10/10/22

SIGNIFICADOS

CRUD =>

C - CREATE - CRIAR

R- READ - LER

U= UPDATE (ATUALIZAR)

D= DELETE (DELETAR)

Comandos

O professor José Carlos passou esse

\* PS>^C ,mas comigo funcionou o debaixo

\* crtl+ c (vai perguntar se quer parar o servidor) => esse comando serve para parar o servidor para ser executado novamente com alguma alteração

\*yarb add nodemon --global ou npm add nodemon --global => faz com que alteração seja atualizada simultaneamente sem precisar dos dois comandos anteriores a esse.

\* yarn add nodemon -D ou npm add nodemon -D => dependência de desenvolvimento só para quem vai desenvolver (11:55 de vídeo usamos esse)

usando o comando (npm add nodemon -D) teremos que fazer um script(package.json) para ele rodar o nodemon

"scripts": {

"start": "node server.js",

"dev ": "nodemon server.js" // script criado agora é só executar no terminal como npm run dev

},

"devDependencies": { // ao rodar o comando ele criou esses campos de dependências que //serão criadas

"nodemon": "^2.0.20"

}

\* npm install uuid ou yarn add uuid => instalado uma dependência que gera um id automático.

(temos que importar para o projeto)

Aula introdutória

Instalando extensões ( thunder client, Code runner,Omni theme(gosta de usar), Eslint => ajuda na importação de arquivos)

Dicas

fs.writeFileSync("D:\Nova pasta\Aulas Full Stack\18º WEB 3.0\Node\Node\Node.js\index.js")

mudando as barras de direção. Selecione uma barra, aperte ctrl + d até selecionar todas as barras e depois delete e substitua de um só vez pela barra oposta que é /

ctrl + alt + n ele roda o arquivo sem apertar a setinha

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Criando um servidor com node somente

/\*const http = require("http");

const hostname = "localhost";

const server = http.createServer((request, response)=>{

response.statusCode = 200;

response.setHeader("Content-type", "text/plain");

response.end("Hello Word");

})

server.listen(3333, hostname, ()=>{

console.log("Server running in localhost: 3333")

})\*/

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*abaixo começo da explicação do express

import express from "express";

import fs from "fs";

import https from "https";

import cors from "cors";

const app = express();

app.use(express.json());

app.use(cors());

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

res.json({message: "Hello Word"});

})

app.listen(3000, () => console.log("api rodando...."));

https.createServer({

cert: fs.readdirSync("D:/Nova pasta/Aulas Full Stack/18º WEB 3.0/Node/Node/Node.js/Criando\_Servidor\_HTTP/certificate.crt"),

key: fs.readdirSync("D:\Nova pasta\Aulas Full Stack\18º WEB 3.0\Node\Node\Node.js\Criando\_Servidor\_HTTP\Key.key")

}, app)

.listen(3001, () => console.log("rodando em https"))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Explicação

\*\*\*\*\*\*\*\*\*\*\*\*\*

const fs = require("fs");// Pedindo para import o módulo /const fs/ lá dentro da biblioteca ("fs")

//fs => significa file system

// Criando um arquivo através do método (writeFileSync) crtl+j para executar o comando abaixo

fs.writeFileSync("D:\Nova pasta\Aulas Full Stack\18º WEB 3.0\Node\Node\Node.js\Explicacao\index.js", "Olá Mundo ! ");

//Criando uma variável(verifyExistsFile) para receber um método(existsSync) que verifica a existência de um arquivo

/\*const verifyExistsFile = fs.existsSync("D:/Nova pasta/Aulas Full Stack/18º WEB 3.0/Node/Node/Node.js/ola.txt ");

if(verifyExistsFile){

console.log("Existe")

}else{

console.log("Não existe")

}

// Próximo passo vamos aprender a ler um arquivo com o módulo fs

// OBSERVAÇÃO: Se não colocar esse comando {encoding : "utf-8"} o terminal vai gerar a informação em byte

// Segue exemplo => <Buffer 5b 0d 0a 20 20 20 20 7b 0d 0a 22 6e 61 6d 65 22 3a 20 22 4a 6f 73 c3 a9 22 0d 0a 20 20 7d 2c 0d 0a 0d 0a 7b 0d 0a 22 6e 61 6d 65 22 3a 20 22 4c 75 63 ... 9 more bytes>

/\*

const userList = fs.readFileSync("D:/Nova pasta/Aulas Full Stack/18º WEB 3.0/Node/Node/Node.js/users.json" , {encoding : "utf-8"});

console.log(userList); ao executar essa linha de comando ele mostra no terminal as informações abaixo. Como resolver ?

