err){

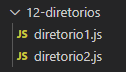
throw(err);

}

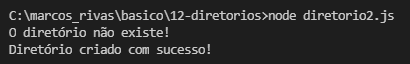
console.log('Diretório criado com sucesso!');

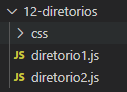
});

}



node diretorio2.js





node diretorio2.js



# Aula 13 - Operações básicas com arquivos em NodeJS

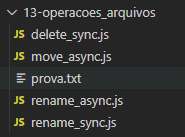
## Renomeando arquivo de forma síncrona

**13-operacoes\_arquivos\rename\_sync.js**

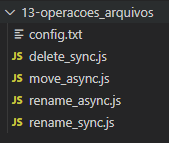
// renomear arquivo de forma síncrona

const fs = require('fs');

fs.renameSync('./prova.txt', './config.txt');



node rename\_sync.js



## Renomeando arquivo de forma assíncrona

**13-operacoes\_arquivos\rename\_async.js**

// renomear arquivos de forma assíncrona

const fs = require('fs');

fs.rename('./config.txt', './prova.txt', (err) => {

if(err){

throw(err);

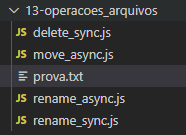
}

console.log('O arquivo foi renomeado com sucesso!');

})

node rename\_async.js





## Movendo arquivo de forma assíncrona

**13-operacoes\_arquivos\move\_async.js**

// mover arquivos de forma assíncrona

const fs = require('fs');

fs.rename('./prova.txt', './src/prova.txt', (err) => {

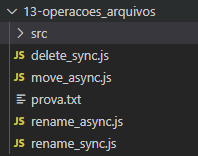
if(err){

throw(err);

}

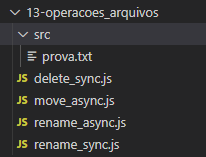
console.log('O arquivo foi movido com sucesso!');

})



node move\_async.js





## Excluindo arquivo de forma síncrona

**13-operacoes\_arquivos\delete\_sync.js**

// deletando um arquivo de forma síncrona

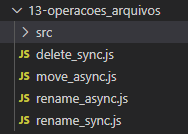
const fs = require('fs');

fs.unlinkSync('./src/prova.txt');

console.log('O arquivo foi excluído!');

node delete\_sync.js





# Aula 14 - Ler streams com NodeJS

**14-ler\_streams\ler\_stream.js**

const fs = require('fs');

// let conteudo = fs.readFileSync('./logs.log');

// console.log(`tamanho: ${conteudo.length}`);

let stream = fs.createReadStream('./logs.log', 'utf-8');

let data = '';

stream.once('data', () => {

console.log('Iniciando o stream...');

});

stream.on('data', (chunk) => {

console.log(`${chunk.length} | `);

data += chunk;

});

stream.on('end', () => {

console.log('Fim de stream!\n');

console.log(data.length);

})

node ler\_stream.js

C:\marcos\_rivas\basico\14-ler\_streams>node ler\_stream.js

Iniciando o stream...

65536 |

65536 |

65536 |

65536 |

65536 |

.

.

.

65536 |

65536 |

65536 |

65536 |

65536 |

65536 |

46650 |

Fim de stream!

12105274

# Aula 15 - Escrever streams com NodeJS

**15-escrever\_streams\escrever\_stream.js**

const fs = require('fs');

const readline = require('readline');

let i = readline.createInterface(process.stdin, process.stdout);

i.question('Qual é o seu nome? > ', (nome)=>{

let stream = fs.createWriteStream(`./${nome}.txt`);

stream.write(`Dito isto: ${nome} \n`);

//fs.writeFileSync(`./${nome}.txt`, `Dito isto ${nome} \n`);

process.stdout.write('O que você quer dizer? \n');

i.on('line', (dito)=>{

if(dito.trim() == 'sair'){

stream.close();

i.close();

}else{

stream.write(dito.trim() + '\n');

//fs.appendFileSync(`./${nome}.txt`, dito.trim() + '\n');

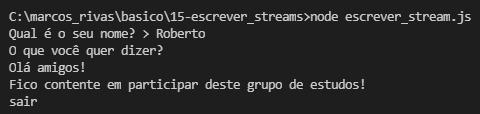
}

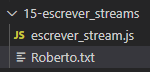
});

});



node escrever\_stream.js





É criado arquivo de texto com o nome de Roberto.txt e com o seguinte conteúdo:

**Roberto.txt**

Dito isto: Roberto

Olá amigos!

Fico contente em participar deste grupo de estudos!

# Aula 16 - Primeira aplicação completa em NodeJS

**16-exercicio01\tutorial\document.js**

const fs = require('fs');

const os = require('os');

class Document{

constructor(dir){

this.\_content = '';

this.\_isSaved = false;

this.\_filename = '';

this.\_dir = dir;

}

exists(name){

return fs.existsSync(`${this.\_dir}/${name}`);

}

append(text){

this.\_content += os.EOL + text;

this.\_isSaved = false;

}

saveas(name){

fs.writeFileSync(`${this.\_dir}/${name}`, this.\_content);

this.\_filename = name;

this.\_isSaved = true;

}

save(){

fs.writeFileSync(`${this.\_dir}/${this.\_filename}`, this.\_content);

this.\_isSaved = true;

this.\_filename = this.\_filename;

}

getContent(){

return this.\_content;

}

hasName(){

if(this.\_filename != ''){

return true;

}else{

return false;

}

}

getName(){

return this.\_filename;

}

isSaved(){

return this.\_isSaved;

}

open(name){

this.\_content = fs.readFileSync(`${this.\_dir}/${name}`, 'UTF-8');

this.\_filename = name;

this.\_isSaved = true;

return this.\_content;

}

}

module.exports = Document;

**16-exercicio01\tutorial\messages.js**

const messages = {

fileSaved: 'Arquivo gravado, pode seguir editando...',

fileNotSaved : 'Arquivo não gravado, pode seguir editando...',

requestFileName: '\* Nome do arquivo: ',

replaceFile: 'Deseja sustituir o arquivo?(y/n): ',

fileNotFound: 'O arquivo não existe!',

fileExists: '\*\*\* Já existe esse arquivo! \*\*\*'

};

module.exports = messages;

**16-exercicio01\tutorial\directory.js**

const fs = require('fs');

const path = require('path');

class Directory{

constructor(){

this.\_path = \_\_dirname;

this.createDocsDir();

}

createDocsDir(){

this.\_path = path.join(this.\_path, 'docs', '');

if(!fs.existsSync('./docs')){

fs.mkdirSync('./docs');

}

}

getPath(){

return this.\_path;

}

getShortPath(){

const paths = path.parse(this.\_path);

let delimiter = "/";

if(paths.dir.indexOf(delimiter) < 0){

delimiter = `\\`;

}

//const dirs = paths.dir.split(delimiter);

return `${paths.root}...${delimiter}${paths.name}`;

}

getFilesInDir(){

const files = fs.readdirSync(this.\_path);

let n = 0;

console.log(`

====================================

Localização: ${this.getShortPath()}

====================================`);

files.forEach(file => {

if(file != '.DS\_Store'){

console.log(` ${file}`);

n++;

}

});

}

}

module.exports = Directory;

**16-exercicio01\tutorial\app.js**

const readline = require('readline');

const Messages = require('./messages');

const Document = require('./document');

const Directory = require('./directory');

const dir = new Directory();

let interface = readline.createInterface(process.stdin, process.stdout);

const tools =`Comandos: :q = sair, :sa = gravar como, :s = salvar

--------------------------------------`

const pantalla = `

================

Editor de texto

================\n

Escolha uma opção:\n

1 Criar novo documento

2 Abrir documento

3 Fechar editor\n> `;

mainScreen();

function mainScreen(){

process.stdout.write('\033c');

interface.question(pantalla, (res) =>{

switch(res.trim()){

case '1':

createFile();

break;

case '2':

openFileInterface();

break;

case '3':

interface.close();

break;

default:

mainScreen();

}

});

}

function readCommands(file){

interface.on('line', (input)=>{

switch(input.trim()){

case ':sa':

saveAs(file);

break;

case ':q':

interface.removeAllListeners('line');

mainScreen();

break;

case ':s':

save(file);

break;

default:

file.append(input.trim());

}

})

}

function createFile(){

let file = new Document(dir.getPath());

renderInterface(file);

readCommands(file);

}

function save(file){

if(file.hasName()){

file.save()

renderInterface(file, `${Messages.fileSaved}\n`);

}else{

saveAs(file);

}

}

function saveAs(file){

interface.question(Messages.requestFileName, (name) =>{

if(file.exists(name)){

console.log(Messages.fileExists);

interface.question(Messages.replaceFile, (confirm)=>{

if(confirm == 'y'){

file.saveas(name);

renderInterface(file, `${Messages.fileSaved}\n`);

}else{

renderInterface(file, `${Messages.fileNotSaved}\n`);

}

});

}else{

file.saveas(name);

renderInterface(file, `${Messages.fileSaved}\n`);

}

});

}

function openFile(file, name){

content = file.open(name);

renderInterface(file);

readCommands(file);

}

function renderInterface(file, mensaje){

process.stdout.write('\033c');

(file.getName() == '') ? console.log(`| Untitled |`) : console.log(`| ${file.getName()} |`);

console.log(tools);

if(mensaje != null) console.log(mensaje);

console.log(file.getContent());

}

function openFileInterface(){

let file = new Document(dir.getPath());

dir.getFilesInDir();

interface.question(Messages.requestFileName, (name) =>{

if(file.exists(name)){

openFile(file, name);

}else{

console.log(Messages.fileNotFound);

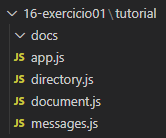
interface.removeAllListeners('line');

mainScreen();

}

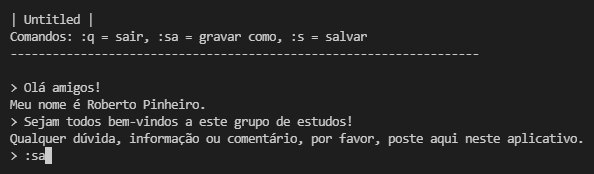
});

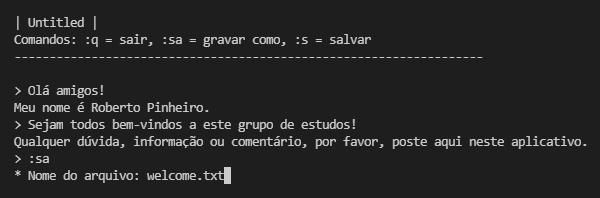
}

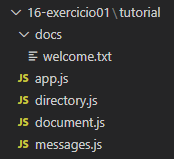


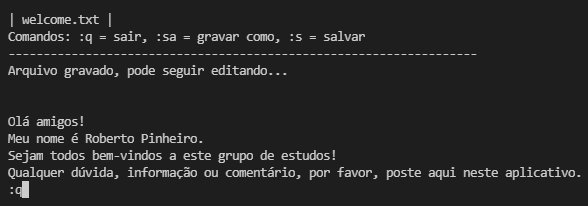
node app.js

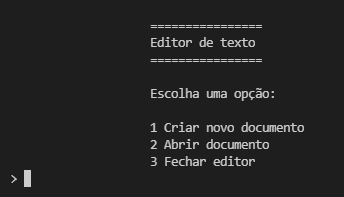


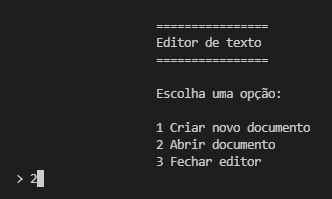


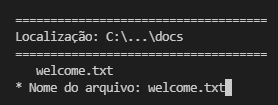


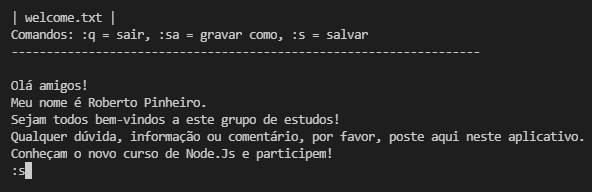


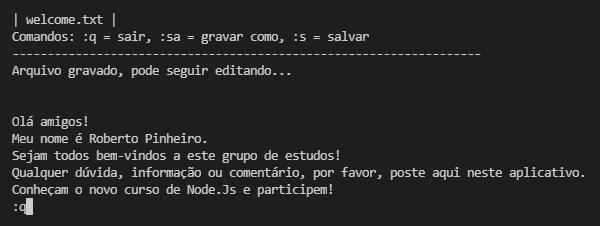


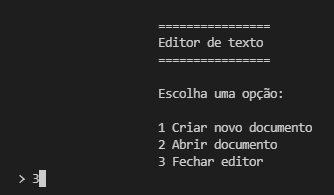












# Aula 17 - Fazer requisições HTTP com NodeJS

**17-requisicao\_http\requisicao.js**

const http = require('http');

const options ={

hostname: 'localhost',

port:80,

path: '/curso/49.%20carrito/tutorial/api/productos/api-productos.php?categoria=juguetes',

method: 'GET'

};

const req = http.request(options, res =>{

console.log(`status code: ${res.statusCode}`);

console.log('Headers: %j', res.headers);

let body = '';

res.on('data', chunk =>{

body += chunk;

});

res.on('end', () =>{

console.log('\n\nResultados:')

console.log(body);

});

});

req.on('error', err =>{});

req.end();

# Aula 18 - Criar um servidor web em 6 linhas de código

**18-servidor\_web\server.js**

const http = require('http');

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

res.writeHead(200, {'content-type': 'text/html'});

res.end(`

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

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

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

<title>Documento HTML</title>

</head>

<body>

<h1>Olá Mundo!</h1>

<p>

Bem-vindo a meu servidor!

</p>

</body>

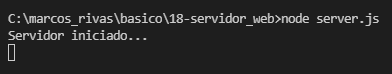
</html>

`);

}).listen(3000);

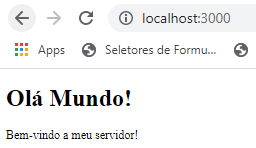
console.log('Servidor iniciado...');

node server.js

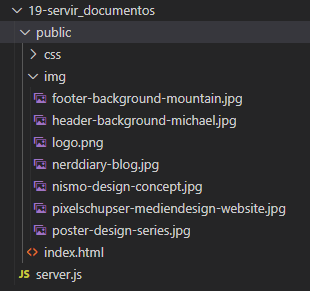


- No browser:

localhost:3000



# Aula 19 - Carregar arquivos de um website



**19-servir\_documentos\server.js**

const http = require('http');

const fs = require('fs');

const path = require('path');

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

console.log(`${req.method} solicita ${req.url}`);

if(req.url == '/'){

fs.readFile('./public/index.html', 'UTF-8', (err, html) =>{

res.writeHead(200, {'Content-Type': 'text/html'});

res.end(html);

});

}else if(req.url.match(/.css$/)){

const reqPath = path.join(\_\_dirname, 'public', req.url);

const fileStream = fs.createReadStream(reqPath, 'UTF-8');

res.writeHead(200, {'Content-Type': 'text/css'});

fileStream.pipe(res);

}else if(req.url.match(/.jpg$/)){

const reqPath = path.join(\_\_dirname, 'public', req.url);

const fileStream = fs.createReadStream(reqPath);

res.writeHead(200, {'Content-Type': 'image/jpg'});

fileStream.pipe(res);

}else{

res.writeHead(404, {'Content-Type': 'text/plain'});

res.end('404 ERROR');

}

}).listen(3000);

console.log('Servidor iniciado...');

**19-servir\_documentos\public\index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Mi primer sitio web</title>

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

</head>

<body>

<div id="main-container">

<section id="intro-container">

<div class="center-align">

<nav>

<div id="logo"><img src="img/logo.png" alt=""></div>

<div id="menu-lista">

<ul>

<li><a href="#about-section">Acerca de..</a></li>

<li><a href="#work">Proyectos</a></li>

<li><a href="#footer">Contacto</a></li>

</ul>

</div>

</nav>

<div class="section-columns">

<div>

<h2 class="white">Mi primer sitio web en la vida</h2>

<div class="btn-section">

<a href="#about-section" class="btn btn-white">Conóceme más</a>

</div>

</div>

<div id="image-bg">

</div>

</div>

</div>

</section>

<div class="rectangle bg-white skew-left"></div>

<div class="section" id="about-section">

<div class="center-align">

<div class="section-columns">

<div>

<h2>Diseño.<br />Desarrollo.<br/>Inspiración.</h2>

</div>

<div id="bio">

<h3>Mi nombre es <span>Marcos Rivas</span> , soy un ingeniero en Sistemas Computacionales, <span>trabajo</span> <span>en</span> <span>Microsoft</span>, y me dedico al <span>diseño web</span><span> como</span><span> hobbie.</span></h3>

<p>Tengo mi <span>propio canal de Youtube</span> que trata sobre tutoriales y cursos para aprender <span>diseño y programación.</span></p>

<p><span>Gasto mucho de mi tiempo en dibujar</span>, que es otro de mis pasatiempos favoritos, así como seguir aprendiendo de las <span>tecnologías actuales.</span></p>

<p><span>Puedes seguirme</span> en <a href="#">Twitter</a> o <a href="#">Facebook</a>, o directamente en el <a href="#">canal de Youtube</a></p>

<div class="btn-section">

<a href="#work" class="btn btn-black">Ver proyectos sobresalientes</a>

</div>

</div>

</div>

</div>

</div>

<div class="rectangle bg-gray skew-right"></div>

<div class="section bg-gray" id="work">

<div class="center-align">

<div class="section-columns" >

<div>

<h2 class="white">Posters.<br />Sitios Web.<br/>Tutoriales.</h2>

<div class="btn-section">

<a href="" class="btn btn-white">Ver proyectos sobresalientes</a>

</div>

</div>

</div>

</div>

</div>

<div class="section bg-gray" id="work02">

<div class="center-align">

<div class="section-columns" >

<div>

<h2 class="white">Opinión.<br />Impactante.<br/>Blog.</h2>

<div class="btn-section">

<a href="" class="btn btn-white">Ver Proyecto</a>

</div>

</div>

</div>

</div>

</div>

<div class="section bg-gray" id="work03">

<div class="center-align">

<div class="section-columns" >

<div>

<h2 class="white">Patrón<br />Diseño.<br/>Móvil.</h2>

<div class="btn-section">

<a href="" class="btn btn-white">Ver Proyecto</a>

</div>

</div>

</div>

</div>

</div>

<div class="section bg-gray" id="work04">

<div class="center-align">

<div class="section-columns" >

<div>

<h2 class="white">Innovación.<br />Fresco.<br/>Color.</h2>

<div class="btn-section">

<a href="" class="btn btn-white">Ver Proyecto</a>

</div>

</div>

</div>

</div>

</div>

<div class="rectangle bg-black skew-left"></div>

<div class="section bg-black" id="footer">

<div class="center-align">

<div class="section-columns" >

<div>

<h2 class="white">Sigamos en Contacto.</h2>

<div class="btn-section">

<a href="mailto: contacto@misitio.com" class="btn btn-white">Mándame un mensaje</a>

</div>

</div>

<div>

<h3>Contáctame si quieres conocerme y saber cómo podemos colaborar en tu siguiente proyecto.</h3>

<p>También puedes estar conmigo por medio de las redes sociales, enterarte de mi más reciente trabajo y conocerme un poco más.</p>

<div class="btn-section">

<a href="http://www.facebook.com/misitio/" class="btn btn-white">Facebook</a>

<a href="http://www.twitter.com/misitio/" class="btn btn-white">Twitter</a>

<a href="http://www.instagram.com/misitio/" class="btn btn-white">Instagram</a>

</div>

</div>

</div>

</div>

</div>

</div>

</body>

</html>

**19-servir\_documentos\public\css**

body{

padding: 0;

margin: 0;

font-family: Arial;

}

span{

position: relative;

}

span::after{ /\* #fffcd6 \*/

background-color: #fffcd6;

content: '';

width: 100%;

height: 50%;

position: absolute;

bottom: -4px;

left: 0;

z-index: -1;

}

p a{ /\* #184DA8 \*/

color: #184DA8;

text-decoration: none;

}

p a:hover{ /\* #184DA8 \*/

text-decoration: underline;

}

.center-align{

margin: 0 auto;

width: 1100px;

}

.black{color: black;}

.white{color: white;}

.section-columns{

}

.section-columns > div{

display: inline-block;

vertical-align: top;

width: 49%;

}

.section-columns h2{

font-size: 72px;

}

#intro-container .section-columns h2{

padding-top: 120px;

padding-bottom: 30px;

}

.section-columns h3{

font-size: 26px;

font-weight: lighter;

line-height: 40px;

}

.section-columns p{

font-weight: lighter;

font-size: 16px;

line-height: 30px;

padding-bottom: 10px;

}

.section-columns #image-bg{

background-image: url('../img/header-background-michael.jpg');

background-repeat: no-repeat;

background-size: 750px;

background-position: -140px -80px;

height: 600px;

}

.section-columns #image-bg-02{

}

.section-columns #bio{

height: 600px;

}

#work, #work02, #work03, #work04{

background-repeat: no-repeat;

background-size: 750px;

background-position: right center;

height: 600px;

padding-top: 150px;

margin-top: -50px;

}

#work {background-image: url('../img/poster-design-series.jpg');}

#work02{background-image: url('../img/nerddiary-blog.jpg');}

#work03{background-image: url('../img/nismo-design-concept.jpg');}

#work04{background-image: url('../img/pixelschupser-mediendesign-website.jpg');}

#footer{

margin-top: -50px;

padding-top: 50px;

height: 550px;

background-image: url('../img/footer-background-mountain.jpg');

background-size: 100%;

background-position: center bottom;

background-repeat: no-repeat;

}

#footer h3{color: white; padding-top: 70px; margin: 0;}

#footer p {color: gray; padding-bottom: 30px;}

.rectangle{

height: 100px;

margin-top: -50px;

}

.bg-white{

background-color: white;

}

.bg-gray{

background-color: #666;

}

.bg-black{

background-color: #000;

}

.skew-left{

transform: skewY(-4deg);

}

.skew-right{

transform: skewY(4deg);

}

.btn{

border: solid 1px;

padding: 20px 40px;

text-decoration: none;

margin-right: 10px;

}

.btn-black{

border-color: black;

color: black;

}

.btn-black:hover{

background-color: black;

border-color: white;

color: white;

}

.btn-white{

border-color: white;

color: white;

}

.btn-white:hover{

background-color: white;

border-color: black;

color: black;

}

#intro-container{

background-color: black;

padding-top: 100px;

