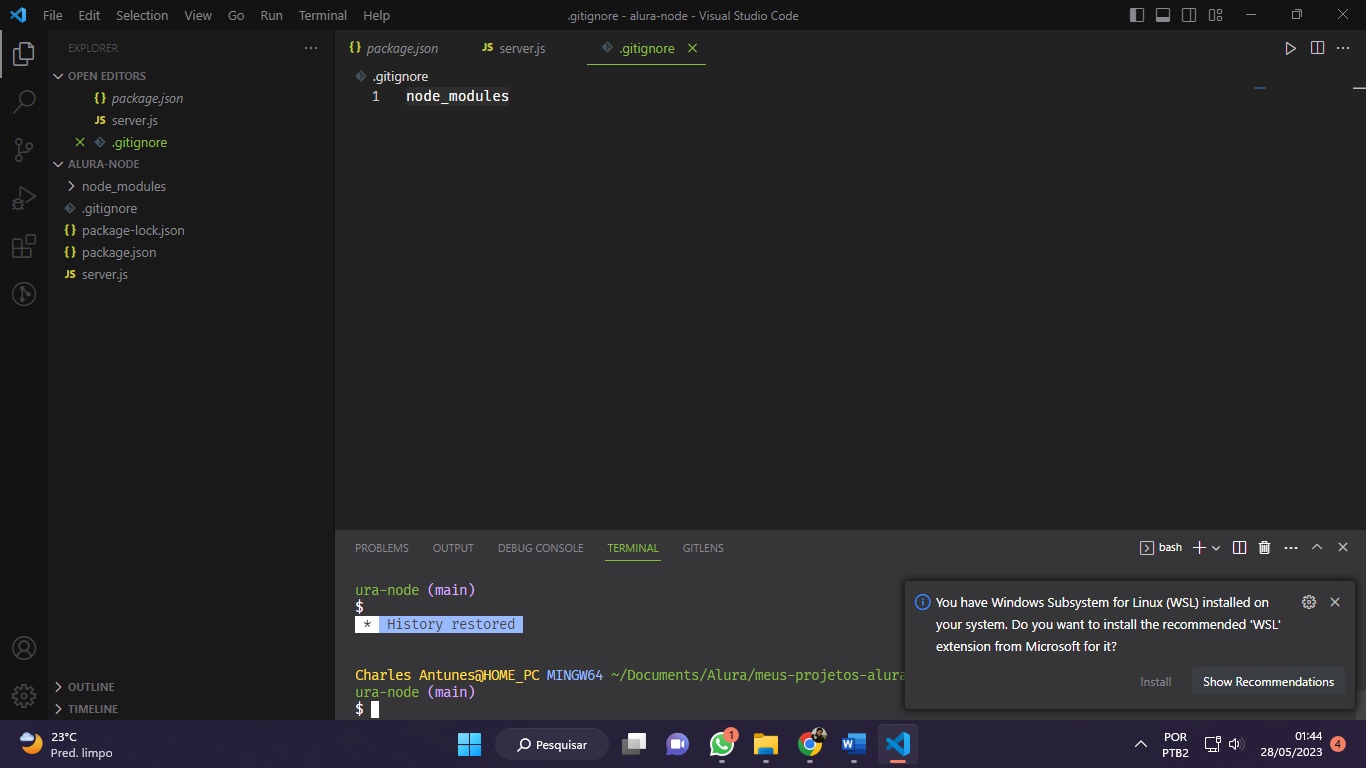
================================ Projeto ========================================

