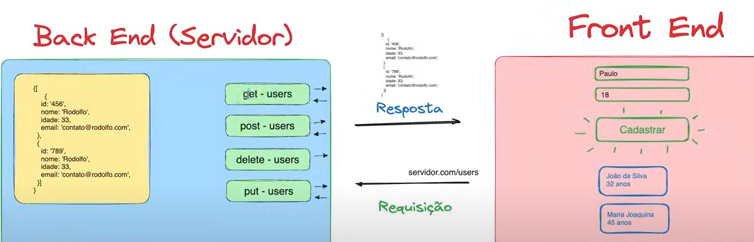
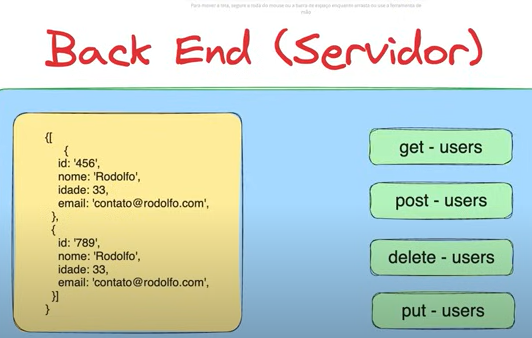
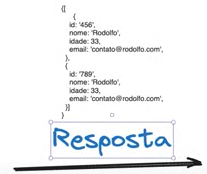
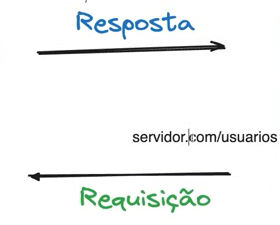
**Criando uma API do ZERO com Node.js e Banco de D**

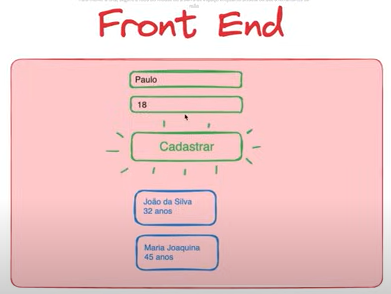
**Aplicação para desenhar projeto:**  <https://excalidraw.com/>

**Link curso:** <https://www.youtube.com/watch?v=PyrMT0GA3sE>









**Instale o Node.js:** <https://nodejs.org/en>

**Instale o NPM:** <https://www.npmjs.com/>

Verificar se o **node** ou **npm** já está instalado: node –v / npm –v

**Crie uma pasta para o projeto**:

Após instalação do **node**.**js** e **npm**

Iniciando projeto, abra o terminal e digite: **npm init –y ->** para iniciar o nosso projeto node

Ao iniciar irá criar o arquivo: **package.json**

Este arquivo terá todas as informações do nosso projeto

Agora vamos criar o primeiro arquivo do projeto, com o nome “**server.js”**

**Instalar a biblioteca “Express”**

comando: **npm i express**

**Documentação**: <https://expressjs.com/>

A biblioteca "**Express**" do npm é um framework web rápido e minimalista para Node.js, amplamente utilizado para criar aplicativos web e APIs. Aqui estão algumas das principais funcionalidades e usos do Express:

1. \*\*Roteamento\*\*:

- Define rotas para diferentes métodos HTTP (GET, POST, PUT, DELETE, etc.) e caminhos URL.

- Suporta middleware que pode ser aplicado a rotas específicas ou globalmente.

2. \*\*Middleware\*\*:

- Permite a adição de funções intermediárias para processar solicitações HTTP.

- Pode ser usado para várias tarefas, como autenticação, manipulação de cookies, análise de corpos de solicitações, manipulação de sessões, entre outras.

3. \*\*Gerenciamento de solicitações e respostas\*\*:

- Facilita o manuseio de dados da solicitação (query strings, parâmetros de URL, corpos de solicitações JSON/form).

- Fornece métodos para enviar respostas (HTML, JSON, arquivos, redirecionamentos, etc.).

4. \*\*Template Engines\*\*:

- Suporta a integração com motores de template como Pug, EJS, Handlebars, permitindo a renderização de páginas HTML dinâmicas.

5. \*\*Modularidade\*\*:

- Incentiva a criação de aplicativos modulares, permitindo que você divida seu aplicativo em diferentes arquivos e módulos.

6. \*\*Integração com Banco de Dados\*\*:

- Pode ser facilmente integrado com várias bibliotecas de banco de dados, como Mongoose (para MongoDB), Sequelize (para SQL), entre outras.

7. \*\*Segurança\*\*:

- Suporta a implementação de práticas de segurança, como a configuração de cabeçalhos HTTP apropriados, proteção contra CSRF e XSS, entre outras.

8. \*\*Facilidade de Testes\*\*:

- Facilita a criação de testes para aplicativos e APIs, permitindo a simulação de solicitações HTTP e a verificação das respostas.

### Exemplo Simples de Uso do Express

Aqui está um exemplo simples de um servidor HTTP básico usando Express:

// Importando a biblioteca Express

const express = require('express');

// Criando uma aplicação Express

const app = express();

// Definindo uma rota para o caminho raiz ("/")

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

res.send('Olá, mundo!');

});

// Definindo uma rota para o caminho "/sobre"

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

res.send('Esta é a página sobre.');

});

// Iniciando o servidor na porta 3000

app.listen(3000, () => {

console.log('Servidor rodando na porta 3000');