}

#intro-container nav{

margin: 0;

padding: 0;

}

#intro-container nav div{

display: inline-block;

vertical-align: top;

}

#intro-container nav #logo{

width: 9%;

}

#intro-container nav #menu-lista{

width: 89%;

text-align: right;

}

#intro-container nav #menu-lista ul{

margin: 0;

padding: 0;

}

#intro-container nav #menu-lista ul li{

display: inline-block;

padding: 0;

margin: 0;

list-style: none;

}

#intro-container nav #menu-lista ul li a{

color: #777;

text-decoration: none;

}

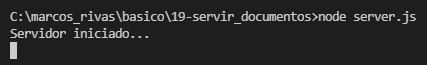
#intro-container nav #menu-lista ul li a:hover{

color: white;

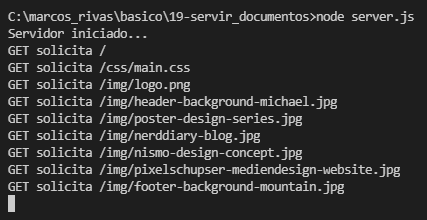
text-decoration: white;

}

node server.js



localhost:3000



- No browser:



# Aula 20 - Criar uma API estática em NodeJS

**20-api\_estatica\server.js**

const http = require('http');

const pedidos = [

{'nome': 'sopa', 'quantidade': 2, 'status': 'servido', 'mesa': 5},

{'nome': 'corte de carne', 'quantidade': 2, 'status': 'processo', 'mesa': 4},

{'nome': 'sopa', 'quantidade': 1, 'status': 'pago', 'mesa': 3},

{'nome': 'salada', 'quantidade': 1, 'status': 'pago', 'mesa': 5},

{'nome': 'massa', 'quantidade': 4, 'status': 'em processo', 'mesa': 1},

];

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

if(req.url == '/'){

res.writeHead(200, {'Content-Type': 'text/json'});

res.end(JSON.stringify(pedidos));

}else if(req.url == '/pedidos-processo'){

pedidosEmProcesso(res);

}else if(req.url == '/pedidos-sopa'){

pedidosPorNome('sopa', res);

}else{

res.writeHead(404, {'Content-Type': 'text/plain'});

res.end('Não encontrada a requisição requerida','latin1');

}

}).listen(3000);

console.log('servidor iniciado...');

const pedidosEmProcesso = (res) =>{

const arrayRes = pedidos.filter(item =>{

return item.status == 'em processo';

});

res.writeHead(200, {'Content-Type': 'text/json'});

res.end(JSON.stringify(arrayRes));

}

const pedidosPorNome = (nome, res) =>{

const arrayRes = pedidos.filter(item =>{

return item.nome == nome;

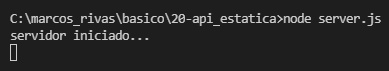
});

res.writeHead(200, {'Content-Type': 'text/json'});

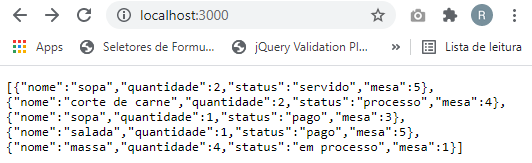
res.end(JSON.stringify(arrayRes));

}

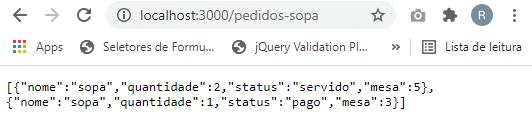
node server.js



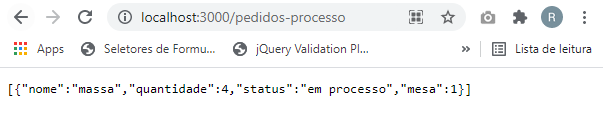
http://localhost:3000



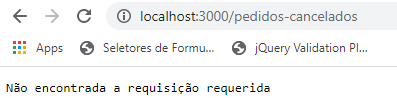
http://localhost:3000/pedidos-sopa



http://localhost:3000/pedidos-processo



http://localhost:3000/pedidos-cancelados



# Aula 21 - Manipulação de requisições POST

**21-requisicao\_post\server.js**

const http = require('http');

const fs = require('fs');

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

if(req.method == 'GET'){

res.writeHead(200, {'Content-Type': 'text/html'});

fs.createReadStream('./form.html', 'UTF-8').pipe(res);

}else if(req.method == 'POST'){

let body = '';

req.on('data', chunk =>{body+= chunk;});

req.on('end', () =>{

res.writeHead(200, {'Content-Type': 'text/html'});

res.end(`

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

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

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

<title>Resultados</title>

</head>

<body>

<h1>Dados recebidos do formulário</h1>

<p>${body}</p>

</body>

</html>

`);

});

}

}).listen(3000);

console.log('Servidor iniciado...');

**21-requisicao\_post\form.html**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

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

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

<title>Formulario</title>

</head>

<body>

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

<h1>Formulário de contato</h1>

<p>

Nome: <br/>

<input type="text" name="nome">

</p>

<p>

Email: <br/>

<input type="text" name="email">

</p>

<p>

País de origem: <br/>

<input type="text" name="pais">

</p>

<p>

<input type="submit" value="Enviar formulário">

</p>

</form>

</body>

</html>

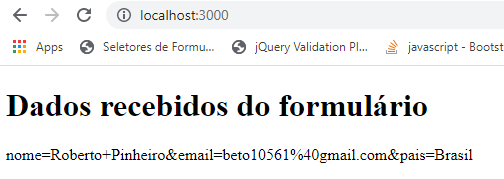
node server.js



- No browser:

localhost:3000

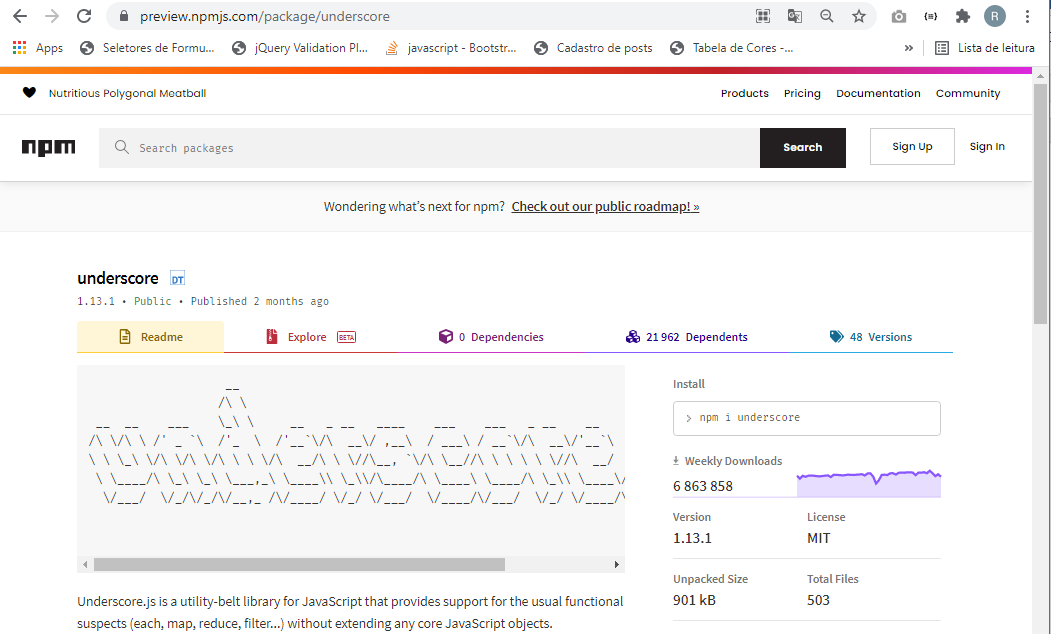




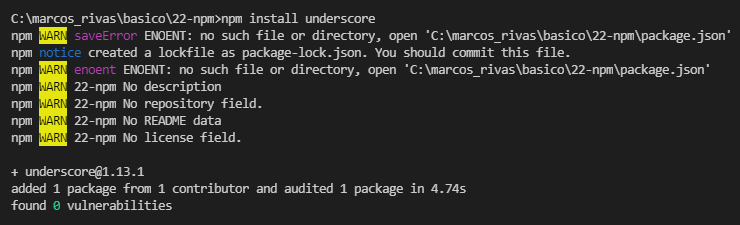
# Aula 22 - Tutorial básico de NPM

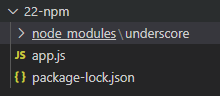
## Instalando o pacote underscore

https://preview.npmjs.com/package/underscore



npm install underscore





**22-npm\app.js**

const \_ = require('underscore');

const lista = [

{'id': 1, 'nome': 'Marcos', 'idade': 27},

{'id': 2, 'nome': 'Regina', 'idade': 26},

{'id': 3, 'nome': 'Bruno', 'idade': 22}

];

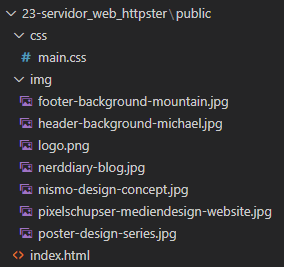
const res = \_.findWhere(lista, {'idade': 22});

console.log(res);

node app.js



# Aula 23 - Servidor web básico com httpster



**23-servidor\_web\_httpster\public\index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Mi primer sitio web</title>

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

</head>

<body>

<div id="main-container">

<section id="intro-container">

<div class="center-align">

<nav>

<div id="logo"><img src="img/logo.png" alt=""></div>

<div id="menu-lista">

<ul>

<li><a href="#about-section">Acerca de..</a></li>

<li><a href="#work">Proyectos</a></li>

<li><a href="#footer">Contacto</a></li>

</ul>

</div>

</nav>

<div class="section-columns">

<div>

<h2 class="white">Mi primer sitio web en la vida</h2>

<div class="btn-section">

<a href="#about-section" class="btn btn-white">Conóceme más</a>

</div>

</div>

<div id="image-bg">

</div>

</div>

</div>

</section>

<div class="rectangle bg-white skew-left"></div>

<div class="section" id="about-section">

<div class="center-align">

<div class="section-columns">

<div>

<h2>Diseño.<br />Desarrollo.<br/>Inspiración.</h2>

</div>

<div id="bio">

<h3>Mi nombre es <span>Marcos Rivas</span> , soy un ingeniero en Sistemas Computacionales, <span>trabajo</span> <span>en</span> <span>Microsoft</span>, y me dedico al <span>diseño web</span><span> como</span><span> hobbie.</span></h3>

<p>Tengo mi <span>propio canal de Youtube</span> que trata sobre tutoriales y cursos para aprender <span>diseño y programación.</span></p>

<p><span>Gasto mucho de mi tiempo en dibujar</span>, que es otro de mis pasatiempos favoritos, así como seguir aprendiendo de las <span>tecnologías actuales.</span></p>

<p><span>Puedes seguirme</span> en <a href="#">Twitter</a> o <a href="#">Facebook</a>, o directamente en el <a href="#">canal de Youtube</a></p>

<div class="btn-section">

<a href="#work" class="btn btn-black">Ver proyectos sobresalientes</a>

</div>

</div>

</div>

</div>

</div>

<div class="rectangle bg-gray skew-right"></div>

<div class="section bg-gray" id="work">

<div class="center-align">

<div class="section-columns" >

<div>

<h2 class="white">Posters.<br />Sitios Web.<br/>Tutoriales.</h2>

<div class="btn-section">

<a href="" class="btn btn-white">Ver proyectos sobresalientes</a>

</div>

</div>

</div>

</div>

</div>

<div class="section bg-gray" id="work02">

<div class="center-align">

<div class="section-columns" >

<div>

<h2 class="white">Opinión.<br />Impactante.<br/>Blog.</h2>

<div class="btn-section">

<a href="" class="btn btn-white">Ver Proyecto</a>

</div>

</div>

</div>

</div>

</div>

<div class="section bg-gray" id="work03">

<div class="center-align">

<div class="section-columns" >

<div>

<h2 class="white">Patrón<br />Diseño.<br/>Móvil.</h2>

<div class="btn-section">

<a href="" class="btn btn-white">Ver Proyecto</a>

</div>

</div>

</div>

</div>

</div>

<div class="section bg-gray" id="work04">

<div class="center-align">

<div class="section-columns" >

<div>

<h2 class="white">Innovación.<br />Fresco.<br/>Color.</h2>

<div class="btn-section">

<a href="" class="btn btn-white">Ver Proyecto</a>

</div>

</div>

</div>

</div>

</div>

<div class="rectangle bg-black skew-left"></div>

<div class="section bg-black" id="footer">

<div class="center-align">

<div class="section-columns" >

<div>

<h2 class="white">Sigamos en Contacto.</h2>

<div class="btn-section">

<a href="mailto: contacto@misitio.com" class="btn btn-white">Mándame un mensaje</a>

</div>

</div>

<div>

<h3>Contáctame si quieres conocerme y saber cómo podemos colaborar en tu siguiente proyecto.</h3>

<p>También puedes estar conmigo por medio de las redes sociales, enterarte de mi más reciente trabajo y conocerme un poco más.</p>

<div class="btn-section">

<a href="http://www.facebook.com/misitio/" class="btn btn-white">Facebook</a>

<a href="http://www.twitter.com/misitio/" class="btn btn-white">Twitter</a>

<a href="http://www.instagram.com/misitio/" class="btn btn-white">Instagram</a>

</div>

</div>

</div>

</div>

</div>

</div>

</body>

</html>

**23-servidor\_web\_httpster\public\css\main.css**

body{

padding: 0;

margin: 0;

font-family: Arial;

}

span{

position: relative;

}

span::after{ /\* #fffcd6 \*/

background-color: #fffcd6;

content: '';

width: 100%;

height: 50%;

position: absolute;

bottom: -4px;

left: 0;

z-index: -1;

}

p a{ /\* #184DA8 \*/

color: #184DA8;

text-decoration: none;

}

p a:hover{ /\* #184DA8 \*/

text-decoration: underline;

}

.center-align{

margin: 0 auto;

width: 1100px;

}

.black{color: black;}

.white{color: white;}

.section-columns{

}

.section-columns > div{

display: inline-block;

vertical-align: top;

width: 49%;

}

.section-columns h2{

font-size: 72px;

}

#intro-container .section-columns h2{

padding-top: 120px;

padding-bottom: 30px;

}

.section-columns h3{

font-size: 26px;

font-weight: lighter;

line-height: 40px;

}

.section-columns p{

font-weight: lighter;

font-size: 16px;

line-height: 30px;

padding-bottom: 10px;

}

.section-columns #image-bg{

background-image: url('../img/header-background-michael.jpg');

background-repeat: no-repeat;

background-size: 750px;

background-position: -140px -80px;

height: 600px;

}

.section-columns #image-bg-02{

}

.section-columns #bio{

height: 600px;

}

#work, #work02, #work03, #work04{

background-repeat: no-repeat;

background-size: 750px;

background-position: right center;

height: 600px;

padding-top: 150px;

margin-top: -50px;

}

#work {background-image: url('../img/poster-design-series.jpg');}

#work02{background-image: url('../img/nerddiary-blog.jpg');}

#work03{background-image: url('../img/nismo-design-concept.jpg');}

#work04{background-image: url('../img/pixelschupser-mediendesign-website.jpg');}

#footer{

margin-top: -50px;

padding-top: 50px;

height: 550px;

background-image: url('../img/footer-background-mountain.jpg');

background-size: 100%;

background-position: center bottom;

background-repeat: no-repeat;

}

#footer h3{color: white; padding-top: 70px; margin: 0;}

#footer p {color: gray; padding-bottom: 30px;}

.rectangle{

height: 100px;

margin-top: -50px;

}

.bg-white{

background-color: white;

}

.bg-gray{

background-color: #666;

}

.bg-black{

background-color: #000;

}

.skew-left{

transform: skewY(-4deg);

}

.skew-right{

transform: skewY(4deg);

}

.btn{

border: solid 1px;

padding: 20px 40px;

text-decoration: none;

margin-right: 10px;

}

.btn-black{

border-color: black;

color: black;

}

.btn-black:hover{

background-color: black;

border-color: white;

color: white;

}

.btn-white{

border-color: white;

color: white;

}

.btn-white:hover{

background-color: white;

border-color: black;

color: black;

}

#intro-container{

background-color: black;

padding-top: 100px;

}

#intro-container nav{

margin: 0;

padding: 0;

}

#intro-container nav div{

display: inline-block;

vertical-align: top;

}

#intro-container nav #logo{

width: 9%;

}

#intro-container nav #menu-lista{

width: 89%;

text-align: right;

}

#intro-container nav #menu-lista ul{

margin: 0;

padding: 0;

}

#intro-container nav #menu-lista ul li{

display: inline-block;

padding: 0;

margin: 0;

list-style: none;

}

#intro-container nav #menu-lista ul li a{

color: #777;

text-decoration: none;

}

#intro-container nav #menu-lista ul li a:hover{

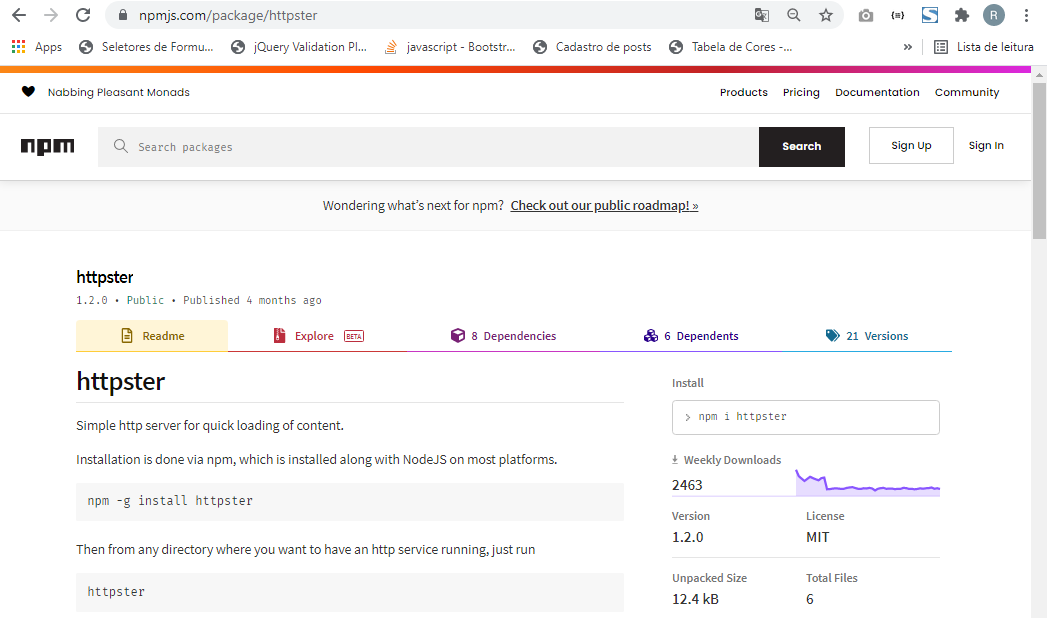
color: white;

text-decoration: white;

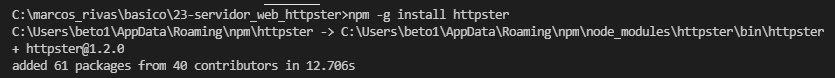
}

## Instalando o pacote httpster

https://www.npmjs.com/package/httpster



npm -g install httpster

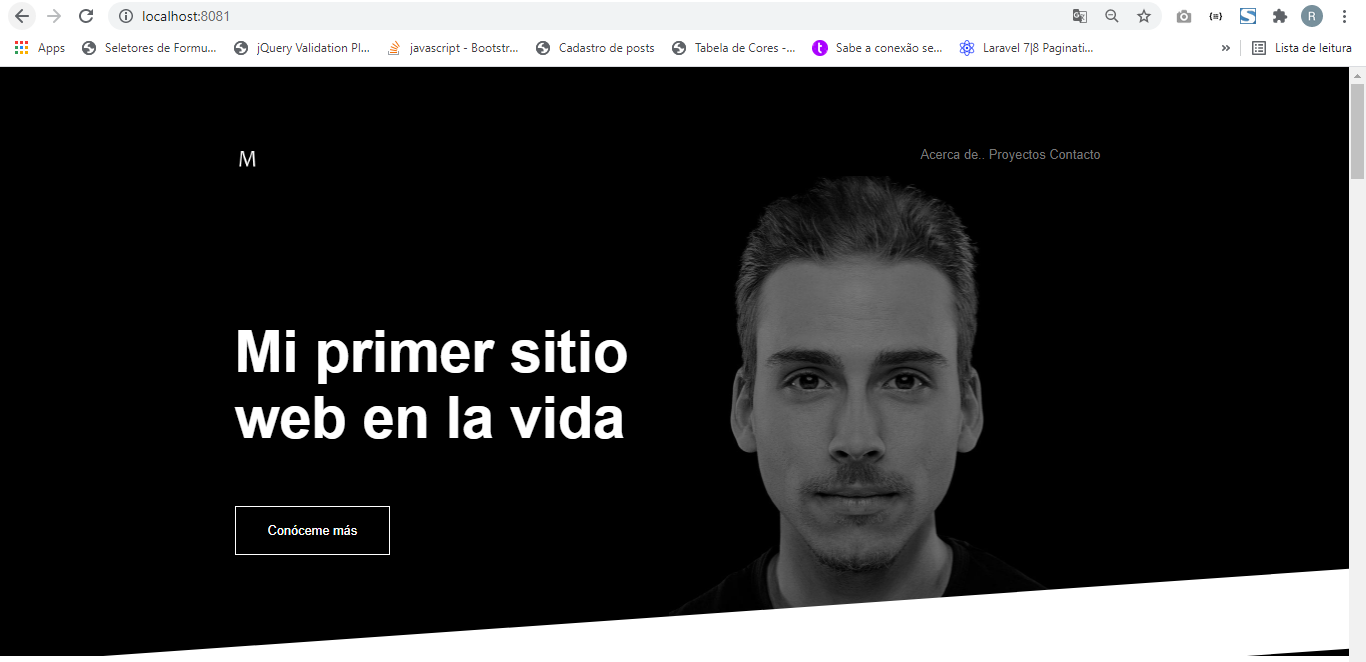


httpster -p 8081 -d ./public



- No browser:

localhost:8081



# Aula 24 - Reinício automático de servidor

**24-reinicio\index.js**

const http = require('http');

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

res.writeHead(200, {'content-type': 'text/html'});

res.end(`

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

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

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

<title>Document</title>

</head>

<body>

<h1>Website de tutoriais</h1>

<h3>Bem-vindo ao meu site!</h3>

</body>

</html>

`);

