# Utilizando Node.js com PostgreSQL DIO

# Instalação do PostgreSQL no Windows

# <https://www.youtube.com/watch?v=Yp2BLdDfU4A>

# SENHA. ediberto

# pgAdmin 4

# senha ediberto

# O que é e como trabalhar com PostgreSQL

# <https://www.youtube.com/watch?v=U3J3oykUzs8>

# 

# Como criar e publicar seu primeiro servidor com NodeJS | Getting Started #7

# https://www.youtube.com/watch?v=163jmMqvGmY

# BAIXAR O Nodes js site: nodes.org/em/

# 

# BAIXA E INSTALA O 16.13.2 LTS

# 

# INSTALAR O VISUAL STUDIO CODE

# Entrar no vscode

# 

# 

# VERIFICAR A VERSÃO DO NODES INSTALADO E DO NPM

# 

# Vai para a pasta onde será criado o projeto

# Instalar o npm

# 

# Instalar o expo-cli

# 

# Cria o projeto em branco

# 

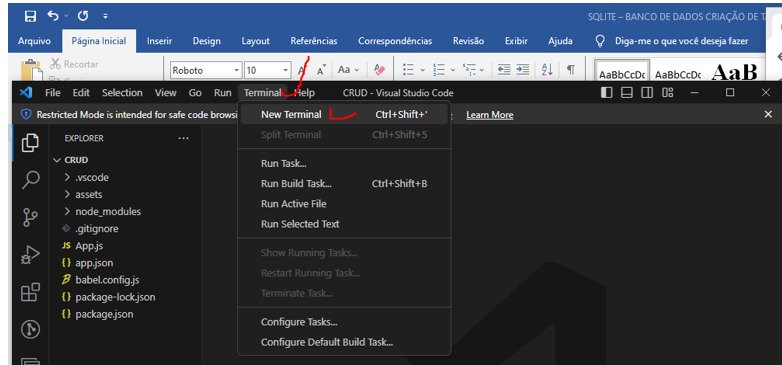
# VAI PARA A PASTA DO PROJETO CRIADA

# 

# ABRE O PROJETO NO VSCODE

# 

# ABRE O TERMINAL



**Fechar o folder do projeto criado.**

Entrar no vscode e entra no Terminal

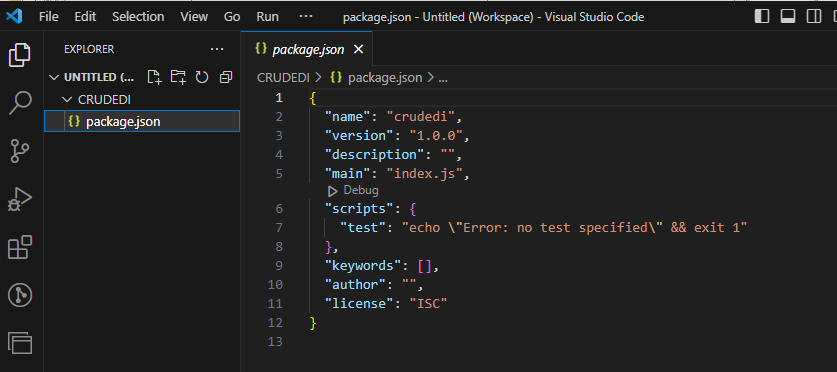
# Criar um novo folder

# 

Abre a área do Terminal

# Iniciar o novo projeto nodes

# 



**Criar o index.js**

const http = require("http");

const hostname = "localhost";

const port = 3333;

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

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

  res.end("ALÔ MUNDO!!");

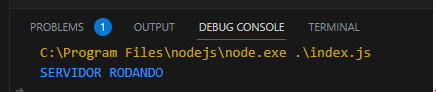
});

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

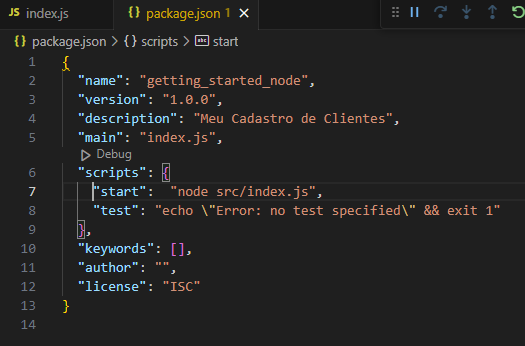
  console.log("SERVIDOR RODANDO");

});