});

```

Neste exemplo:

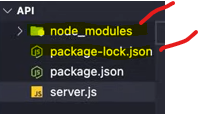
- `require('express')` importa a biblioteca Express.

- `app.get()` define rotas para os caminhos especificados.

- `app.listen()` inicia o servidor e o faz ouvir na porta 3000.

O Express é amplamente utilizado devido à sua simplicidade, flexibilidade e grande conjunto de funcionalidades que facilitam o desenvolvimento de aplicativos web e APIs robustas.

Após instalar o Express, irá criar os arquivos abaixo:



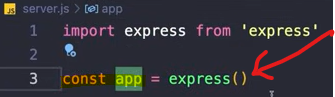
No arquivo **server.js** vamos importar o **Express**



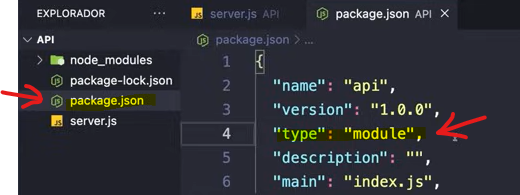
Antes para importar utilizava esta forma antiga, mas está sendo descontinuada:

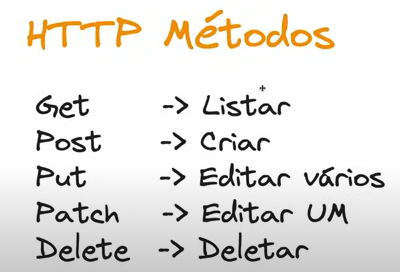


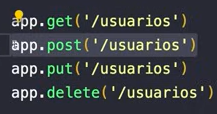
Crie uma váriavel para poder utilizar a bliblioteca Express:



Acesse o arquivo **package.json** e adicione **“type”: “module**” para dizer que está usando a forma nova de import do express



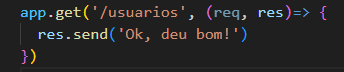






Ou





Avisar o servidor qual porta vai ser utilizada:



Iniciar servidor:

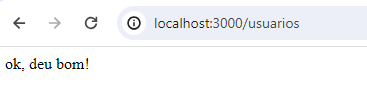


Para parar o servidor é “**CRTL + c**”

E para evitar que pare em cada alteração feita no código durante o desenvolvimento, basta iniciar usando o comando: “***node --watch server.js”*** *desta forma automaticamente vai restartar o servidor*

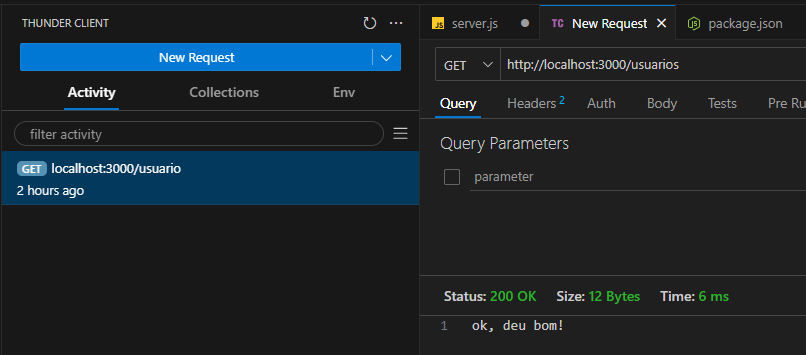
******

Testando rota

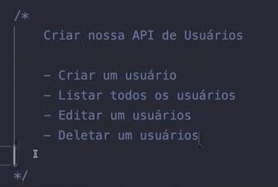


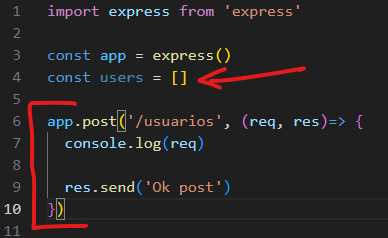
Instale o **Thunder Client** para testar a rota pelo **get, post, put e delete**

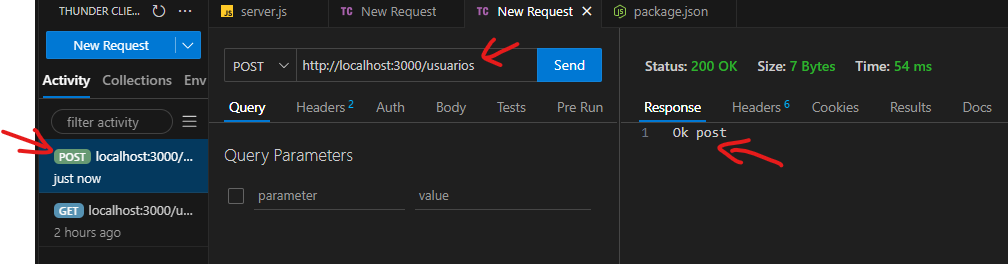




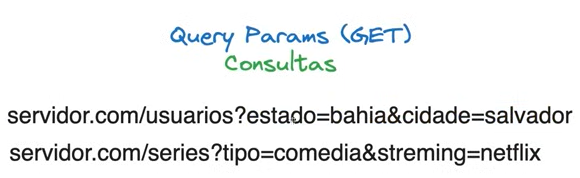
Próximas etapas do curso:







**EXISTEM TRÊS FORMAS PARA ENVIAR REQUISIÇÕES:**



**Route Params** é sempre para enviar uma informação apenas

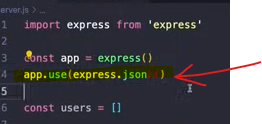


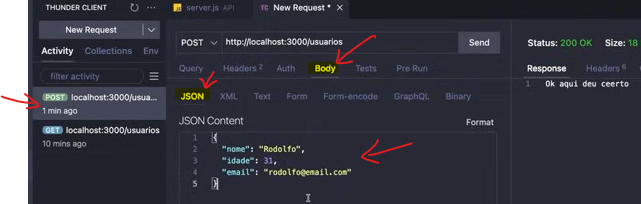
E por ultimo, para enviar informações mais delicadas ou um volume maior de infrmações utilizamdo **Body Params**

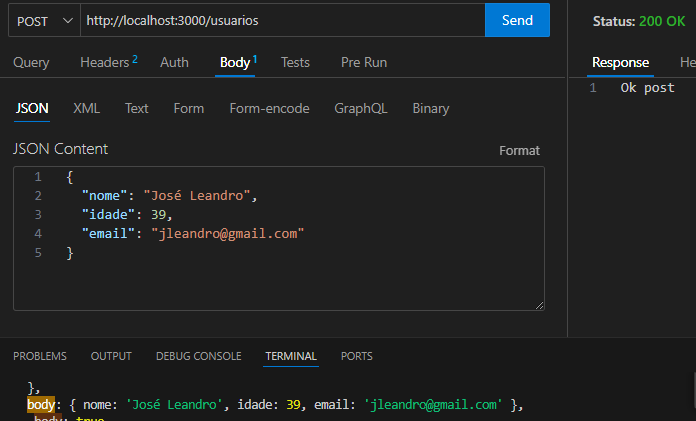


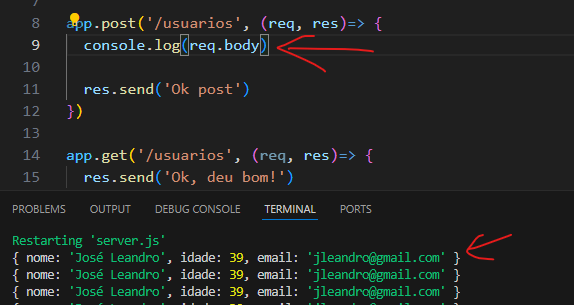
Simular um cadastro

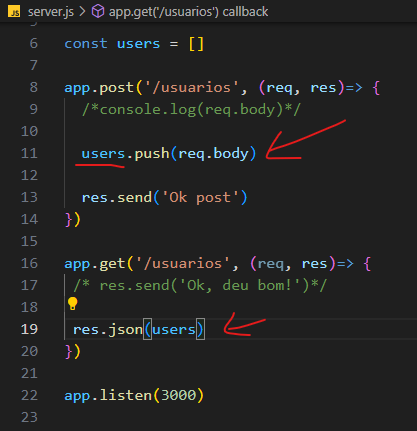
Antes do teste adicione o código para informar o “**Express”** para que use o “**json”**



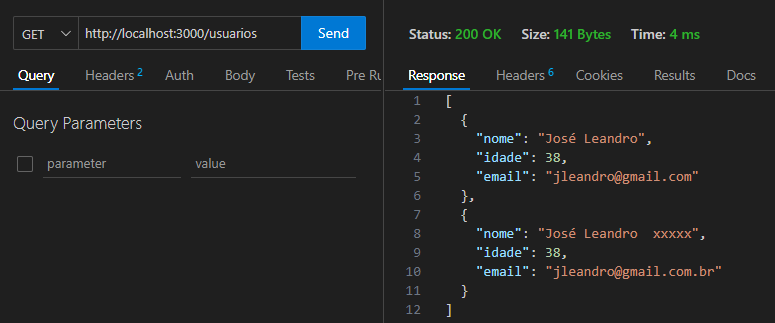


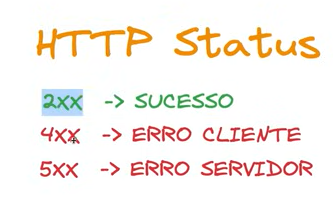




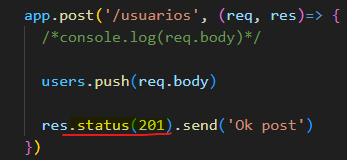


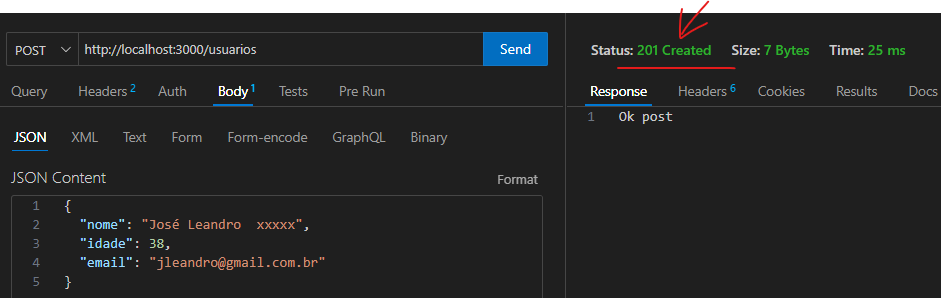
Consultando lista

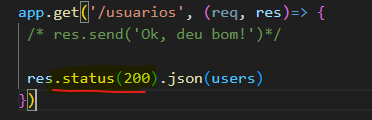




Adicionando status de retorno com sucesso **“Push”**



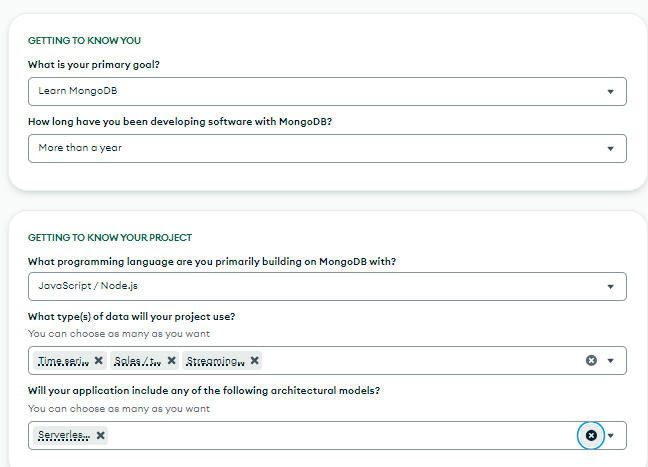