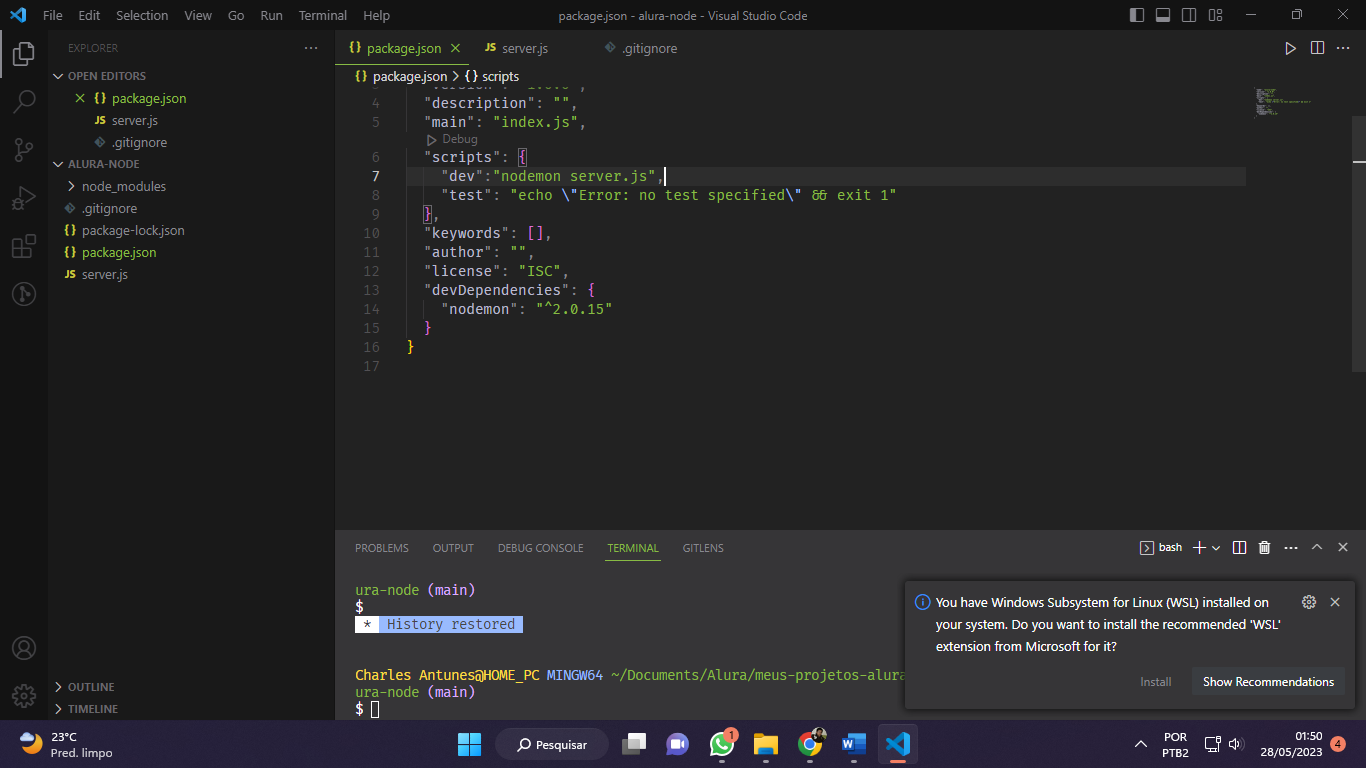
* Dependências para iniciar e salvar automaticamente o projeto, mostra se contém erros no projeto.

npm install [nodemon@2.0.15](mailto:nodemon@2.0.15) -D

obs. -D é para mostrar que é apenas usado no desenvolvimento e quando for para produção ele não vai, vai gerar a pasta node\_modules, onde todas as bibliotecas baixadas vão ser guardadas, por isso criamos a pasta/arquivo .gitignore e adicionamos dentro o node\_modules.



* Ativando o nodemon no package.json, adicionando na “scripts” uma linha com "dev":"nodemon server.js"



Obs. Para executar o nodemon, basta executar npm run dev onde o dev foi adicionado no scripts do package.json

