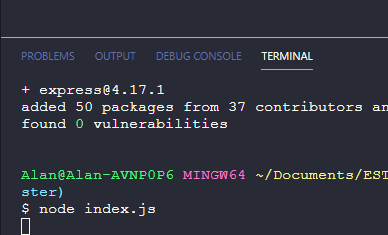
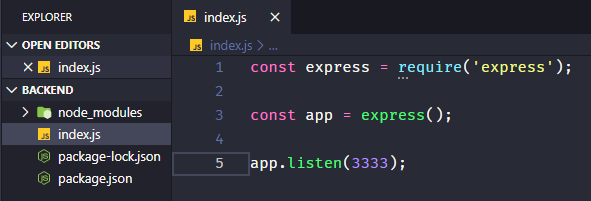
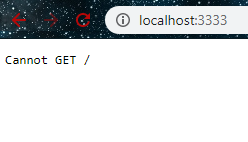
**Omnistack11**

**CRIANDO APLICAÇÃO EM NODE**

Pasta: Backend

Iniciar projeto em node: npm init -y

Instalar express: npm install express

Rodando servidor: node index.js

**Para rodar o servidor, posso usar os comandos:**

node src/index.js ou npm start

**Métodos:**

**GET:** Busca/ Listar uma informação do back-end

**POST:** Criar uma informação no back-end

**PUT:** Alterar uma informação do back-end

**DELETE:** Deletar uma informação do back-end

**Parâmetros (Funções):**

**request:** guarda todos os dados que vem através da requisição do usuário

**response:** é responsável por retornar uma resposta para o usuário

**Tipos de parâmetros:**

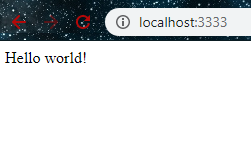
**Query Params:** Parâmetros nomeados enviados na rota após "?" (Filtros, paginação)

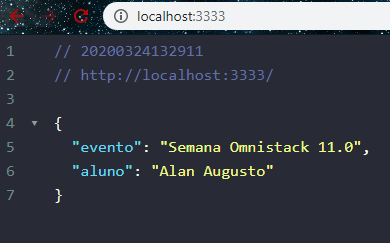
**Route Params:** Parâmetros utilizados para identificar recursos

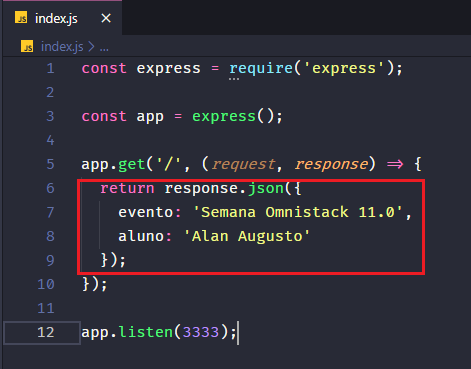
**Request Body:** Corpo da requisição, utilizado para criar ou alterar recursos

Criando uma rota GET(listar), retornando uma mensagem pelo response(responsável por retornar uma resposta ao usuário):

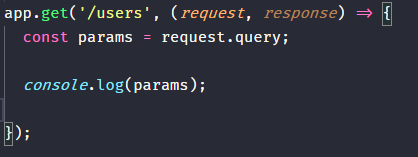


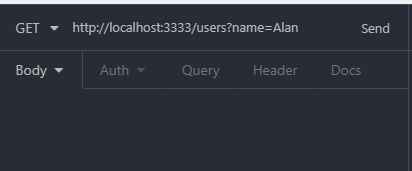


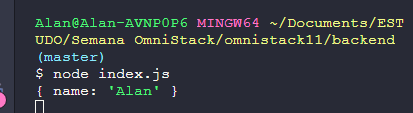
Criando uma rota GET(listar), retornando dados pelo response(responsável por retornar uma resposta ao usuário) no formato json:



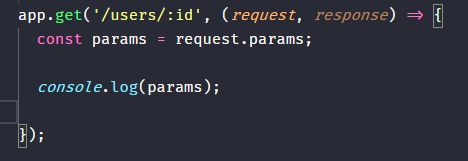
Buscar/ Listar todos os parâmetros que vem através do query params (?name=Alan)