}).listen(3000);

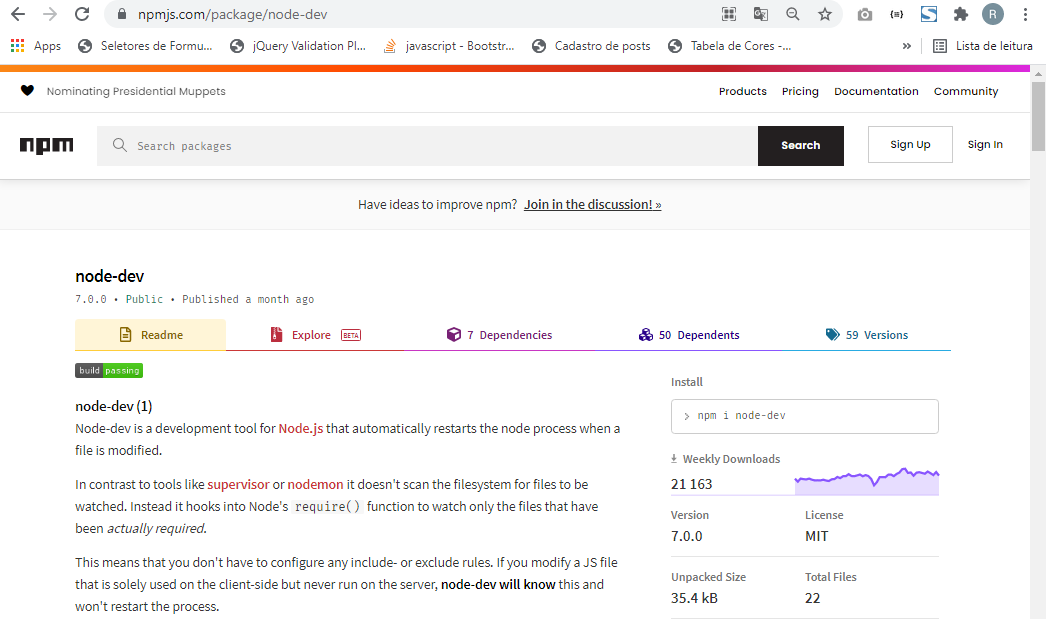
console.log('Servidor iniciado...');

node index.js

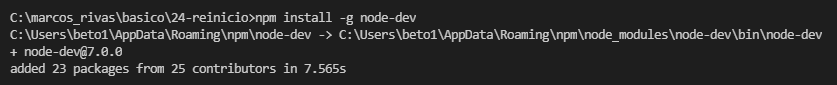


## Instalando o pacote node-dev

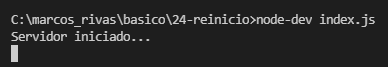
https://www.npmjs.com/package/node-dev



npm install -g node-dev



node-dev index.js



**24-reinicio\index.js**

const http = require('http');

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

res.writeHead(200, {'content-type': 'text/html'});

res.end(`

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

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

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

<title>Document</title>

</head>

<body>

<h1>Website de tutoriais</h1>

<h3>Meu nome é Roberto Pinheiro</h3>

<h3>Bem-vindo ao meu site!</h3>

</body>

</html>

`);

}).listen(3000);

console.log('Servidor iniciado...');

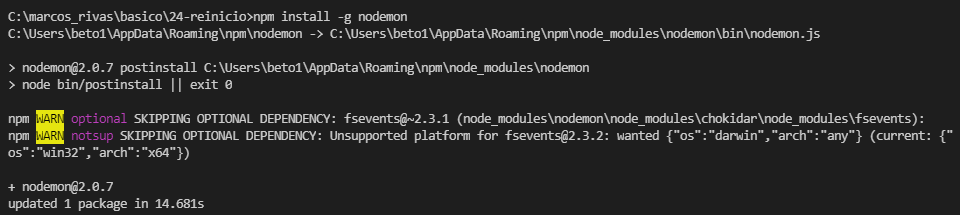
- No browser:

localhost:3000

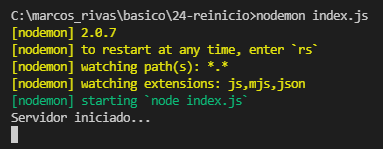


## Instalando o pacote nodemon

npm install -g nodemon



nodemon index.js



**24-reinicio\index.js**

const http = require('http');

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

res.writeHead(200, {'content-type': 'text/html'});

res.end(`

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

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

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

<title>Document</title>

</head>

<body>

<h1>Website de tutoriais</h1>

<h3>Tutoriais de desenvolvimento web</h3>

<h3>Bem-vindo ao meu site!</h3>

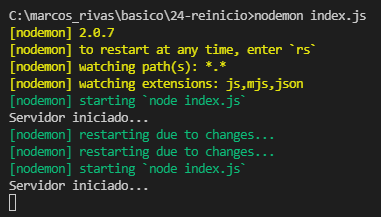
</body>

</html>

`);

}).listen(3000);

console.log('Servidor iniciado...');

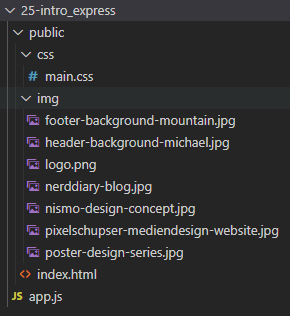


- No browser:

localhost:3000

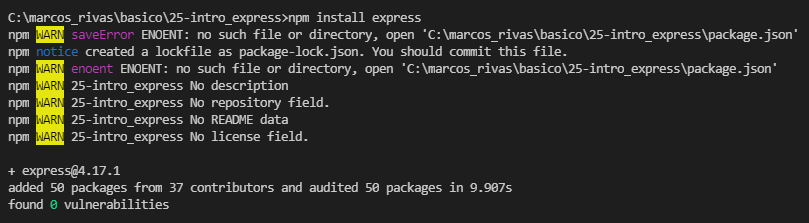


# Aula 25 - Introdução a Express



## Instalando o pacote Express

npm install express



**25-intro\_express\app.js**

const express = require('express');

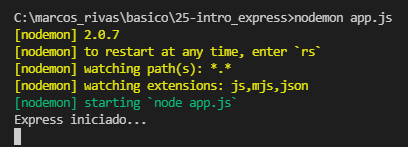
var app = express();

app.use(express.static('./public'));

app.listen(3000);

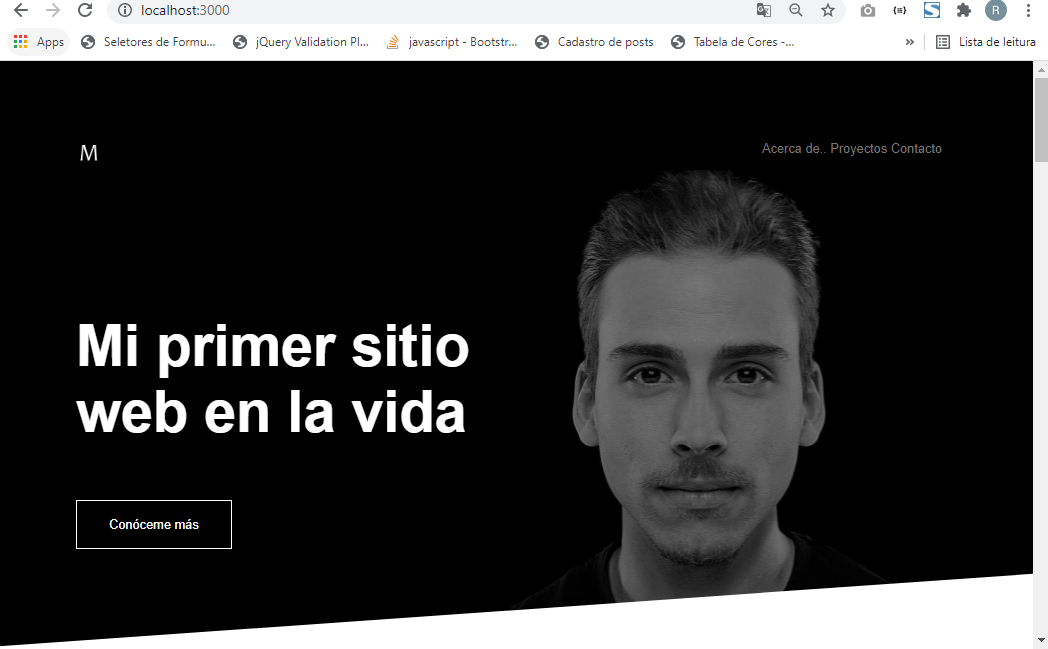
console.log('Express iniciado...');

nodemon app.js

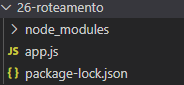


- No browser:

localhost:3000



# Aula 26 - Roteamento em Express



**26-roteamento\app.js**

const express = require('express');

const port = 3000;

const app = express();

const pedidos = [

{'nome': 'sopa', 'quantidade': 2, 'status': 'servido', 'mesa': 5},

{'nome': 'corte de carne', 'quantidade': 2, 'status': 'processo', 'mesa': 4},

{'nome': 'sopa', 'quantidade': 1, 'status': 'pago', 'mesa': 3},

{'nome': 'salada', 'quantidade': 1, 'status': 'pago', 'mesa': 5},

{'nome': 'massa', 'quantidade': 4, 'status': 'em processo', 'mesa': 1},

];

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

res.send('Olá a todos!');

});

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

res.send('Página inicial');

});

app.get('/pedidos-em-processo', (req, res) =>{

pedidosEmProcesso(res);

});

app.get('/pedidos-por-nome/:nome', (req, res) =>{

pedidosPorNome(req.params.nome,res);

});

app.listen(port, () =>{

console.log(`Servidor iniciado na porta ${port}`);

});

const pedidosEmProcesso = (res) =>{

const arrayRes = pedidos.filter(item =>{

return item.status == 'em processo';

});

res.json(arrayRes);

}

const pedidosPorNome = (nome, res) =>{

const arrayRes = pedidos.filter(item =>{

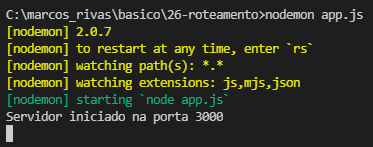
return item.nome == nome;

});

res.json(arrayRes);

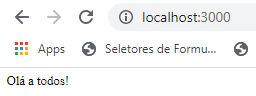
}

nodemon app.js

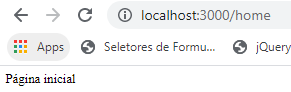


- No browser:

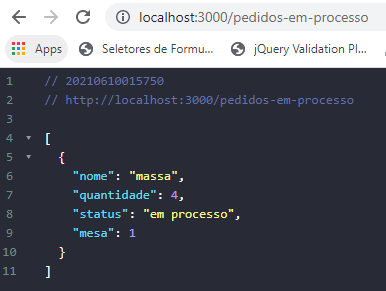
http://localhost:3000/



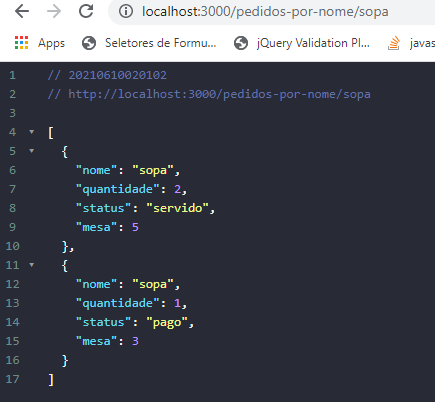
http://localhost:3000/home



http://localhost:3000/pedidos-em-processo



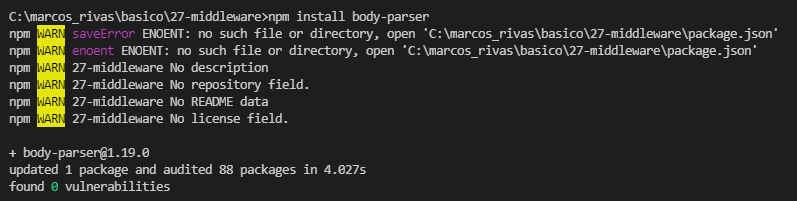
http://localhost:3000/pedidos-por-nome/sopa



# Aula 27 - Exemplo de middleware em Express

## Instalando body-parser

npm install body-parser



**27-middleware\app.js**

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

const urlencodedParser = bodyParser.urlencoded({extended: false});

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

res.send(`<h1>Teus dados são: <br/>

Nome: ${req.query.nome} <br/>

Sobrenome: ${req.query.sobrenome} </h1>`);

console.log(req.query);

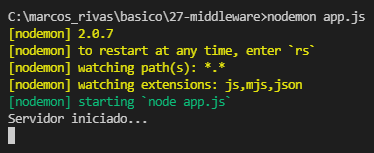
});

app.listen(3000, () =>{

console.log('Servidor iniciado...');

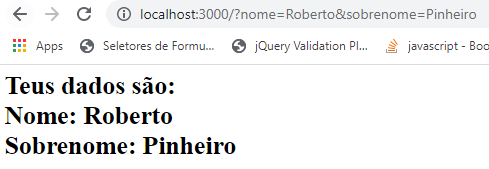
});

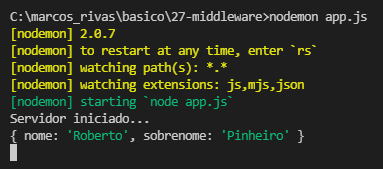
nodemon app.js



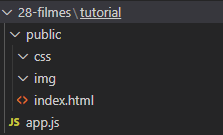
- No browser:

http://localhost:3000/?nome=Roberto&sobrenome=Pinheiro



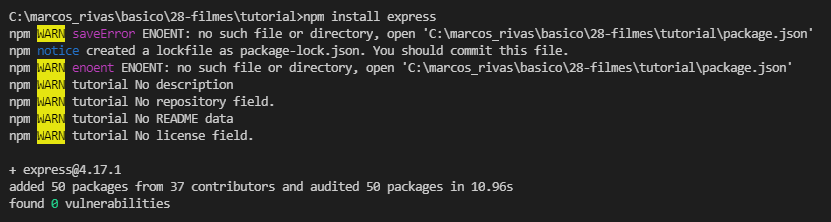


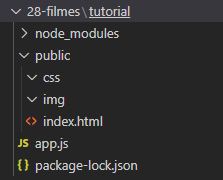
# Aula 28 - Exemplo de Express JS com AJAX - Parte 1/2



## Instalando Express

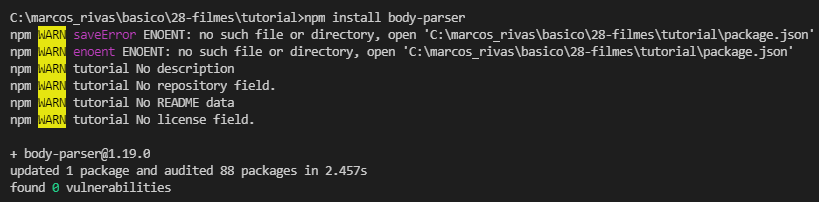
npm install express





## Instalando body-parser

npm install body-parser



Obs.: Nodemon já havia sido instalado de forma global.

**28-filmes\tutorial\app.js**

const express = require('express');

const bodyParser = require('body-parser');

const path = require('path');

const fs = require('fs');

const app = express();

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

app.use(express.static('./public'));

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

res.setHeader('Content-type', 'text/html');

res.sendFile('./public/index.html');

})

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

const file = fs.readFileSync('./filmes.json', 'UTF-8');

console.log(file);

res.setHeader('Content-type', 'text/json');

res.send(file);

});

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

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

const nome = req.body.nome;

const rating = req.body.rating;

// abrir arquivo

let file = fs.readFileSync('./filmes.json', 'UTF-8');

// converte-lo em um array

const json = JSON.parse(file);

// inserir um novo elemento

json.filmes.push({"nome": nome, "rating": parseInt(rating)});

// gravar json no arquivo

file = fs.writeFileSync('./filmes.json', JSON.stringify(json));

res.send('Dados gravados com sucesso!');

});

app.listen(3001, () =>{

console.log('Servidor iniciado...');

});

**28-filmes\tutorial\filmes.json**

{"filmes": [{"nome": "Harry Potter 4","rating": 9}]}

**28-filmes\tutorial\public\index.html**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

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

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

<title>Filmes</title>

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

</head>

<body>

<div id="main-container">

<h1>Registro de filmes</h1>

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

<p>

<label for="nome">Nome do filme</label><br/>

<input type="text" name="nome" id="nome">

</p>

<p>

<label for="rating">Qualificação</label><br/>

<input type="range" min="0" max="10" name="rating" id="rating">

</p>

<input type="submit" value="Registrar novo filme" id="bSubmit">

</form>

<div id="filmes-container">

</div>

</div>

</body>

</html>

**28-filmes\tutorial\public\css\main.css**

body{

font-family: Arial, Helvetica, sans-serif;

padding: 0;

margin: 0;

}

form{

padding-bottom: 20px;

}

#main-container{

margin: 0 auto;

width: 400px;

}

input[type="text"]{

padding: 10px;

width: 350px;

}

input[type="button"]{

color: white;

background-color: rgba(18, 69, 163, 255);

padding: 10px;

border: 0;

}

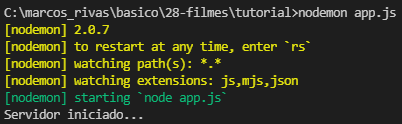
#filmes-container div{

padding: 10px 0px;

border-bottom: solid 1px #ccc;

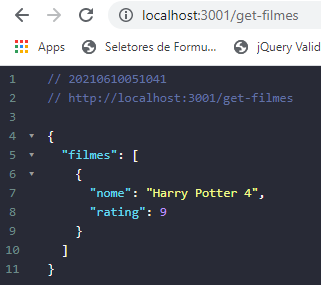
}

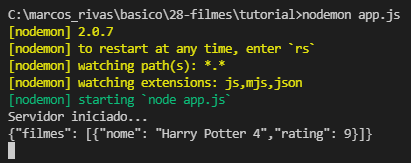
nodemon app.js



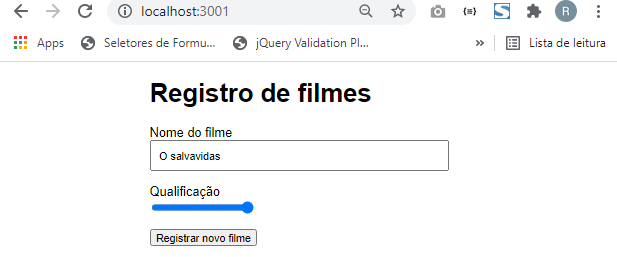
- No browser:

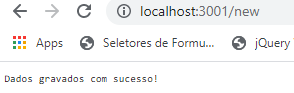
http://localhost:3001/get-filmes





localhost:3001

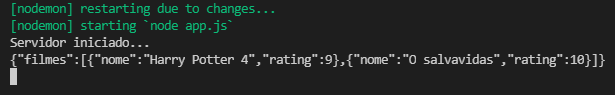




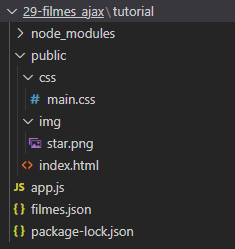
- No browser:

http://localhost:3001/get-filmes





# Aula 29 - Exemplo de Express JS com AJAX - Parte 2/2



**29-filmes\_ajax\tutorial\app.js**

const express = require('express');

const bodyParser = require('body-parser');

const path = require('path');

const fs = require('fs');

const app = express();

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

app.use(express.static('./public'));

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

res.setHeader('Content-type', 'text/html');

res.sendFile('./public/index.html');

})

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

const file = fs.readFileSync('./filmes.json', 'UTF-8');

console.log(file);

res.setHeader('Content-type', 'text/json');

res.send(file);

});

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

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

const nome = req.body.nome;

const rating = req.body.rating;

// abrir arquivo

let file = fs.readFileSync('./filmes.json', 'UTF-8');

// converte-lo em um array

const json = JSON.parse(file);

// inserir um novo elemento

json.filmes.push({"nome": nome, "rating": parseInt(rating)});

// gravar json no arquivo

file = fs.writeFileSync('./filmes.json', JSON.stringify(json));

res.send('Dados gravados com sucesso!');

});

app.listen(3001, () =>{

console.log('Servidor iniciado...');

});

**29-filmes\_ajax\tutorial\public\index.html**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

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

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

<title>Filmes</title>

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

</head>

<body>

<div id="main-container">

<h1>Registro de filmes</h1>

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

<p>

<label for="nome">Nome do filme</label><br/>

<input type="text" name="nome" id="nome">

</p>

<p>

<label for="rating">Qualificação</label><br/>

<input type="range" min="0" max="10" name="rating" id="rating">

</p>

<input type="submit" value="Registrar novo filme" id="bSubmit">

</form>

<div id="filmes-container">

</div>

</div>

</body>

</html>

<script>

const bSubmit = document.querySelector('#bSubmit');

bSubmit.addEventListener('click', () =>{

//validar campos

const nome = document.querySelector('#nome').value;

const rating = document.querySelector('#rating').value;

if(nome.trim() === '' || rating.trim() === '') return false;

// enviar requisição POST a /new

fetch('/new', {

method: 'POST',

headers: {'Content-type': 'application/json'},

body: JSON.stringify({nome: nome, rating: rating})

})

.then(res => res.text())

.then(data =>{

// mostrar mensagem de erro/sucesso

alert(data);

//atualizar lista de filmes

loadFilmes();

});

});

function loadFilmes(){

fetch('/get-filmes', {method: 'GET'})

.then(res => res.json())

.then(data =>{

const filmes = document.querySelector('#filmes-container');

let html = '';

data.filmes.forEach(filme =>{

html += `<div>${filme.nome} <img src="img/star.png" width="16" />${filme.rating}</div>`

});

filmes.innerHTML = html;

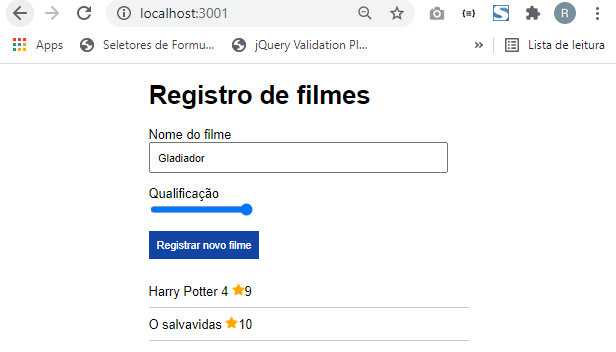
});