EXECUTA O index.js – PARA INICIAR O SERVIDOR



COLOCAR O NOME DO SISTEMA NO package.json



**Package.json**

{

  "name": "getting\_started\_node",

  "version": "1.0.0",

  "description": "Meu Cadastro de Clientes",

  "main": "index.js",

  "scripts": {

    "start":  "node src/index.js",

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

  },

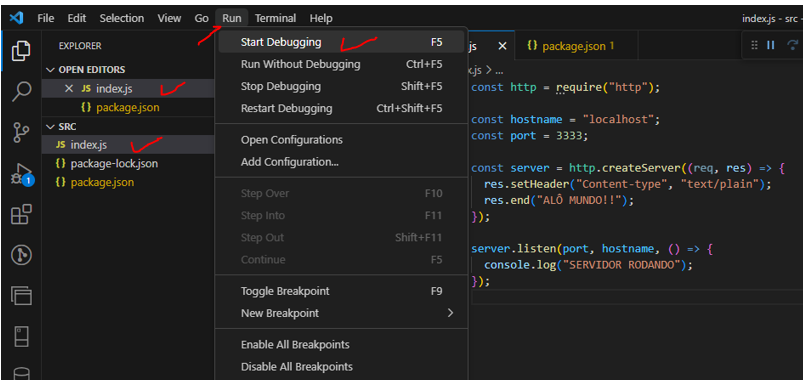
  "keywords": [],

  "author": "",

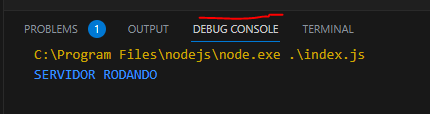
  "license": "ISC"