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

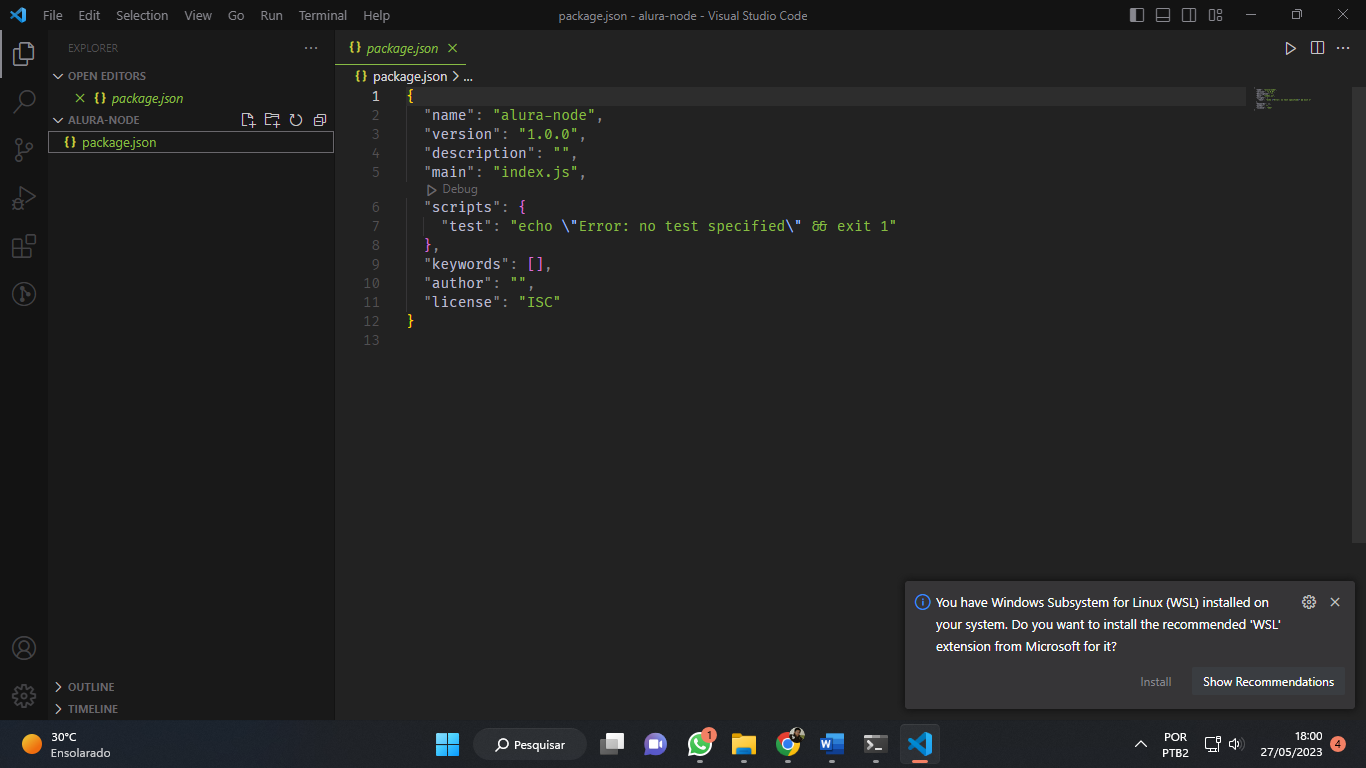
1 – No CMD

mkdir alura-node = criando a pastar do projeto

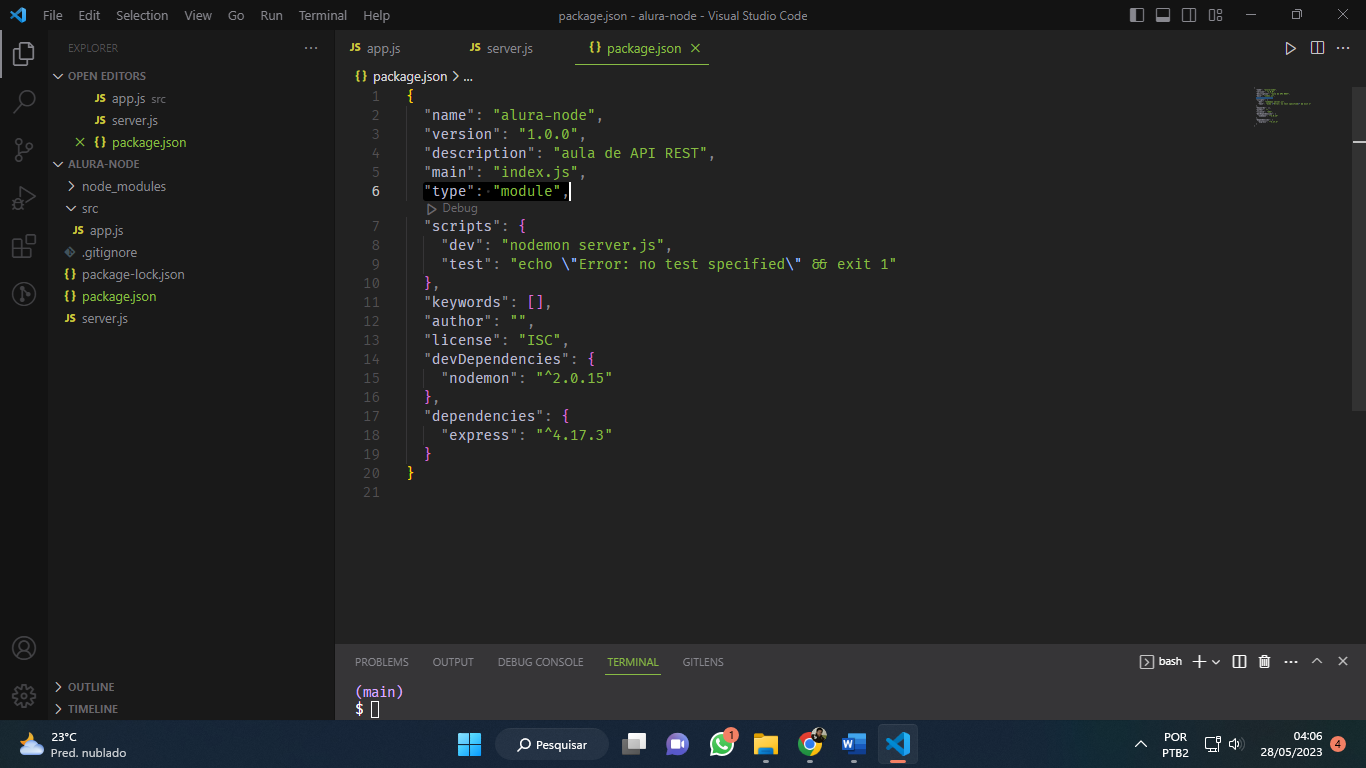
cs mkdir alura-node = entrando na pasta do projeto