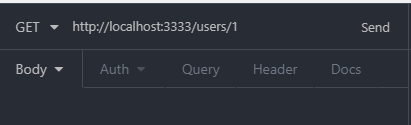


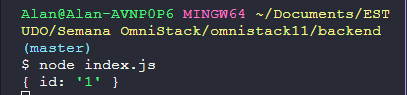


****

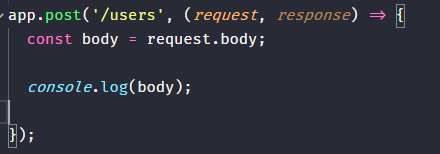
Buscar/ Listar todos os parâmetros que vem através do route params ...users/1)

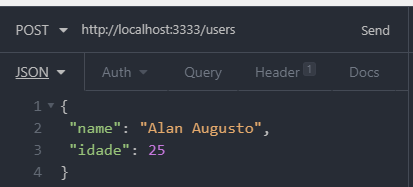
****

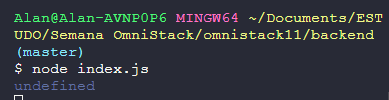
****

****

Criar um usuário por exemplo pelo request.body (.../users)

****

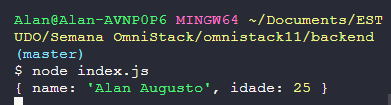
****

****

**Underfined:** porque precisamos avisar que os dados são em formato json, conform abaixo:

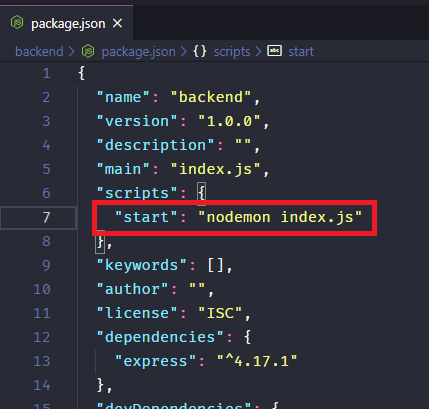


E agora temos a requisição no console:



**Instalar nodemon:** npm intall nodemon -D

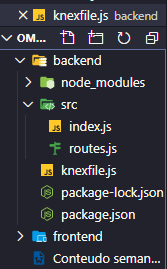
**E configurar package abaixo:**



Criar pasta src

Colocar index.js dentro

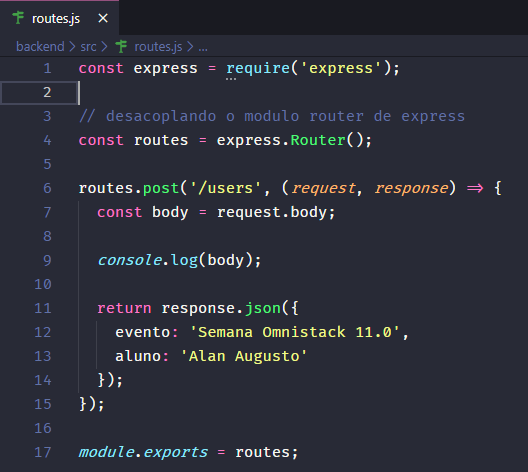
Criar arquivo routes.js e para colocar as rotas e colocar dentro da pasta src