}

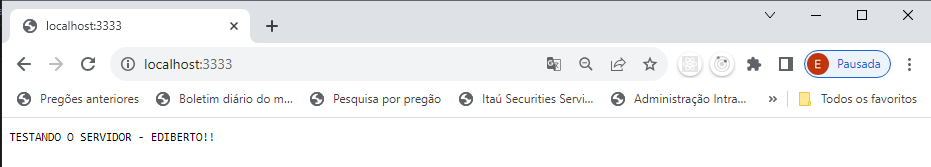
RODA



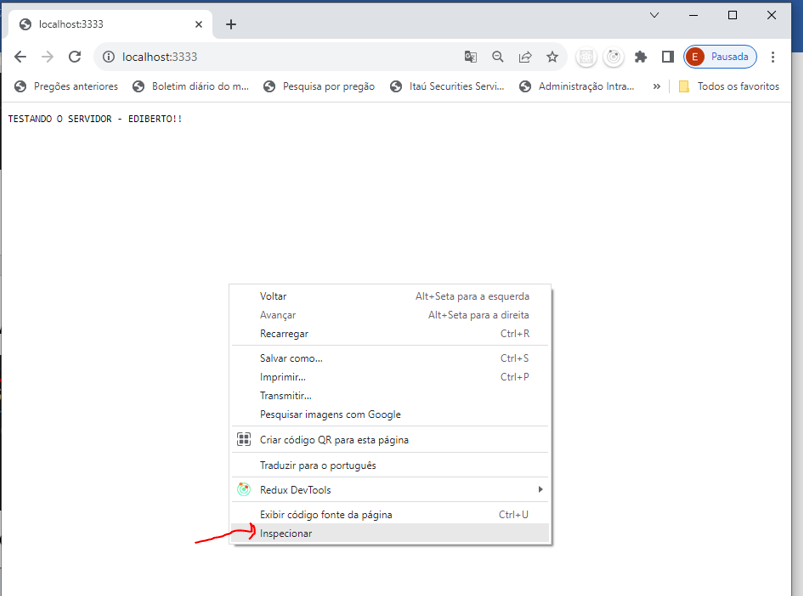
**SAÍDA**

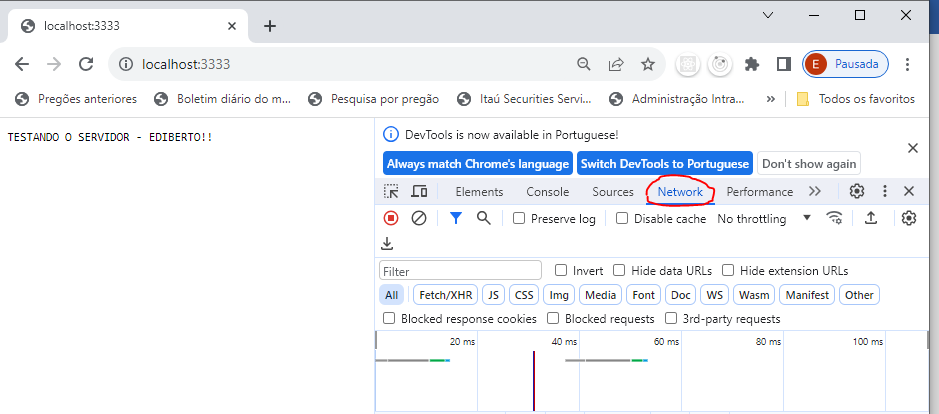


**TESTANDO O SERVIDOR NO BROWSER**

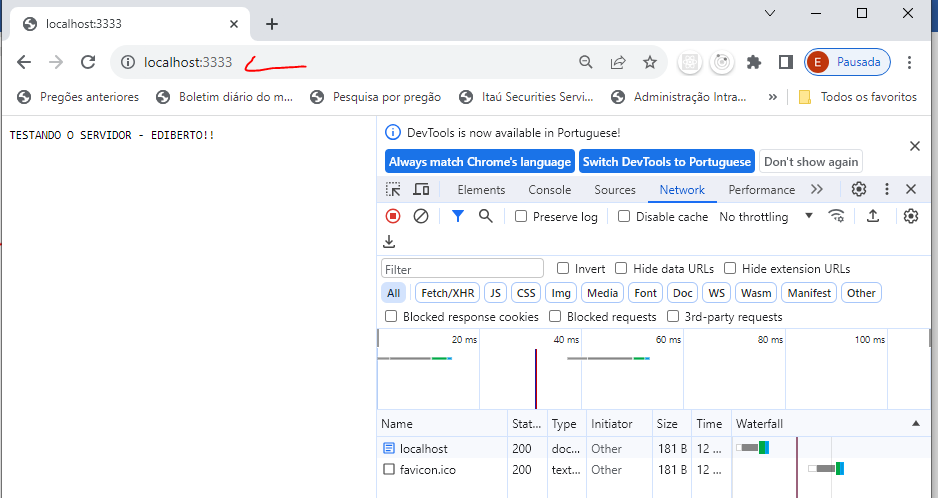


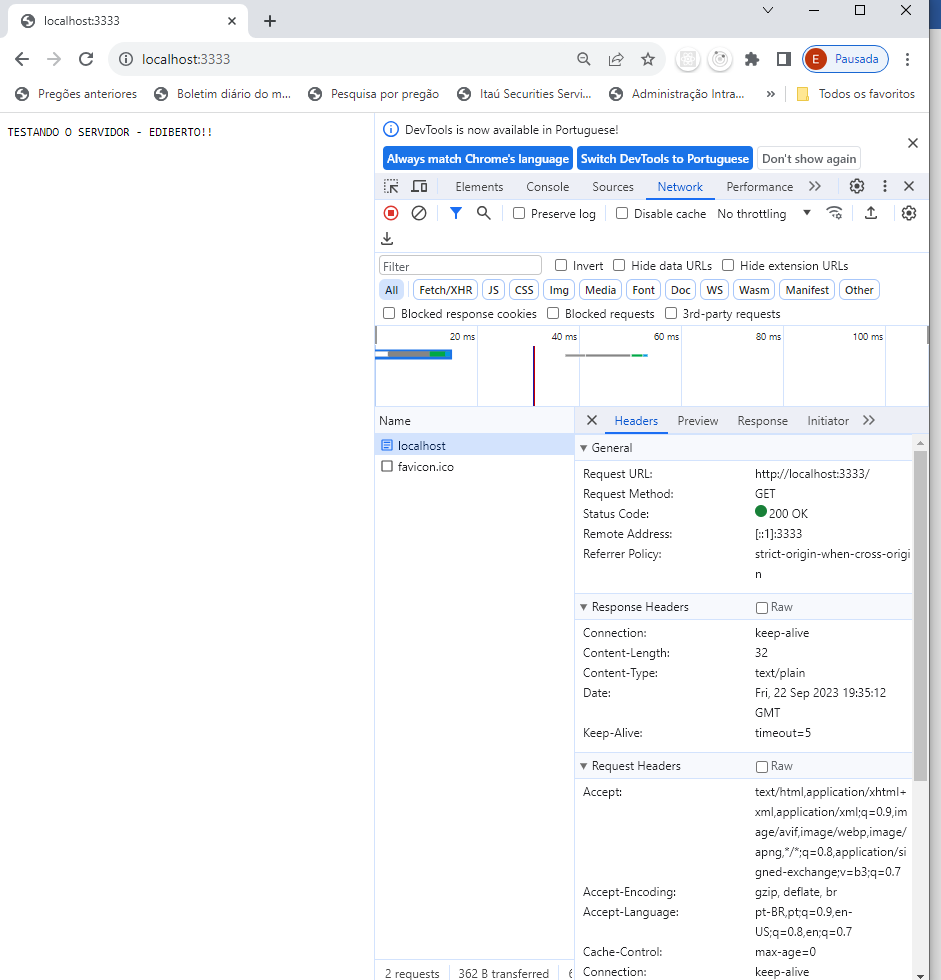
**DENTRO DO BROWSER PODEMOS VER AS REQUISIÇÕES REALIZADAS NO SERVIDOR**

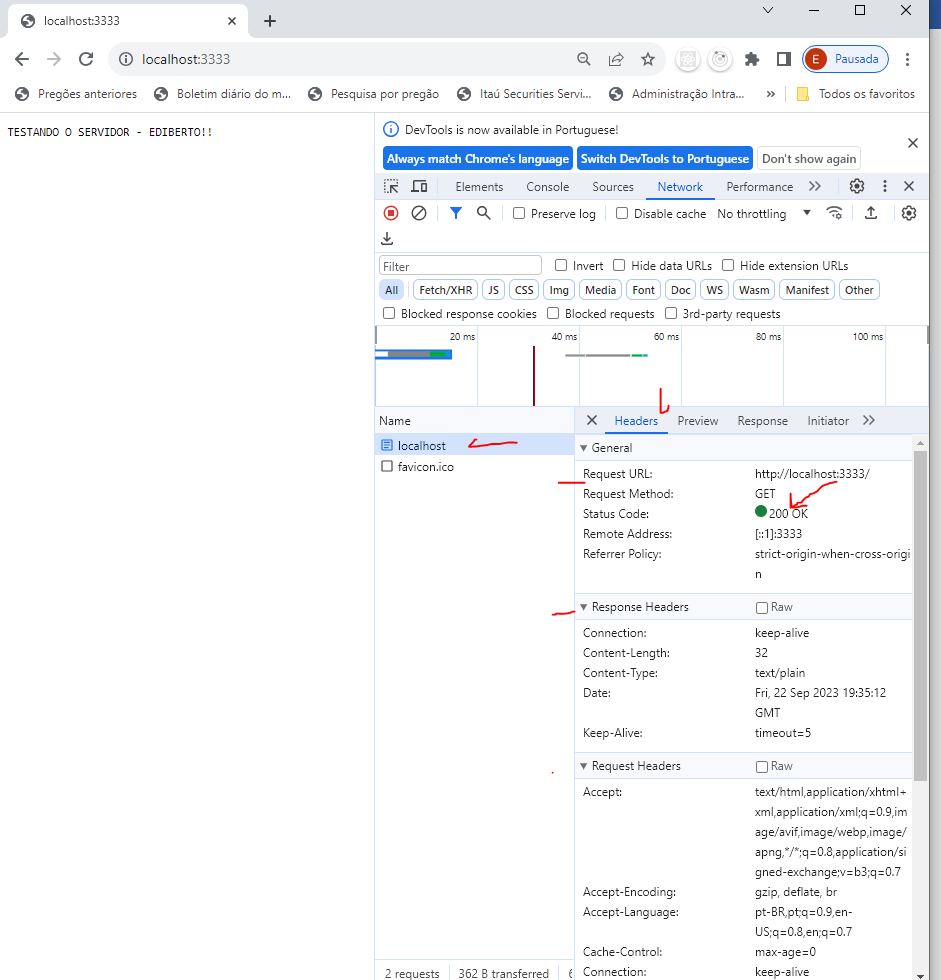




