**Node.js**

Criando uma API do Zero!

**CRIAÇÕES E INSTALAÇÕES NECESSÁRIAS PARA TODOS OS PROJETOS**

**Passo 1:** Abrir no CMD na pasta ‘Projetos’ e criar a pasta do projeto com o comando mkdir + nome\_do\_projeto ou dentro da pasta ‘Projetos’ clicar com o botão direito / nova pasta.

**Passo 2:** Chamar a pasta do projeto criado com o comando cd + nome\_do\_projeto

**Passo 3:** Abrir o VSCODE com o comando code .

**Passo 4:** Com o CMD ainda aberto ou no Terminal do VSCODE, precisamos iniciar nosso projeto Node, então o comando para startar o trabalho é **npm init –y** (o -y significa que a inicialização vai ser respondido tudo com yes/sim).

Após a conclusão da inicialização abrirá uma arquivo automaticamente chamado package.json (nesse arquivo encontraremos todas as informações do nosso projeto, incluindo o formato que se deve rodar no servidor local e também as dependências instaladas neste caso o express).   
  
Além disso, precisamos incluir uma informação no package.json. Abaixo de **“version”**, incluir a seguinte informação “type”: “module”,

**Passo 5:** Agora precisamos de uma biblioteca e a mais utilizada é a EXPRESS do site npm. Para ativar a biblioteca basta dar o comando **npm i express**  no CMD ou no Terminal do VSCODE e aguardar a instalação.

Após a instalação surgirá 2 novos aquivos, um chamado node\_modules (que é todas as bibliotecas necessárias para rodar o express) e outro arquivo chamado package-lock.json

**Passo 6:** Com o VSCODE aberto, criar um arquivo chamado server.js

**CHEGOU A HORA DE FINALMENTE A CODAR.**

**Passo 7:** Agora, no arquivo server.js precisamos importar a biblioteca Express, criar uma variável padrão para usar a biblioteca e criar uma rota.  
Mas o que é uma ROTA?  
  
**ROTA** é a comunicação entre o Back End e o Front End. Porém não da para ir criando ROTAS sem conhecer os **Tipos de Rotas /** **HTTP Métodos**.

***Get*** *=> Mostrar / Listar*

***Post*** *=> Criar*

***Put*** *=> Editar vários*

***Patch*** *=> Editar um*

***Delete*** *=> Deletar*

As rotas precisam de um endereço, exemplo [www.lojadoseuzé.com/xxxxx](http://www.lojadoseuzé.com/xxxxx)

Após o / , essa sim será nossa rota de direção.

* Para fazer tudo isso, basta seguir os passos abaixo:

/\* impotação da biblioteca\*/

import express from 'express'

/\* variável que será utilizada para pegar tudo que esta dentro da biblioteca express \*/

const app = express()

/\* Rota usuario + Requisição + Resposta. \*/

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

*res*.send('Ok, deu bom')

})

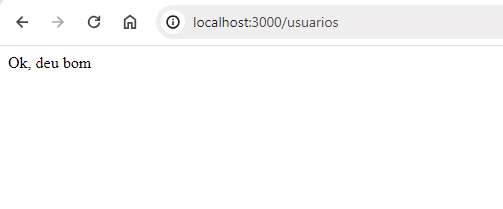
/\* Precisamos avisar em qual porta nosso servidor vai rodar \*/

app.listen(3000)

