

**INICIANDO A APLICAÇÃO**

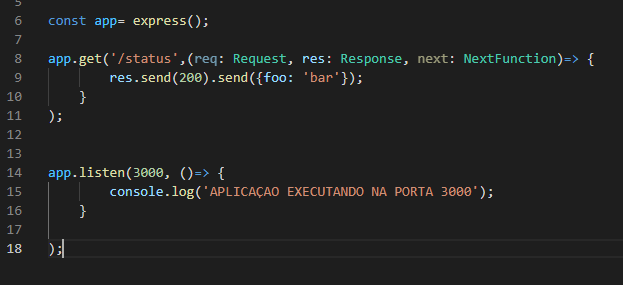




Express = biblioteca de rotas http

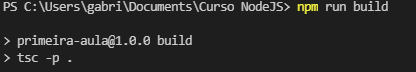
PARA APLICAÇÃO



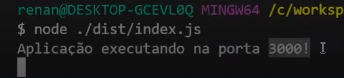


**EXECUÇÃO**

**Iniciando a aplicação**





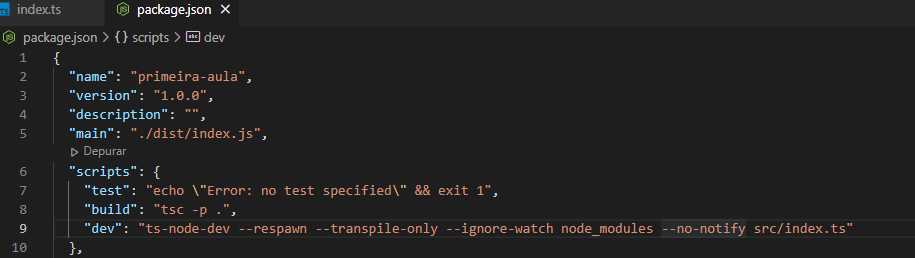
[Express](https://expressjs.com/) é o framework Node mais popular e a biblioteca subjacente para uma série de outros frameworks do Node. O Express oferece soluções para:

* Gerenciar requisições de diferentes verbos HTTP em diferentes URLs.
* Integrar "view engines" para inserir dados nos templates.
* Definir as configurações comuns da aplicação web, como a porta a ser usada para conexão e a localização dos modelos que são usados para renderizar a resposta.
* Adicionar novos processos de requisição por meio de "middleware" em qualquer ponto da "fila" de requisições.

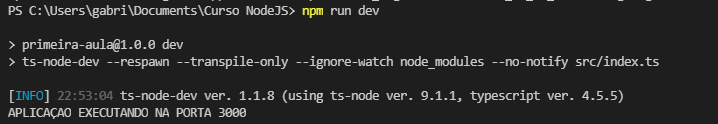
O *Express*é bastante minimalista, no entanto, os desenvolvedores têm liberdade para criar pacotes de middleware específicos com o objetivo de resolver problemas específicos que surgem no desenvolvimento de uma aplicação. Há bibliotecas para trabalhar com cookies, sessões, login de usuários, parâmetros de URL, dados em requisições POST, cabeçalho de segurança e tantos outros. Você pode achar uma lista de pacotes de middleware mantidos pela equipe Express em [Express Middleware](http://expressjs.com/en/resources/middleware.html) (juntamente com uma lista de pacotes populares desenvolvidos por terceiros).

**MELHORAR A EXECUÇÃO**







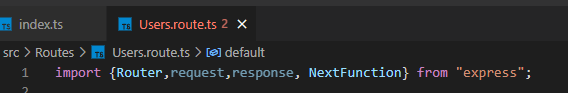




CRIANDO AS ROTAS

****



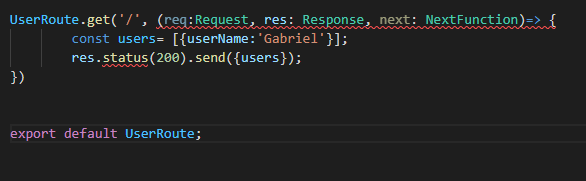
****



Importamos, o Router, request, response, nextfunction

****

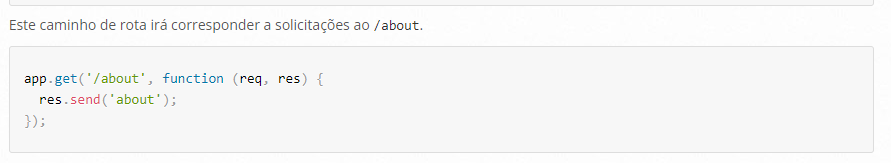
Definimos a Rotar com UserRoute = Router()

****

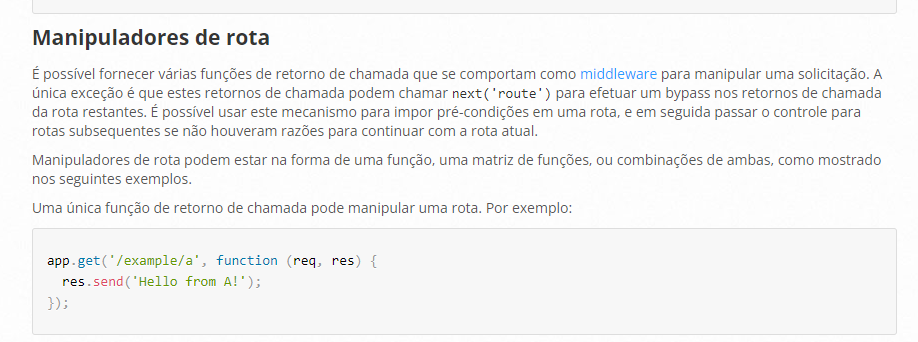
Aplicamos o método GET através da sintave UserRoute.get(

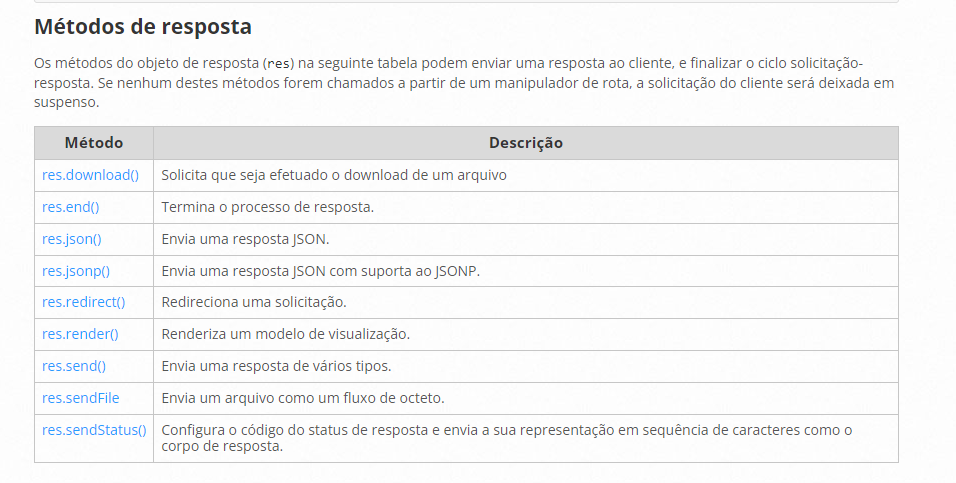
( (req:Request, res: Response, next: NextFunction)=> )













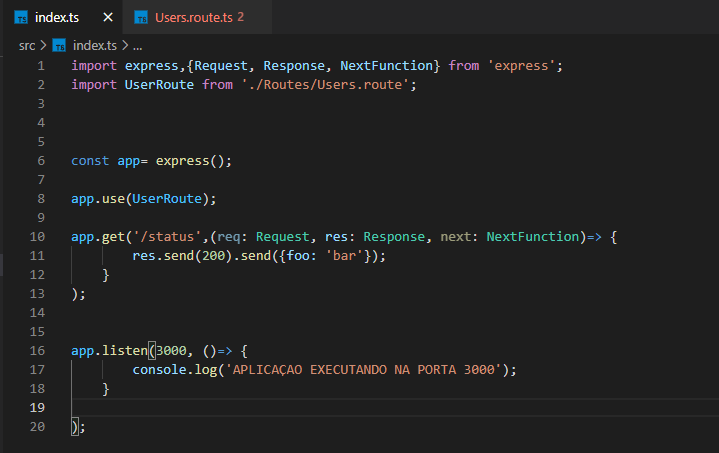
Arrow function com método de cada atributo

const users= [{userName:'Gabriel'}];

        res.status(200).send({users});

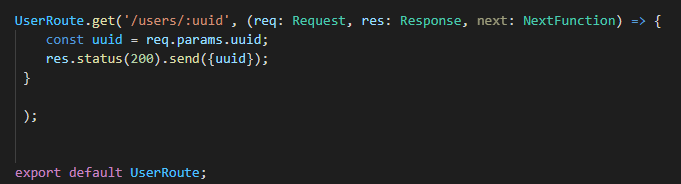
send para enviar users

e depois exportamos export default UserRoute;