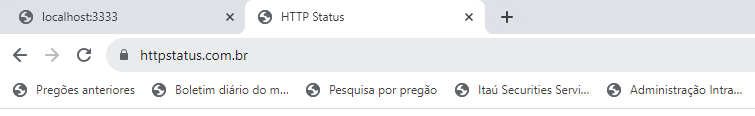
**Dar um enter na frente de 3333**

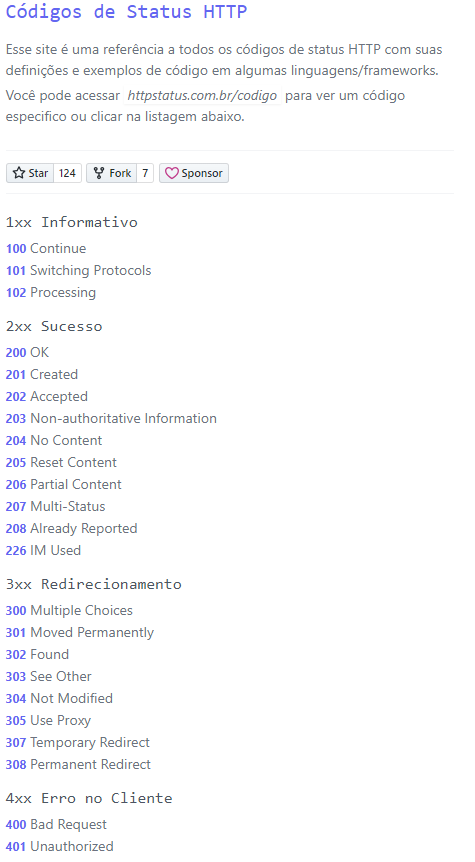


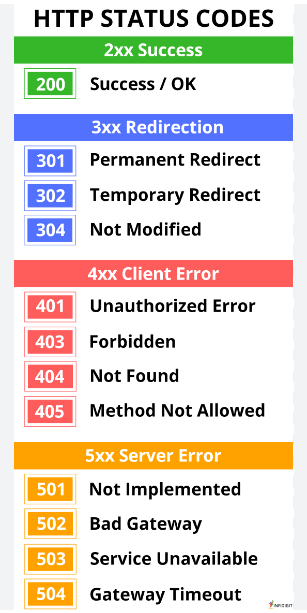


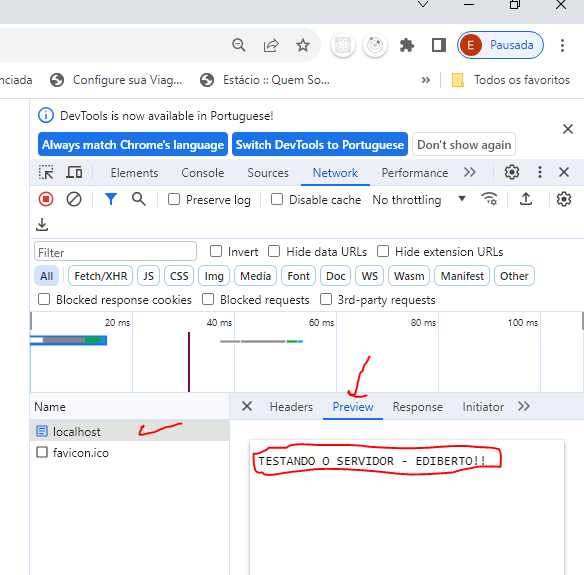


OBS. VEJA OS PADRÕES DE STATUS DE SERVIDORES









INSTALAR O EXPRESS (OBS. PAREI AQUI E VOU PARA O PRÓXIMO VIDEO)