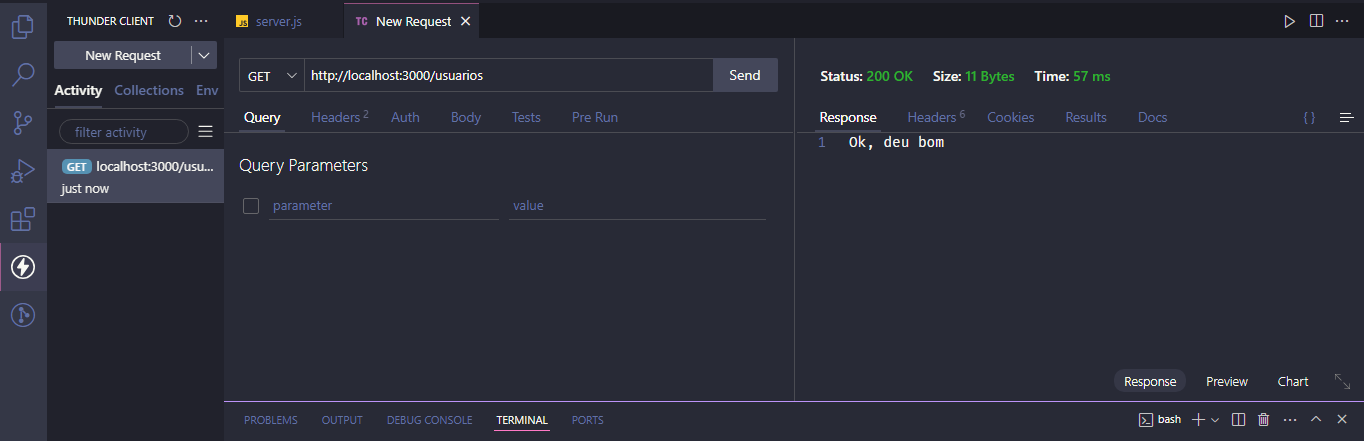
**Passo 8:** Para fazer rodar no navegador o código, basta colocar o seguinte comando node server.js (Se não aparecer nada é porque esta ok)

Existe outro comando pra caso o servidor der algum erro, dar restart automático que é node –-watch server.js

Agora abrir o navegador de sua preferencia e colocar o endereço do servidor local da sua máquina: [**https://localhost:3000/usuarios**](https://localhost:3000/usuarios)que vai aparecer na tela as informações de ***res/resposta***



Se NÃO quiser rodar no navegador, existe uma extensão no VSCODE chamada **THUNDER CLIENT**, com essa extensão também conseguimos acompanhar o processo de desenvolvimento .



**OBJETIVO => CRIAR NOSSA API DE USUÁRIOS.**

**Passo 9:** Nosso objetivo é criar nossa API de Usuários, pra isso precisamos fazer o seguinte:

***Post*** *- Criar um usuário*

***Get*** *- Listar todos os usuários*

***Patch*** *- Editar um usuário*

***Delete*** *- Deletar um usuário*

/\* impotação da biblioteca\*/

import express from 'express'

/\* variável que será utilizada para pegar tudo que esta dentro da biblioteca express \*/

const app = express()

/\* variável que será utilizada para lista de usuários \*/

const users = []

/\* Rota Post usuario + Requisição + Resposta. \*/

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

     console.log(*req*)

   // Resposta

*res*.send('Ok aqui deu certo')

})

/\* Rota Get usuario + Requisição + Resposta. \*/

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

    // Resposta

*res*.send('Ok, deu bom')

})

/\* Precisamos avisar em qual porta nosso servidor vai rodar \*/

app.listen(3000)

Iniciaremos com **Post**:

Para Criar uma usuário, por exemplo em uma tela de cadastro de login, precisamos enviar essas informações do usuário para algum local e existem 3 formas de enviar essas **Request’s** (Requisição/Informação) .

**1° Forma: Query Params (GET) - Consultas**

www.servidor.com/usuarios**?**estado=bahia**&**cidade=salavador

www.servidor.com/series**?**tipo=comedia**&**streming=netflix

Esses pontos de interrogação **?** são parâmetros por meio da URL, tudo que estiver depois do ponto de interrogação se tornará variáveis.

O **&** comercial significa separar as informações.

O **Query Params** é usado para fazer pesquisas, informações que não são relevantes.

**2° Forma: Route Params (GET, PUT, DELETE) – Buscar, Editar algo específico, deletar.**

Get servidor.com/usuários/22

Put servidor.com/usuários/22

Delete servidor.com/usuários/22

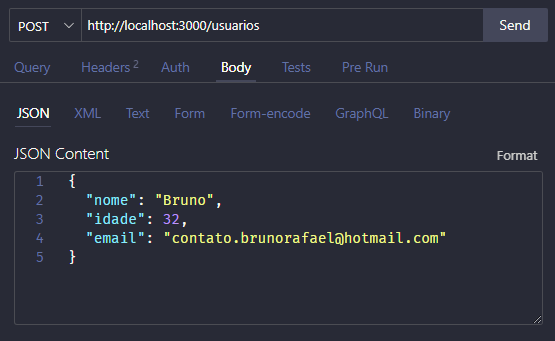
O **Route Params** é usado para acessar uma informação especifica, o numero 22 no exemplo acima significa o ID do usuário.