**Definindo outra rota**



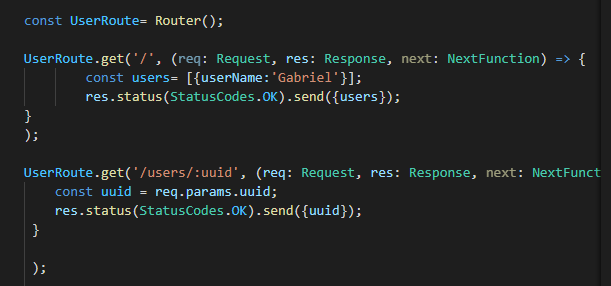




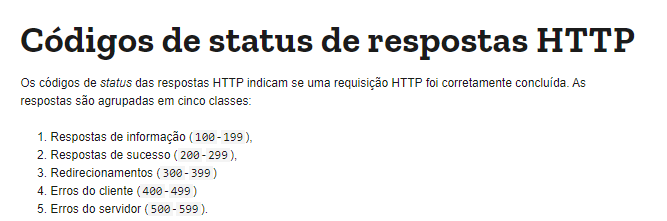


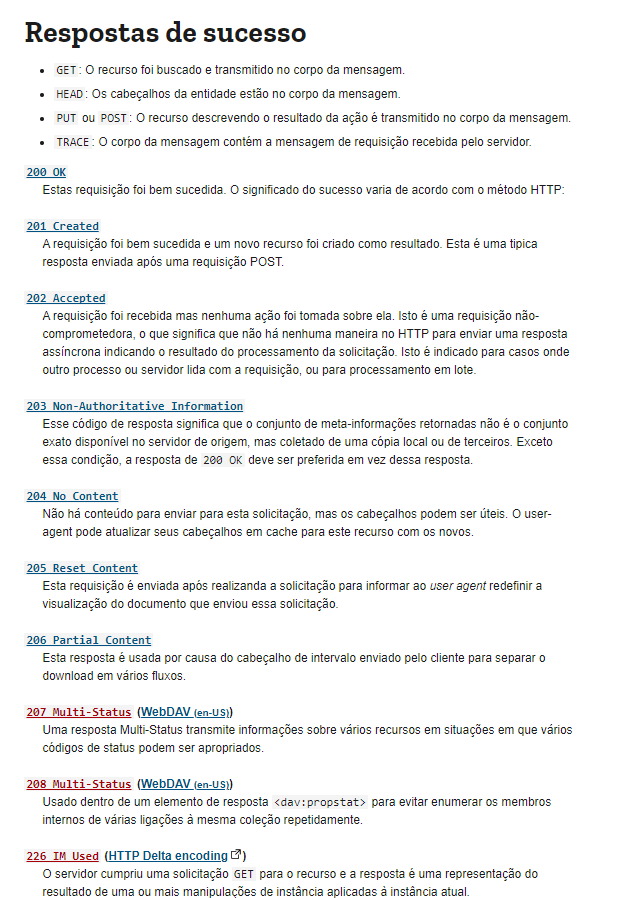






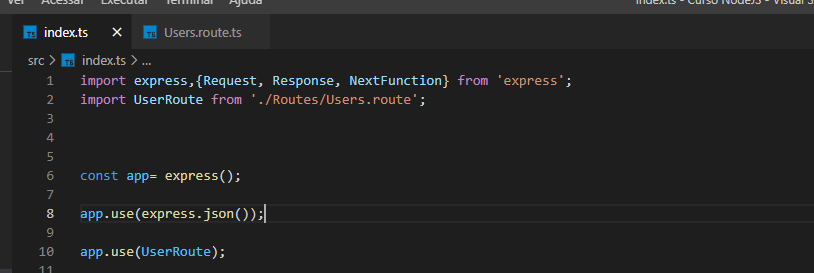






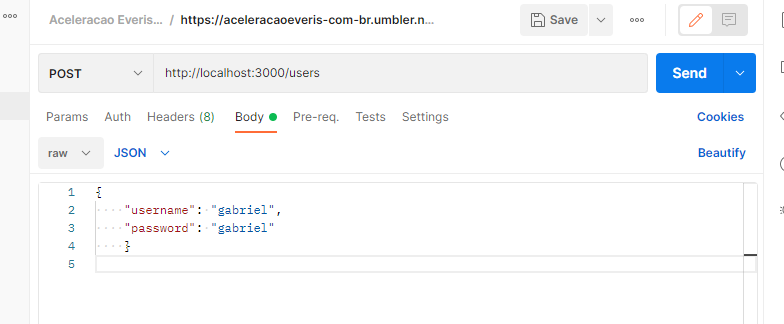
[Códigos de status de respostas HTTP - HTTP | MDN (mozilla.org)](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status)

**DEFININDO A ROTA POST**

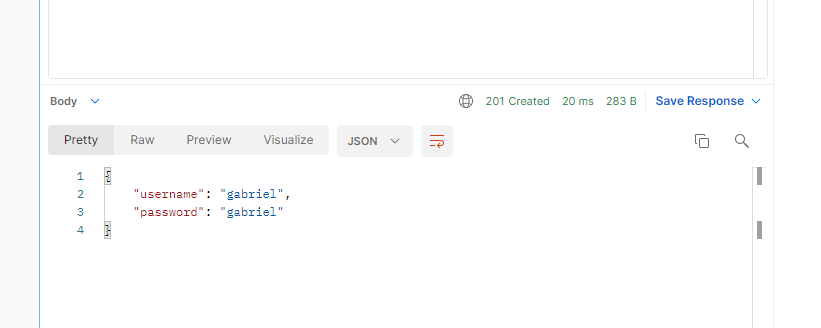


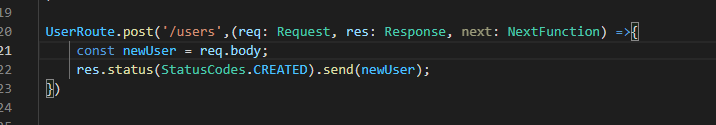


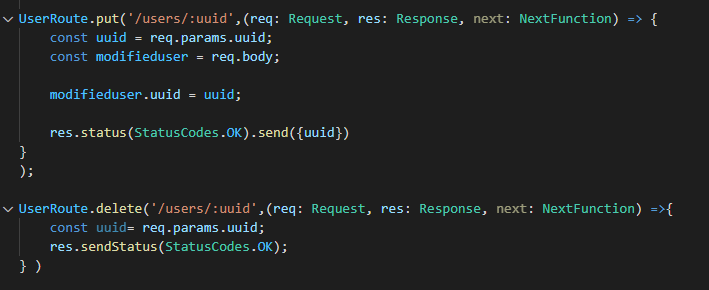
Middleware responsável por interpretar o contente type e utilizar o json como content type aceito