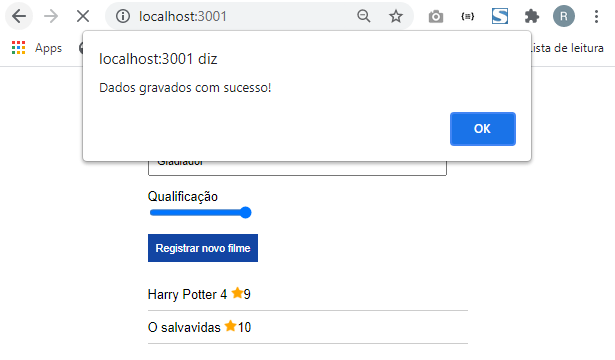
}

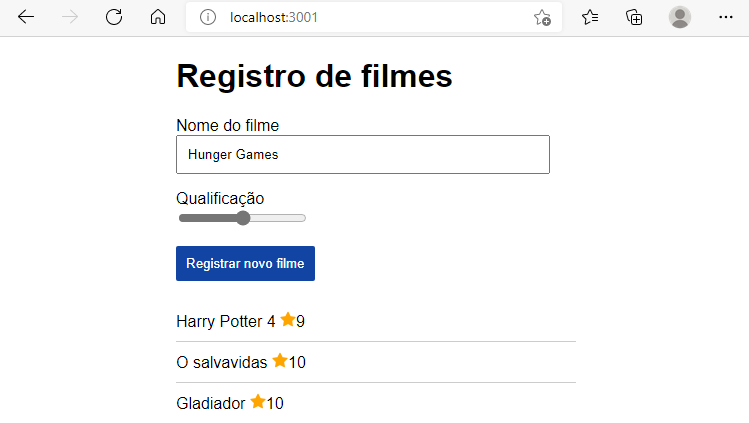
</script>

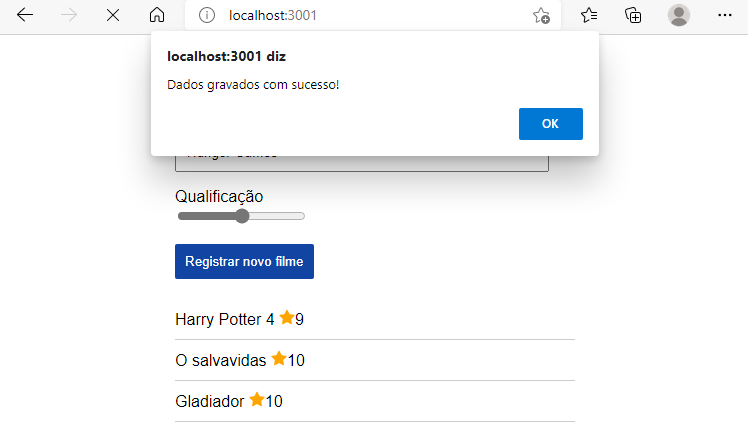
- No browser:

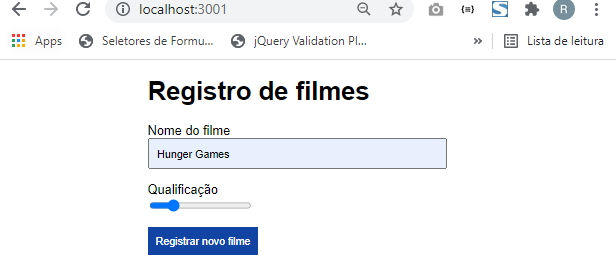
localhost:3001

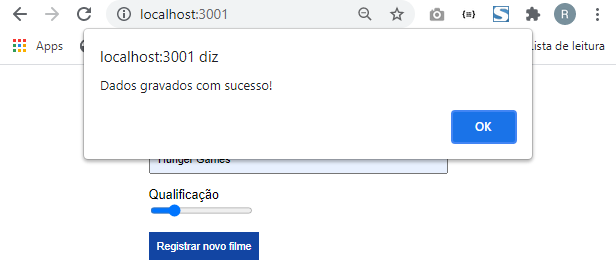


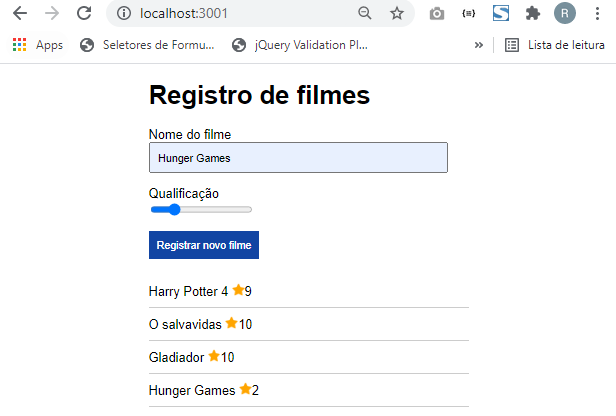








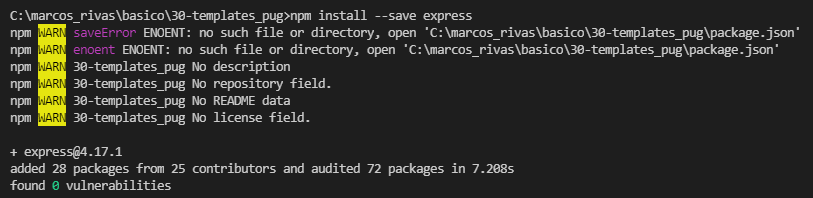




# Aula 30 - Tutorial de Pug para fazer templates dinâmicos

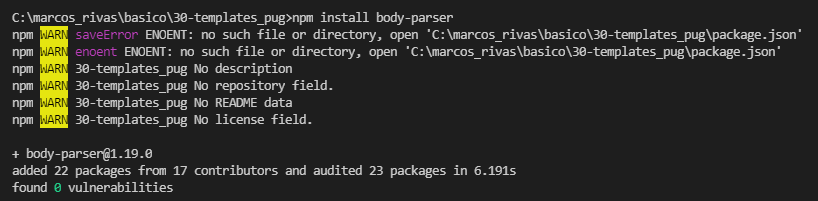
## Instalando o Express

npm install --save Express



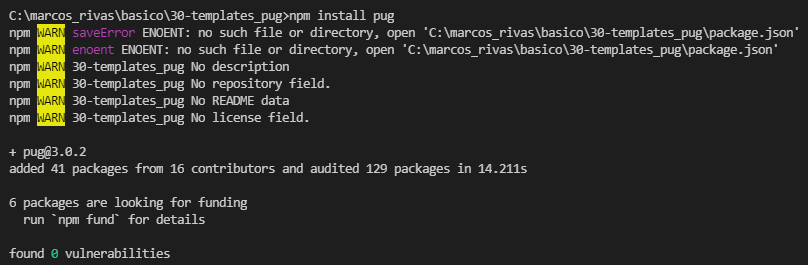
## Instalando o body-parser

npm install body-parser

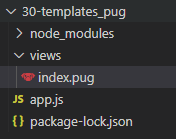


## Instalando pug

npm install pug



- No diretório raiz da aplicação crie uma pasta chamada "views" e dentro dela um arquivo chamado "index.pug":



**30-templates\_pug\app.js**

const express = require('express');

const app = express();

const bodyParser = require('body-parser');

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

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

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

res.render('index', {titulo: "Minha primeira app", nome: "Roberto Pinheiro"});

});

app.listen(3002, () =>{

console.log('Servidor iniciado...');

});

**30-templates\_pug\views\index.pug**

html

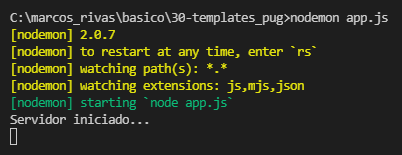
head

title= titulo

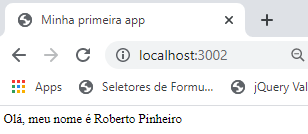
body

div Olá, meu nome é #{nome}

node app.js



localhost:3002



# Aula 31 - Tutorial de Mustache para fazer templates dinâmicas

## Instalando mustache-express



npm install mustache-express



**31-templates\_mustache\app.js**

const express = require('express');

const app = express();

const bodyParser = require('body-parser');

const mustacheExpress = require('mustache-express');

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

app.engine('.mustache', mustacheExpress());

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

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

const dados = [

{nome: 'Helena', idade: 10},

{nome: 'Luis', idade: 40}

];

res.render('index',{

titulo: 'Minha primeira app',

nome: 'Juan Perez',

dados: dados

});

});

app.listen(3002, () =>{

console.log('Servidor iniciado...');

});

**31-templates\_mustache\views\index.mustache**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

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

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

<title>{{titulo}}</title>

</head>

<body>

Olá, meu nome é {{nome}}.

{{#dados}}

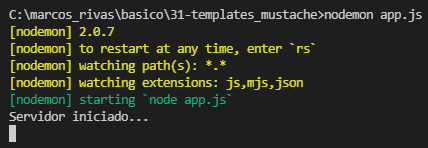
<li>{{nome}} - {{idade}}</li>

{{/dados}}

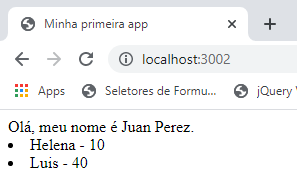
</body>

</html>

node app.js

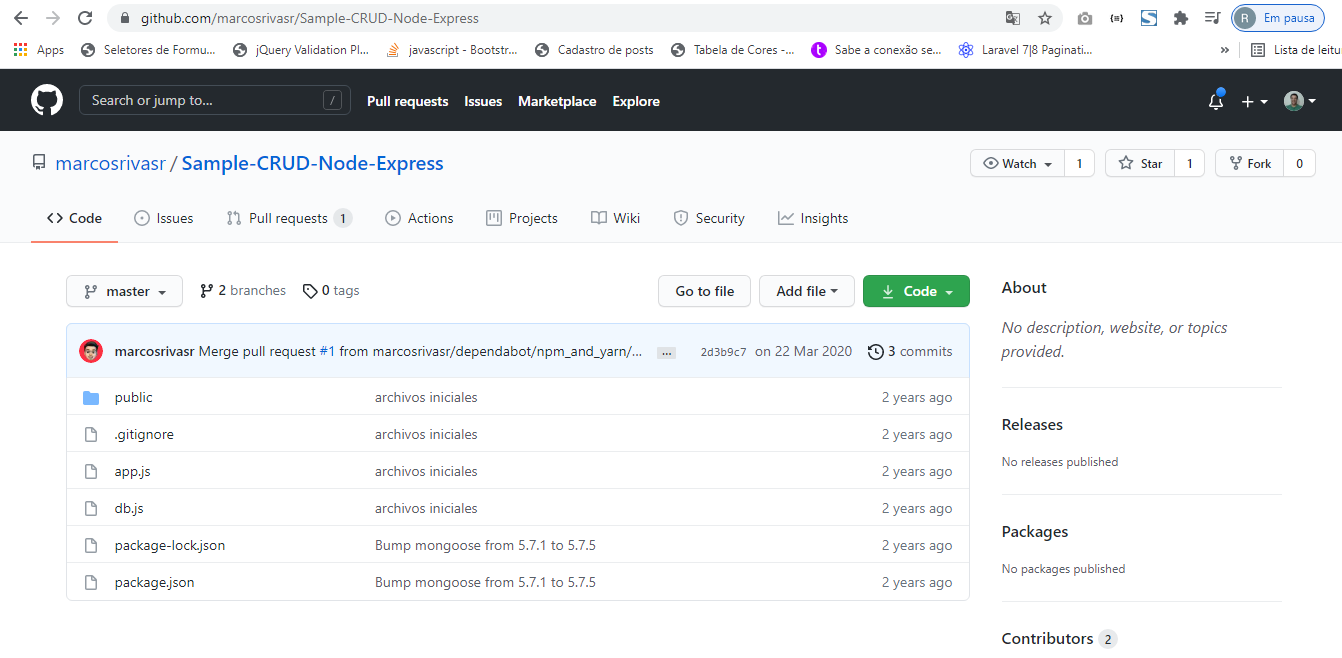


localhost:3002



# Aula 32 - CRUD - Conexão a base de dados

https://github.com/marcosrivasr/Sample-CRUD-Node-Express



## Instalando e configurando o Mongo Express

O Mongo Express é uma interface para gerenciamento de banco de dados.

- Instale o Mongo Express de forma global em seu computador:

npm install -g mongo-express

- Acesse a pasta:

C:\Users\beto1\AppData\Roaming\npm\node\_modules\mongo-express

- Crie um arquivo chamado "config.js" e copie o conteúdo do arquivo "config.default.js" para dentro dele.

- Dentro deste arquivo há um trecho que permite alterar o nome de usuário e a senha para acessar o aplicativo:

basicAuth: {

username: getFileEnv(basicAuthUsername) || 'admin',

password: getFileEnv(basicAuthPassword) || 'pass',

},

- Por segurança é recomendado alterá-lo, porém se isso não for feito, os valores padrões são:

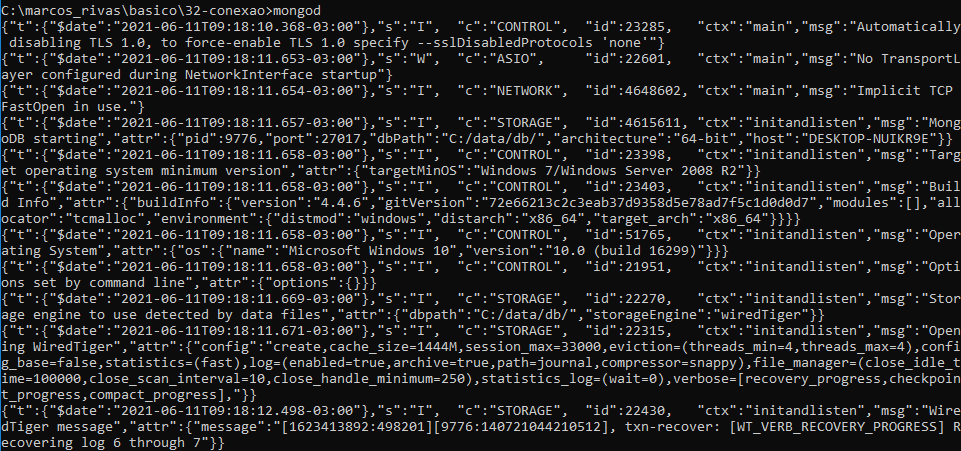
user: admin

password: pass

## Rodando o Mongo Express

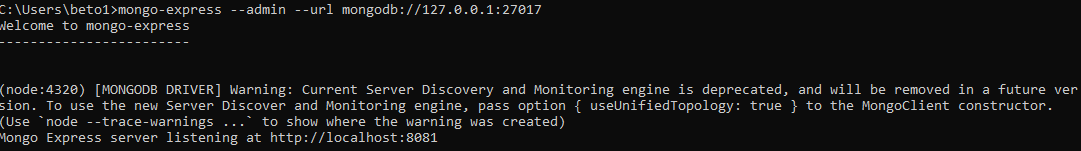
- Inicialmente rode o servidor do Mongo. No terminal entre com o comando:

mongod



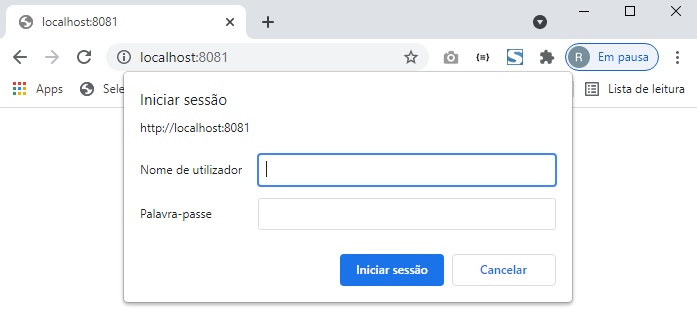
- Para rodar o Mongo Express, no terminal, entre com o comando:

mongo-express --admin --url mongodb://127.0.0.1:27017

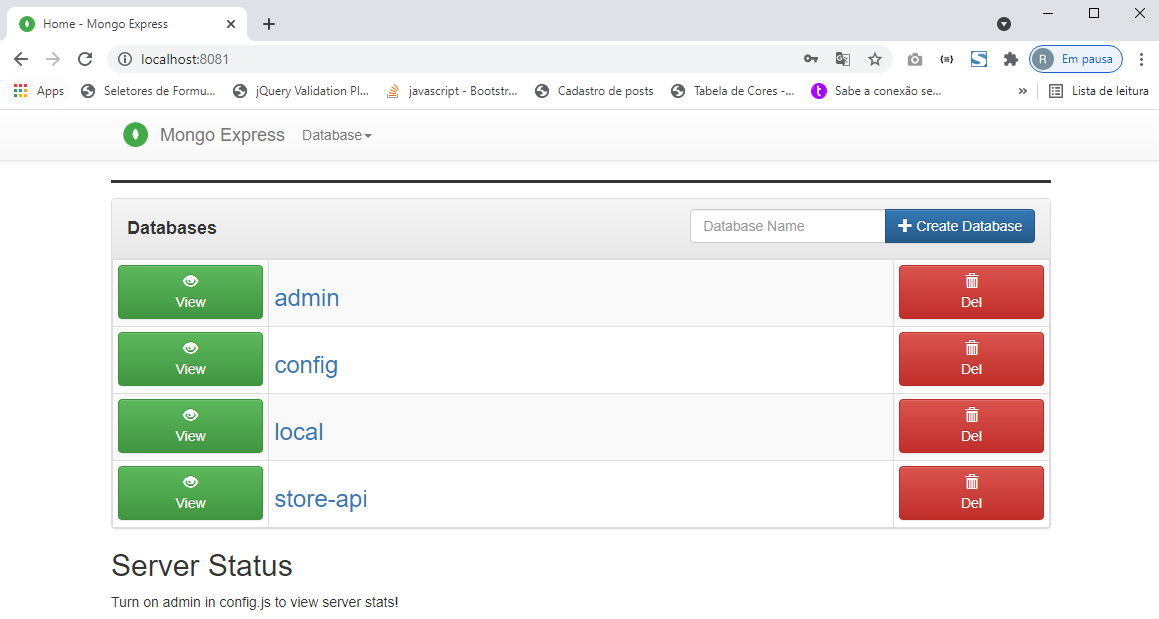


- No browser:

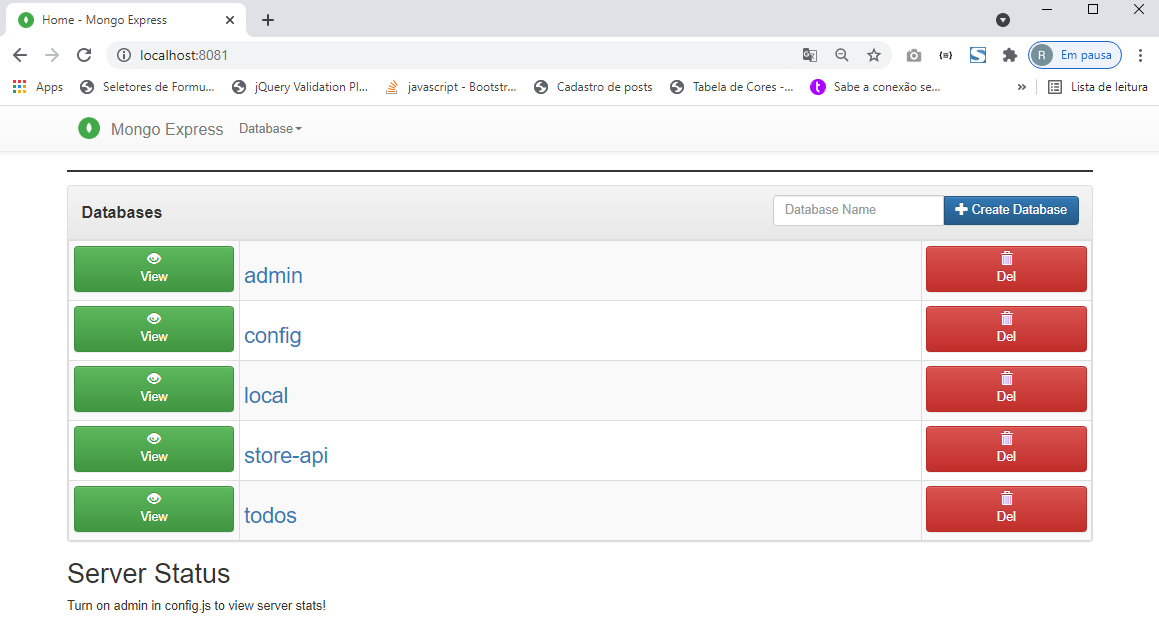
localhost:8081



- Entre com os valores configurados e clique no botão "Iniciar sessão"



- Crie um database com o nome "todos"

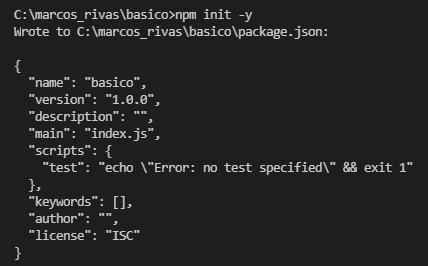


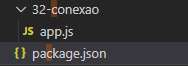
- No Visual Studio Code, na pasta raiz do projeto, adicione um arquivo chamado "app.js":



- No terminal, entre com o comando:

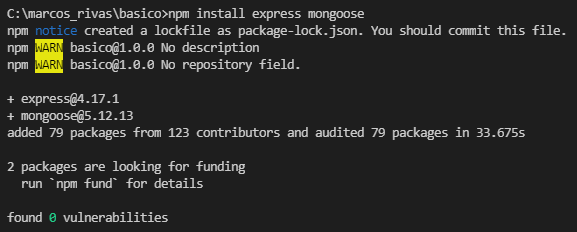
npm init -y





## Instalando os pacotes Express e Mongoose

npm install express mongoose



package.json

{

"name": "basico",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

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

},

"keywords": [],

"author": "",

"license": "ISC",

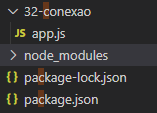
"dependencies": {

"express": "^4.17.1",

"mongoose": "^5.12.13"