Index.js



routes.js



**CRIANDO E CONFIGURANDO BANCO DE DADOS**

**SQLite usando o KNEX**

**O KNEX.JS é uma Query Builder, que nos permite criar querys para buscar dados do banco, utilizando códigos javascript:**

npm install knex

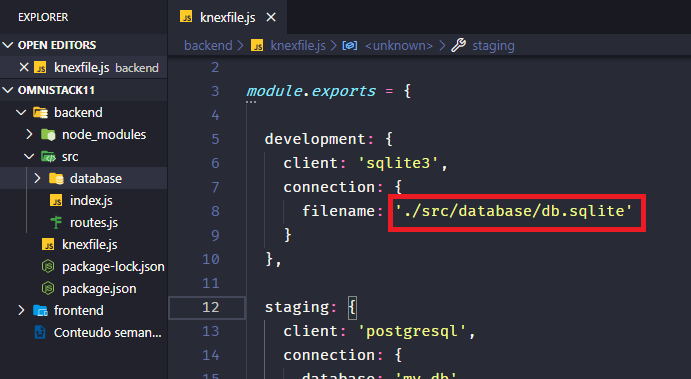
**Instalar banco SQLite por ser do mercado atual:**

npm install sqlite3

**Iniciar banco:**

npx knex init

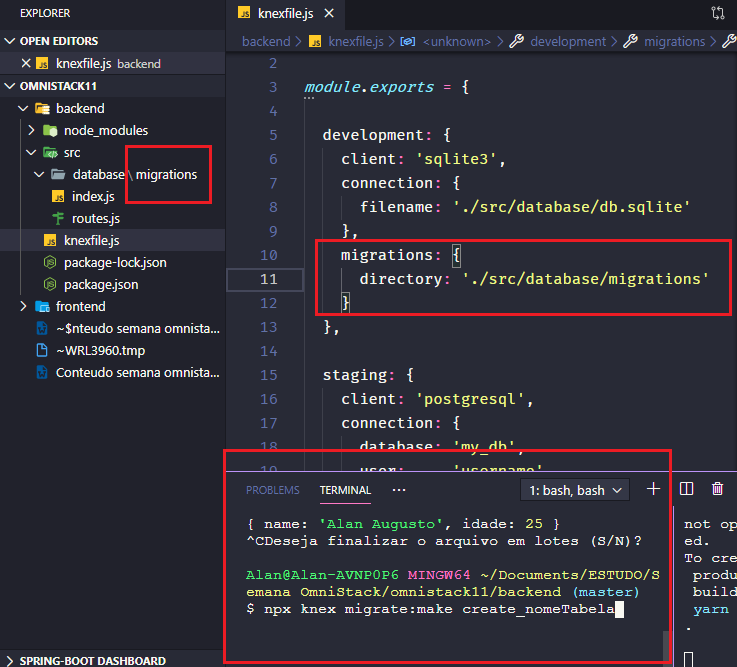
Abrir arquivo knexfile.js, alterar configuração e criar pasta database:



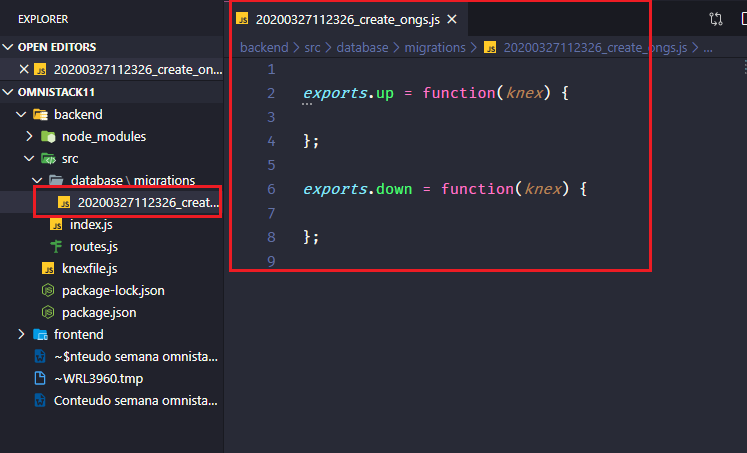
**CRIANDO A PRIMEIRA MIGRATIONS**

**Conceito:** crias tabelas e manter um histórico das tabelas criadas, alteradas, como data e horário, como se fosse um controle de versão.