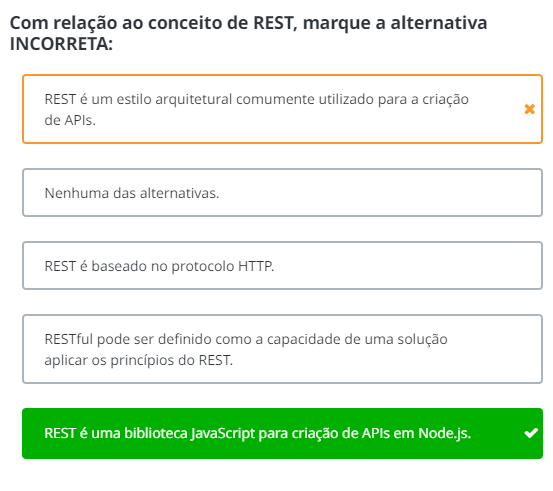






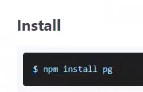






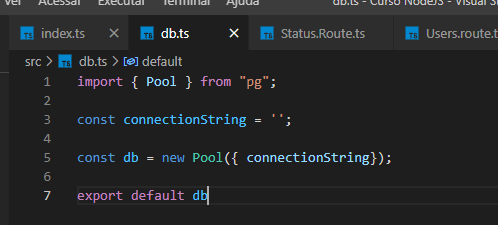


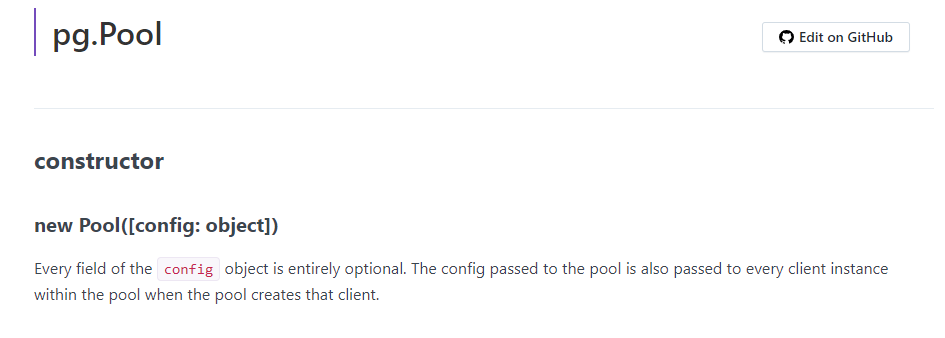


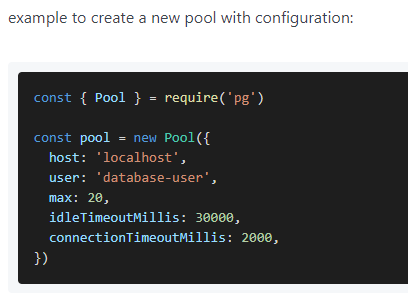




**Configurando o banco de dados DEFININDO A POOL**



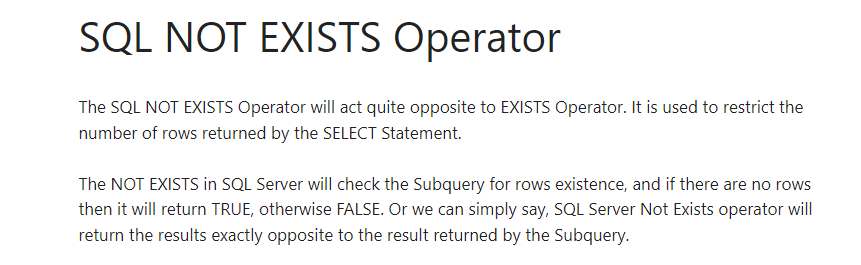




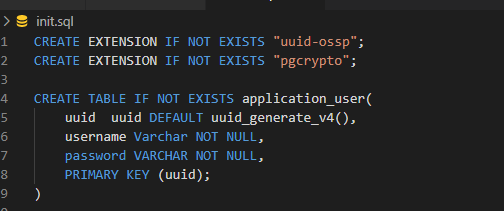




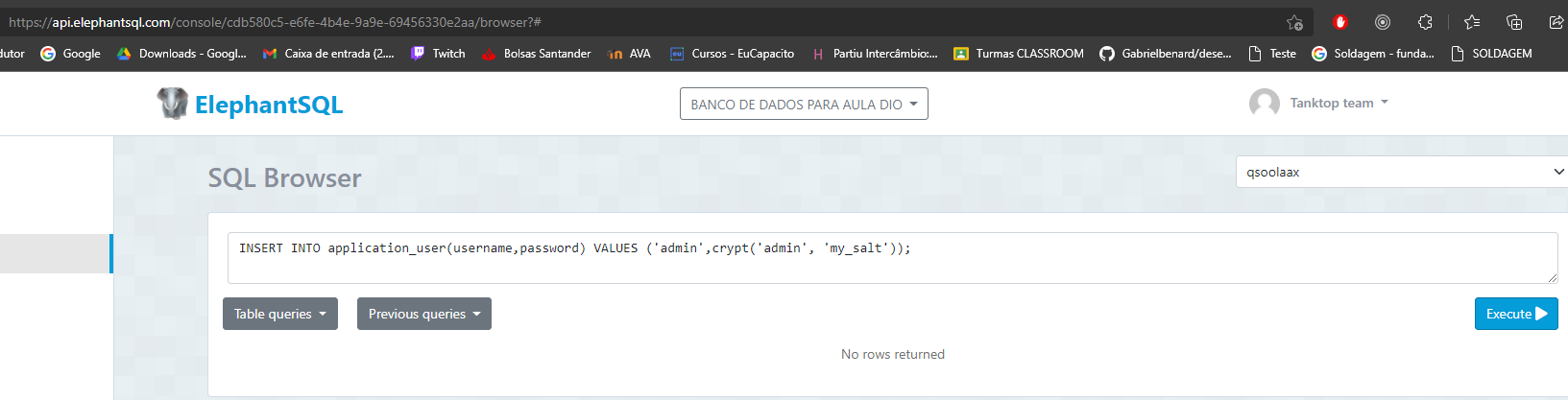
**CRIANDO NOSSAS TABELAS E INSERINDO DADOS**