}

}



## Instalando o body-parser de forma global

npm install -g body-parser

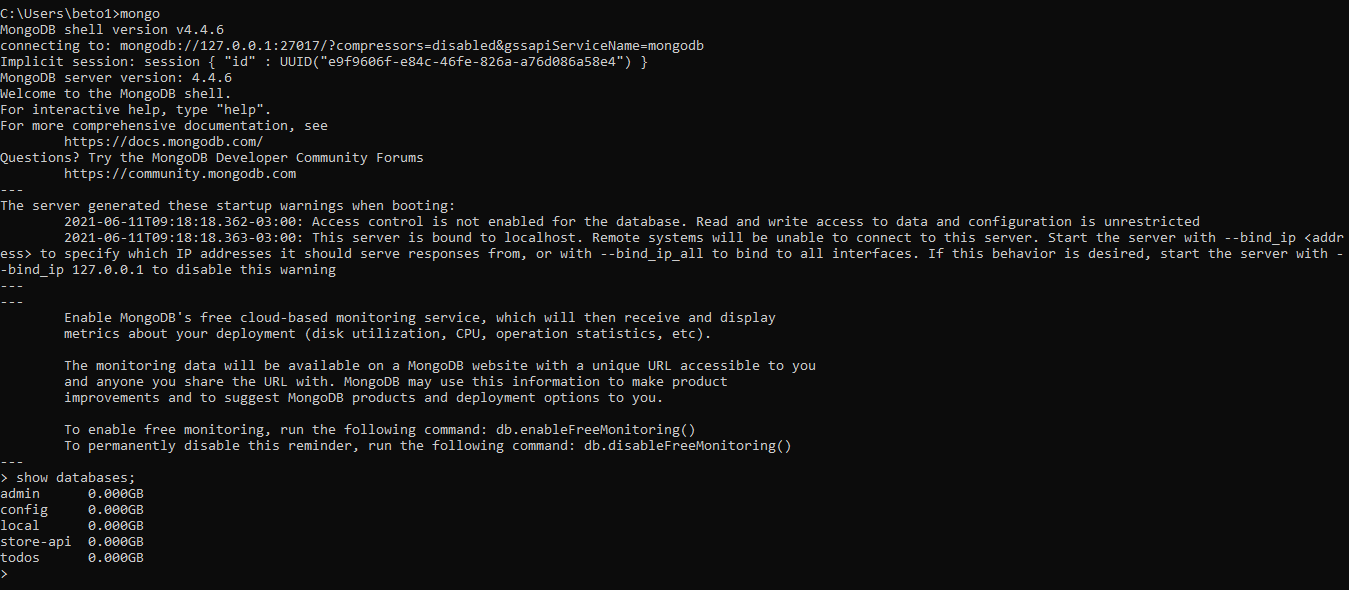


## Criando um usuário para o BD "todos" no Mongo Express

- No terminal, entre com o comando:

mongo

show databases;



use todos;



db.createUser({

user: 'dev',

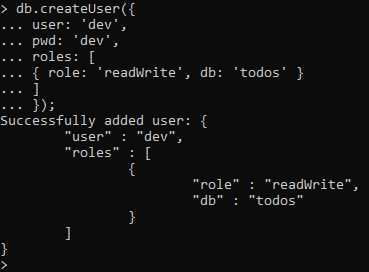
pwd: 'dev',

roles: [

{ role: 'readWrite', db: 'todos' }

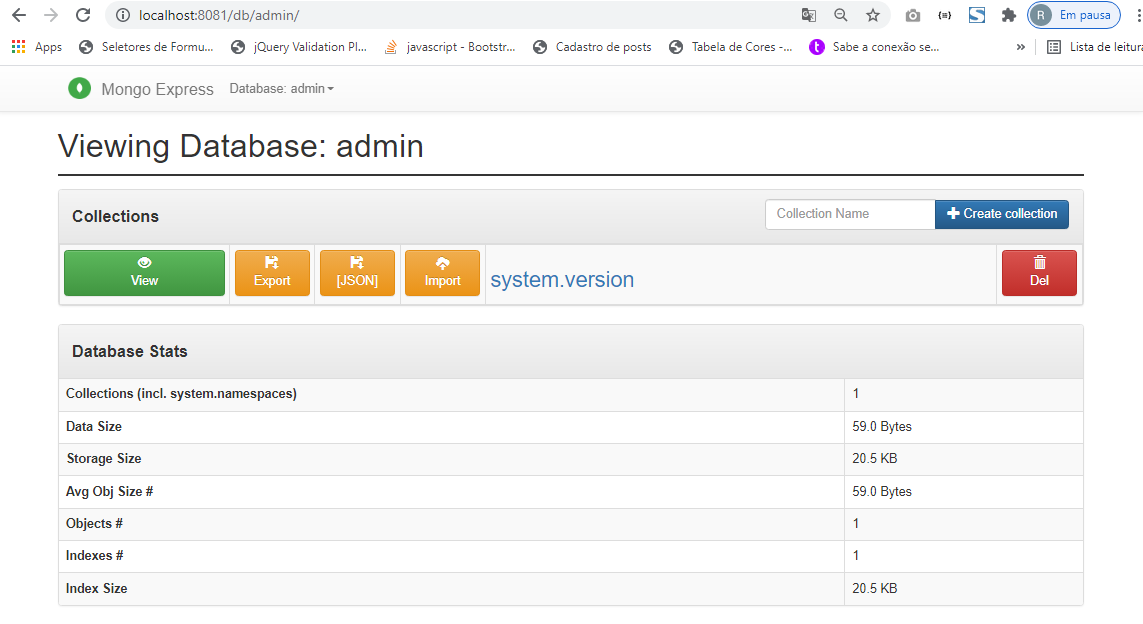
]

});

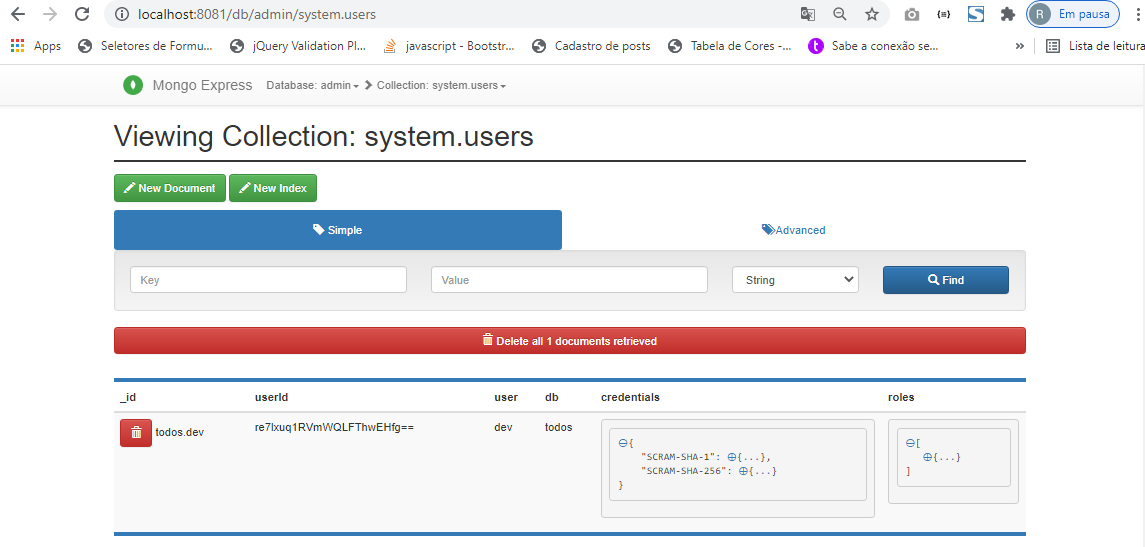


- No Mongo Express:

- Clique no link "admin":



- Clique no link "system.users":



## Criando a conexão com a base de dados

**app.js**

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const app = express();

// conexão com o banco de dados

mongoose.connect('mongodb://dev:dev@localhost:27017/todos', {useNewUrlParser: true, useUnifiedTopology: true});

const connection = mongoose.connection;

connection.once('open', () =>{

console.log('Conectado a base de dados!');

});

connection.on('error', (err) =>{

console.log('Erro ao se conectar a base de dados: ', err);

});

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

res.json({response: 'success'});

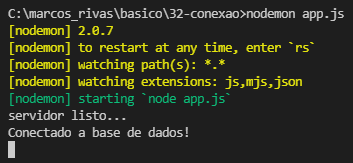
});

app.listen(3000, () =>{

console.log('servidor listo...');

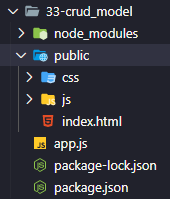
});

node app.js



# Aula 33 - CRUD - Criando o modelo

- Dentro da pasta raiz do projeto adicione uma subspasta chamada "public" e dentro desta pasta adicione um arquivo chamado "index.html" e duas subpastas: "css" e "js":



**app.js**

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const app = express();

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

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

// conexão com o banco de dados

mongoose.connect('mongodb://dev:dev@localhost:27017/todos', {useNewUrlParser: true, useUnifiedTopology: true});

const connection = mongoose.connection;

connection.once('open', () =>{

console.log('Conectado a base de dados!');

});

connection.on('error', (err) =>{

console.log('Erro ao se conectar a base de dados: ', err);

});

//modelo

const Todo = mongoose.model('Todo', {text: String, completed: Boolean});

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

res.json({response: 'success'});

});

app.listen(3000, () =>{

console.log('servidor listo...');

});

**33-crud\_model\public\index.html**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8" />

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

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

<title>CRUD Node.js + Express.js + MongoDB</title>

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

</head>

<body>

<div id="main-container">

<form action="/add" method="post">

<input

type="text"

name="text"

id=""

autocomplete="off"

placeholder="Insira um novo todo"

/>

</form>

<div id="todos"></div>

</div>

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

</body>

</html>

**33-crud\_model\public\css\style.css**

body {

font-family: Arial, Helvetica, sans-serif;

font-size: 24px;

}

#main-container {

margin: 0 auto;

width: 600px;

}

#main-container form input {

font-size: 24px;

padding: 20px;

width: 558px;

}

#todos .todo {

width: 100%;

margin: 10px 0;

}

#todos .todo div {

display: inline-block;

padding: 10px 5px;

}

#todos .todo .checkbox-container {

width: 25px;

}

#todos .todo .checkbox-container input {

transform: scale(2);

}

#todos .todo .text-container {

color: black;

width: 495px;

}

#todos .todo .completed {

color: #ccc;

text-decoration: line-through;

}

#todos .todo .actions-container a {

color: #ccc;

font-weight: bolder;

text-decoration: none;

padding: 10px;

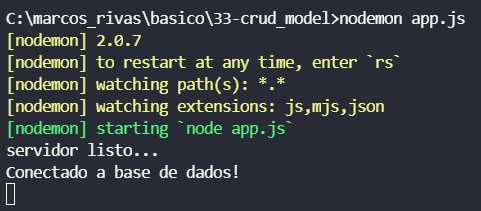
}

#todos .todo .actions-container a:hover {

color: rgb(30, 71, 136);

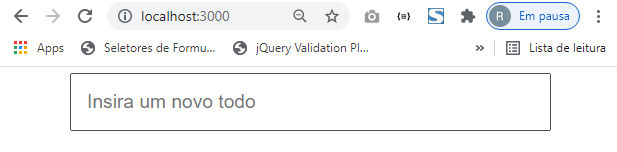
}

nodemon app.js



- No browser:

localhost:3000



# Aula 34 - CRUD - create

**34-crud\_create\app.js**

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const app = express();

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

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

// conexão com o banco de dados

mongoose.connect('mongodb://dev:dev@localhost:27017/todos', {useNewUrlParser: true, useUnifiedTopology: true});

const connection = mongoose.connection;

connection.once('open', () =>{

console.log('Conectado a base de dados!');

});

connection.on('error', (err) =>{

console.log('Erro ao se conectar a base de dados: ', err);

});

//modelo

const Todo = mongoose.model('Todo', {text: String, completed: Boolean});

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

const todo = new Todo({text: req.body.text, completed: false});

todo.save()

.then(doc =>{

console.log('Dado inserido com sucesso!', doc);

//res.redirect('/');

res.json({response: 'success'});

})

.catch(err =>{

console.log('Erro ao inserir ', err.message);

res.status(400).json({response: 'failed'});

});

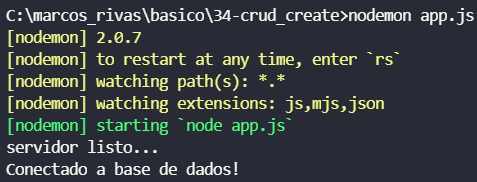
});

app.listen(3000, () =>{

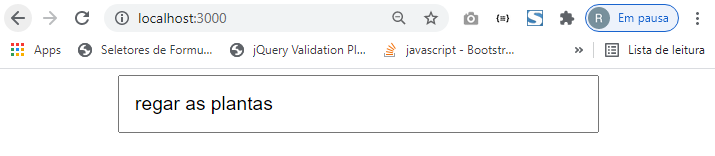
console.log('servidor listo...');

});

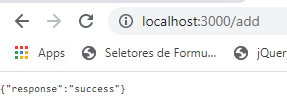
nodemon app.js

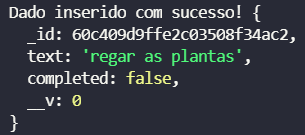


localhost:3000



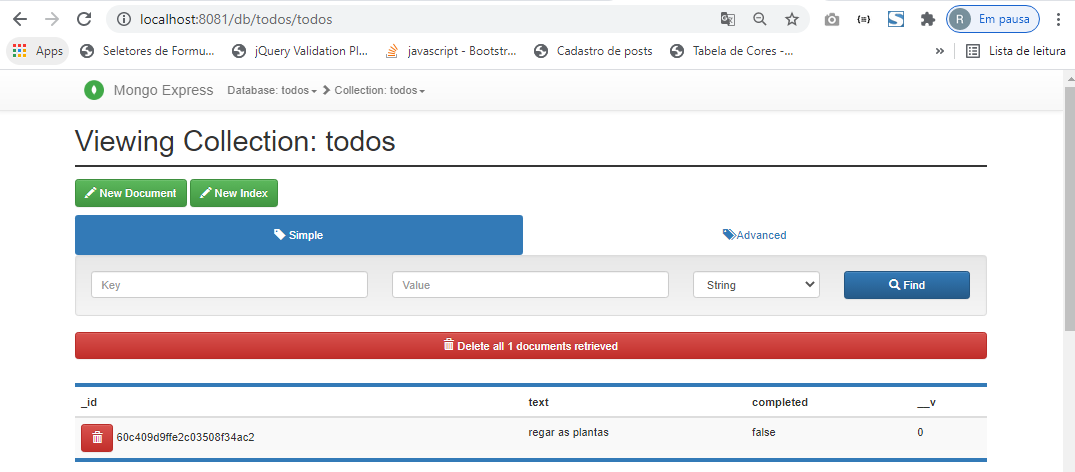
http://localhost:3000/add



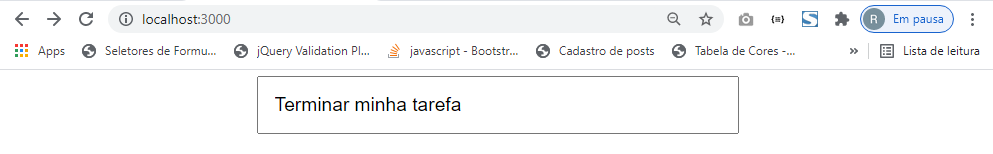


- No Mongo Express:

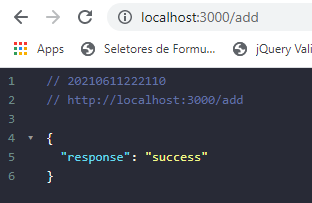
mongo-express --admin --url mongodb://127.0.0.1:27017

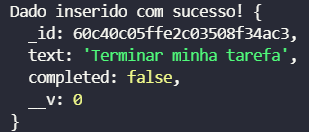


localhost:3000

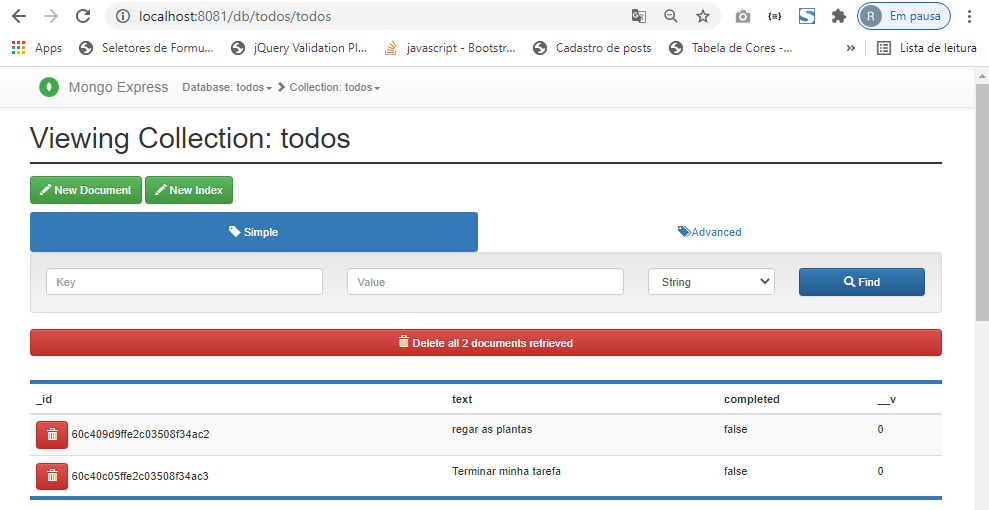


localhost:3000/add





- No Mongo Express:



# Aula 35 - CRUD - consulta

**app.js**

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const app = express();

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

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

// conexão com o banco de dados

mongoose.connect('mongodb://dev:dev@localhost:27017/todos', {useNewUrlParser: true, useUnifiedTopology: true});

const connection = mongoose.connection;

connection.once('open', () =>{

console.log('Conectado a base de dados!');

});

connection.on('error', (err) =>{

console.log('Erro ao se conectar a base de dados: ', err);

});

//modelo

const Todo = mongoose.model('Todo', {text: String, completed: Boolean});

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

const todo = new Todo({text: req.body.text, completed: false});

todo.save()

.then(doc =>{

console.log('Dado inserido com sucesso!', doc);

//res.redirect('/');

res.json({response: 'success'});

})

.catch(err =>{

console.log('Erro ao inserir ', err.message);

res.status(400).json({response: 'failed'});

});

});

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

Todo.find({},'text completed')

.then(doc =>{

res.json({response: 'success', data: doc});

})

.catch(err =>{

console.log('Erro ao consultar elementos!', err.message);

res.status(400).json({response: 'failed'});

});

});

app.listen(3000, () =>{

console.log('servidor listo...');

});

http://localhost:3000/getall



# Aula 36 - CRUD - update

**app.js**

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const app = express();

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

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

// conexão com o banco de dados

mongoose.connect('mongodb://dev:dev@localhost:27017/todos', {useNewUrlParser: true, useUnifiedTopology: true});

const connection = mongoose.connection;

connection.once('open', () =>{

console.log('Conectado a base de dados!');

});

connection.on('error', (err) =>{

console.log('Erro ao se conectar a base de dados: ', err);

});

//modelo

const Todo = mongoose.model('Todo', {text: String, completed: Boolean});

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

const todo = new Todo({text: req.body.text, completed: false});

todo.save()

.then(doc =>{

console.log('Dado inserido com sucesso!', doc);

//res.redirect('/');

res.json({response: 'success'});

})

.catch(err =>{

console.log('Erro ao inserir ', err.message);

res.status(400).json({response: 'failed'});

});

});

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

Todo.find({},'text completed')

.then(doc =>{

res.json({response: 'success', data: doc});

})

.catch(err =>{

console.log('Erro ao consultar elementos!', err.message);

res.status(400).json({response: 'failed'});

});

});

app.get('/complete/:id/:status', (req, res) =>{

const id = req.params.id;

const status = req.params.status == 'true'; // converter para booleano

Todo.findByIdAndUpdate({\_id: id},{$set: {completed: status}})

.then(doc =>{

res.json({response: 'success'});

})

.catch(err =>{

console.log('Error aol atualizar dado ', err.message);

res.status(400).json({response: 'failed'});

});

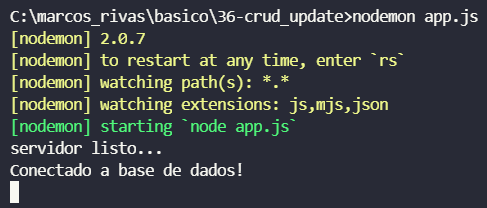
});

app.listen(3000, () =>{

console.log('servidor listo...');

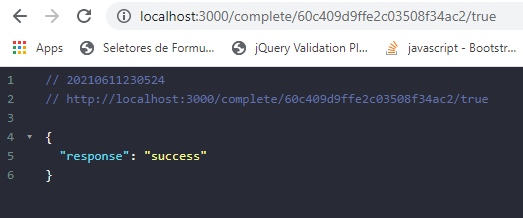
});

nodemon app.js



- No browser:

http://localhost:3000/complete/60c409d9ffe2c03508f34ac2/true



http://localhost:3000/getall



# Aula 37 - CRUD - delete

**app.js**

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const app = express();

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

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

// conexão com o banco de dados

mongoose.connect('mongodb://dev:dev@localhost:27017/todos', {useNewUrlParser: true, useUnifiedTopology: true});

const connection = mongoose.connection;

connection.once('open', () =>{

console.log('Conectado a base de dados!');

});

connection.on('error', (err) =>{

console.log('Erro ao se conectar a base de dados: ', err);

});

//modelo

const Todo = mongoose.model('Todo', {text: String, completed: Boolean});

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

const todo = new Todo({text: req.body.text, completed: false});

todo.save()

.then(doc =>{

console.log('Dado inserido com sucesso!', doc);

//res.redirect('/');

res.json({response: 'success'});

})

.catch(err =>{

console.log('Erro ao inserir ', err.message);

res.status(400).json({response: 'failed'});

});

});

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

Todo.find({},'text completed')

.then(doc =>{

res.json({response: 'success', data: doc});

})

.catch(err =>{

console.log('Erro ao consultar elementos!', err.message);

res.status(400).json({response: 'failed'});

});

});

app.get('/complete/:id/:status', (req, res) =>{

const id = req.params.id;

const status = req.params.status == 'true'; // converter para booleano

Todo.findByIdAndUpdate({\_id: id},{$set: {completed: status}})

.then(doc =>{

res.json({response: 'success'});

})

.catch(err =>{

console.log('Erro ao atualizar dado ', err.message);

res.status(400).json({response: 'failed'});