**Antes de criarmos a primeira migrations, vamos criar a pasta migrations e configurar o arquivo knexfile.js apontando para a pasta em seguida dar o comando:** npx knex migrate:make create\_nomeTabela

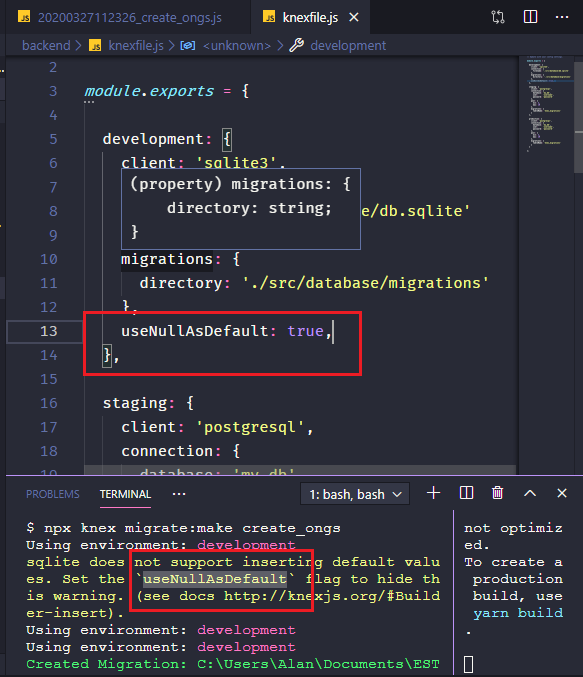
****

**Migration criada**

****

**Configurar o valor padrão das colunas para serem sempre nulos (null), valor vazio, no arquivo knexfile.js,**

**Porque por padrão o SQLite não suporta a incersão o default values, de valores padrão para as colunas no banco de dados.**

****

**Criando tabelas com a estrutura do knex**

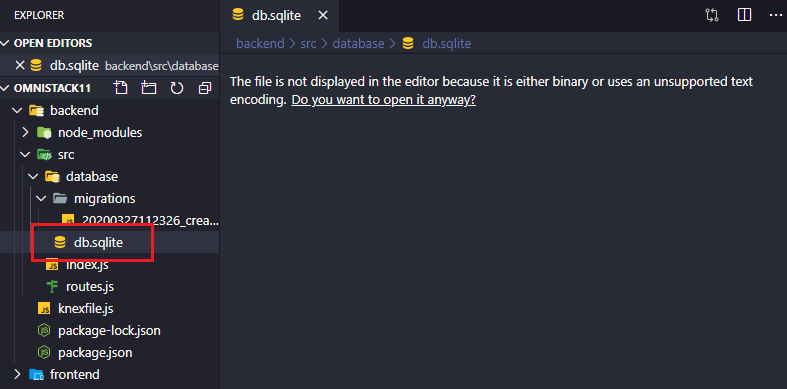
O método up é responsável por criar a tabela

O Método down para excluir a tabela caso queira alterar algo.