# Utilizando Node.js com PostgreSQL DIO

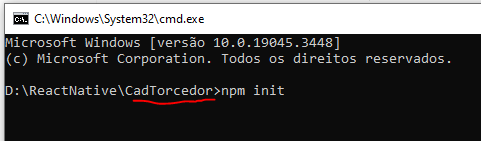
# <https://www.youtube.com/watch?v=qRhzfZKmcts>

# 

NESTE PONTO, SUPONHAMOS QUE O NOSSO BANCO JÁ FOI CRIADO.



INICIAR UM PROJETO na Pasta CadTorcedor dentro de ReactNative

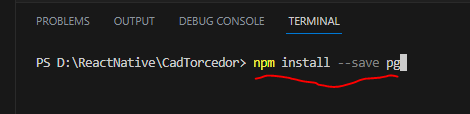


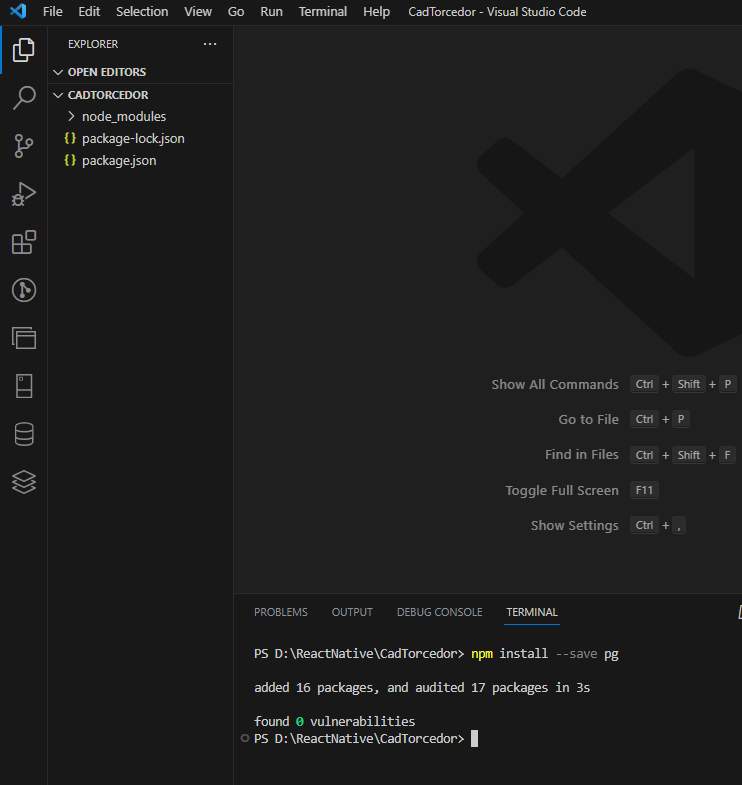
ENTER NAS OPÇÕES E yes na ultima

ABRI NO vscode

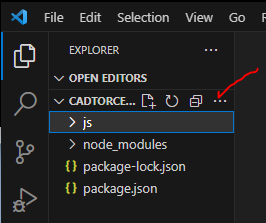


OBS. **pg** É O DRIVE PARA O NODES CONECTAR COM O POSTGRESQL

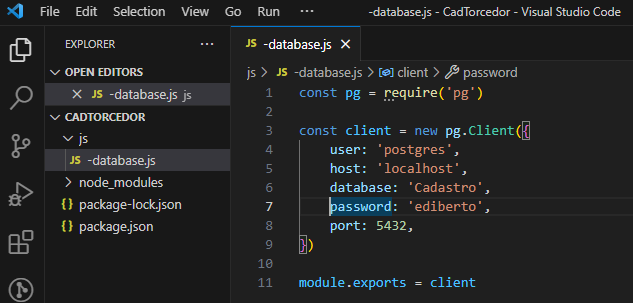




CRIAR A PASTA js



DENTRO DA PASTA **js**: CRIAR O ARQUIVO **-database.js**



**\_database.js**

const pg = require('pg')

const client = new pg.Client({

    user: 'postgres',

    host: 'localhost',

    database: 'Cadastro',

    password: 'ediberto',

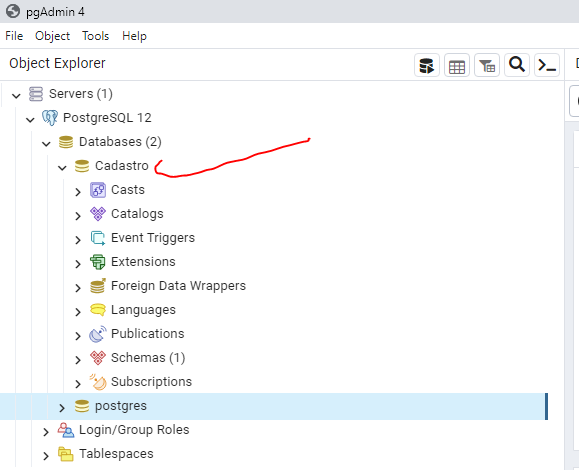
    port: 5432,

})

