**1º Preparar o ambiente, instalando node, yarn, vscode**

-Node (foi via linha de comando) via Chocolatey – cinst install node –version coloque a versão

-Yarn (foi via linha de comando) via Chocolatey

Ferramentas: Dracula Theme, Material Icon Theme, RocketSeat Snipptes do ReactJS, nome da Font: Firacode

Depois em file preferences>settings (Abre as configurações do nosso vscode), COLOQUE NO MODO JSON, e altere o nome da font para Fira Code que foi instalado, e o: “editor.fontLigatures”: true

Terminal igual o dele, explica no post do blog rocketseat

**2º Começar criar api com nodejs**

1º Criar uma pasta do projeto, e depois disso dar um yarn init, para criar o arquivo package.json, a onde fica localizado todas as dependências do projeto.

Com isso, podemos instalar o express que é responsável pelas rotas dentro do node

*yarn add express*

Crie uma pasta SRC e um arquivo server.js

**2.1º** Fazemos um require para um const:

const express = require(‘express’);

Esse require vai lá nas nosso node modules e importa o nosso express para um const

**2.2º** Logo depois, podemos instanciar transformando-o em um objeto para uma outra const

const server = express();

Agora podemos utilizar todos os métodos que estão no server que são do express:

Principais, é o listen, get, use

**2.3º** Agora definir uma rota para que o node possa startar

Utilizando o objeto server com a função listen:

server.listen(3333);

com isso quando start o projeto com node, ele inciará nessa porta

**2.4º** Lembrando que temos que definir uma rota primeiramente, utilizaremos o server e o método get e com isso passamos como primeira propriedade a rota (‘/’), e depois uma arrow function com as propriedades req, res para podermos manipular, e tratar a req do client e devolvermos uma resposta

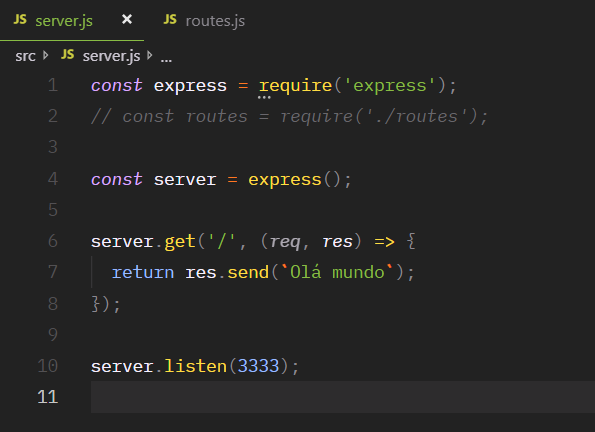
server.get(‘/’, (req, res) => {

return res.send(`Hello World`);

});

Note que nessa função utilizamos a propriedade res, para implementar o método send, que retornará nossa frase.

Seu código ficará assim:



**2.5º** Agora precisamos refator esse código, afim de deixar apenas um arquivo responsável pelas as rotas. Só que antes disso, está chato toda vez termos que ficar startando o projeto com node e depois ter que parar e dar node novamente, para que isso fique automática precisamos de uma dependência o: nodemon

Para isso basta digitar o comando:

yarn add nodemon

**2.6º** Com nodemon instalado, basta irmos no package.json, na primeira seção, e adicionar um script. Com esse script podemos adicionar qualquer coisa e quando chamarmos ele será ativo. Então vamos colocar para o nodemon

"scripts": {

"dev": "nodemon src/server.js"

}

Então quando chamarmos o comando yarn dev, ele irá startar o nodemon, assim podemos abrir outro terminal e deixar o projeto rodando e quando salvarmos ele automática restarta o projeto

**2.7º** Agora vamos refatorar a parte de rotas, para isso crie um outro arquivo na pasta src chamado routes.js

E lá, vamos importar também o express

const express = require (‘express’);

**2.8º** Vamos também utilizar as funções de rotas que está dentro do objeto express, para isso coloque-a dentro de uma const

const routes = express.Routes();

**2.9º** Agora é simples, basta utilizar o método get que está em routes para poderar renderizar (buscar - get) do routes e aplica as configurações de rota como propriedade e a arrow function com req e res, e utilize o método res com o método json, para renderizar um json

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

return res.json(**`**Olá**`**);

});

**2.10** Agora vamos construir um json e vamos trabalhar uma requisição do usuário que vai enviar o seu nome, e vamos responder o nome dela, dando ola.

Basta adicionar nesse método json um template string, utilizando o objeto req que vou passado como propriedade, o método query, com o nome da req que foi passado. Lembrando de dar um nome para esse json, e dentro de uma template string. Como nesse exemplo

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

return res.json({message: **`**Olá ${req.query.name}**`**});

});

**2.11** Para finalizar, vamos exportar esse serviço de rotas para utilizar dentro do nosso server.js, só adicionar

*module*.*exports* = routes;

então todo o código do routes.js ficara assim:

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

*const* routes = express.Router();

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

return res.json({message: **`**Olá ${req.query.name}**`**});

});

*module*.*exports* = routes;

E sendo assim, vamos voltar no nosso server.js, importar nosso routes com require adicionado o caminho, e fazer usando utilizando o objeto server.

*const* routes = require('./routes');

server.use(routes);

O arquivo server.js ficará assim:

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

*const* routes = require('./routes');

*const* server = express();

server.use(routes);

server.listen(3333);