npm init -y = criando o projeto, -y quer dizer que as perguntas de criação não aparecem, já vão ser respondida

code . = abrir o vcsCode



* No package.json, vamos cria informação “type”:”module”, para informar que vamos trabalhar com módulos.



O projeto vem com esse arquivo como principal para iniciar, ele usa o “main” como rota para o arquivo principal que é o index.js

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

2 – Criando o servidor local, criar na raiz da pasta o arquivo server.js e importando os componentes nativos.

Dentro do **server.js**

*const* http = require('http');

*const* port = 3000;  *//porta da comunicação do servidor local, depois tem de mudar e criar uma variável de ambiente que irá definir a porta.*

*//criando o servidor*

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

    res.writeHead(200, {'Content-type' : 'text/plain'}); *//resposta é um status escrito na cabeça da requisição(writeHead), o tipo de resposta é um texto.*

    res.end('Curso de Node') *//resposta no body do html, texto teste, depois será mudado*

})

*//mostrando a porta que está usando*

server.listen(port,()=>{

    console.log(**`**Servidor escutando em http://localhost:${port}**`**)

})

Terminal

$ node server.js

Servidor escutando em <http://localhost:3000>

Obs.: clicando no link, abre a página html com a palavra - Curso de Node

Ao instalar o Node, já temos à nossa disposição alguns módulos embarcados para começar a desenvolver nossas bibliotecas e aplicações. Para conhecer mais sobre esses recursos nativos, acesse a documentação oficial referente às APIs nesse [link](https://nodejs.org/api/).

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

3 – Dentro do server.js, definindo as **rotas**, depois tem de separar as rotas em um arquivo separado.

Adicionado rotas e alteração na res.end().

*const* rotas = {

    '/': 'Curso de Nodejs',

    '/livros': 'Entrei na pag. de livros',

    '/autores': 'Listagem de autores'

}

*//criando o servidor*

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

    res.writeHead(200, {'Content-type' : 'text/plain'}); *//resposta é um status escrito na cabeça da requisição(writeHead), o tipo de resp é um texto.*

    res.end(rotas[req.url]);

})

Terminal

$ node server.js

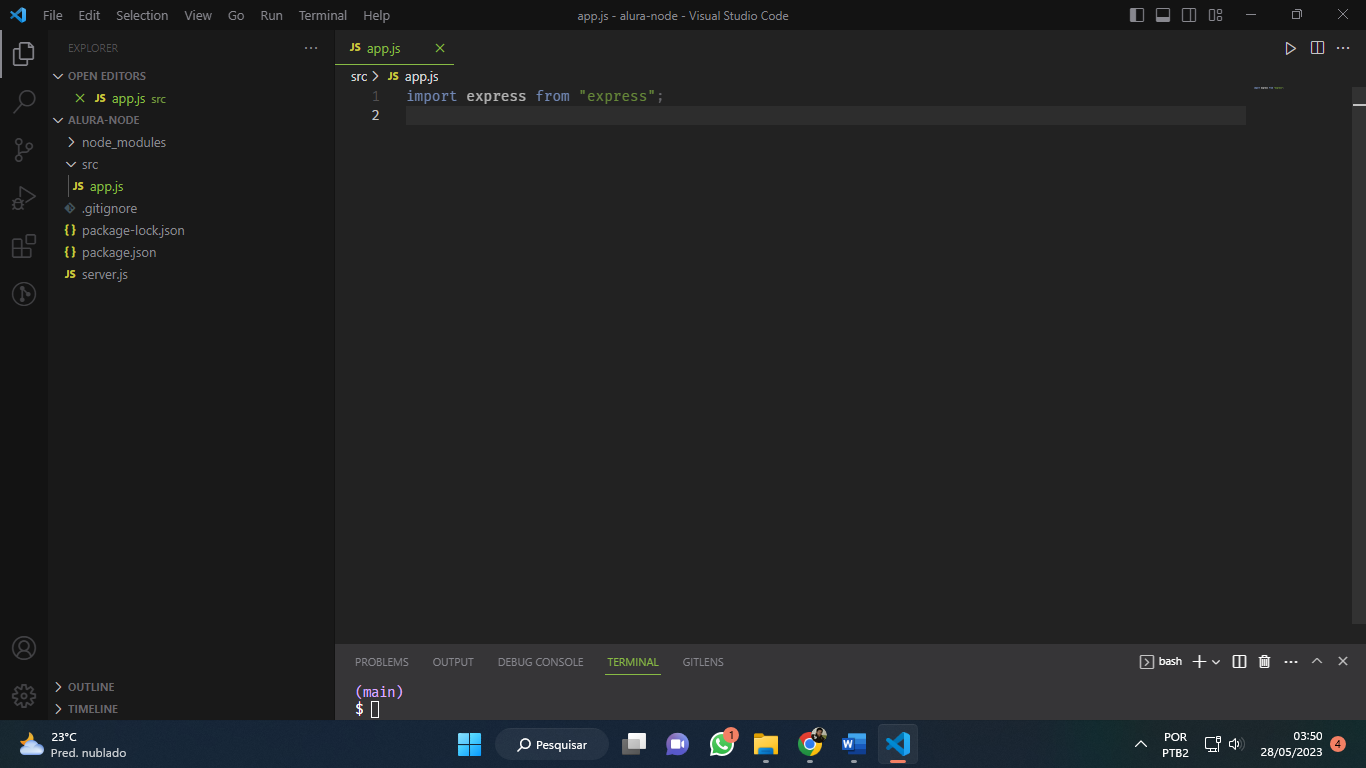
Servidor escutando em <http://localhost:3000>