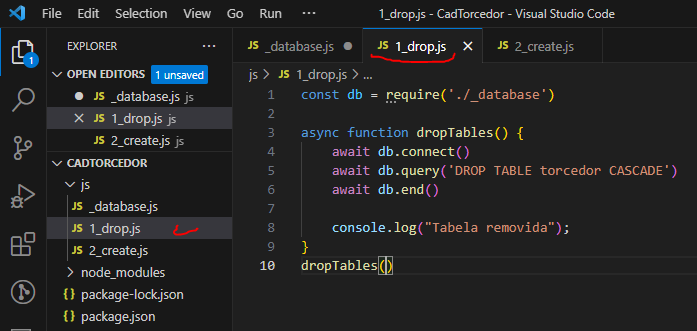
module.exports = cliente

ABRIR O BANCO DE DADOS NO pgAdmim 4

Obs. **Veja que o Banco de dados Cadastro foi criado**



DENTRO DA PASTA **js**: CRIAR O ARQUIVO **1\_drop.js. neste caso será removida a tabela torcedor, caso exista.**



Código do arquivo 1\_drop.js

const db = require('./\_database')

async function dropTables() {

    await db.connect()

    await db.query('DROP TABLE torcedor CASCADE')

    await db.end()

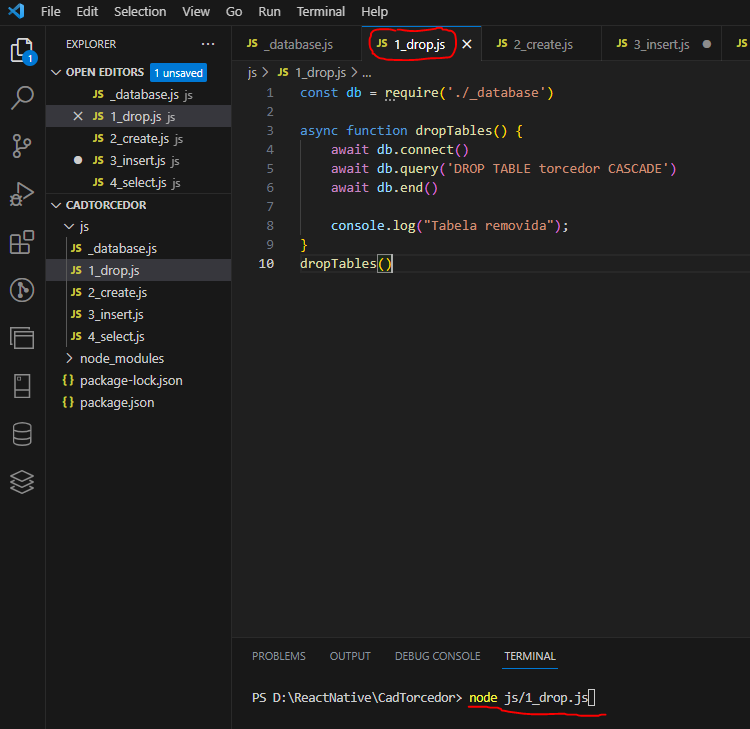
    console.log("Tabela removida");

}

dropTables()