});

});

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

const id = req.params.id;

Todo.findByIdAndDelete({\_id: id})

.then(doc =>{

//res.json({response: 'success'});

res.redirect('/');

})

.catch(err =>{

console.log('Erro ao excluir dado ', err.message);

res.status(400).json({response: 'failed'});

});

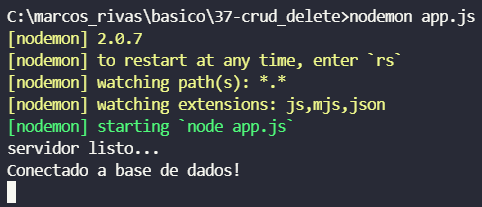
});

app.listen(3000, () =>{

console.log('servidor listo...');

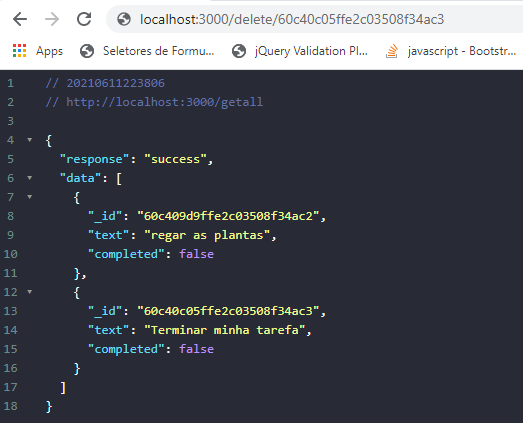
});

nodemon app.js



- No browser:

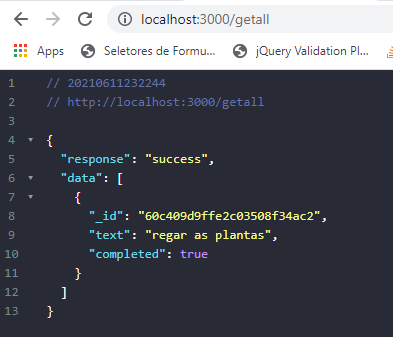
http://localhost:3000/delete/60c40c05ffe2c03508f34ac3



localhost:3000



http://localhost:3000/getall



# Aula 38 - CRUD - completo

- Dentro da pasta "public/js" adicione um arquivo chamado "main.js":

**38-crud\_completo\public\js\main.js**

window.addEventListener('DOMContentLoaded', (e) =>{

updateTodos();

});

const mapCheckboxes = () =>{

document.querySelectorAll('.complete-checkbox').forEach(item => {

item.addEventListener('click', async e => {

const id = e.target.parentNode.parentNode.id;

let classes = e.target.parentNode.parentNode.childNodes[3].className.replace('completed', '').trim();

const completed = e.target.checked;

const res = await updateTodo(id, completed);

if(res.response === 'success'){

if(completed){

e.target.parentNode.parentNode.childNodes[3].className += 'completed';

} else {

e.target.parentNode.parentNode.childNodes[3].className += 'classes';

}

}

});

});

}

const updateTodos = () => {

fetch('http://localhost:3000/getall')

.then(res => res.json())

.then(data =>{

if(data.response === 'success'){

const todos = data.data;

document.querySelector('#todos').innerHTML = '';

todos.forEach(todo =>{

document.querySelector('#todos').innerHTML += `

<div class='todo' id='${todo.\_id}'>

<div class="checkbox-container"><input type="checkbox" class="complete-checkbox" name="" id="" ${(todo.completed === true)? 'checked': ''}></div>

<div class="text-container ${(todo.completed === true)? 'completed': ''}">${todo.text}</div>

<div class="actions-container"><a href='/delete/${todo.\_id}'>X</a></div>

</div>

`;

});

mapCheckboxes();

}

})

.catch(err => console.error(err));

}

const updateTodo = async (id, completed) =>{

const res = await fetch('http://localhost:3000/complete/' + id + '/' + completed)

.then(res => res.json());

return res;

}

document.querySelector('#formulario').addEventListener('submit', e => {

e.preventDefault();

const text = document.querySelector('#text').value;

if(text === ''){

return false;

}

fetch('http://localhost:3000/add', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({text: text})

})

.then(res => res.json())

.then(data => {

if(data.response === 'success'){

updateTodos();

document.querySelector('#text').value = '';

}

});

});

**38-crud\_model\public\index.html**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8" />

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

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

<title>CRUD Node.js + Express.js + MongoDB</title>

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

</head>

<body>

<div id="main-container">

<form id="formulario" action="/add" method="post">

<input

type="text"

name="text"

id="text"

autocomplete="off"

placeholder="Insira um novo todo"

/>

</form>

<div id="todos"></div>

</div>

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

</body>

</html>

**38-crud\_completo\app.js**

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const app = express();

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

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

// conexão com o banco de dados

mongoose.connect('mongodb://dev:dev@localhost:27017/todos', {useNewUrlParser: true, useUnifiedTopology: true});

const connection = mongoose.connection;

connection.once('open', () =>{

console.log('Conectado a base de dados!');

});

connection.on('error', (err) =>{

console.log('Erro ao se conectar a base de dados: ', err);

});

//modelo

const Todo = mongoose.model('Todo', {text: String, completed: Boolean});

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

const todo = new Todo({text: req.body.text, completed: false});

todo.save()

.then(doc =>{

console.log('Dado inserido com sucesso!', doc);

// res.redirect('/');

res.json({response: 'success'});

})

.catch(err =>{

console.log('Erro ao inserir ', err.message);

res.status(400).json({response: 'failed'});

});

});

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

Todo.find({},'text completed')

.then(doc =>{

res.json({response: 'success', data: doc});

})

.catch(err =>{

console.log('Erro ao consultar elementos!', err.message);

res.status(400).json({response: 'failed'});

});

});

app.get('/complete/:id/:status', (req, res) =>{

const id = req.params.id;

const status = req.params.status == 'true'; // converter para booleano

Todo.findByIdAndUpdate({\_id: id},{$set: {completed: status}})

.then(doc =>{

res.json({response: 'success'});

})

.catch(err =>{

console.log('Erro ao atualizar dado ', err.message);

res.status(400).json({response: 'failed'});

});

});

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

const id = req.params.id;

Todo.findByIdAndDelete({\_id: id})

.then(doc =>{

//res.json({response: 'success'});

res.redirect('/');

})

.catch(err =>{

console.log('Erro ao excluir dado ', err.message);

res.status(400).json({response: 'failed'});

});

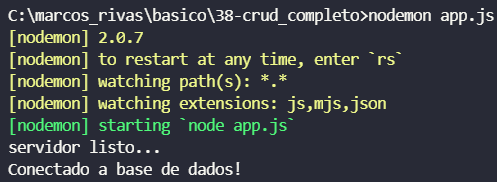
});

app.listen(3000, () =>{

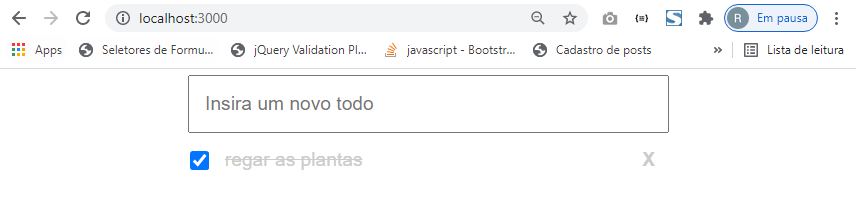
console.log('servidor listo...');

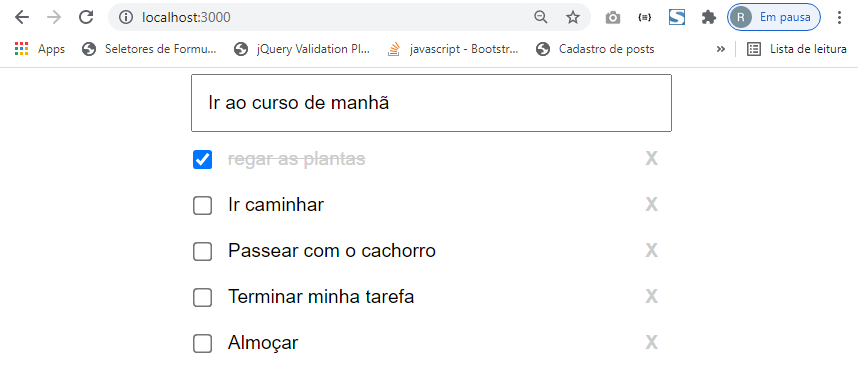
});

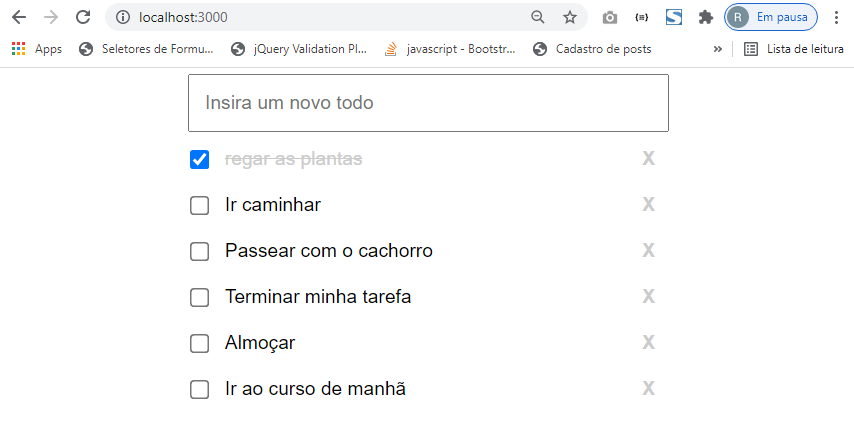
nodemon app.js

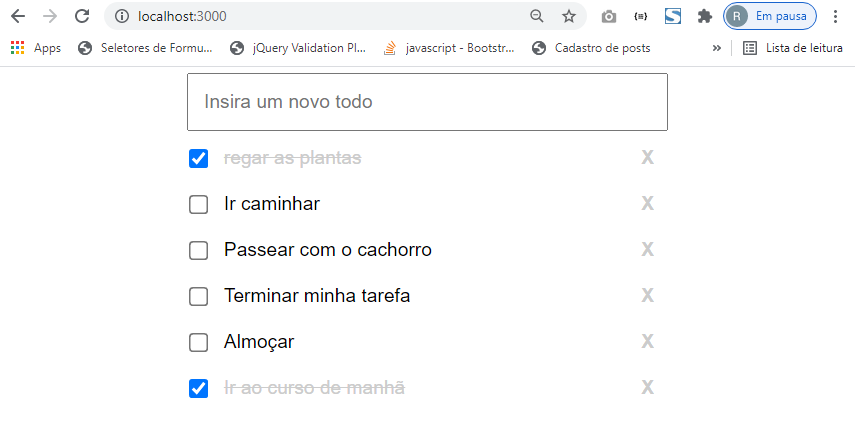


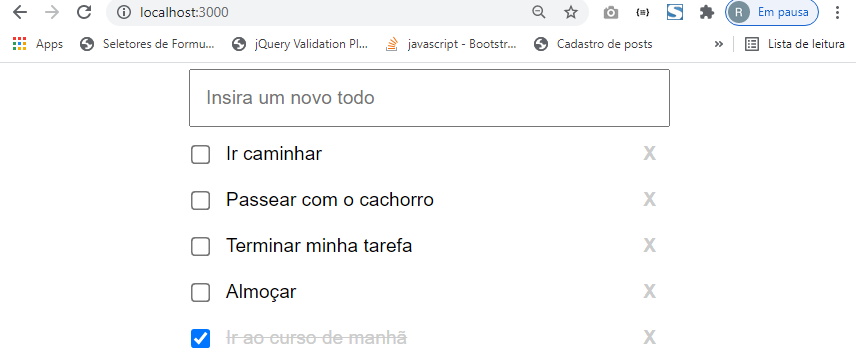
localhost:3000





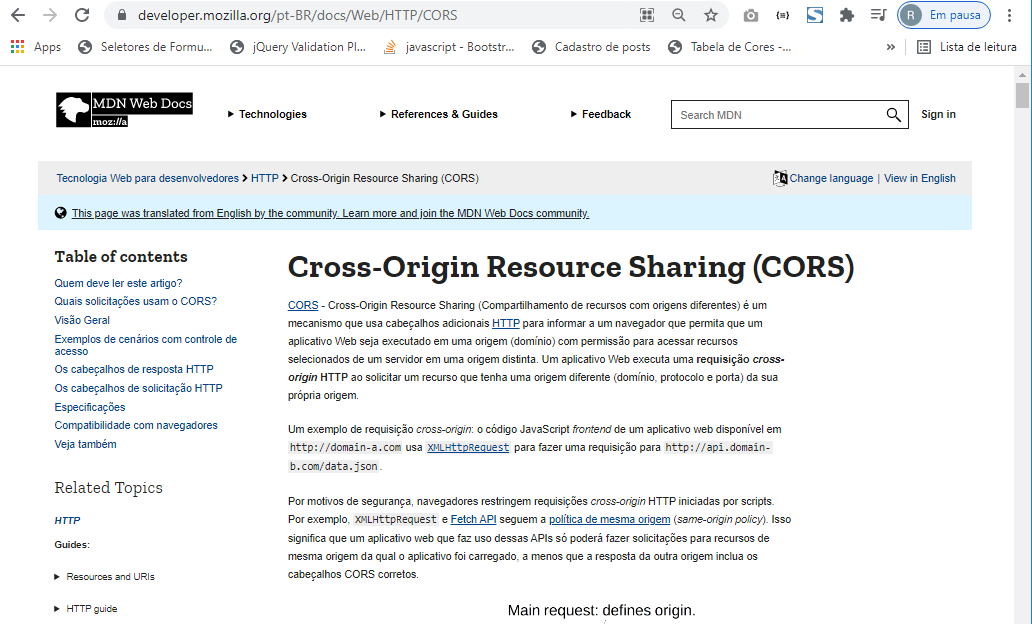






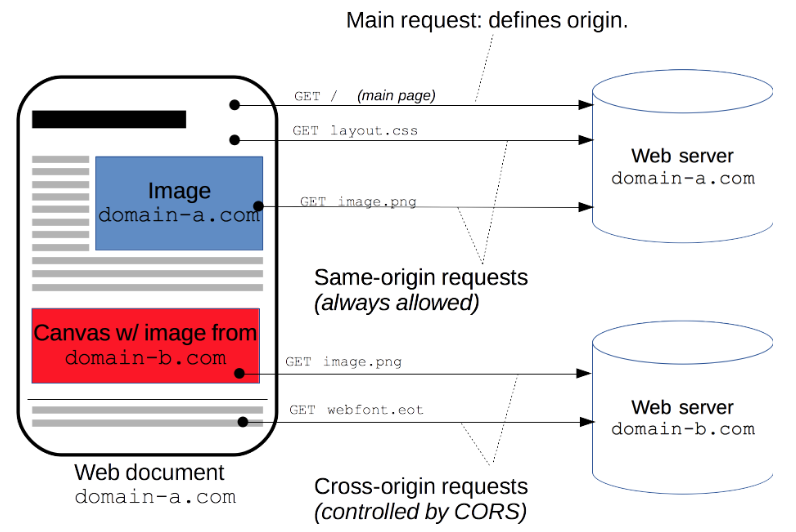
# Aula 39 - O que é CORS e como configurá-lo?

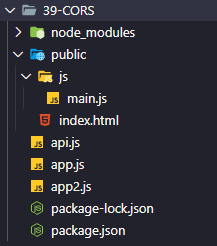
https://developer.mozilla.org/pt-BR/docs/Web/HTTP/CORS



Cross-Origin Resource Sharing (Compartilhamento de recursos com origens diferentes) é um mecanismo que usa cabeçalhos adicionais HTTP para informar a um navegador que permita que um aplicativo Web seja executado em uma origem (domínio) com permissão para acessar recursos selecionados de um servidor em uma origem distinta. Um aplicativo Web executa uma requisição cross-origin HTTP ao solicitar um recurso que tenha uma origem diferente (domínio, protocolo e porta) da sua própria origem.

Por motivos de segurança, navegadores restringem requisições cross-origin HTTP iniciadas por scripts. Por exemplo, XMLHttpRequest e Fetch API seguem a política de mesma origem (same-origin policy). Isso significa que um aplicativo web que faz uso dessas APIs só poderá fazer solicitações para recursos de mesma origem da qual o aplicativo foi carregado, a menos que a resposta da outra origem inclua os cabeçalhos CORS corretos.





**39-CORS\app.js**

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

app.use(express.static('./public/'));

app.listen(3000, () =>{

console.log('Servidor 1 iniciado...');

});

**39-CORS\public\index.html**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8" />

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

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

<title>CORS</title>

</head>

<body>

<h1>Olá Mundo!</h1>

<script>

fetch('http://localhost:3001')

.then((res) => res.json())

.then((data) => {

console.log(data);

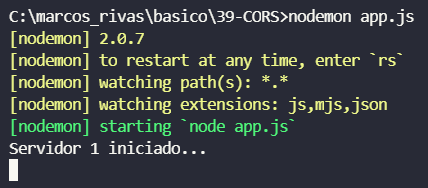
});

</script>

</body>

</html>

node app.js



**39-CORS\app2.js**

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

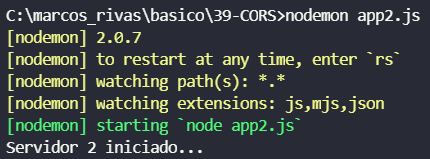
app.use(express.static('./public/'));

app.listen(3002, () =>{

console.log('Servidor 2 iniciado...');

});

node app2.js



**39-CORS\api.js**

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

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

res.json({message: 'ok'});

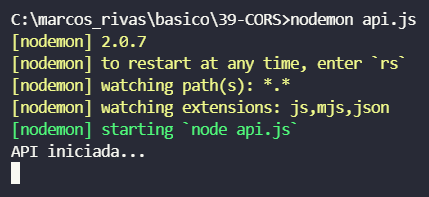
});

app.listen(3001, () =>{

console.log('API iniciada...');

});

nodemon api.js



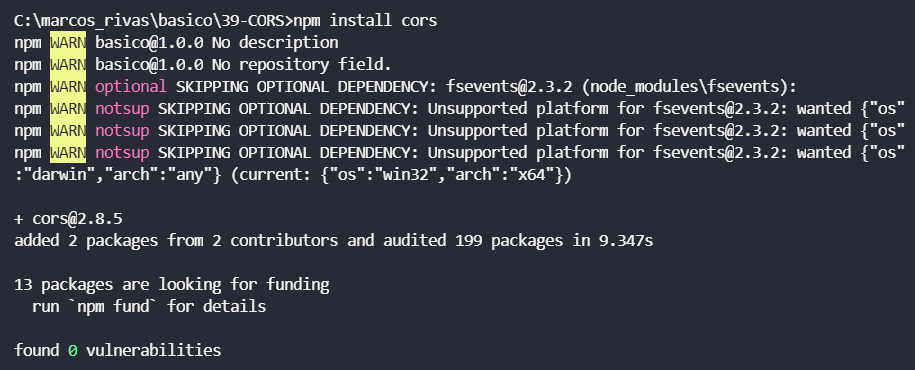
- No browser:

localhost:3000



# Instalando CORS

npm install cors



**39-CORS\api.js**

const express = require('express');

const bodyParser = require('body-parser');

const cors = require('cors');

const app = express();

app.use(cors());

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

res.json({message: 'ok'});

});

app.listen(3001, () =>{

console.log('API iniciada...');

});

- No browser:

localhost:3000



**39-CORS\api.js**

const express = require('express');

const bodyParser = require('body-parser');

const cors = require('cors');

const app = express();

app.use(cors());

var whitelist = ['http://localhost:3000']

var corsOptions = {

origin: function (origin, callback) {

if (whitelist.indexOf(origin) !== -1) {

callback(null, true)

} else {

callback(new Error('Not allowed by CORS'))

}

}

}

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

res.json({message: 'ok'});

});

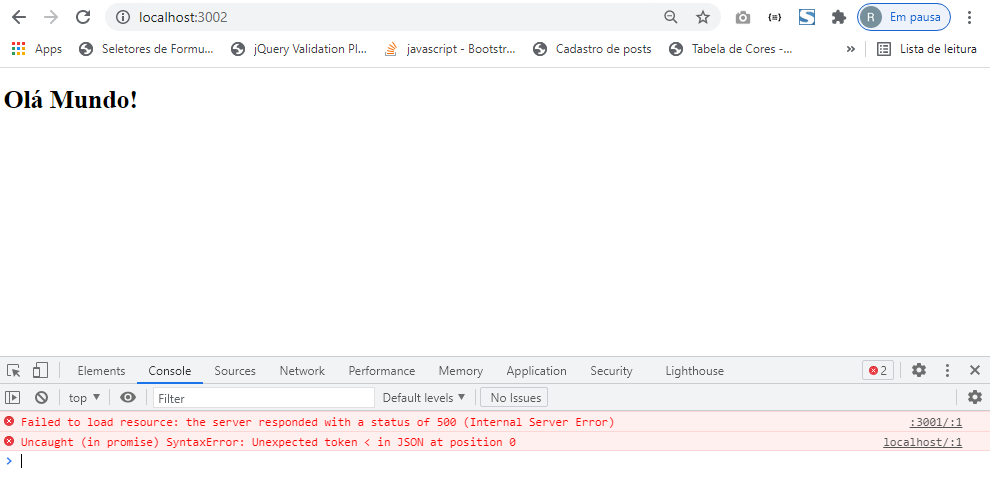
app.listen(3001, () =>{

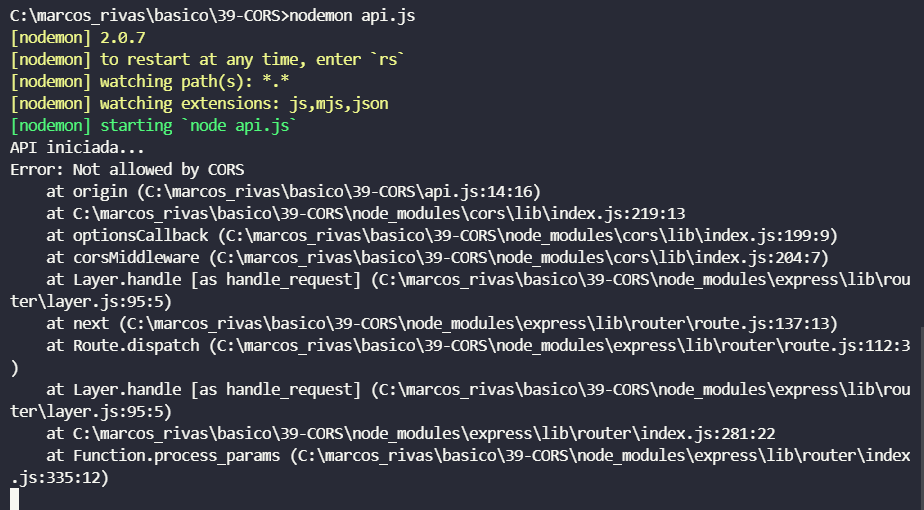
console.log('API iniciada...');