Obs. ao abrir a página, será direcionado para a rota home(‘/’) com o texto - Curso de Nodejs, depois na página, altere o final da url as duas rotas declaradas, 1°= <http://localhost:3000/livros>, 2º= <http://localhost:3000/autores> . Será exibido -Entrei na pag. de livros e Listagem de autores.

4 – Express , instalando o framework/biblioteca

npm install [express@4.17.3](mailto:express@4.17.3)

* padronizando o projeto com pasta src e dentro, criando o arquivo app.js



* trabalhando no arquivo app.js

import express from "express";

*const* app = express();

*//simulando uma biblioteca para testar a visualizar dos livros na página.*

*const* livros = [

    {

        id: 1,

        "titulo" : "Senhor dos Aneis"

    },

    {

        id: 2,

        "titulo" : "O Hobit"

    }

]

*//rotas com as infos*

app.get('/', (*req*,*res*)=>{  *//buscar as inf da home*

    res.status(200).send('Curso de Nodejs com API'); *// resposta e msg*

})

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

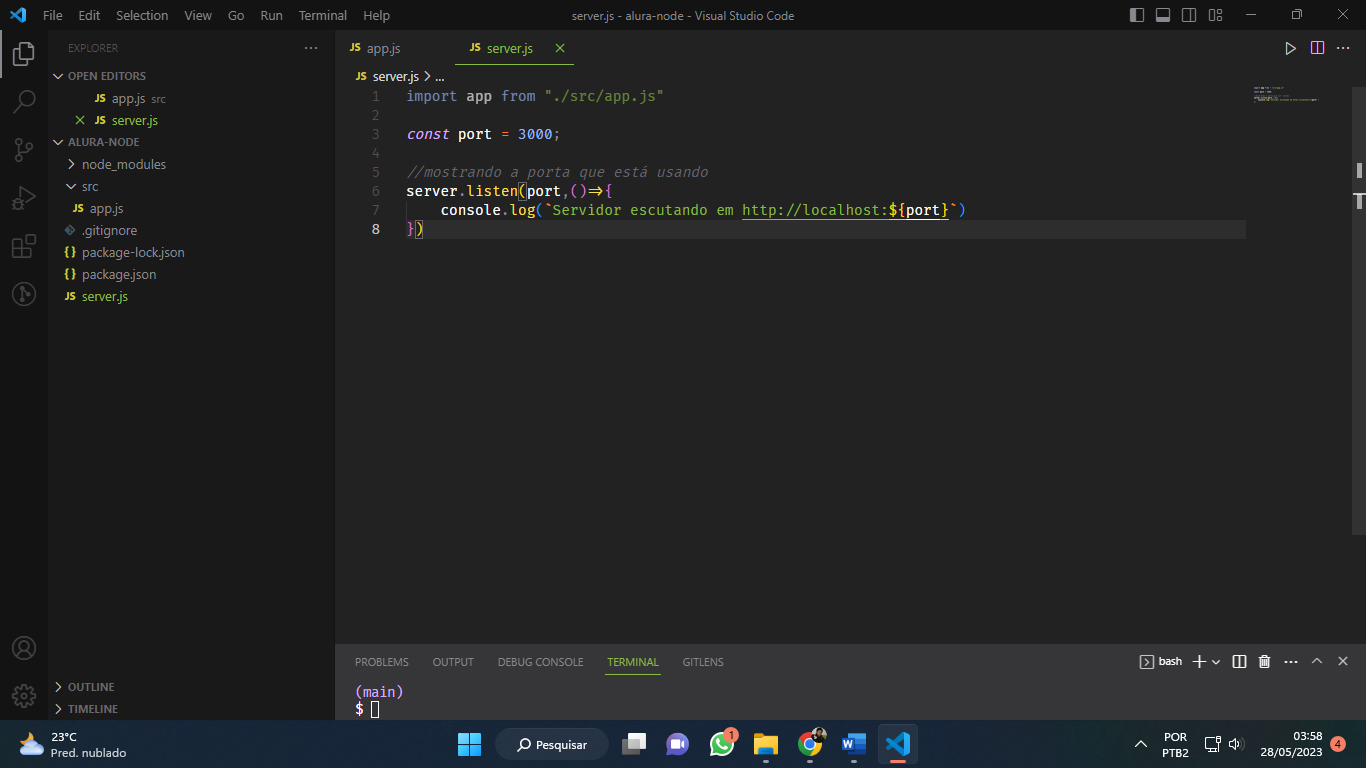
    res.status(200).json(livros) *//convertendo p json os arquivos recebidos*