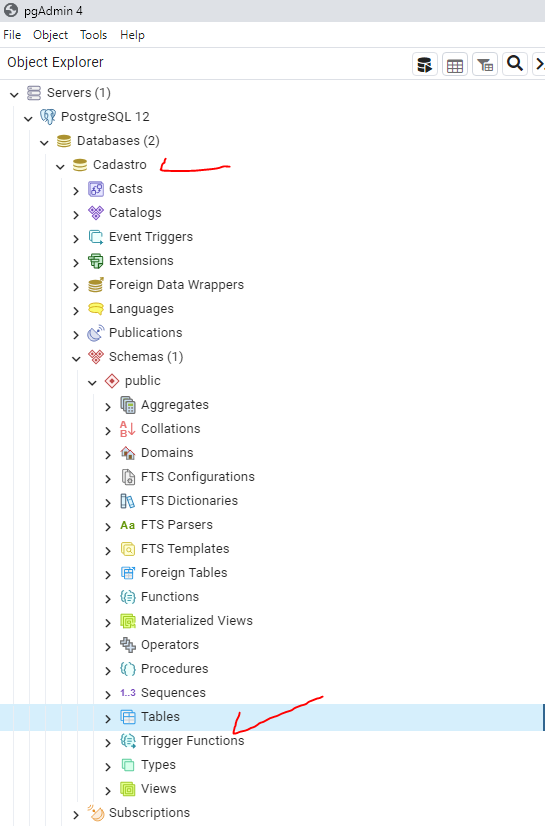
RODAR o arquivo para excluir tabela

**node js/1\_drop.js**

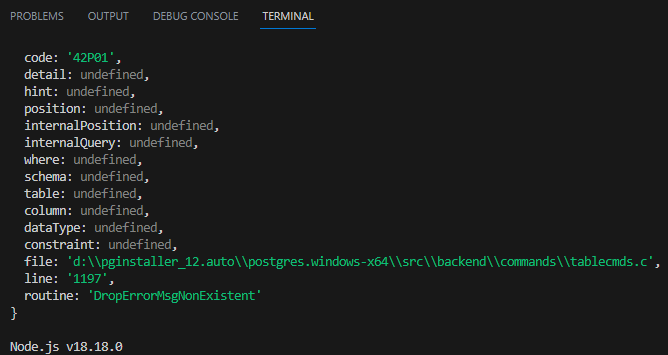


ABRIR O BANCO DE DADOS NO pgAdmim 4

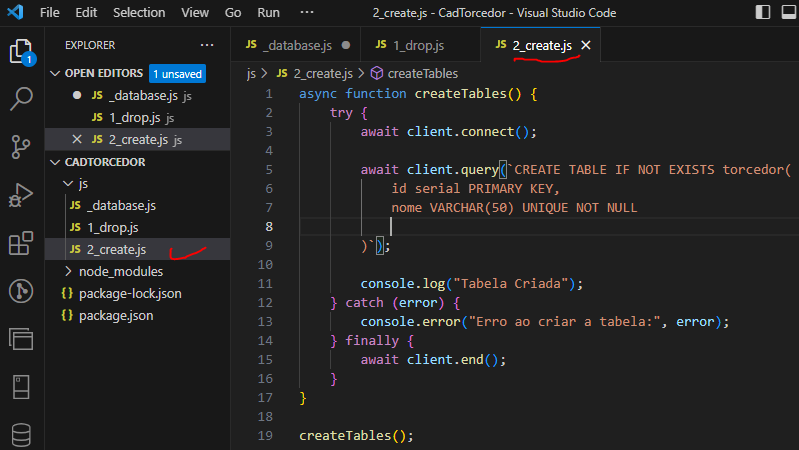
**OBS. VERIFIQUE QUE A TABELA torcedor FOI REMOVIDA**



**Obs. Caso a tabela fornecedor não exista, será exibida a msg abaixo.**



DENTRO DA PASTA **js**: CRIAR O ARQUIVO **2\_create.js - SERÁ CRIADA A TABELA torcedor, CASO NÃO EXISTA NO BANCO DE DADOS Cadastro**



**CRIAÇÃO da Tabela torcedor no banco Cadastro, caso não exista.**

**Código do arquivo 2\_create.js**

const db = require('./\_database');

async function createTables() {

await db.connect();

await db.query(`CREATE TABLE IF NOT EXISTS torcedor(

id serial PRIMARY KEY,

nome VARCHAR(50) UNIQUE NOT NULL

)`);

await db.end();

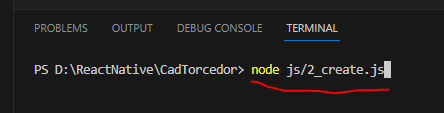
console.log("Tabela Criada");

}

createTables();

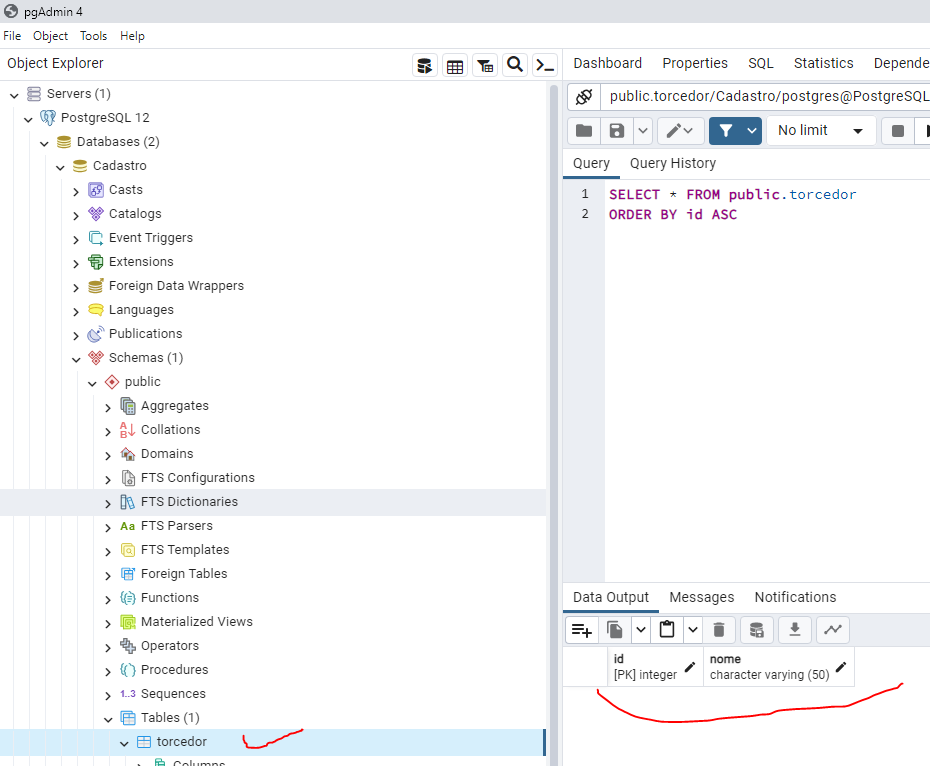
RODAR o arquivo para criar a tabela torcedor no banco