});

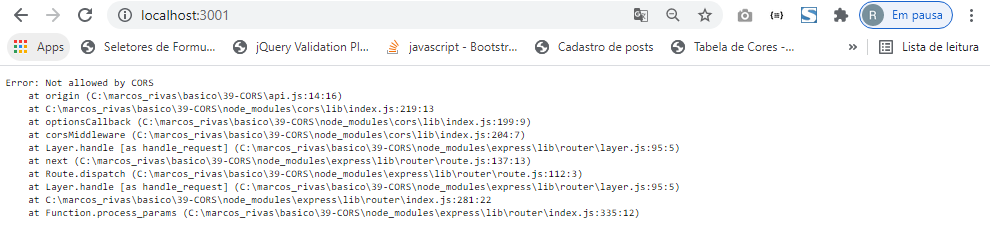
- No browser:

localhost:3002



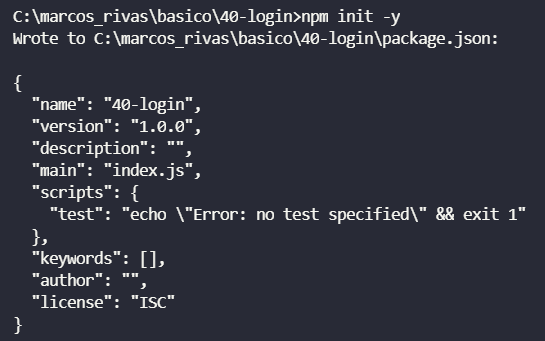


localhost:3001



# Aula 40 - Registro e login completo com NodeJS e MongoDB

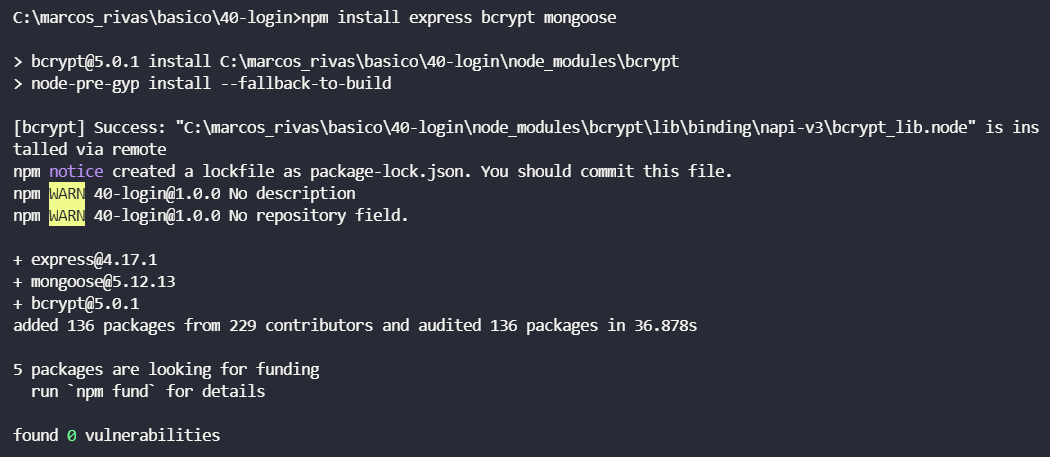
npm init -y

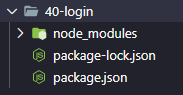


## Instalando express, bcrypt e mongoose

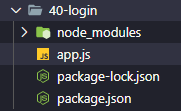
- No terminal entre com o seguinte comando:

npm install express bcrypt mongoose





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



**40-login\app.js**

const express = require('express');

const path = require('path');

const bodyParser = require('body-parser');

const app = express();

const bcrypt = require('bcrypt');

const mongoose = require('mongoose');

app.use(bodyParser.json());

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

app.use(express.static(path.join(\_\_dirname, 'public')));

const mongo\_uri = 'mongodb://dev:dev@localhost/todos';

mongoose.connect(mongo\_uri, function(err) {

if(err) {

throw err;

} else {

console.log(`Successfully connected to ${mongo\_uri}`);

}

})

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

});

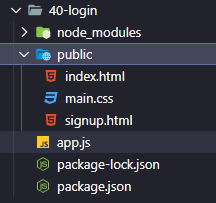
app.listen(3000, () => {

console.log('Server started...');

})

module.exports = app;

- Na pasta raiz do projeto, adicione uma pasta chamada "public" com três arquivos: "index.html, main.css, signup.html":



**40-login\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>Login</title>

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

</head>

<body>

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

<h2>Login</h2>

<div class="section">

<div class="title">Username</div>

<div class="field">

<input type="text" id="username" name="username" />

</div>

</div>

<div class="section">

<div class="title">Password</div>

<div class="field">

<input type="password" id="password" name="password" />

</div>

</div>

<div class="section">

<div class="button"><input type="submit" value="Login" /></div>

<div class="button"><a href="signup.html">Registrar-se</a></div>

</div>

</form>

</body>

</html>

**40-login\public\main.css**

\* {

box-sizing: border-box;

}

body {

font-family: Arial, Helvetica, sans-serif;

}

form {

width: 400px;

margin: 0 auto;

border: solid 1px #ccc;

padding: 20px;

}

input[type='text'],

input[type='password'] {

border: solid 1px #ccc;

border-radius: 3px;

font-size: 18px;

outline: none;

padding: 10px;

width: 100%;

}

input[type='submit'] {

border: 0;

background-color: #003366;

border-radius: 3px;

font-size: 18px;

color: white;

outline: 0;

padding: 10px 30px;

}

.section {

padding: 15px 0;

}

.section .title {

padding: 10px 0;

}

.button {

padding: 10px 0;

text-align: center;

}

.button a {

color: #1a72cb;

}

**40-login\public\signup.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>Registro</title>

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

</head>

<body>

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

<h2>Registro</h2>

<div class="section">

<div class="title">Username</div>

<div class="field">

<input type="text" id="username" name="username" />

</div>

</div>

<div class="section">

<div class="title">Password</div>

<div class="field">

<input type="password" id="password" name="password" />

</div>

</div>

<div class="section">

<div class="button">

<input type="submit" value="Registrar usuário" />

</div>

<div class="button">

<a href="index.html">Login</a>

</div>

</div>

</form>

</body>

</html>

- Rode o servidor do Mongo DB:

mongod

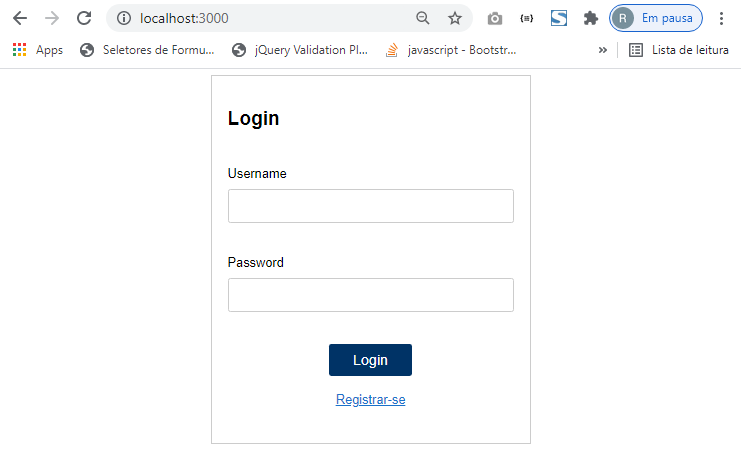
- No terminal do Visual Studio Code, entre com:

nodemon app.js

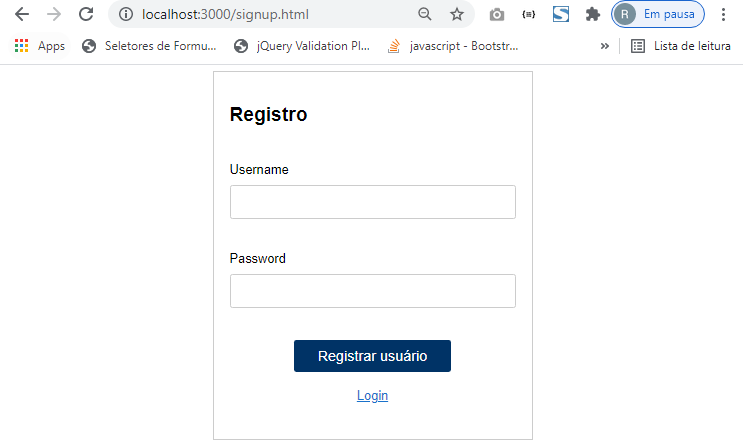


- No browser:

localhost:3000

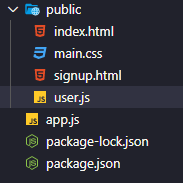


localhost:3000/signup.html



## Criando o Model de usuários

- Adicione a pasta "public" um arquivo chamado "user.js"



**40-login\public\user.js**

const mongoose = require('mongoose');

const bcrypt = require('bcrypt');

const saltRounds = 10;

const UserSchema = new mongoose.Schema({

username: {

type: String,

require: true,

unique: true

},

password: {

type: String,

require: true

}

});

UserSchema.pre('save', function(next){

if(this.isNew || this.isModified('password')){

const document = this;

bcrypt.hash(document.password, saltRounds, (err, hashedPassword) => {

if(err) {

next(err);

} else {

document.password = hashedPassword;

next();

}

});

} else {

next();

}

});

UserSchema.methods.isCorrectPassword = function(password, calback) {

bcrypt.compare(password, this.password, function(err, same){

if(err){

callback(err);

} else {

callback(err, same);

}

});

}

module.exports = mongoose.model('User', UserSchema);

**40-login\app.js**

const express = require('express');

const path = require('path');

const bodyParser = require('body-parser');

const app = express();

const mongoose = require('mongoose');

const User = require('./public/user');

app.use(bodyParser.json());

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

app.use(express.static( path.join(\_\_dirname, 'public')));

const mongo\_uri = 'mongodb://dev:dev@localhost/todos';

mongoose.connect(mongo\_uri, function(err) {

if(err) {

throw err;

} else {

console.log(`Successfully connected to ${mongo\_uri}`);

}

})

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

const {username, password} = req.body;

const user = new User({username, password});

user.save(err => {

if(err){

res.status(500).send('Erro ao registrar o usuário');

} else {

res.status(200).send('Usuário registrado com sucesso!');

}

});

});

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

const {username, password} = req.body;

User.findOne({username}, (err, user) => {

if(err){

res.status(500).send('Erro ao autenticar o usuário!');

} else if(!user) {

res.status(500).send('O usuário não existe!');

} else {

user.isCorrectPassword(password, (err, result) => {

if(err){

res.status(500).send('Erro ao autenticar o usuário!');

} else if(result) {

res.status(200).send('Usuário autenticado com sucesso!');

} else {

res.status(500).send('Usuário e/ou senha incorreta!');

}

});

}

});

});

app.listen(3000, () => {

console.log('Server started...');

})

module.exports = app;

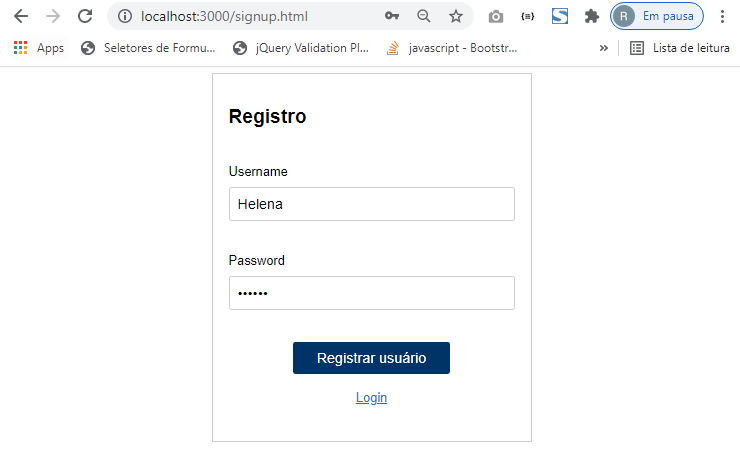
nodemon app.js

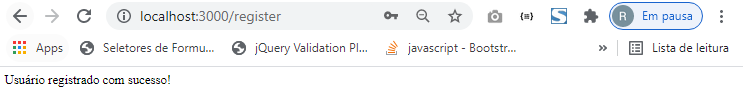


- No browser:

Senha: 123456

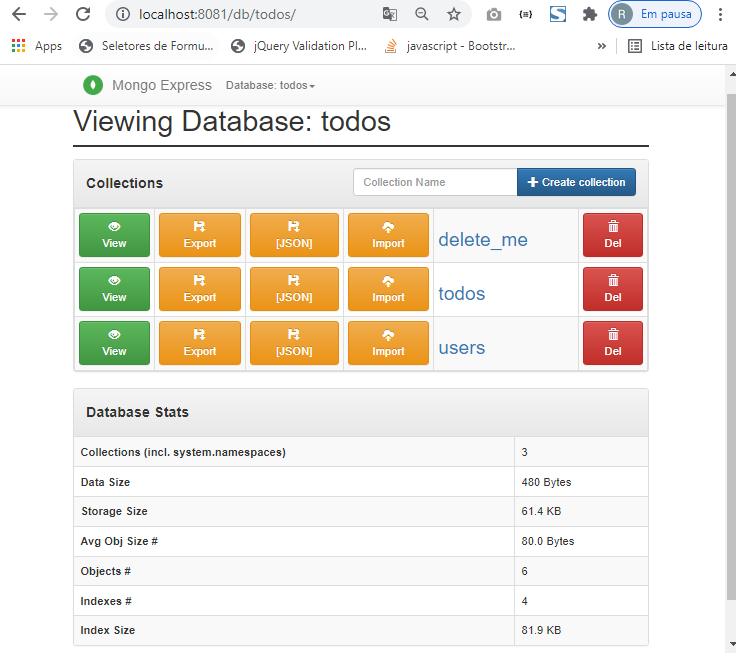
http://localhost:3000/signup.html

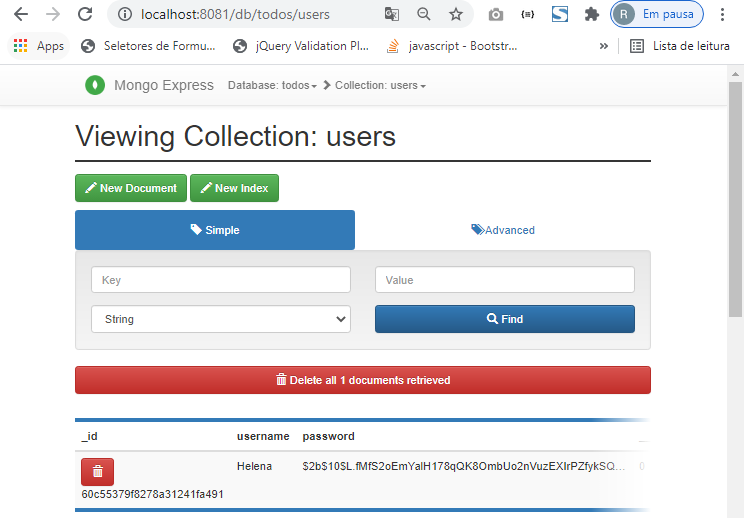




mongo-express --admin --url mongodb://127.0.0.1:27017



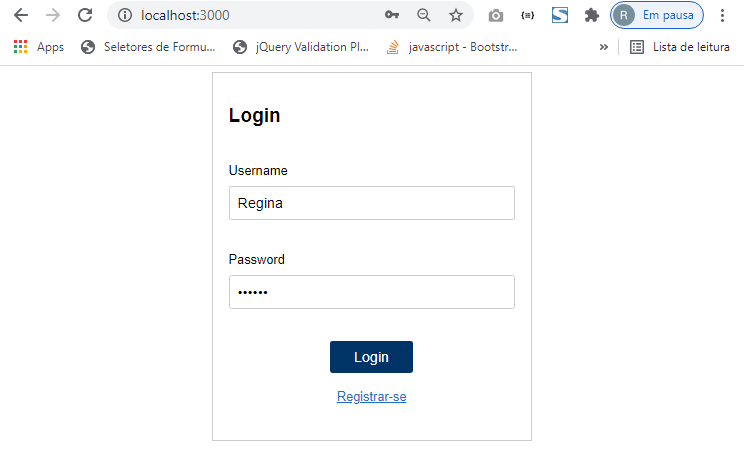


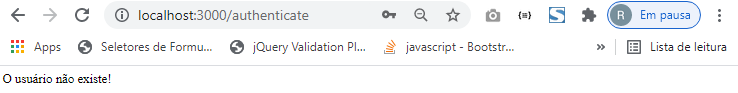


- No browser:

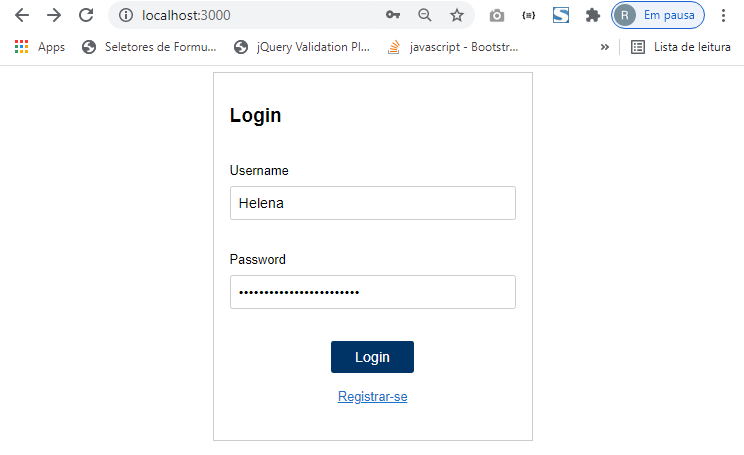
Usuário não existe:

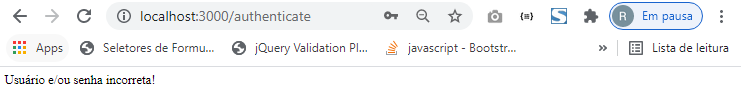
localhost:3000





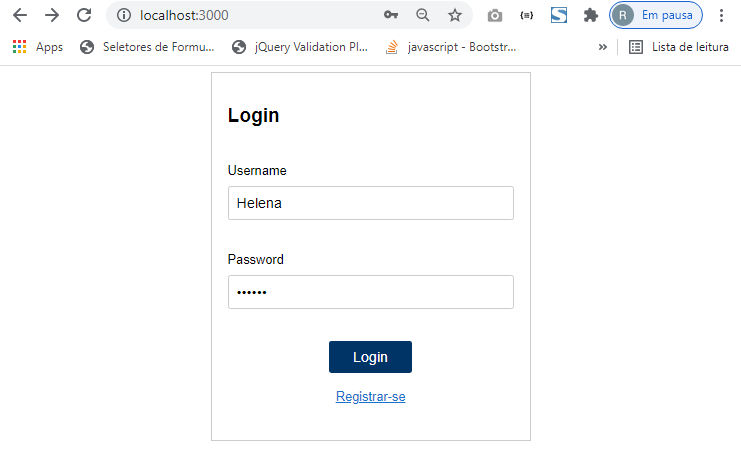
Senha errada:

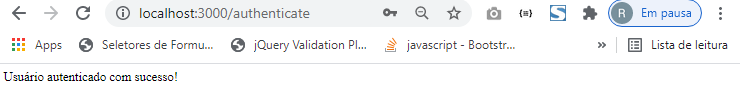




username e password corretos:

Senha: 123456



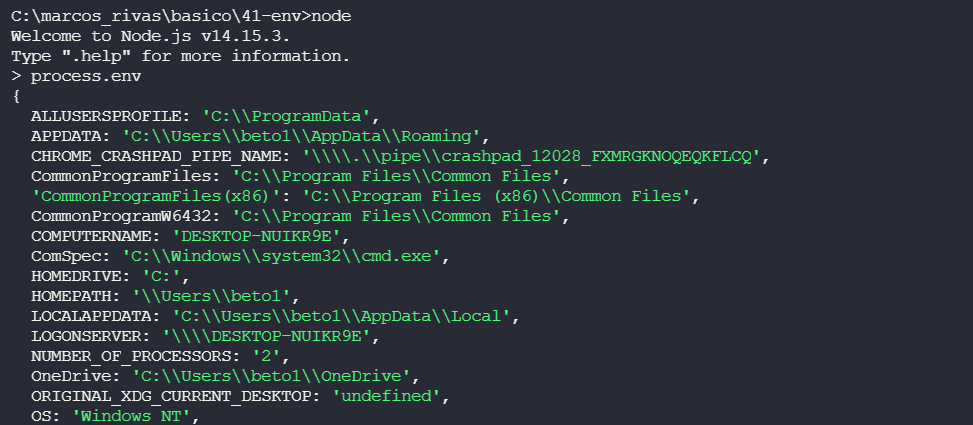


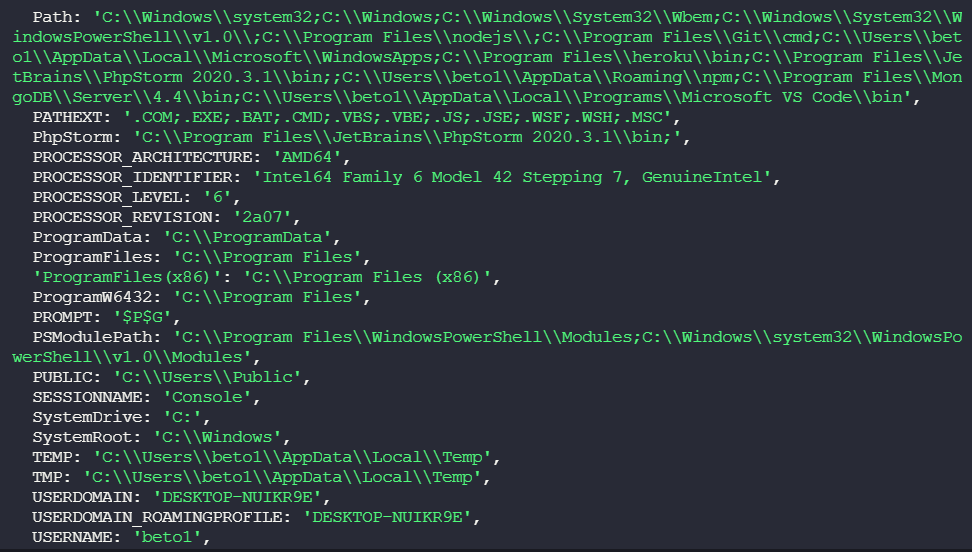
# Aula 41 - Como usar variáveis de ambiente com dotenv