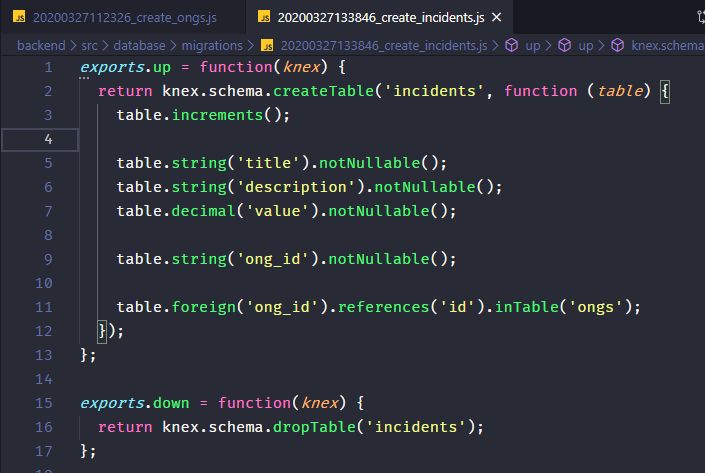


**Comando abaixo para executar a criação da tabela:**

npx knex migrate:latest



**Criando outra tabela “incidents” e referenciando a tabela “ongs”**

****

**Para deletar a tabela criada, usar o comando:**

npx knex migrate:rollback

**ele vai deletar a ultima tabela criada**

**Para listar as tabelas criada, usar o comando:**

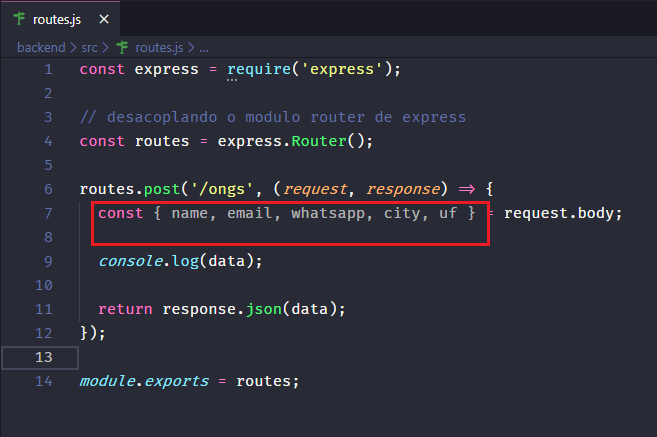
npx knex migrate:status

**Trazendo os dados cadastrados pelo usuário**

****

****

**Dentro do data temos os dados armazenados, e podemos pegar cada dado em uma variável separada, ou seja, desestruturar, como abaixo:**

****

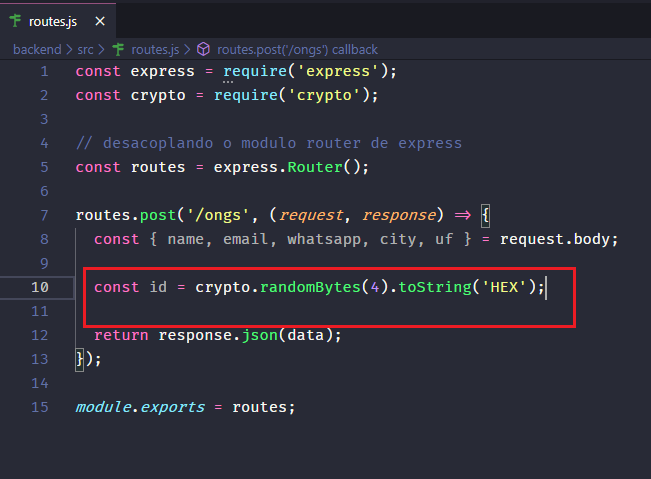
**Agora vamos criar o id da ong, não fazer com que o usuário crie. E vamos utilizar o pacote crypto, que vem junto com o node.**

**O pacote crypto é para criptografia, porém conseguimos utilizá-lo para gerar uma string aleatória, um texto aleatório.**

**randomByts(4) –** cria quatro bytes de caractere

**toString() –** converte estes caractere em string do tipo HEXADECIMAL

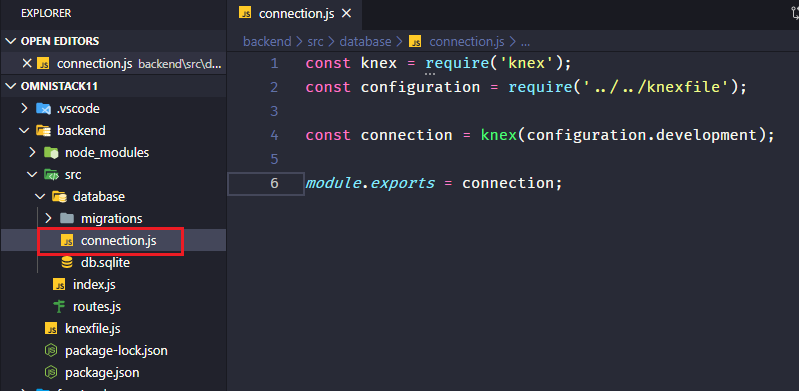
**ou seja, ele vai criar ids contendo caractere e números**