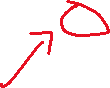




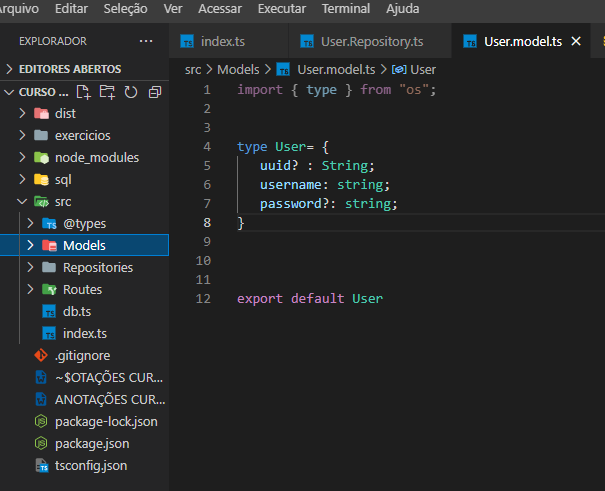






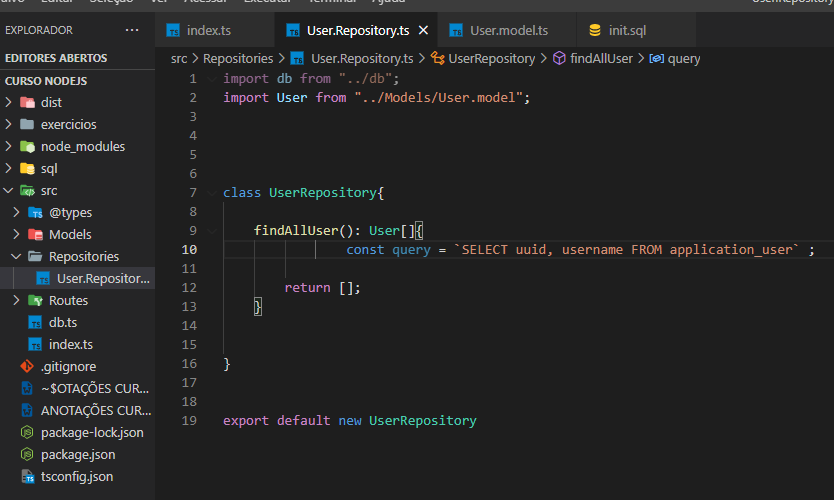


Criando modelos de TABELA

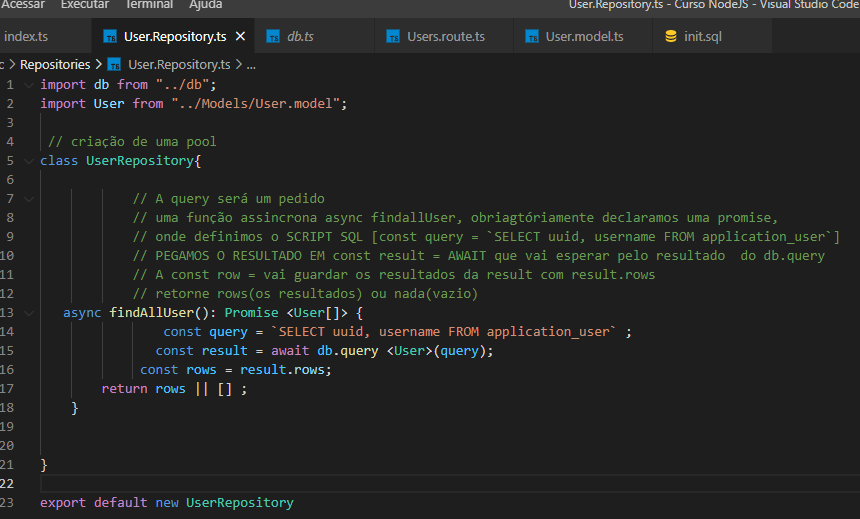




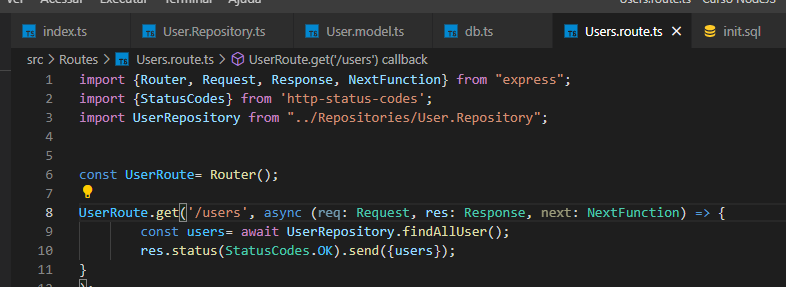
REALIZANDO A CONSULTA DE USUÁRIOS





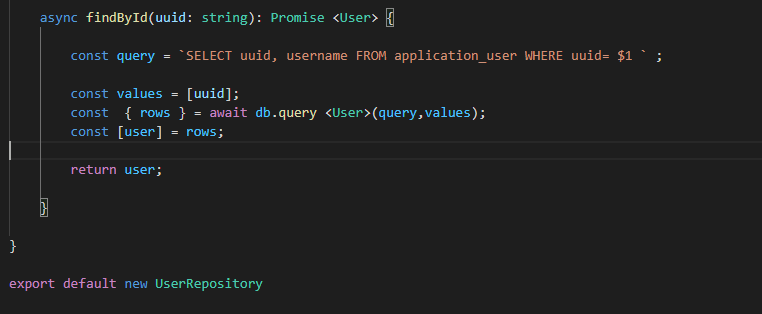




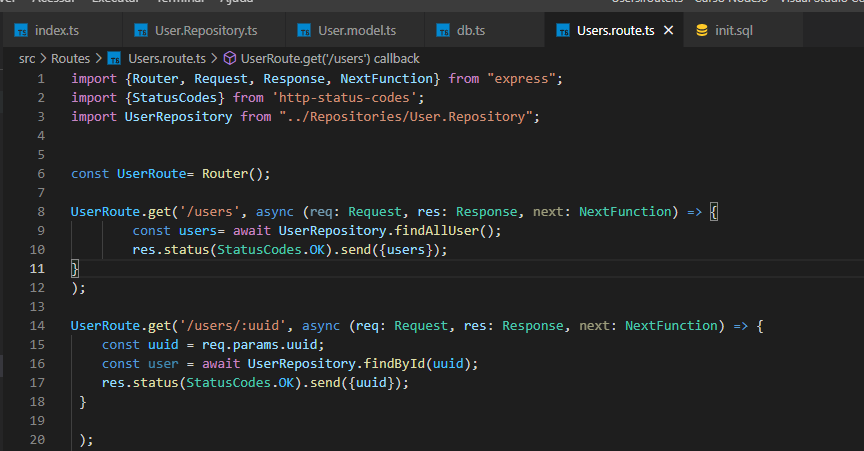




DESENVOLVENDO CONSULTA DE USUÁRIOS POR ID

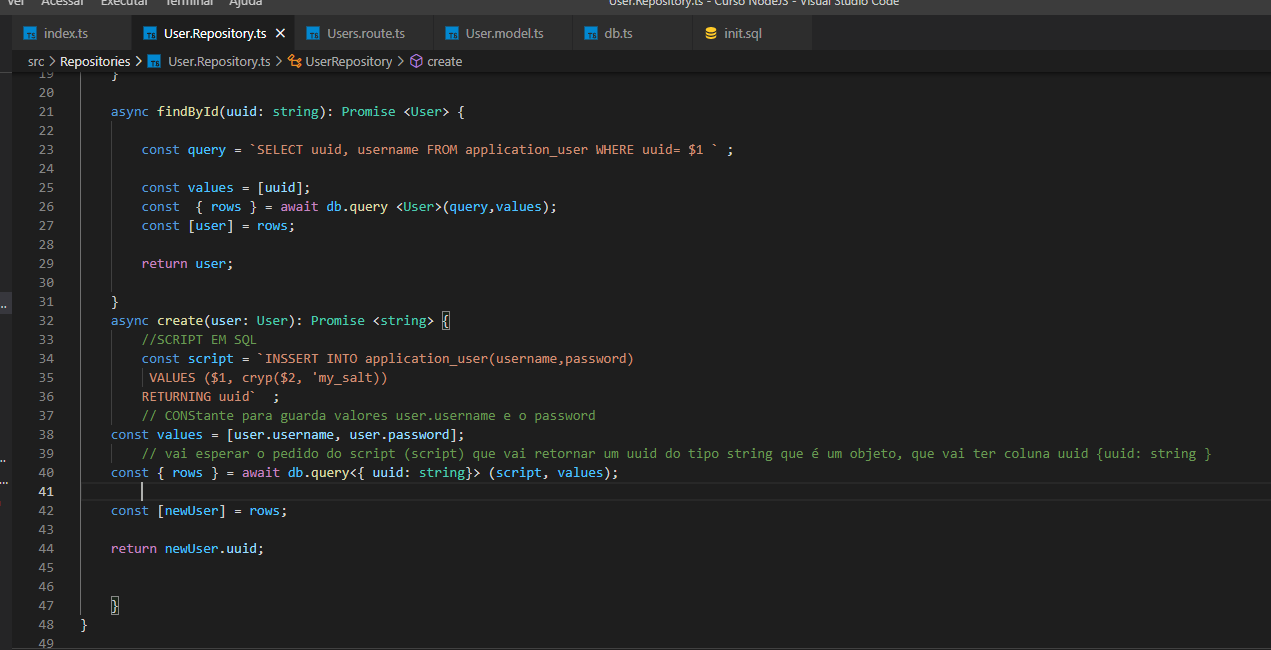




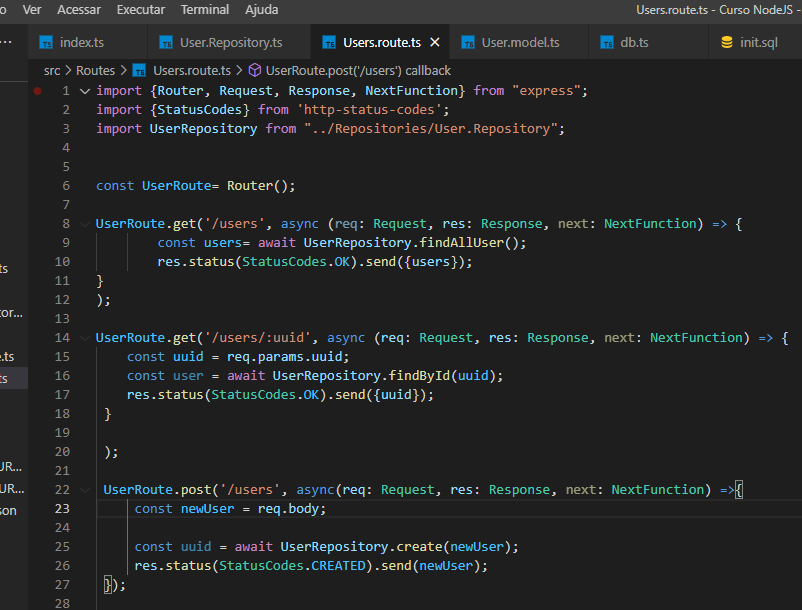


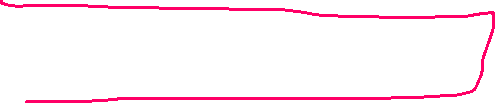


INSERÇÃO DE USUÁRIOS NO BANCO DE DADOS

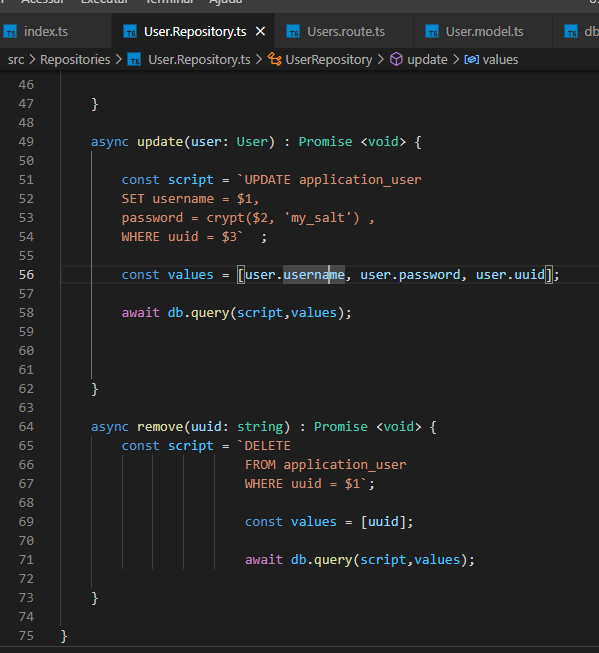