})

export default app  *//exportando para esses dados serem trabalhados no server.js*

* Apagando o tipo de exportação que é o require e colocando o import, também apagando os códigos de teste de server.js.

Obs. Como o express foi importado no app.js, vamos importar apenas o app.js que já vem com o express. Não esquece de da extensão .js



Obs. Apaguei os comentários do port.

* Iniciando os novos códigos, com alteração na port, adicionado duas opções de portas. Outra foi apagando o server e colocando app

import app from "./src/app.js"

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

*//mostrando a porta que está usando*

app.listen(port,()=>{

    console.log(**`**Servidor escutando em http://localhost:${port}**`**)

})

* Inicializando para testar

npm run dev

[nodemon] 2.0.15

[nodemon] to restart at any time, enter **`**rs**`**

[nodemon] watching path(s): \*.\*

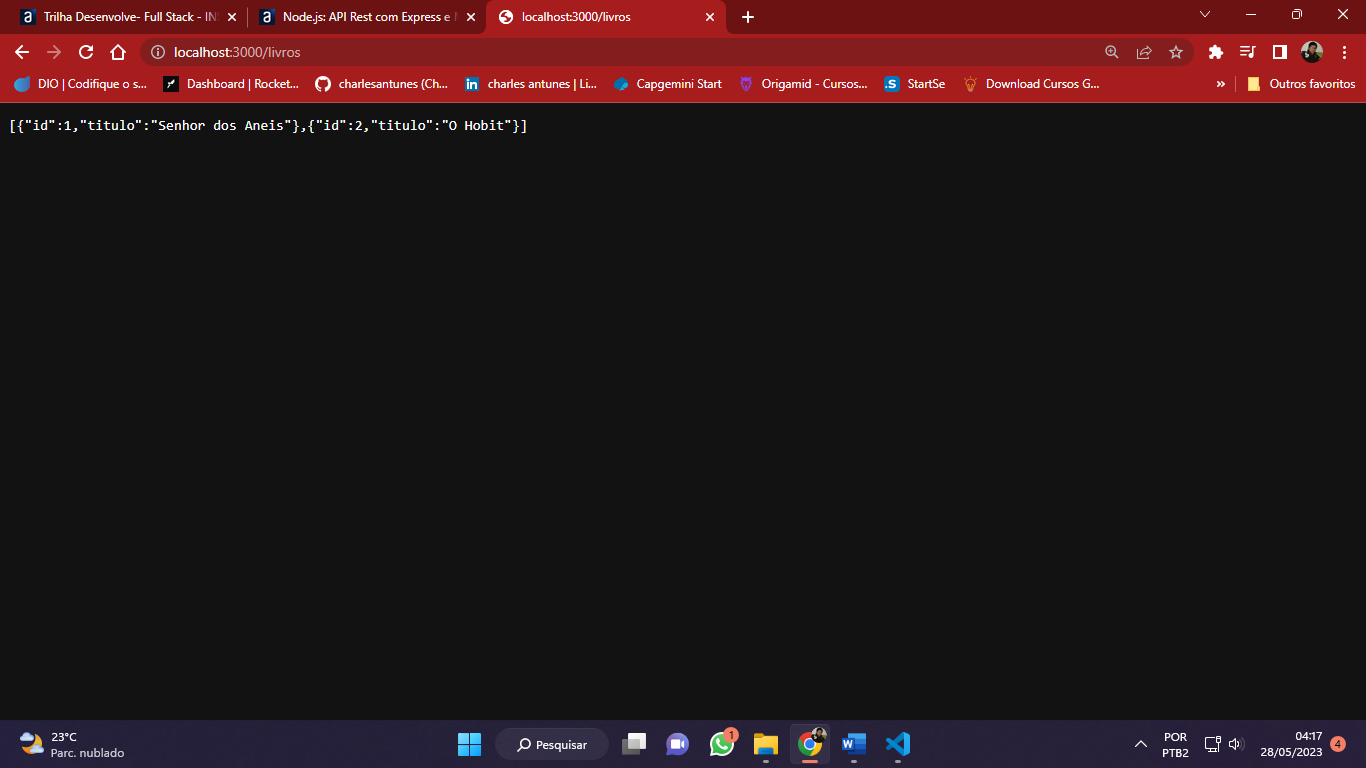
[nodemon] watching extensions: js,mjs,json