****

**Agora vamos fazer a conexão com o banco de dados**

Criaremos um arquivo chamado **connection.js**

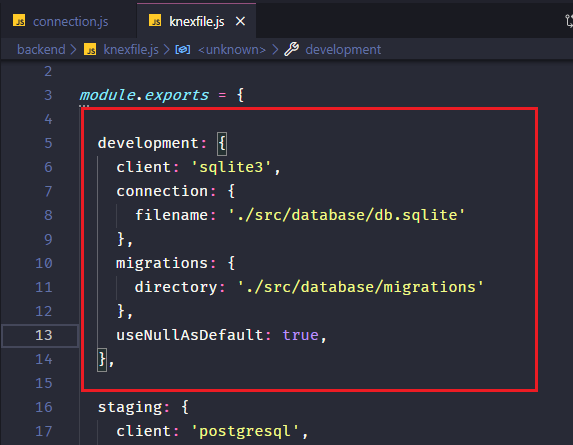
**E configuramos conforme abaixo:**

****

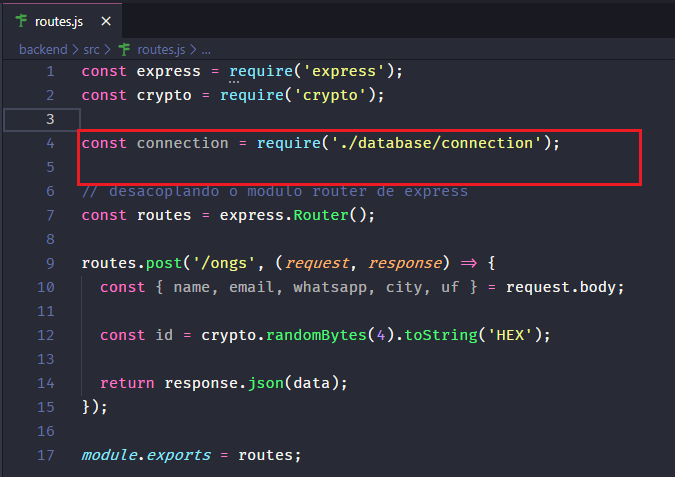
**- Importamos o knex**

**- Importamos as configurações do nosso banco de dados, disponíveis no arquivo knexfile.js**

**- Criando a conexão e passando como parâmetro configuration e o development que é a conexão de desenvolvimento:**

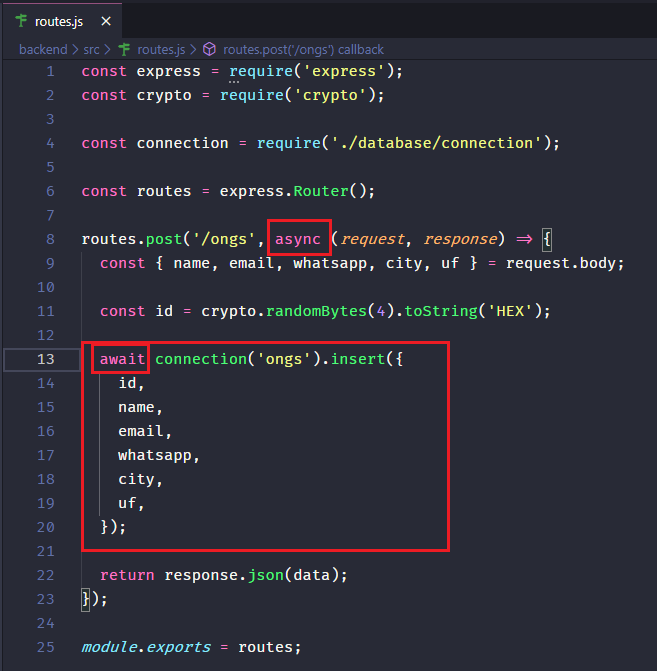
****

**E vamos importar esta conexão dentro do arquivo que quero conectar**

****

**E agora vamos conseguir fazer as operações com o banco de dados**

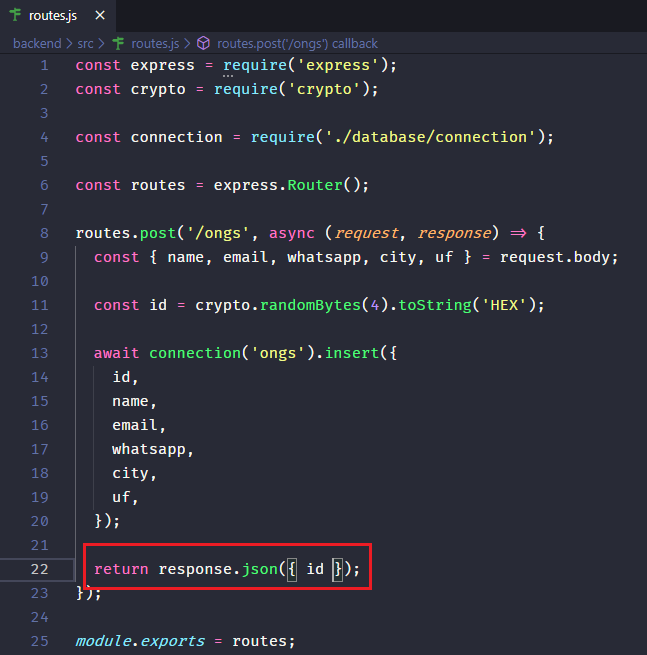