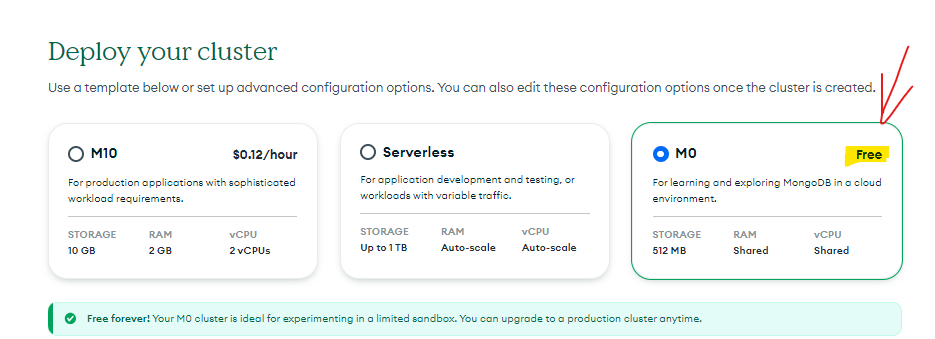




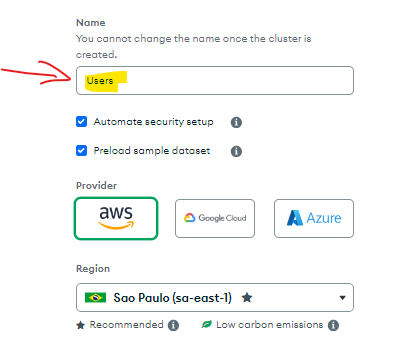
Agora iremos criar o banco de dados, antes crie uma conta no **MongoDB:** <https://www.mongodb.com/pt-br>

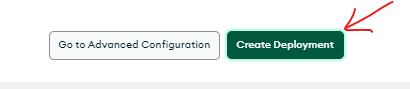
**Email**: [jlrodrigues200984@gmail.com](mailto:jlrodrigues200984@gmail.com) **puk:** motoNumCarEU#





**Nome do banco**

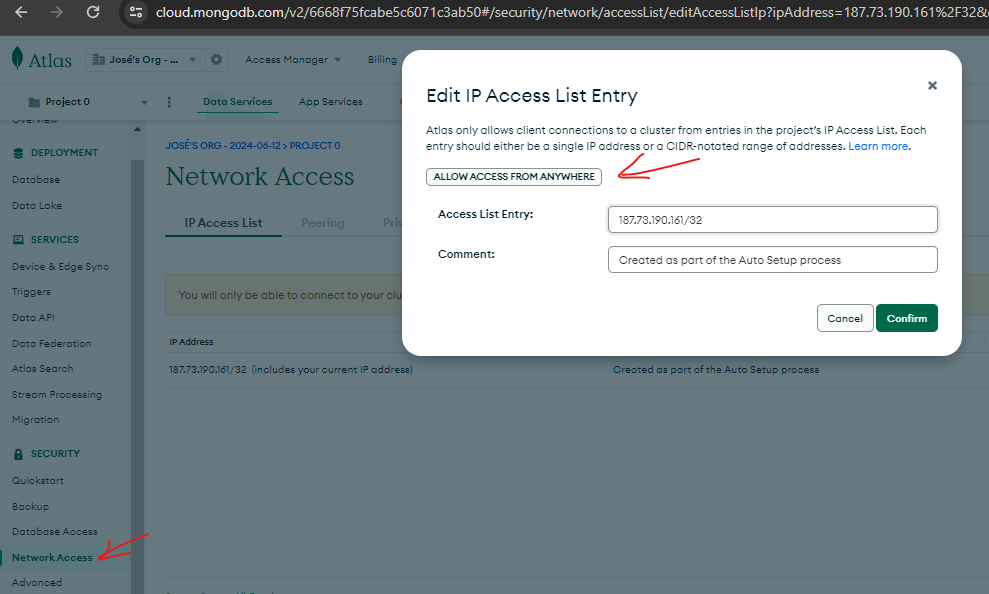


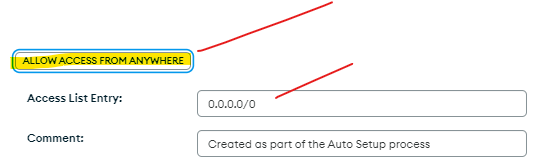


**Usuário BD:** jlrodriguesDev

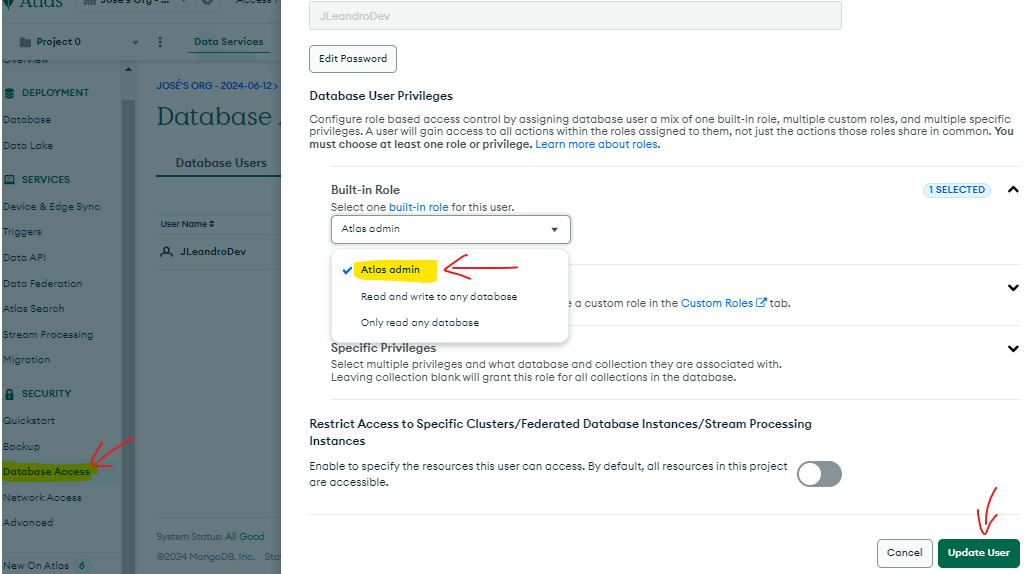
**Puk:** e5xyNAzTJhjNW9Q1

Uma confguração importante para fazer no MongoDB após criar conta



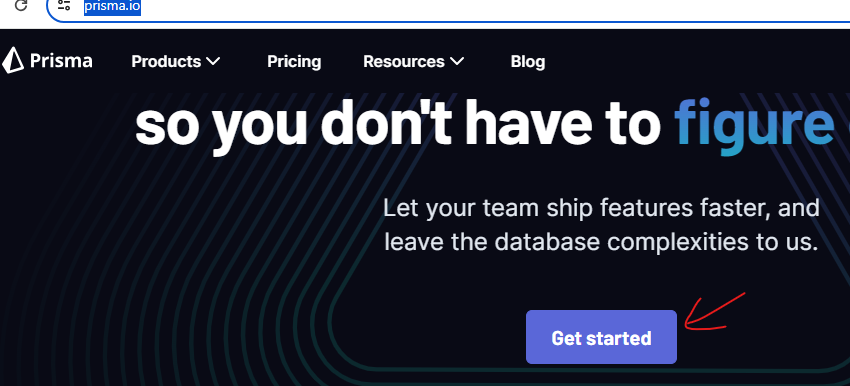


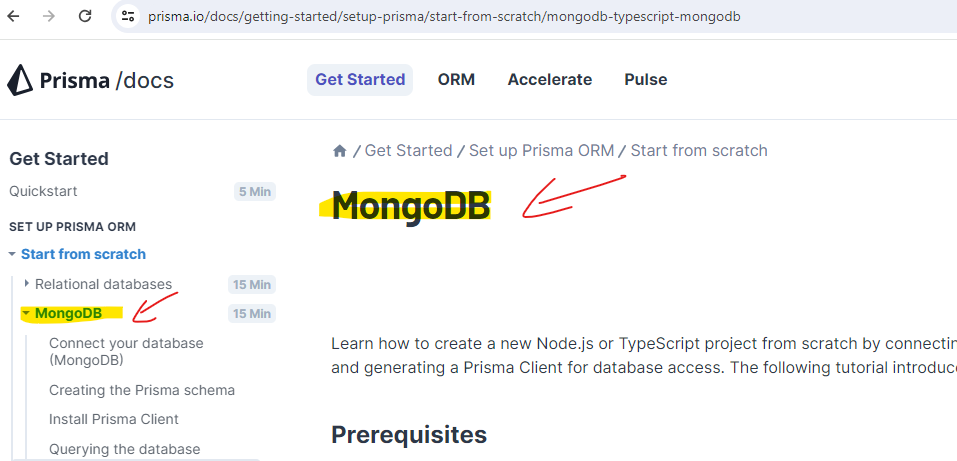
Outra configuração importante é mudar o tipo de usuário para “**Admin”**

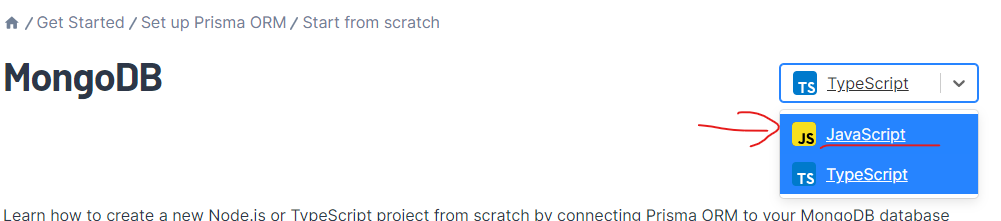


Agora vamos instalar uma biblioteca para conseguir ter acesso ao banco de dados pela aplicação