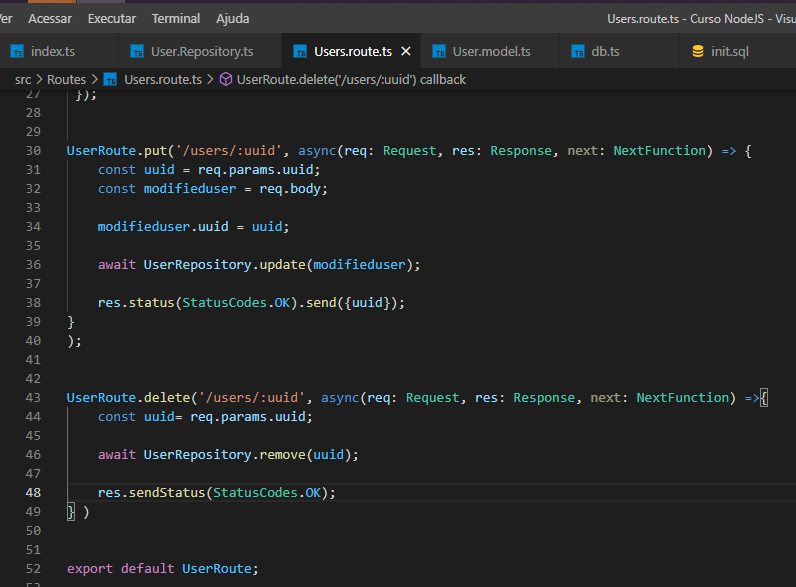




UPTADE E REMOVENDO USUÁRIOS

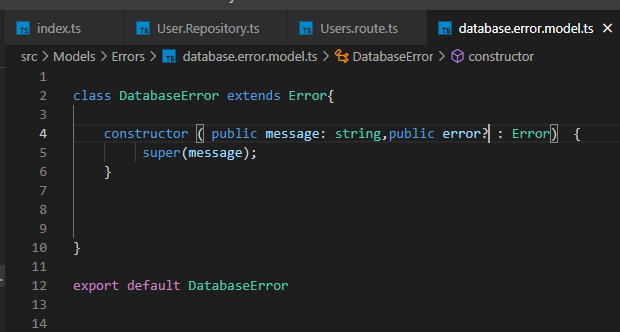






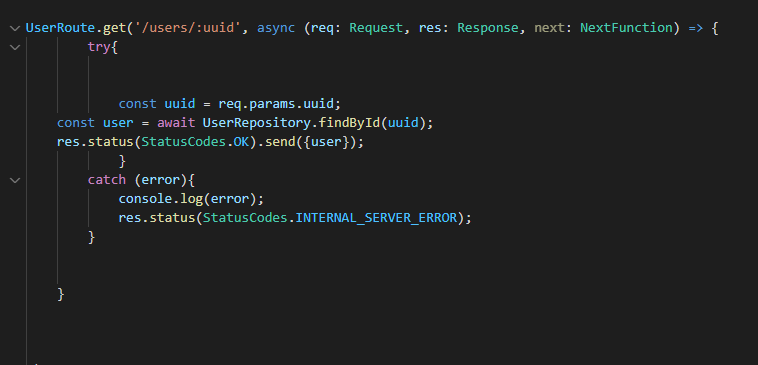


Tratamento de erros



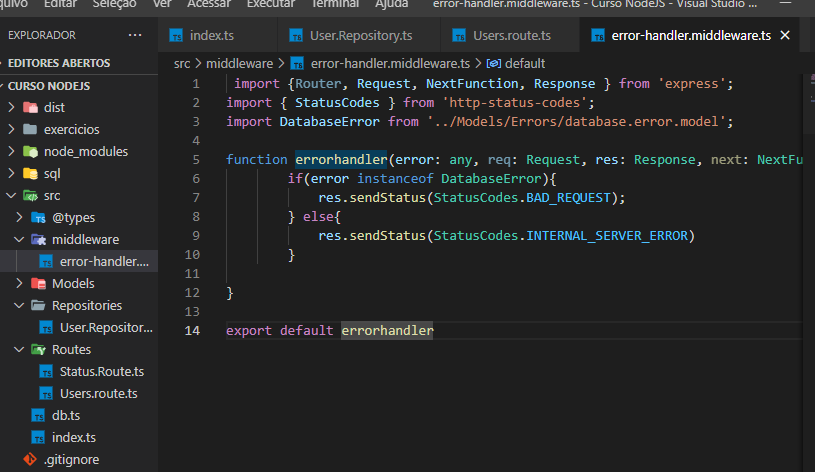




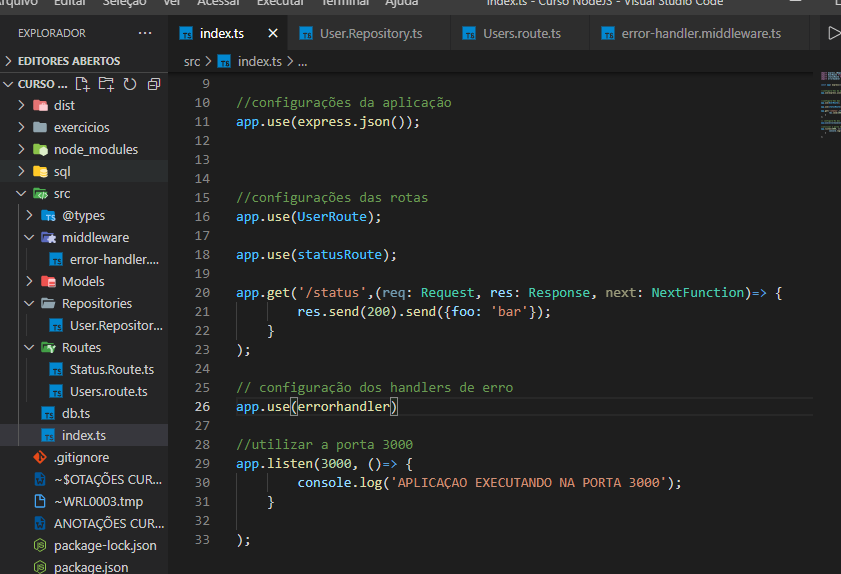




DESENVOLVENDO ERROR HANDLING







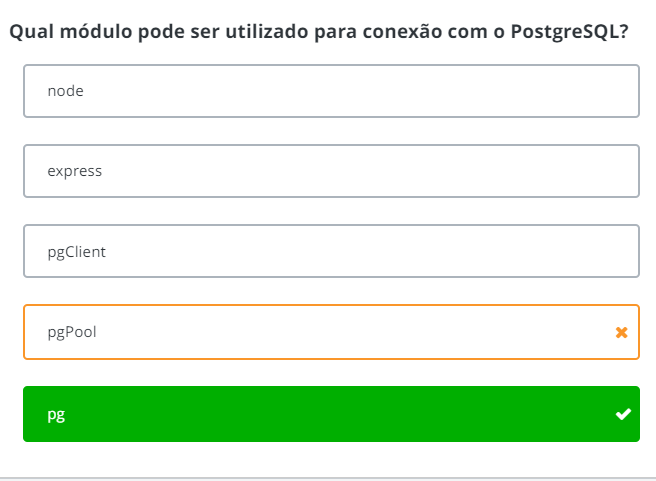




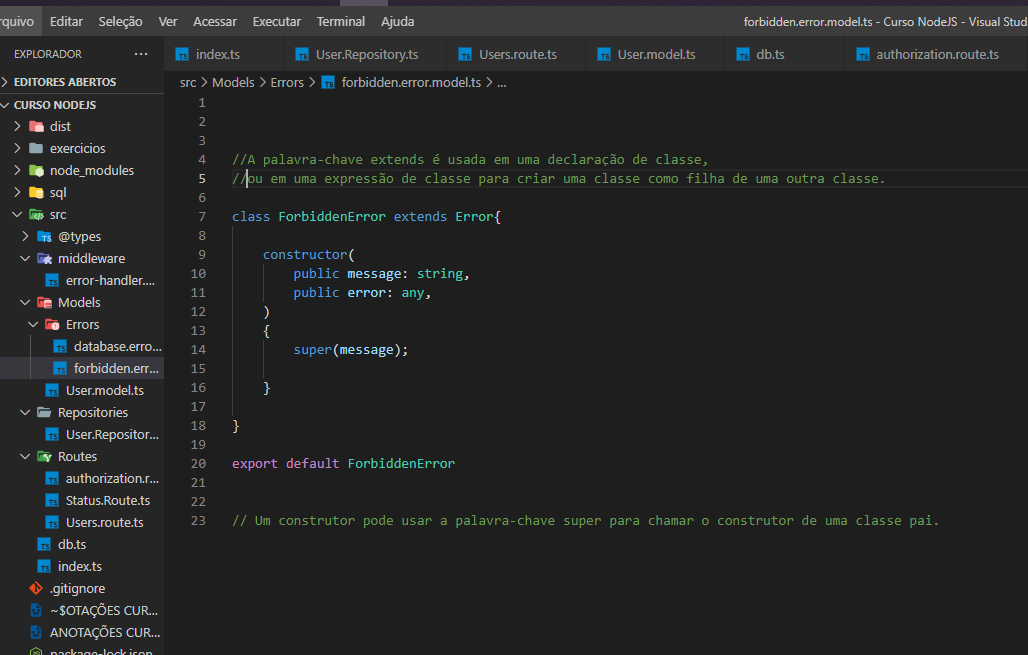




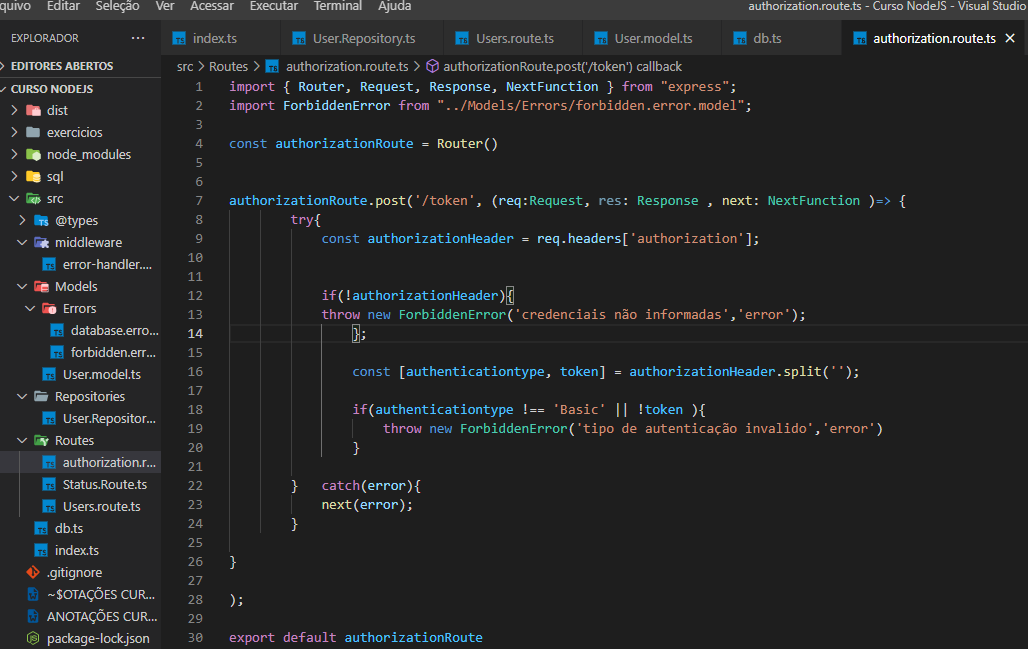




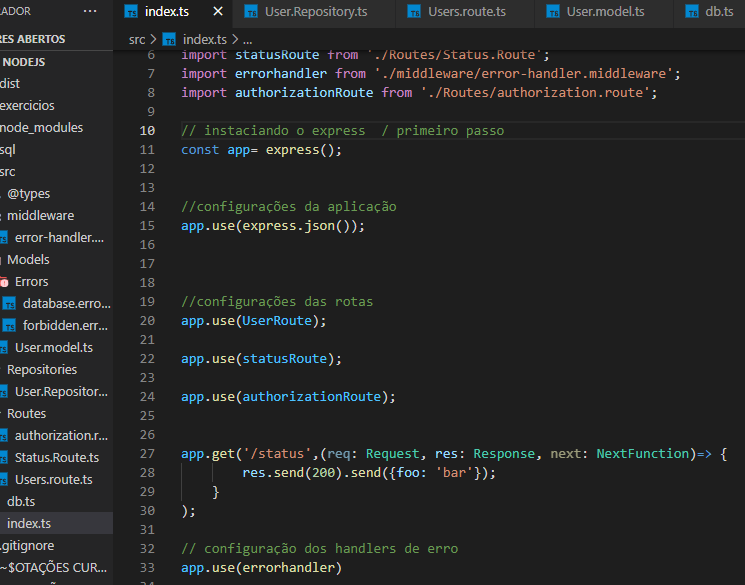
**DESENVOLVENDO ROTA DE AUTORIZAÇÃO E VALIDANDO TOKEN**



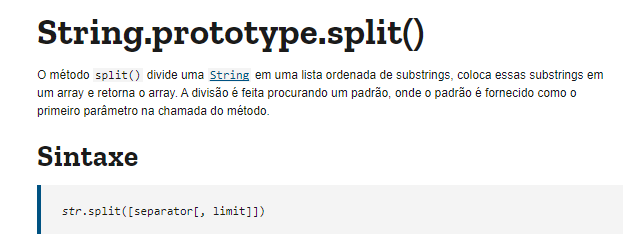




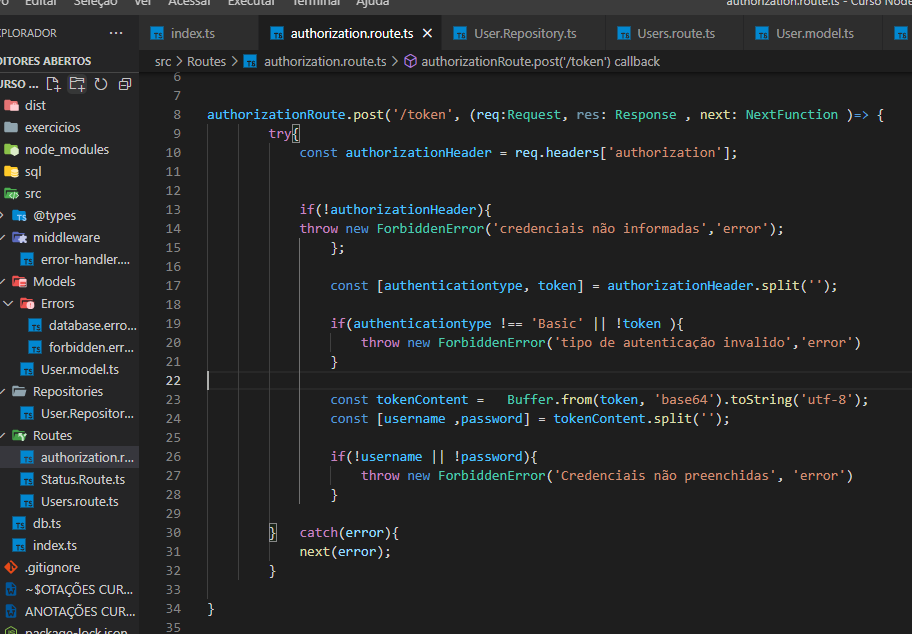








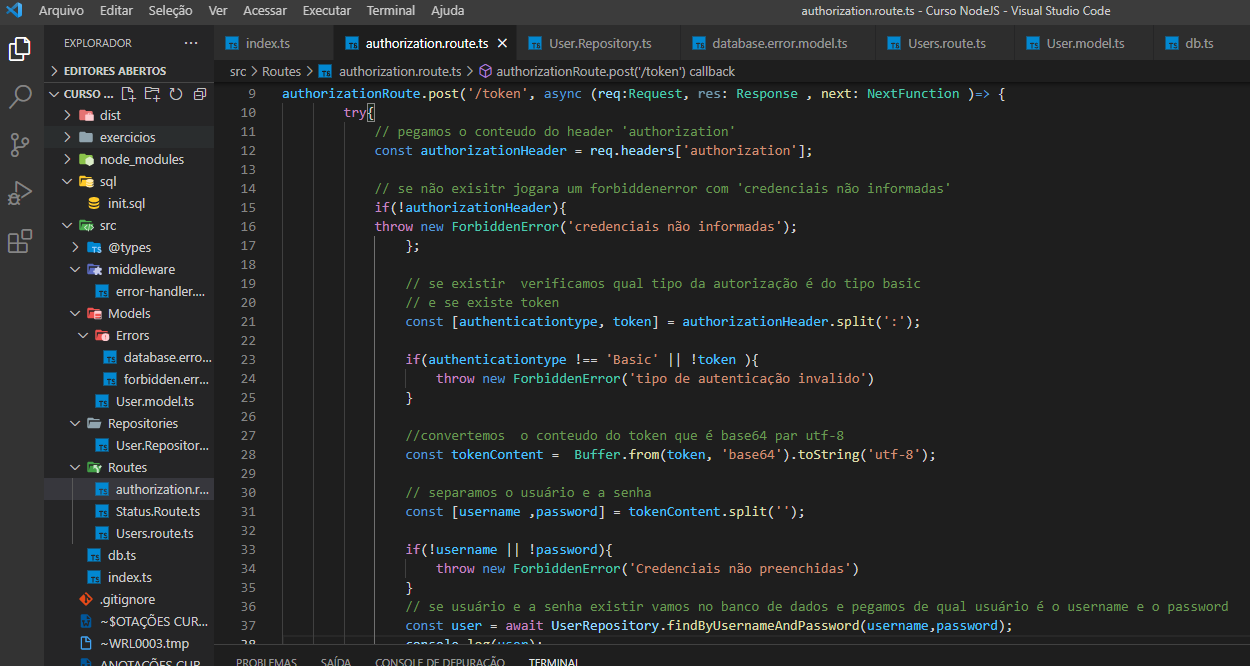
Decodificando token e validando crendencias

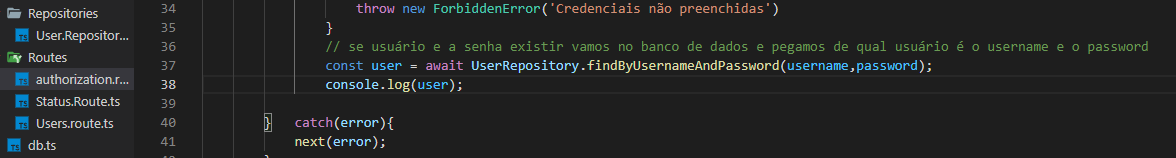




Um buffer é um espaço de memória (tipicamente RAM) que armazena dados binários. No [Node.js](https://nodejs.org/en/about/), podemos acessar esses espaços de memória com a classe Buffer integrada. Os buffers armazenam uma sequência de números inteiros, de maneira similar às [matrizes](https://www.digitalocean.com/community/tutorials/understanding-arrays-in-javascript) em [JavaScript](https://www.digitalocean.com/community/tutorial_series/how-to-code-in-javascript). Diferentemente das matrizes, você não pode alterar o tamanho de um buffer após ele ser criado.