[

{

"name": "José"

},

{

"name": "Lucas"

}

]\*/

/\*

const parse= JSON.parse(userList); Esse comando JSON.parse converte qualquer string para json

parse.map(user=> console.log(user.name));

Agora sim ficou bem mais apresentável

José

Lucas

\*/

Esse comando

JSON.parse =>converte qualquer string para json

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

HTTPS ( o s no final se refere a segurança SSL)

Gerador de SSL (site)=> https://pt.rakko.tools/tools/46/

npm init -y => cria o package.json e dentro está

{

"name": "node",

"version": "1.0.0",

"description": "",

"main": "Node.js",

"scripts": {

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

},

"keywords": [],

"author": "",

"license": "ISC"

}

criando o node\_modules e outros arquivos a partir do comando

npm install express

import cors from "cors"; => npm install cors ( serve para gerenciar a questão de segurança de nosso site)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Criando um servidor app com o express

arquivos server.js

import express from "express";

//Criando um servidor com o nome app

const app = express();

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

res.status(200).json({message: "Hello word"});

})

app.listen(3333, () => console.log("Server running in localhost: 3333"))

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Precisamos criar o módulo no arquivo => package.json

{

"name": "servidor-express",

"version": "1.0.0",

"description": "",

"main": "index.js",

"type": "module", // modulo criado

//script criado para executar o programa sem preecisar direcionando as rotas até ele

"scripts": {

"start": "node server.js"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"express": "^4.18.2",

"node": "^19.0.0"

}

}

AOS 9 MINUTOS ELE EXCLUIU O INDEX.JS

( Ler sobre os códigos 200,201,202,204)

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

res.status(200).json({message: "Hello Word"});

})

app.post ("/todos", (req,res )=> {

res.status(201).send()

})

app.put("/todos", (req,res )=> {

res.status(202).send()

})

app.delete("/todos", (req,res )=> {

res.status(202).send()

})

app.listen(3333, () => console.log("Server running in localhost: 3333"));

18:10 minutos criou um arquivo de regras de negócios com o nome

README.md => arquivos de regras de negócios

aplicando as regras de negócios

app.post ("/todos", (req,res )=> {

const { body } = req;

body.concluded = false; // concluidos

body.id = uuidv4(); // id gerado de forma automática

body.date = new Date(); // data gerada de forma automática

todos.push(body);

res.status(201).send()

})

Achando erros(Exemplos)

app.post ("/todos", (req,res )=> {

const { body } = req;

//console.log(body); deu um erro e para ele achar , ele digitou essa linha de comando par o terminal mostrar o erro

todos.push(body);