**Acesse o link:** <https://www.prisma.io/>

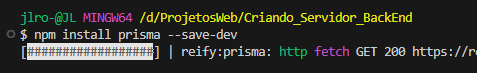


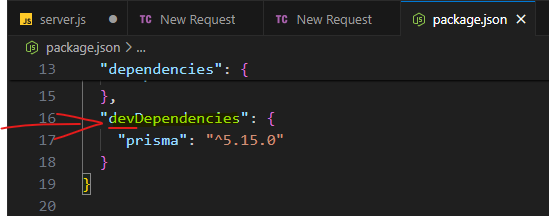




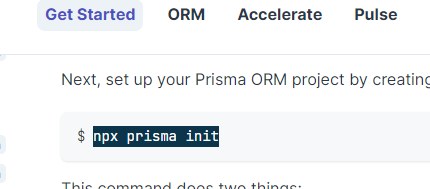
**Comando para instalar a biblioteca,** vai instalar uma dependencia de desenvolvimento

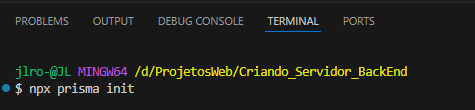




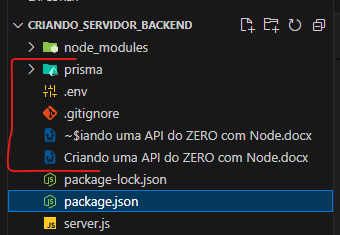


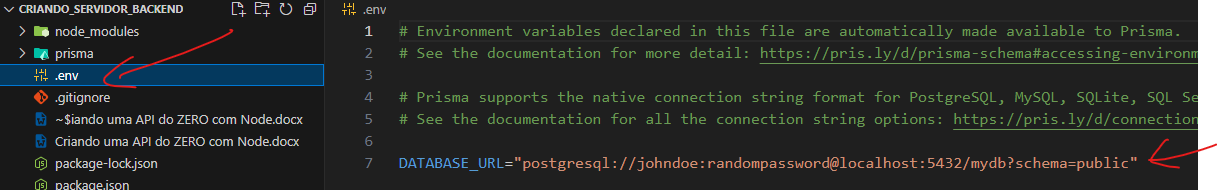
Após instalar aplique o comando:

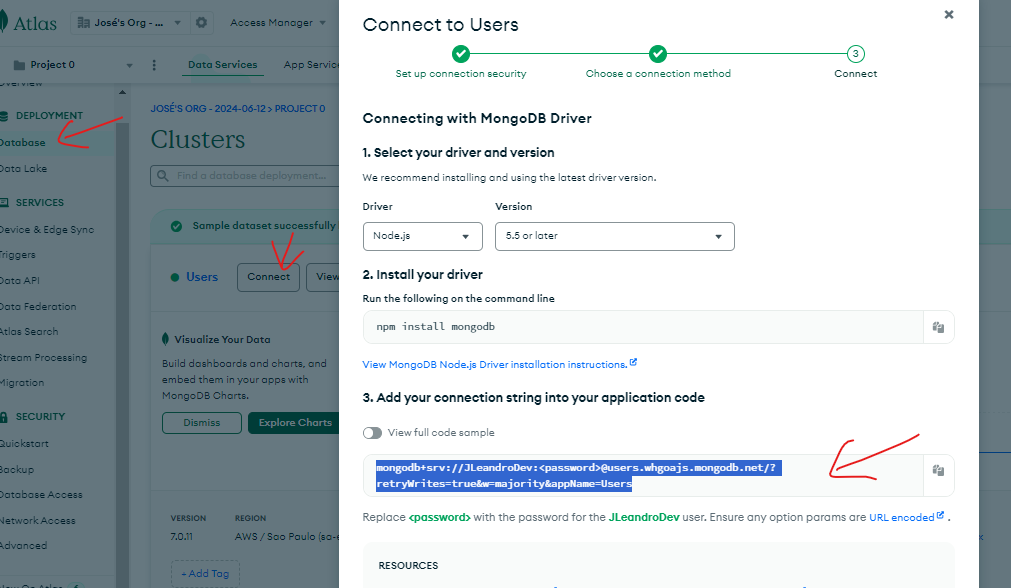




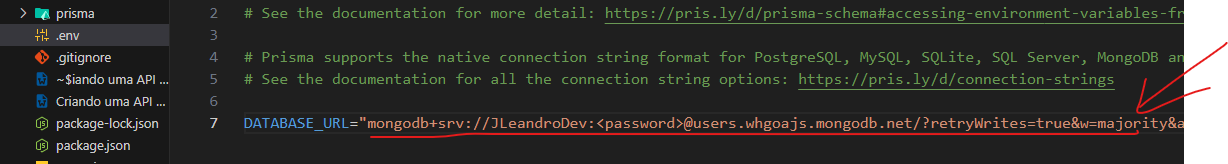
Ele vai criar alguns arquivos na aplicação:



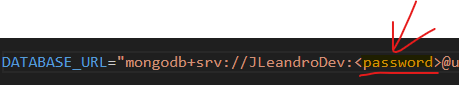


Acesso o MongoDB e copie o código mostrado no print abaixo: 

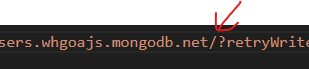
Cole este código no arquivo **env.**



Agora coloque a senha do BD que foi criada, remova todo o “**<password>”** e os sinais “**<** e **>**”