**node js/2\_create.js**



ABRIR O BANCO DE DADOS NO pgAdmim 4

**OBS. VERIFIQUE QUE A TABELA torcedor FOI CRIADA**



DENTRO DA PASTA **js**: CRIAR O ARQUIVO **3\_insert.js - SERÃO INSERIDOS DADOS NA TABELA torcedor.**

**3\_insert.js**

const db = require('./\_database');

async function insertData() {

    await db.connect();

    const queryTorcedor = "INSERT INTO torcedor (nome) VALUES ($1)";

    try {

        await db.query(queryTorcedor, ['Ediberto Silva']);

        await db.query(queryTorcedor, ['Marta Costa']);

        console.log('Dados inseridos');

    } catch (error) {

        console.error('Erro ao inserir dados:', error);

    } finally {

        await db.end();

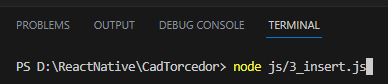
    }

}

insertData();

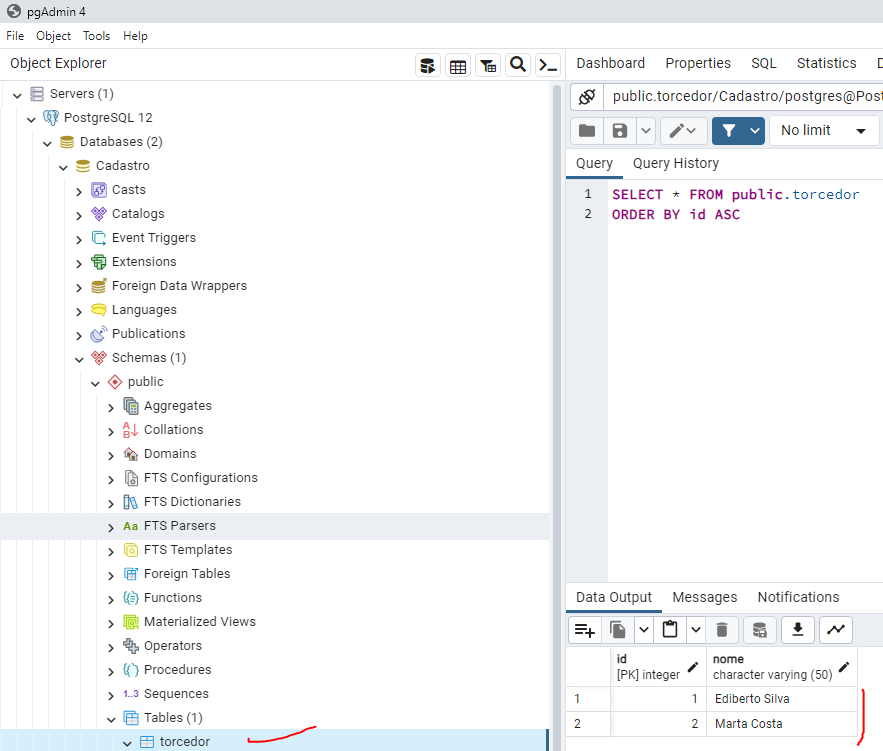
RODAR o arquivo para criar a tabela torcedor no banco

**node js/3\_insert.js**



ABRIR O BANCO DE DADOS NO pgAdmim 4

**OBS. VERIFIQUE QUE OS DOIS REGISTROS FORAM GRAVADOS NA TABELA torcedor**



DENTRO DA PASTA **js**: CRIAR O ARQUIVO **4\_select.js - SERÃO LISTADOS TODOS OS REGISTROS EXISTENTES NA TABELA torcedor.**

**4\_insert.js**

const db = require('./\_database')

async function listData(){

    await db.connect()

    var result

    result = await db.query("SELECT \* FROM torcedor")

    console.log("TORCEDORES")

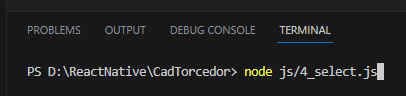
    console.log(result.rows);

}

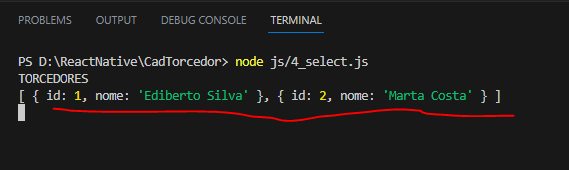
listData()

RODAR o arquivo para listar os registros da tabela torcedor no banco

**node js/4\_select.js**



**RESULTADO. SERÃO EXIBIDOS OS DADOS EM FORMATO DE ARRAY**



OBS. EXEMPLO DE SALVAR REGISTRO POR API

https://www.youtube.com/watch?v=x6ibKvE0GbY&list=PLAxN8g6Knm0camfON299B-vl31IYQhA8Q&index=7