**Agora vamos utilizar o connection e o método insert para inserir dados, especificando qual a tabela e as colunas que desejo inserir**

****

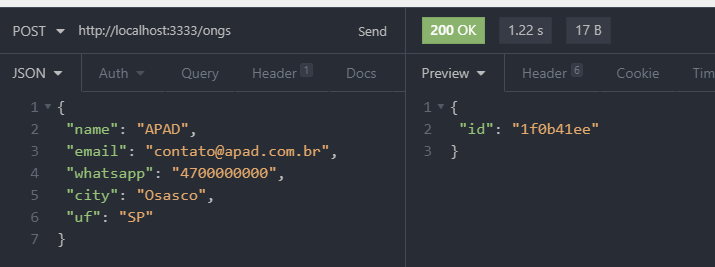
**O async e o await serve para dar uma pausa no momento da inserção do dados finalizar pra então continuar no return, isso pois para não haver uma demora na inserção.**

**E precisamos retornar, devolver a resposta para o cliente, a ong, que neste caso será o id,**

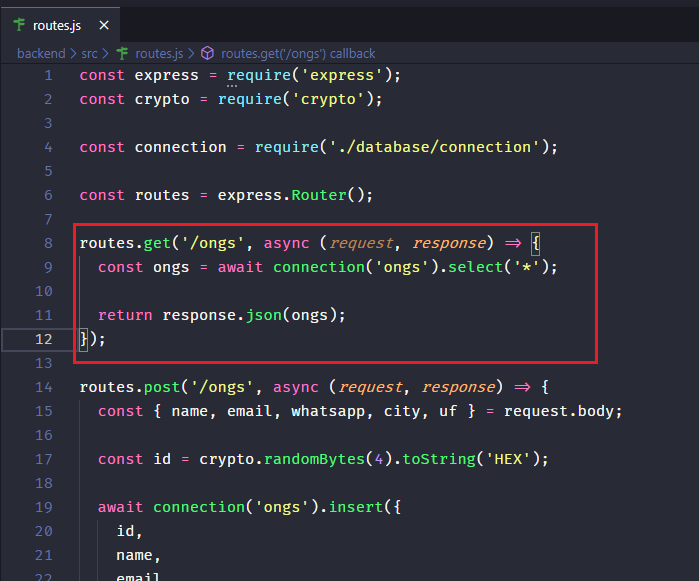
**Porque quando a ong se cadastra dentro da aplicação, ela precisa saber esse id que foi cadastrado pra ela, porque isso vai funcionar como se fosse um cpf, um cnpj dessa ong dentro do nosso sistema, esse id que ela vai utilizar para se comunicar dentro da aplicação**