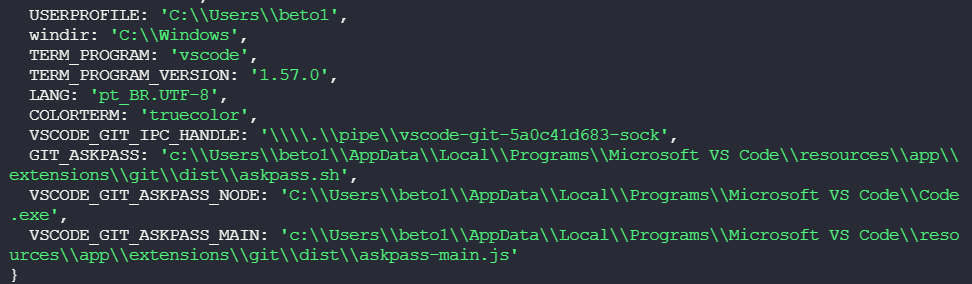
No terminal, entre com o comando:

node

process.env



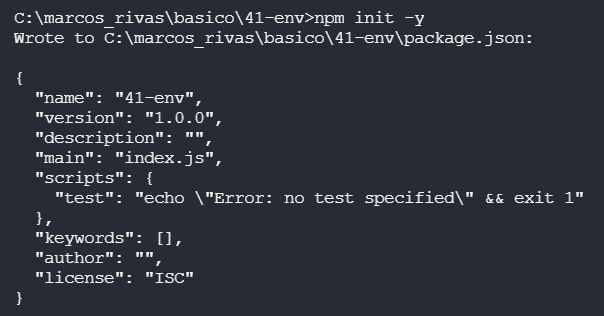




Usos:

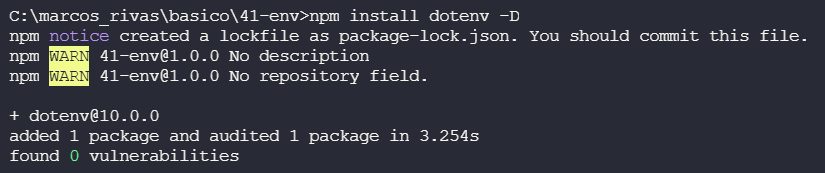
* Informação sensível (exemplo: senhas);
* Informação de configuração;
* Tokens, conexões e ports

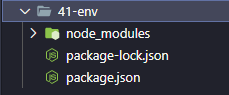
npm init -y



## Instalando dotenv

npm install dotenv -D





**41-env\package.json**

{

"name": "41-env",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

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

},

"keywords": [],

"author": "",

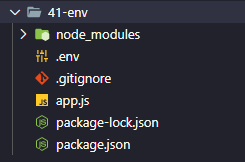
"license": "ISC",

"devDependencies": {

"dotenv": "^10.0.0"

}

}



**41-env\app.js**

if(process.env.NODE\_ENV !== 'production'){

require('dotenv').config();

}

const app = require('http').createServer((req, res) => res.send('Hello World!'));

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

console.log(process.env.TOKEN);

app.listen(PORT, () => {

console.log(`Server is listening on port ${PORT}`);

});

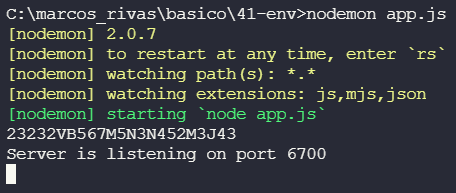
**41-env\.env**

PASSWORD=esteehmeupassword

TOKEN=23232VB567M5N3N452M3J43

PORT=6700

nodemon app.js

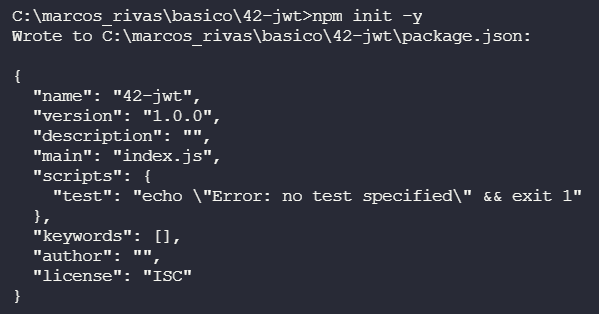


**41-env\.gitignore**

.env

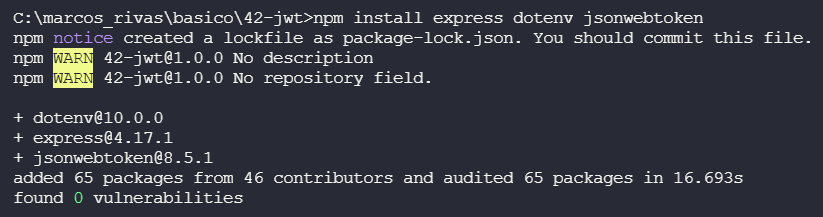
# Aula 42 - Autenticação com JWT (JSON Web Token)

npm init -y



## Instalando as dependências

npm install express dotenv jsonwebtoken



**42-jwt\package.json**

{

"name": "42-jwt",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

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

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"dotenv": "^10.0.0",

"express": "^4.17.1",

"jsonwebtoken": "^8.5.1"

}

}



**42-jwt\index.js**

const express = require('express');

const app = express();

const jwt = require('jsonwebtoken');

require('dotenv').config();

app.use(express.urlencoded({extended: false}));

app.use(express.json());

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

res.send('Hello World!');

})

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

res.send(`<html>

<head>

<title>Login</title>

</head>

<body>

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

Nome de usuário: <input type="text" name="text"><br />

Senha: <input type="password" name="password"><br />

<input type="submit" value="Iniciar sessão"/>

</form>

</body>

</html>`);

});

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

res.json({

tuits: [

{

id: 1,

text: 'Este é meu primeiro tuit',

username: 'vidamrr'

},

{

id: 2,

text: 'A melhor linguagem é HTML',

username: 'patinho\_feliz'

}

]

});

});

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

const {username, password} = req.body;

// consultar BD e validar que existem tanto username como password

const user = {username: username};

const accessToken = generateAccessToken(user);

res.header('authorization', accessToken).json({

message: 'Usuário autenticado!',

token: accessToken

});

});

function generateAccessToken(user){

return jwt.sign(user, process.env.SECRET, {expiresIn: '50m'});

}

function validateToken(req, res, next){

const accessToken = req.headers['authorization'];

if(!accessToken) res.send('Access denied');

jwt.verify(accessToken, process.env.SECRET, (err, user) => {

if(err){

res.send('Access denied, token expired or incorrect');

}else{

next();

}

});

}

app.listen(3000, () => {

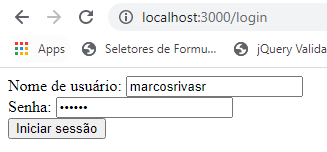
console.log('Servidor iniciado...');

});

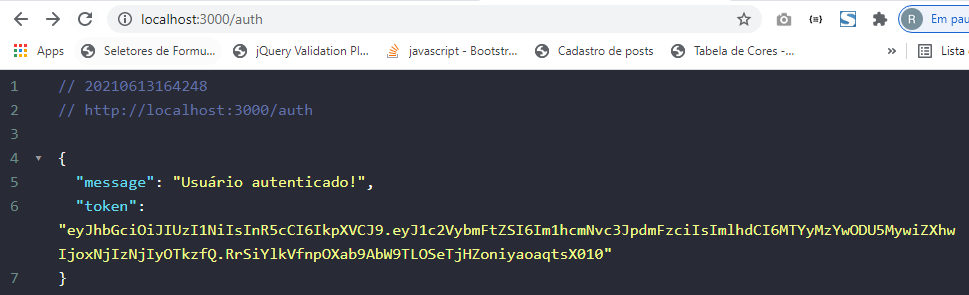
**42-jwt\.env**

SECRET=vidamrr

Senha: marcos



localhost:3000/auth



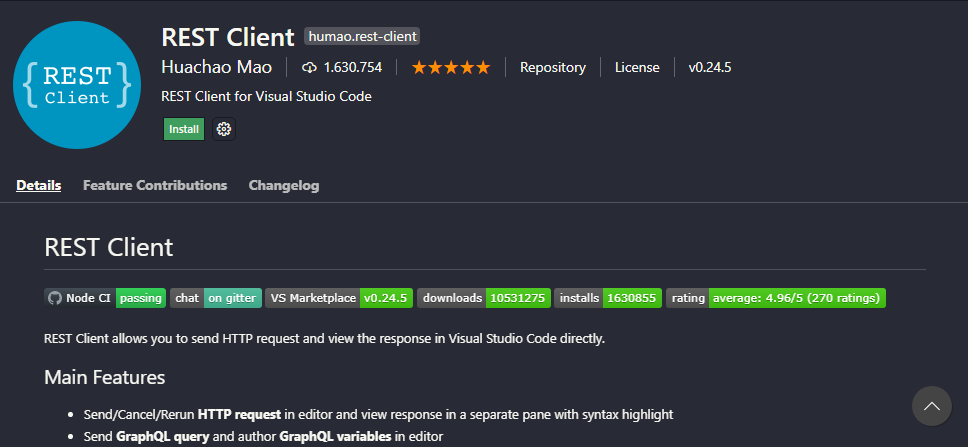
{

"message": "Usuário autenticado!",

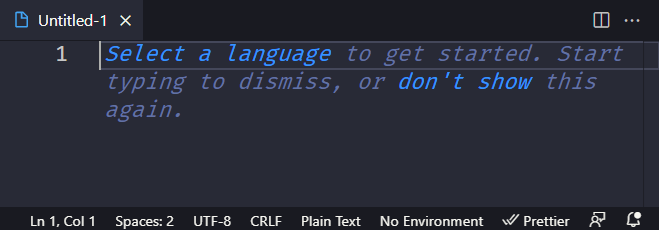
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Im1hcmNvc3JpdmFzciIsImlhdCI6MTYyMzYwODU5MywiZXhwIjoxNjIzNjIyOTkzfQ.RrSiYlkVfnpOXab9AbW9TLOSeTjHZoniyaoaqtsX010"

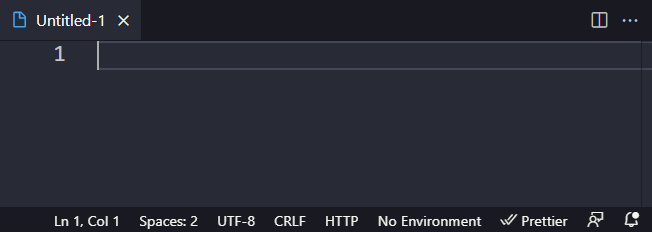
}

- No Visual Studio Code, instale uma extensão chamada "REST Client":

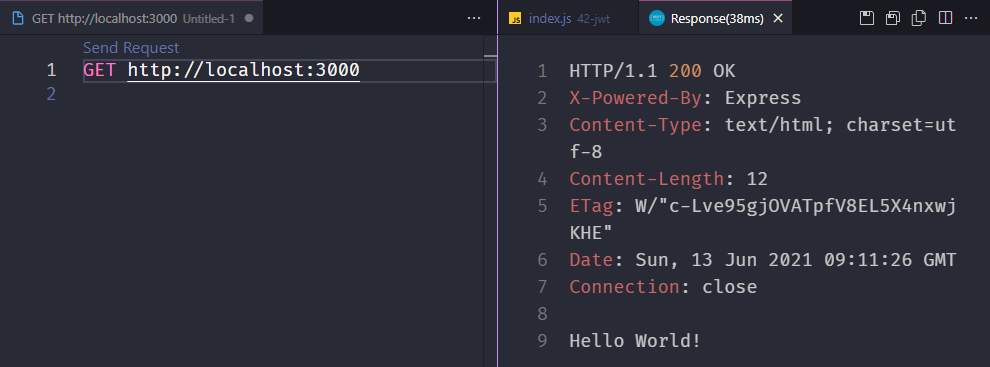


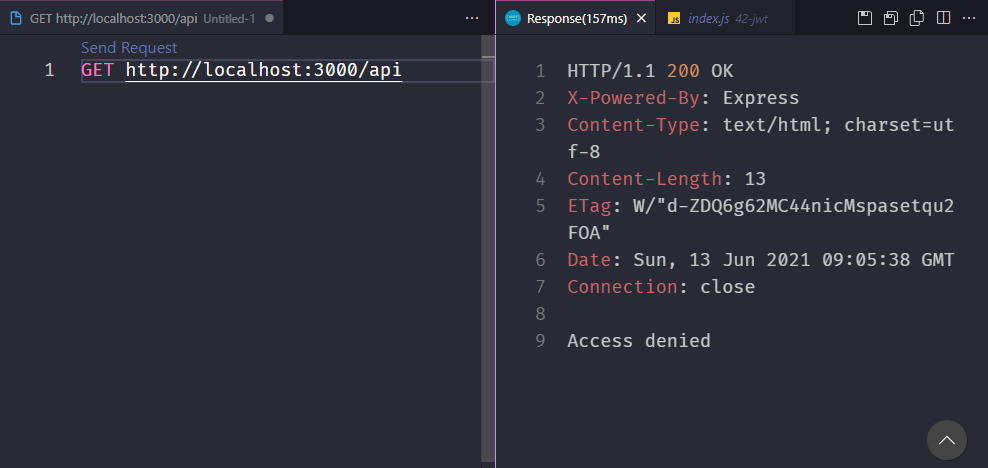
- No Visual Studio Code crie um novo arquivo e o defina como tipo "http":



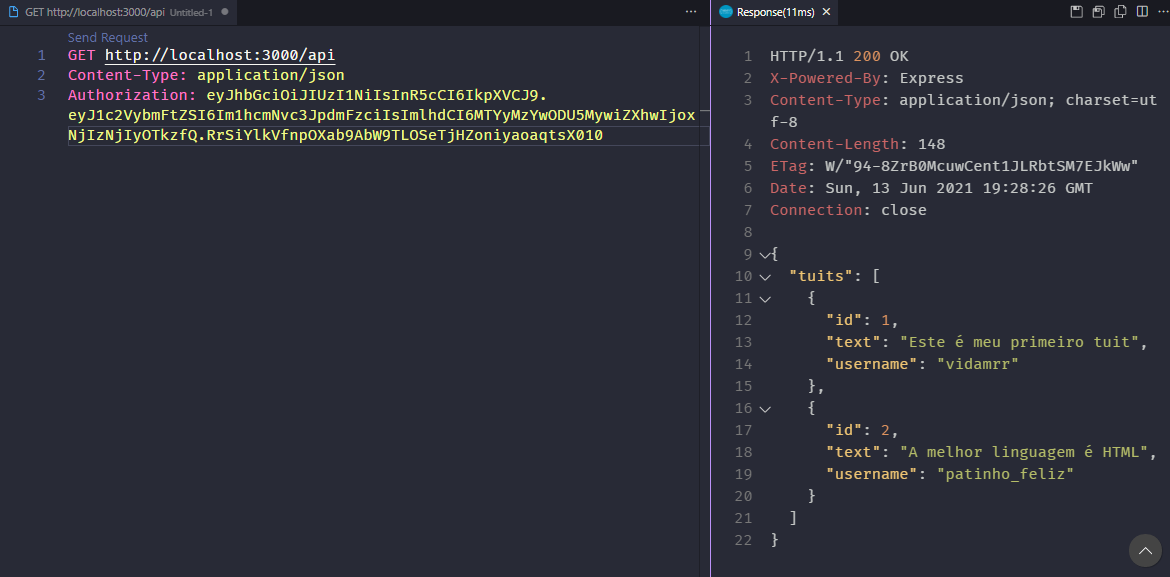


- Entre com:

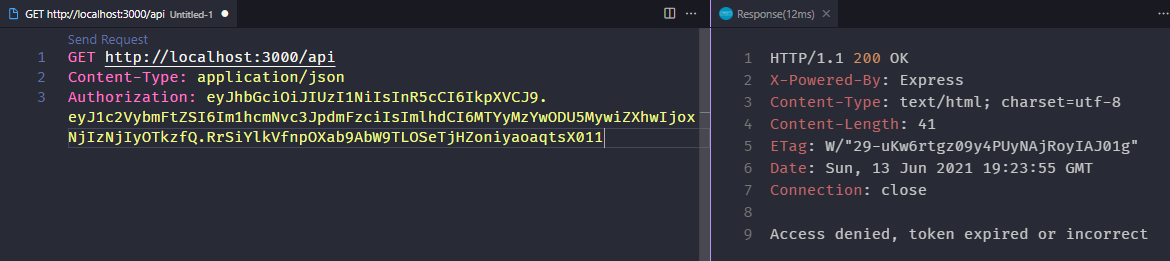




- Passando um token correto e não expirado:



Passando um token incorreto ou inválido:



**42-jwt\index.js**

const express = require('express');

const app = express();

const jwt = require('jsonwebtoken');

require('dotenv').config();

app.use(express.urlencoded({extended: false}));

app.use(express.json());

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

res.send('Hello World!');

})

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

res.send(`<html>

<head>

<title>Login</title>

</head>

<body>

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

Nome de usuário: <input type="text" name="text"><br />

Senha: <input type="password" name="password"><br />

<input type="submit" value="Iniciar sessão"/>

</form>

</body>

</html>`);

});

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

res.json({

tuits: [

{

id: 1,

text: 'Este é meu primeiro tuit',

username: 'vidamrr'

},

{

id: 2,

text: 'A melhor linguagem é HTML',

username: 'patinho\_feliz'

}

]

});

});

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

const {username, password} = req.body;

// consultar BD e validar que existem tanto username como password

const user = {username: username};

const accessToken = generateAccessToken(user);

res.header('authorization', accessToken).json({

message: 'Usuário autenticado!',

token: accessToken

});

});

function generateAccessToken(user){

return jwt.sign(user, process.env.SECRET, {expiresIn: '50m'});

}

function validateToken(req, res, next){

const accessToken = req.headers['authorization'] || req.query.accesstoken;

if(!accessToken) res.send('Access denied');

jwt.verify(accessToken, process.env.SECRET, (err, user) => {

if(err){

res.send('Access denied, token expired or incorrect');

}else{

next();

}

});

}

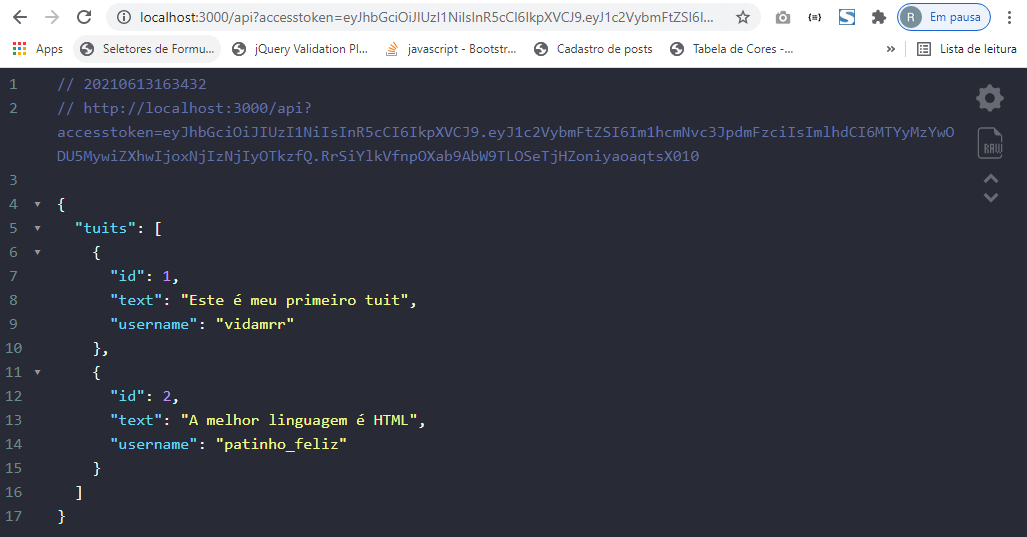
app.listen(3000, () => {

console.log('Servidor iniciado...');

});

- No browser:

http://localhost:3000/api?accesstoken=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Im1hcmNvc3JpdmFzciIsImlhdCI6MTYyMzYwODU5MywiZXhwIjoxNjIzNjIyOTkzfQ.RrSiYlkVfnpOXab9AbW9TLOSeTjHZoniyaoaqtsX010



**42-jwt\index.js**

const express = require('express');

const app = express();

const jwt = require('jsonwebtoken');

require('dotenv').config();

app.use(express.urlencoded({extended: false}));

app.use(express.json());

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

res.send('Hello World!');

})

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

res.send(`<html>

<head>

<title>Login</title>

</head>

<body>

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

Nome de usuário: <input type="text" name="text"><br />

Senha: <input type="password" name="password"><br />

<input type="submit" value="Iniciar sessão"/>

</form>

</body>

</html>`);

});

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

res.json({

username: req.user,

tuits: [

{

id: 1,

text: 'Este é meu primeiro tuit',

username: 'vidamrr'

},

{

id: 2,

text: 'A melhor linguagem é HTML',

username: 'patinho\_feliz'

}

]

});

});

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

const {username, password} = req.body;

// consultar BD e validar que existem tanto username como password

const user = {username: username};

const accessToken = generateAccessToken(user);

res.header('authorization', accessToken).json({

message: 'Usuário autenticado!',

token: accessToken

});

});

function generateAccessToken(user){

return jwt.sign(user, process.env.SECRET, {expiresIn: '50m'});

}

function validateToken(req, res, next){

const accessToken = req.headers['authorization'] || req.query.accesstoken;

if(!accessToken) res.send('Access denied');

jwt.verify(accessToken, process.env.SECRET, (err, user) => {

if(err){

res.send('Access denied, token expired or incorrect');

}else{

req.user = user;

next();

}

});

}

app.listen(3000, () => {

console.log('Servidor iniciado...');

});