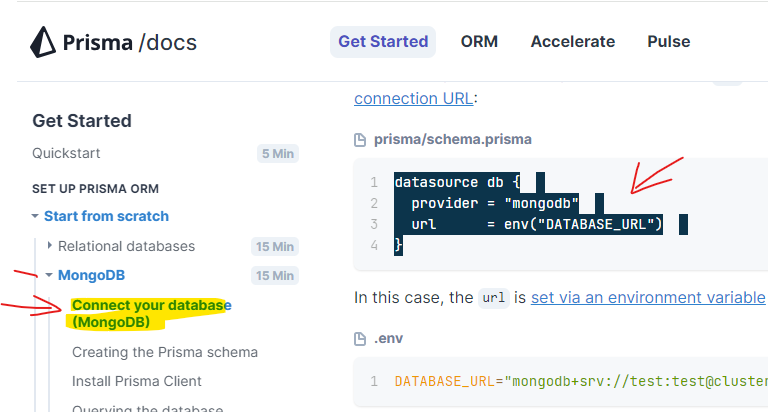


E após o barra “/” coloque o nome do Banco de Dados:

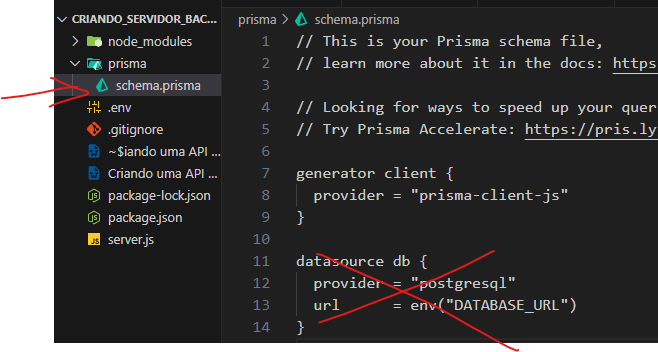




Agora volte no prisma e copie as seguintes linhas:



Agora no schema.prima, delete o código mostrado abaixo:



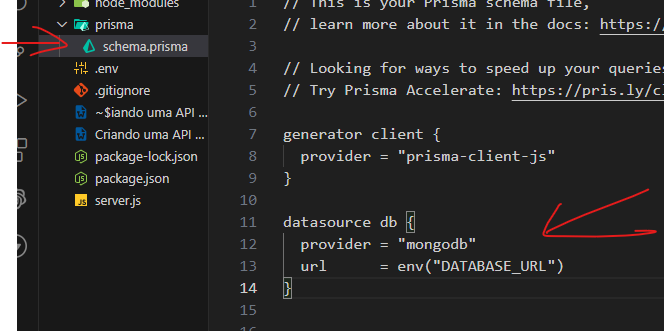
E cole este que foi copiado anteriormente

**datasource db** {

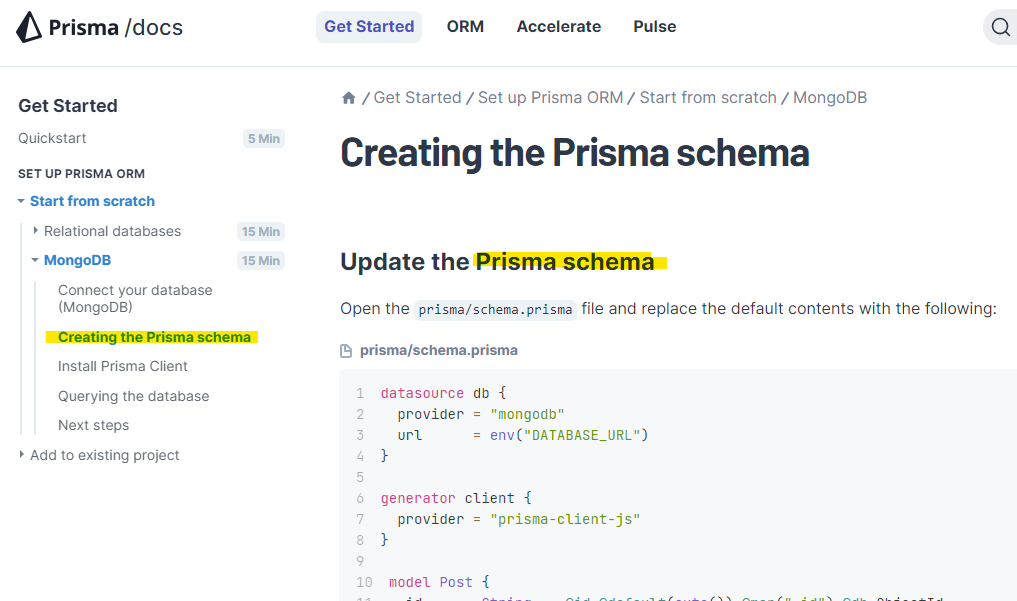
provider = "mongodb"

url = env("DATABASE\_URL")

}



Agora iremos criar o **shema**

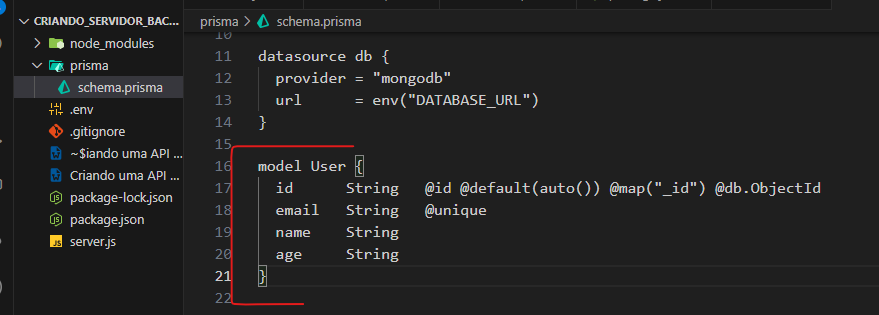


A principio copie todo código do **model user**:



Será copiado e editado conforme a necessidade no arquivo **schema.prisma**

model User {  
 id String @id @default(auto()) @map("\_id") @db.ObjectId  
 email String @unique  
 name String?  
 address Address?  
 posts Post[]  
}

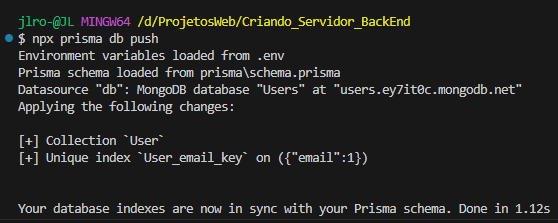


Após finalizar rode o comando