**3° Forma: Body Params (POST, PUT)**

{  
 “nome”: “Bruno”, “id”:22  
}

O **Body Params** é usado para enviar informações pelo Body, e nessa situação é possível enviar informações mais delicadas como por exemplo: Senha do Cartão, Senha do Usuário ou quando são muitas e muitas informações.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Já entendemos as formas que são enviadas as informações, então iremos utilizar a opção **Body** Params para Criar (POST) o usuário.

Na extensão que instalação tem uma opção chamada Body, após clicar vai abrir uma caixa de texto escrito JSON Content, é neste local que vamos criar o usuário.



Agora precisamos enviar esse formato JSON para o meu server. Mas antes de enviar precisamos avisar meu server que vai chegar um arquivo no formato JSON. Então, só na variável app que criamos, abaixo dela colocar a seguinte código: **app.**use (express.json ( ) )

Agora só conferir se as informações adicionadas no Body estão chegando na Request.

/\* importação da biblioteca\*/

import express from 'express'

/\* variável que será utilizada para pegar tudo que esta dentro da biblioteca express \*/

const app = express()

/\* Aviso ao server que o app vai usar o formato json \*/ 🡸

app.use(express.json())

/\* variável que será utilizada para lista de usuários \*/

const users = []

/\* Rota Post usuario + Requisição + Resposta. \*/

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

    console.log(*req*)

    // Resposta

*res*.send('Ok aqui deu certo')

})

/\* Rota Get usuario + Requisição + Resposta. \*/

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

    // Resposta

*res*.send('Ok, deu bom')

})

/\* Precisamos avisar em qual porta nosso servidor vai rodar \*/

app.listen(3000)

Tudo certo, esta chegando na Request nosso Body, agora precisamos tirar o console.log abaixo da ROTA Post e colocar o seguinte comando para salvar os usuários e não sair mais depois de atualizar os códigos:

**users.**push (req**.**body)   
**res**.status (201).json(req.body)  
  
Agora na hora de listar os usuário e mostrar na tela, eu vou abaixo da ROTA Get e coloco o seguinte comando:   
  
**res**.status (200).json(users)

import express from 'express'

const app = express()

app.use(express.json())

const users = []

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

    users.push(*req*.body)

*res*.status(201).json(*req*.body)

})

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

*res*.status(200).json(users)

})

app.listen(3000)

/\* Objetivo => Criar nossa API de usuários

    - Criar um usuário **OK**

    - Listar todos os usuários **OK**

    - Editar um usuário **OK**

    - Deletar um usuário **OK**

\*/

PRONTO, ESTA CRIADA NOSSA API DE USUÁRIOS.  
  
 Agora temos que colocar essa API em um banco de dados e vou utilizar o **Mongo DB.**

**Passo 1:** Entrar no site do banco de dados Mongo DB e fazer o cadastro.

**Passo 2:** Iniciar o banco de dados Free, criar um domínio e uma senha.

**Passo 3:** Lateral esquerda ir na opção Network Access / Editar / Add Current.... / Confirm.  
  
**Passo 4:** Lateral esquerda ir na opção Database Access / Editar / Built-in-role / Atlas Admin / Update User.  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Banco de dados criado e configurado.   
  
Agora precisamos de uma biblioteca para o Mongo DB rodar lá dentro do nosso código. Vou usar a Prisma.io.   
  
**Passo 1:** Get Start.

**Passo 2:** Set up Prima ORM / Start from scratch / Mongo DB / Alterar de TypeScript para JavaScript /

**Passo 3:** Lateral esquerda ir na opção Network Access / Editar / Add Current.... / Confirm.  
  
**Passo 4:** Lateral esquerda ir na opção Database Access / Editar / Built-in-role / Atlas Admin / Update User.

**Passo 5:** Instalar o Prisma com o comando npm install prisma --save-dev

**Passo 6:** Criar arquivos com o comando npx prisma init

**Passo 7:** Retornar até o Mongo DB / Database / Conect / Drivers / Node.js / Add your connection string...

**Passo 8:** Retornar até o Mongo DB / Database / Conect / Drivers / Node.js / Add your connection string...