[nodemon] starting **`**node server.js**`**

Servidor escutando em http:*//localhost:3000*

Obs. Observe que estou usando nodemon e se não aparecer o “servidor escutando em http://localhost:3000 “, significa que tem erro no cod

* Na página, se alterar o final da url vai aparecer os livros com os filmes criados em json



========================= **INICIANDO O CRUD, TUDO JUNTO, SEM ESTRUTURA, APENAS TESTE** =========================

import express from "express";

*const* app = express();

app.use(express.json()) *//converte o entra como json() em objeto e ser tratado*

*//simulando uma biblioteca para testar a visualizar dos livros na página.*

*const* livros = [

    { id: 1, "titulo" : "Senhor dos Aneis" },

    { id: 2, "titulo" : "O Hobit" }

]

* **PESQUISAR TODOS OS LIVROS.**

*//===================== Pesquisa todos os Livros =====================================*

*//rotas com as infos*

app.get('/', (*req*,*res*)=>{  *//buscar as inf da home*

    res.status(200).send('Curso de Nodejs com API'); *// resposta e msg*

})

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

    res.status(200).json(livros) *//convertendo p json os arquivos recebidos*

})

* **ADICIONAR**

*//====================== Adicionar ================================================*

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

    livros.push(req.body)   *//push() = adicionar, o que estiver no corpo, adiciona em livros*

    res.status(201).send('O Livro foi cadastrado com sucesso')

})

export default app  *//exportando para esses dados serem trabalhados no server.js*

obs. Codigo organizado no array livros, adicionado o post e no começo adicionado add.use(express.json()) para converter json() em objeto.

* **ATUALIZAR**

*//====================== Atualizar ================================================*

app.put('/livros/:id', (*req*,*res*)=>{

*let* index = buscaLivros(req.params.id);*// req.params.id é o id do endereço*

    livros[index].titulo = req.body.titulo; *//atualiza o títudo do corpo*

    res.status(201).json(livros)

})

*function* buscaLivros(*id*) { *//func p/ buscar livro pelo id, depois ela é substituida pelo bd*

    return livros.findIndex(*Element* => Element.id == id)*//findIndex() = verifica todos indices*

}

* **PESQUISAR UM LIVRO (essa vai ser inserido no início, junto com o pesquisar todos)**

*//====================== Pesquisa um Livro============================================*

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

*let* index = buscaLivros(req.params.id);

    res.status(200).json(livros[index]);

})

========================================== **NO POSTMAN** ==========================================

* **PESQUISAR TODOS OS LIVROS.**

GET http://localhost:3000/livros

Resposta:

[

    {

        "id": 1,

        "titulo": "Senhor dos Aneis"

    },

    {

        "id": 2,

        "titulo": "O Hobit"

    }

]

* **PESQUISAR UM LIVRO**

GET http://localhost:3000/livros/

Resposta:

{

    "id": 2,

    "titulo": "O Hobit"

}

* **ADICIONAR**

POST http://localhost:3000/livros/

{

    "id": 3,

    "titulo": "Matrix"

}

Resposta:

O Livro foi cadastrado com sucesso

* **ATUALIZAR** (o id vai no URL, 3 era o Matriz)

PUT http://localhost:3000/livros/3

{

    "titulo": "Velozes e furiosos"

}

Resposta:

[

    {

        "id": 1,

        "titulo": "Senhor dos Aneis"

    },

    {

        "id": 2,

        "titulo": "O Hobit"

    },

    {

        "id": 3,

        "titulo": "Steve Jobs"

    }

]

DELETE http://localhost:3000/livros/1

Resposta:

Livro 1 removido com sucesso

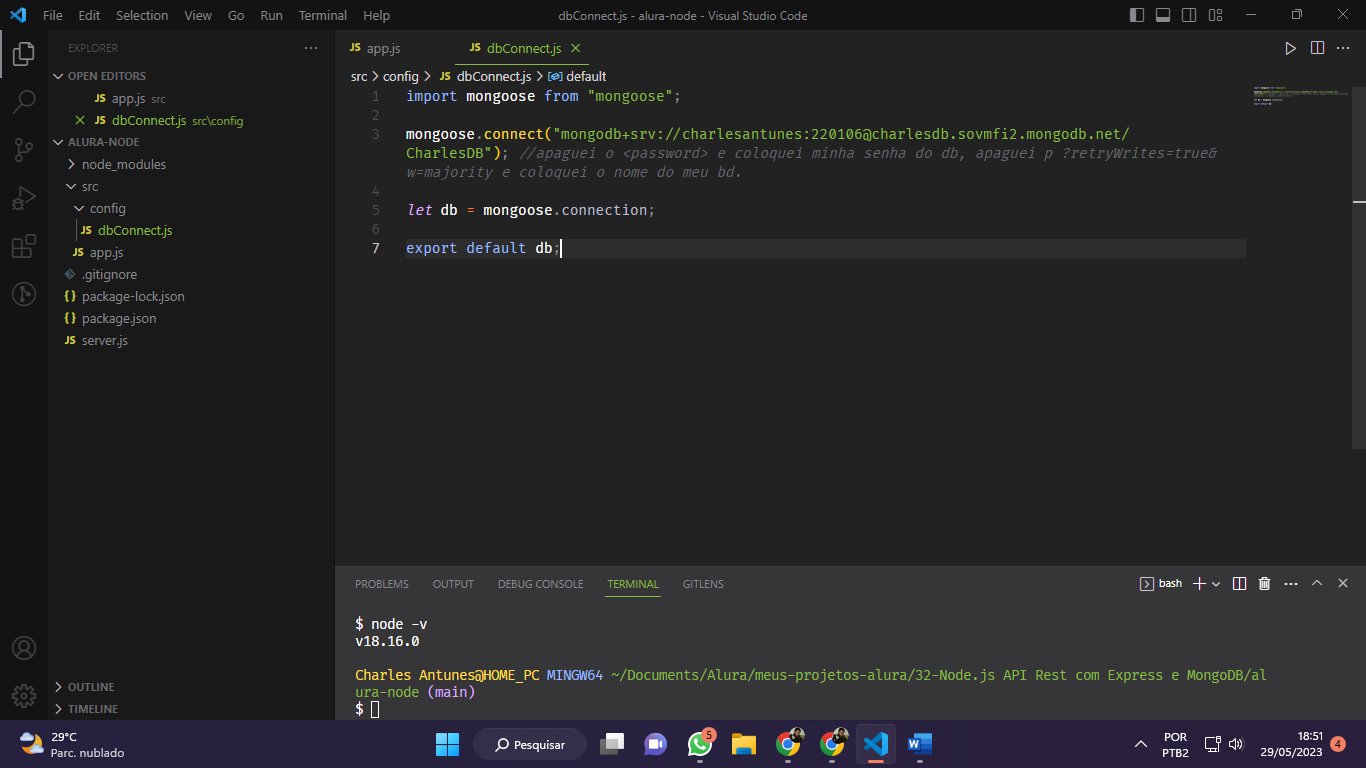
========================================== **FIM DO CRUD TESTE** =========================================

================= REALIZANDO AS MUDANÇAS PARA BOAS PRÁTICAS E COMUNICAR COM MONGODB CLOUD ======================

1. – Instalando o mongoose (auxilia na comunicação com o bd)

npm install mongoose@6.2.6

Dentro da pasta src, criar uma pasta com o nome config e dentro criar o arquivo dbConnect.js, buscando o link de conexão no mongodb/database/connect/drivers.



import mongoose from "mongoose";

mongoose.connect("mongodb+srv://charlesantunes:220106@charlesdb.sovmfi2.mongodb.net/CharlesDB"); *//apaguei o <password> e coloquei minha senha do db, apaguei p ?retryWrites=true&w=majority e coloquei o nome do meu bd.*

*let* db = mongoose.connection;

export default db;

No app.js