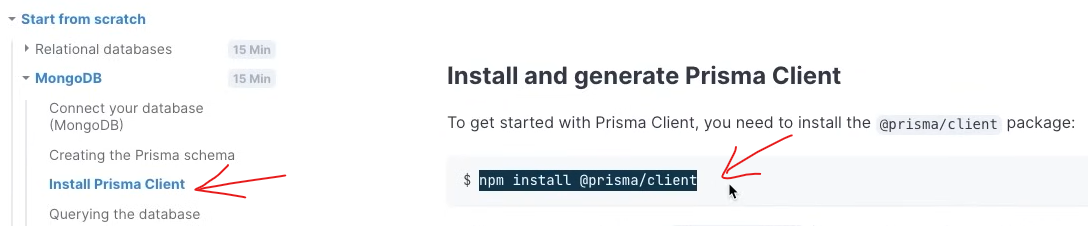




Prisma configurado com sucesso!

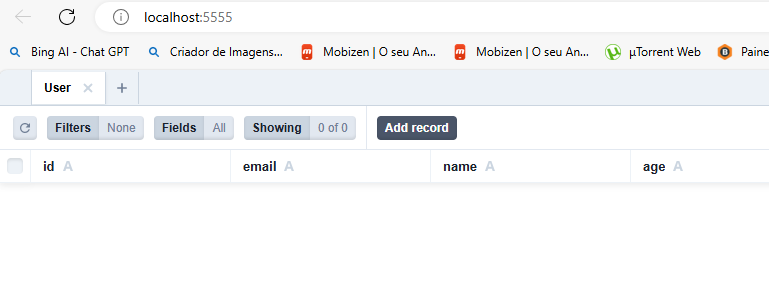


Instalar o Prisma client: npm install @prisma/client

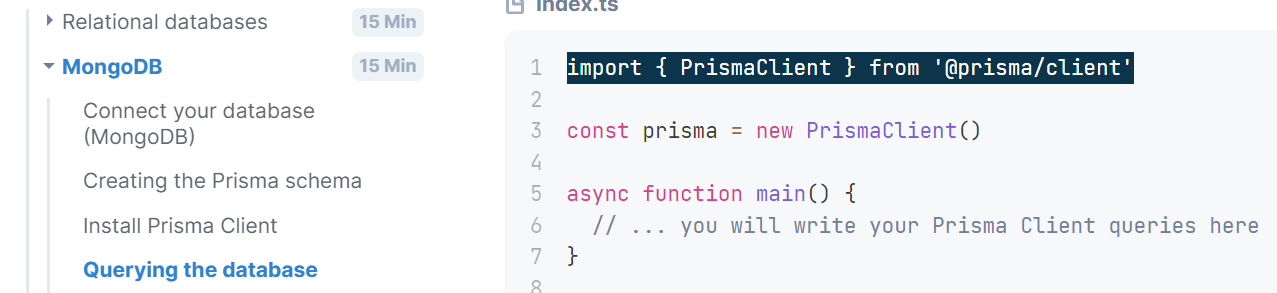


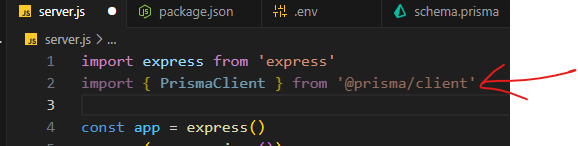
**Após instalar digite:** npx prisma studio

Vai ter acesso ao seu banco



Vamos importar o Prisma Client para o projeto:

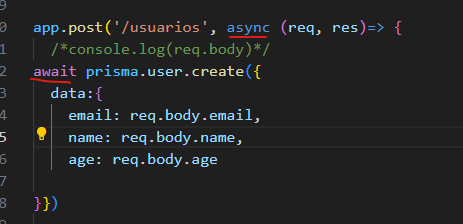




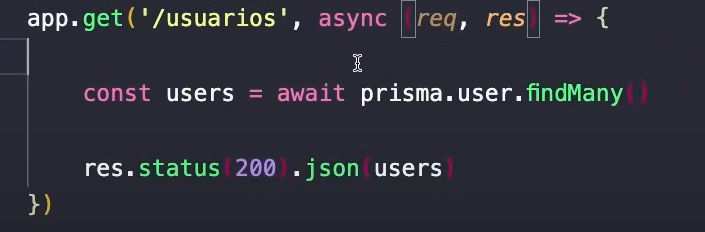
Crie a variável para importar tudo que precisamos dentro do Prisma:

l 

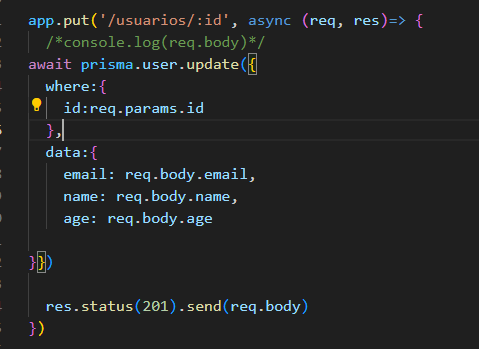
criando **Post** para cadastrar



Criando **Get**



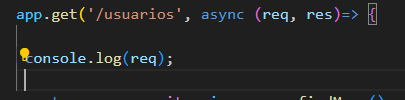
Criando **Put**

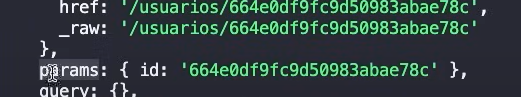


Para analisar parametros

 console.log(req);

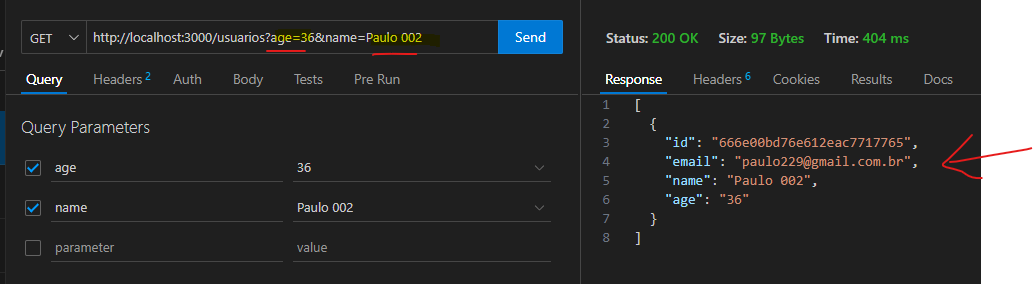
ex:





Usando **Get** com parâmetros





**Parte 2** – Front end