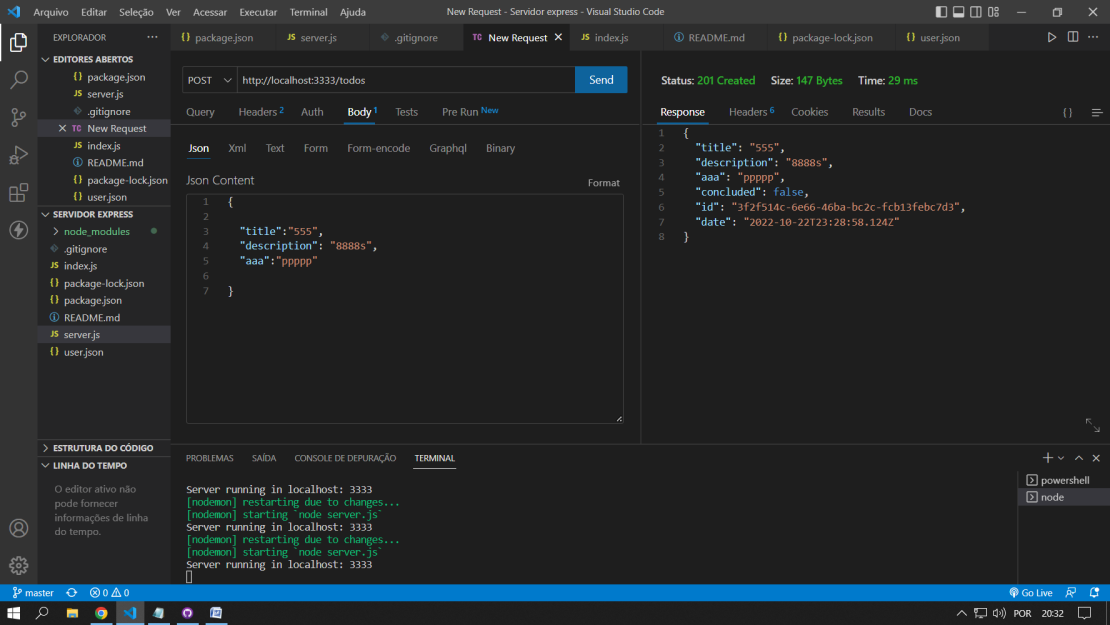
res.status(201).send()

})

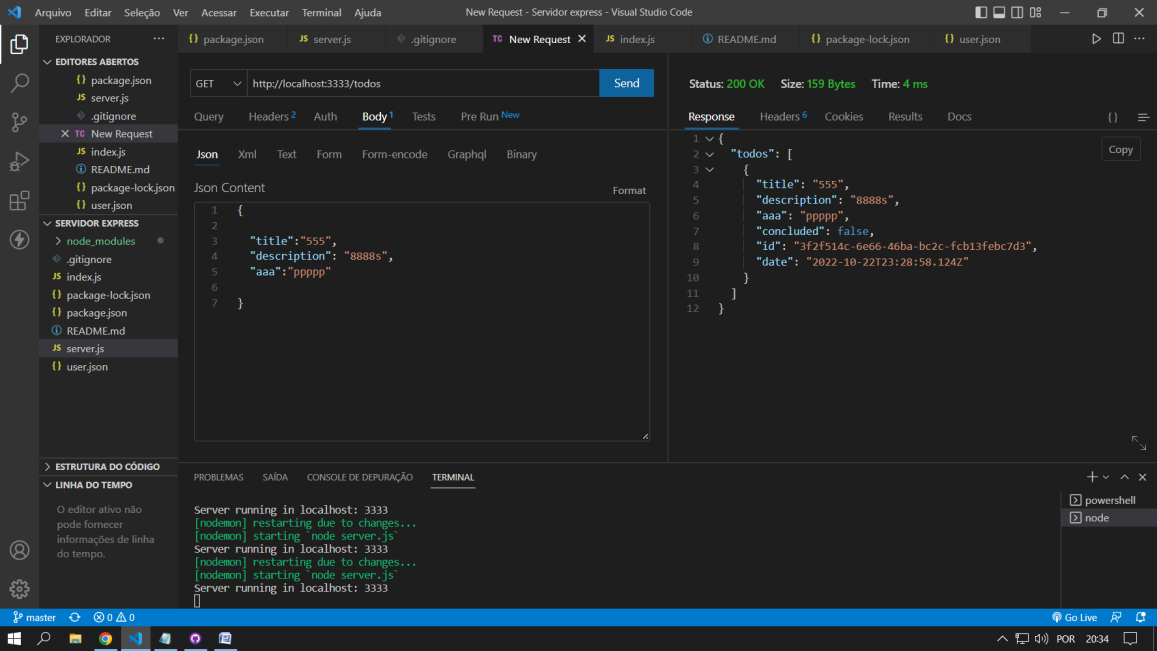
Causa do erro acima: O express não reconhece o json e para isso temos que falar para ele que iremos trabalhar com json e para isso ele utilizou o(app.use(express.json());) midor que é uma espécie de intermediador dentro do Express

Usando o thunder a extenção

Post



GET



PUT