**BUSCANDO O USUÁRIO E SENHA CONFORME SOLICITAÇÃO**



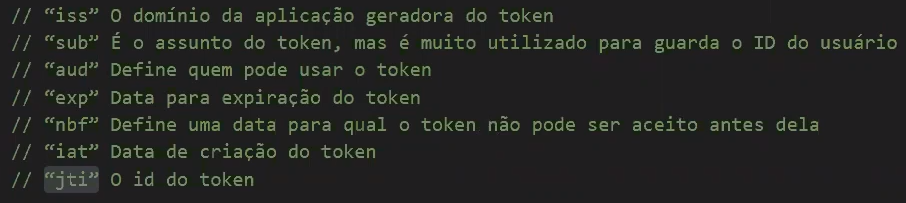


Instalação da jwt para geração de tokens

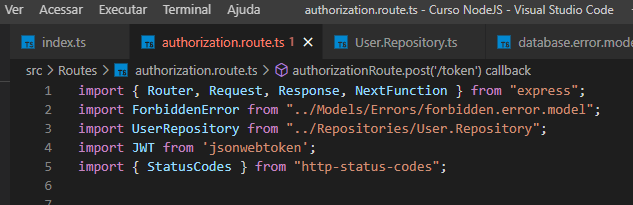


PARA TYPESCRIPT

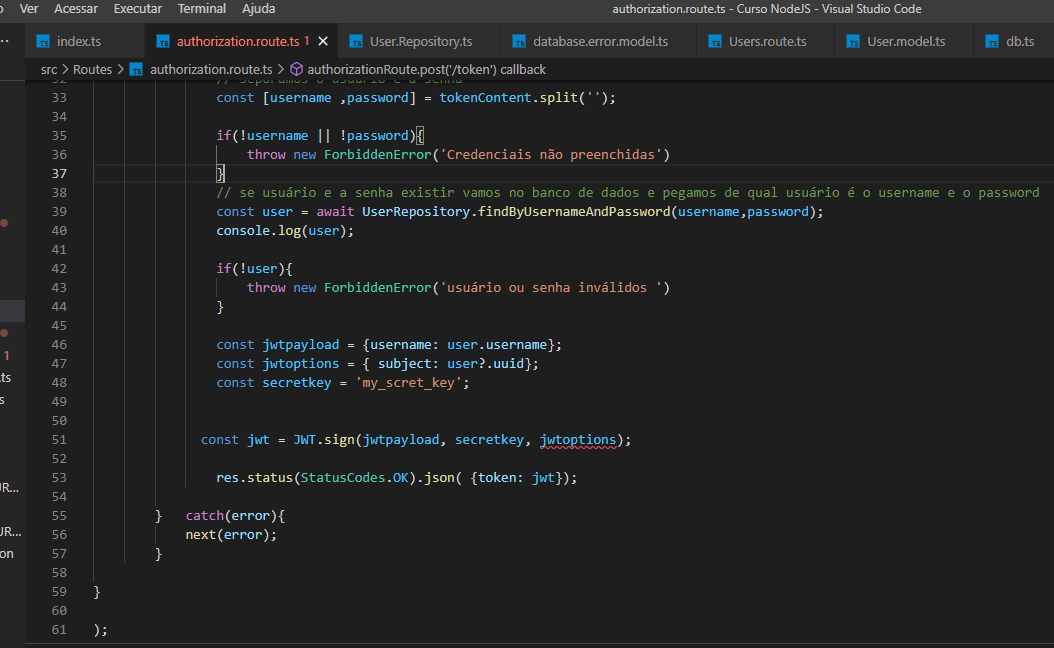




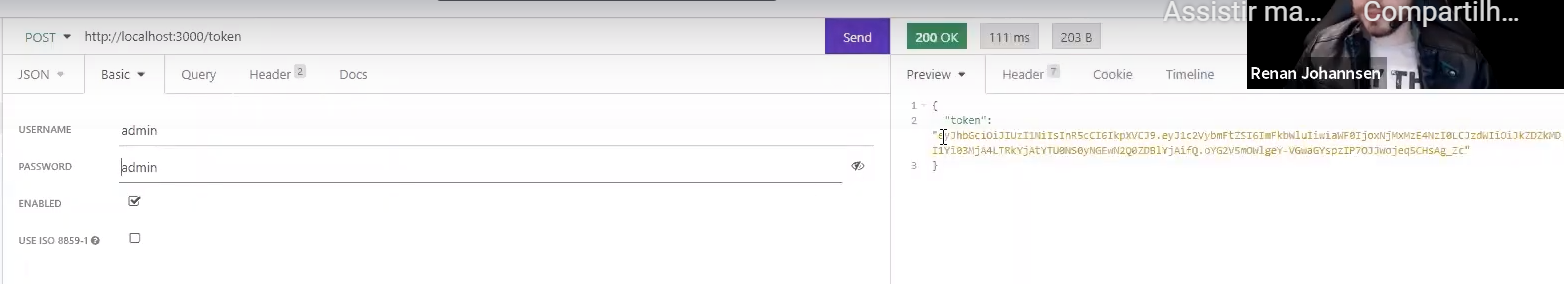
Criando o token

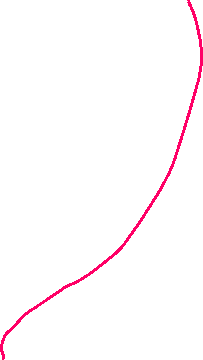


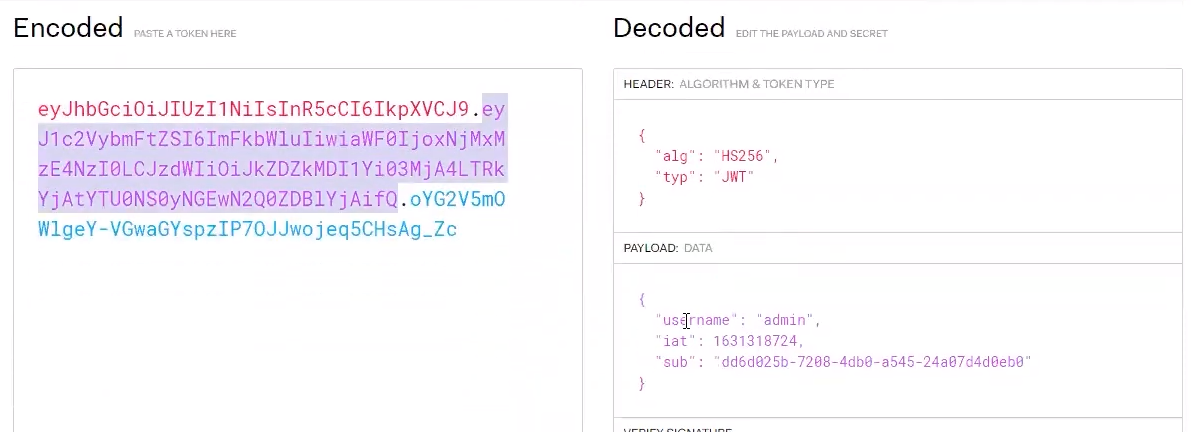






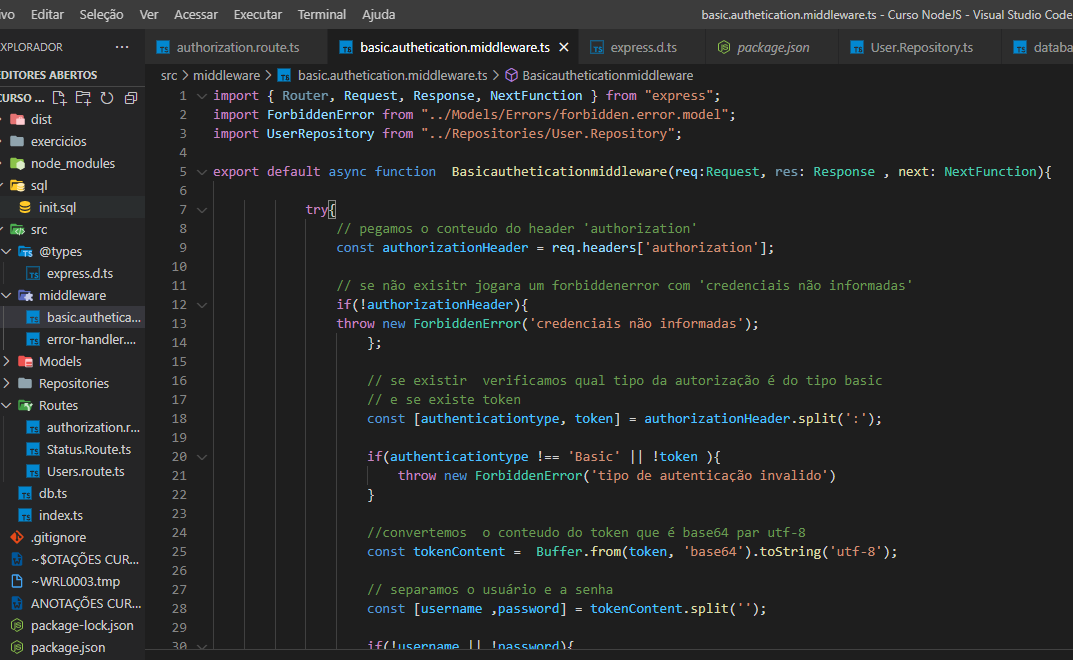


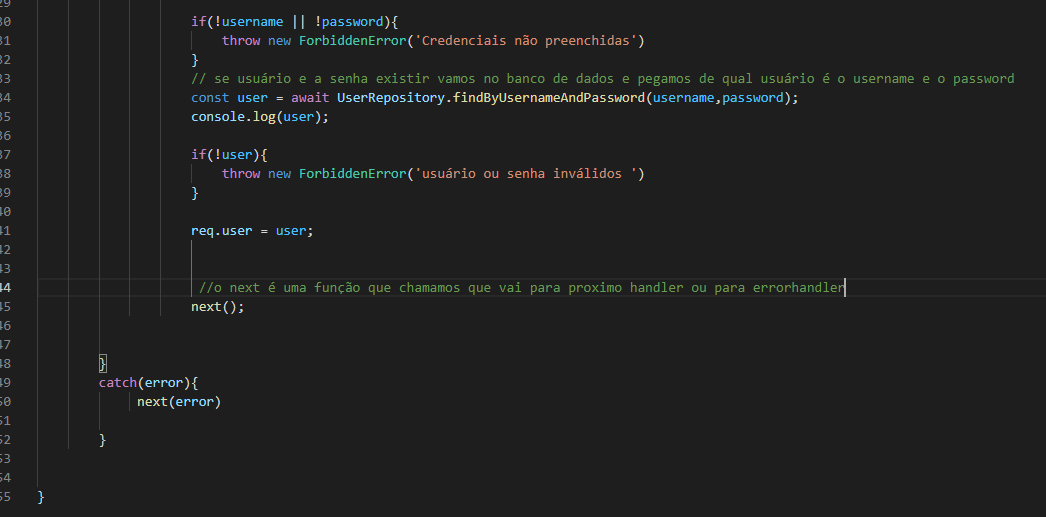


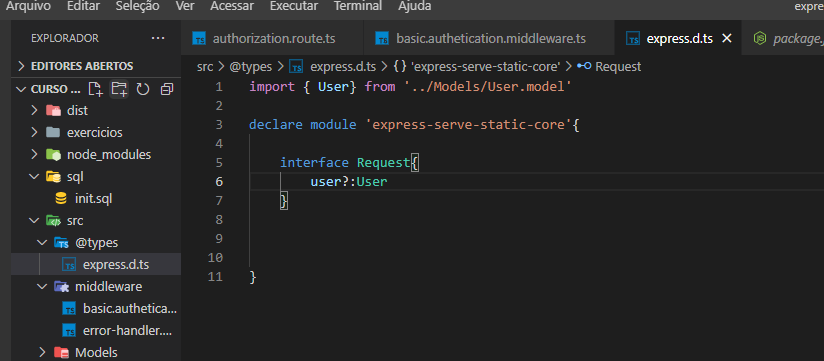


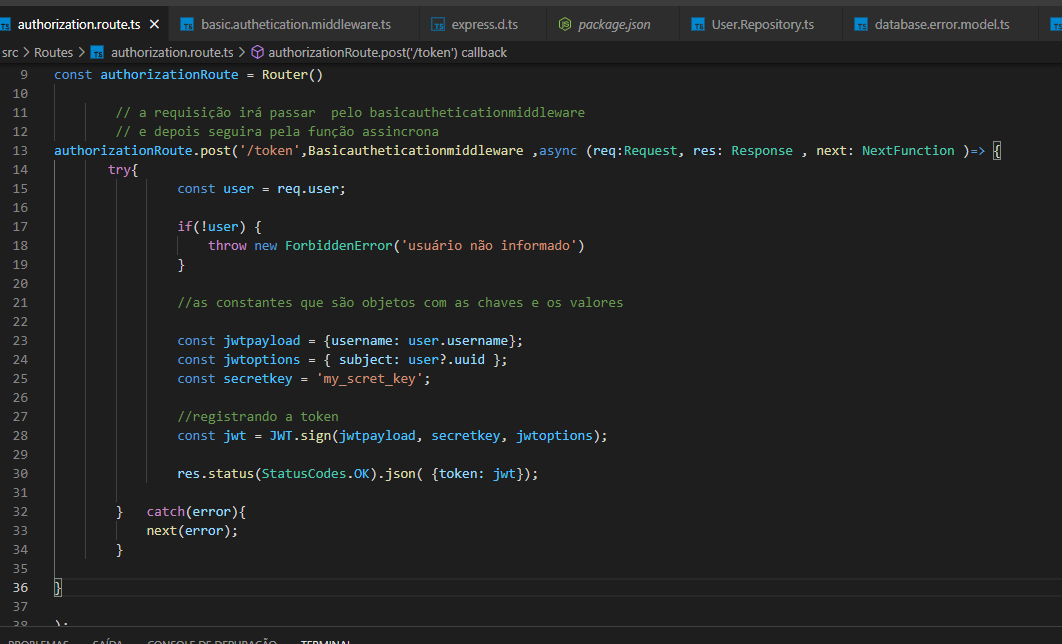


Desenvolvendo um middleware para reutilizar as funções de autenticação

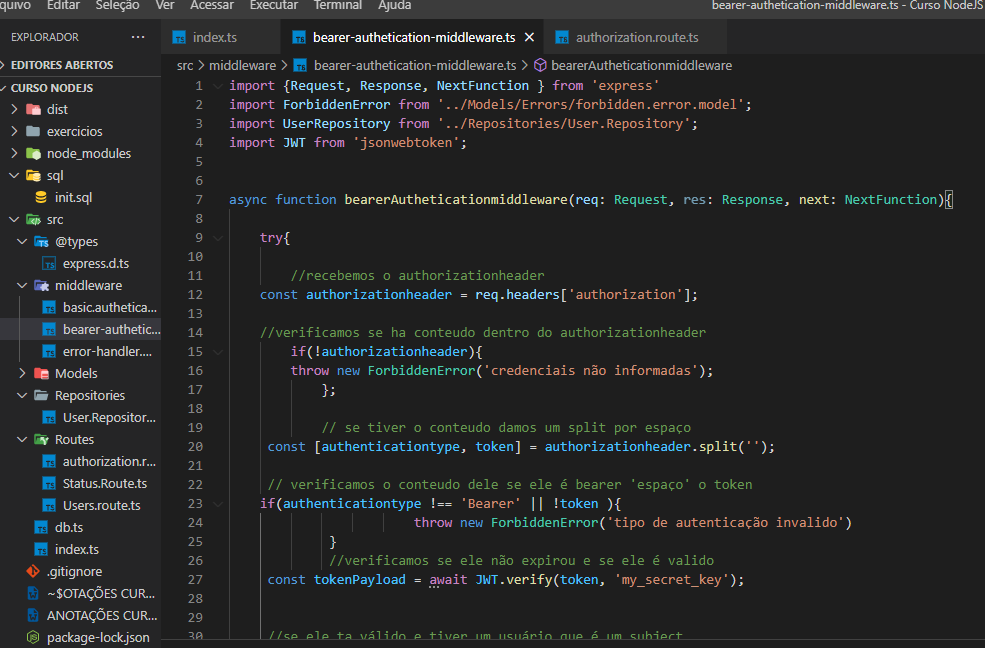


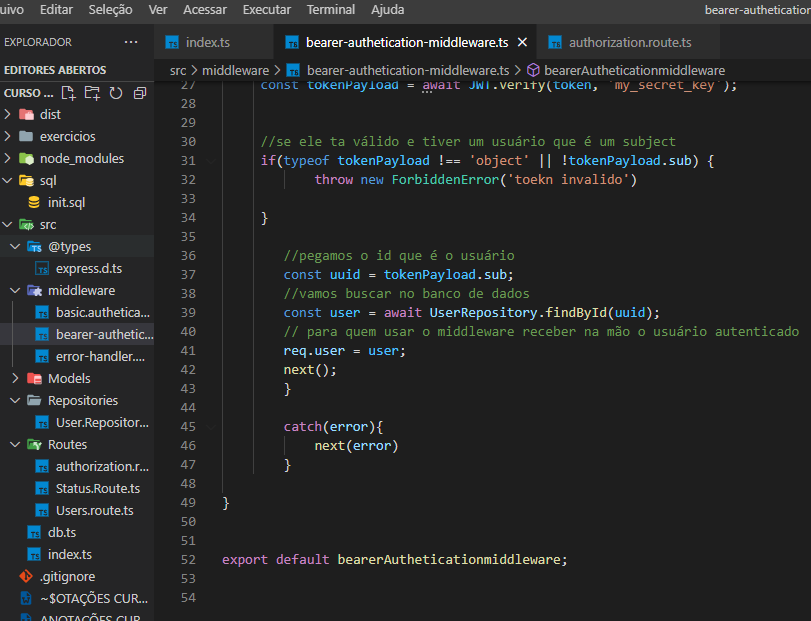






Desenvolvendo a busca de usuários validada por token





Desenvolvendo a rota de validação de token