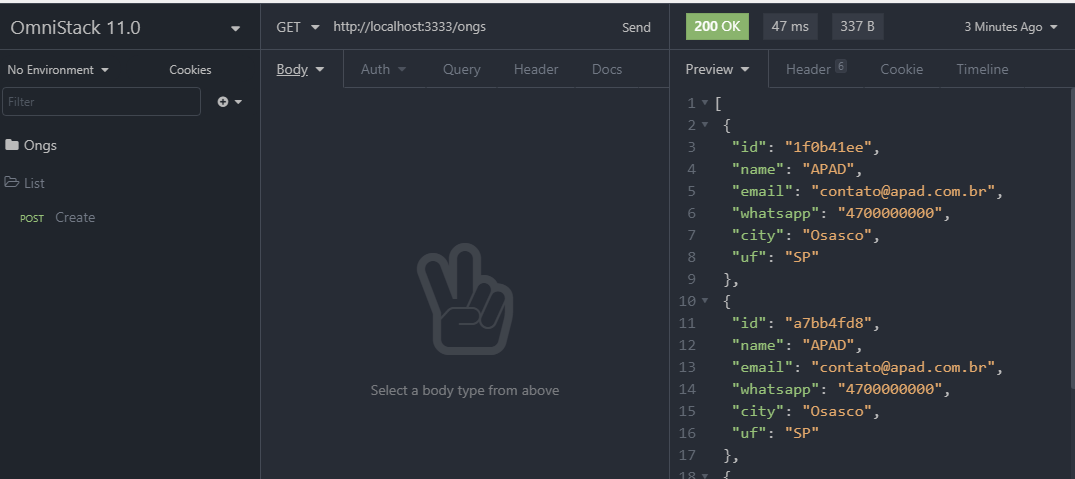
****

**Testando no insomnia**

****

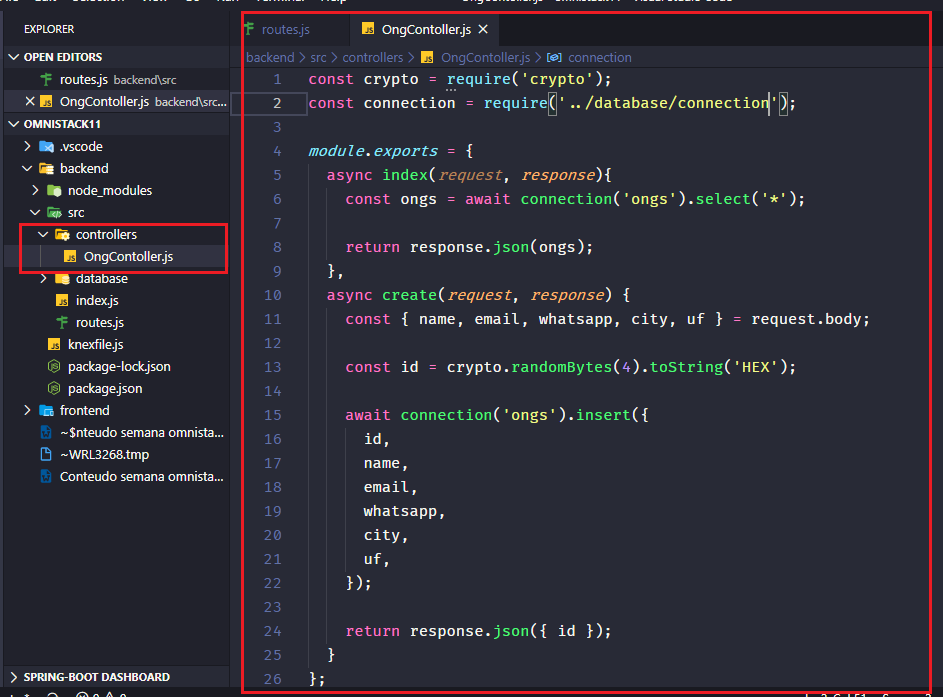
**Agora vamos criar uma rota para listar todas as ong do banco de dados**

****

****

**Agora vamos organizar mais**

**Criar uma pastar controllers onde passaremos nossas querys, começamos exportando um objeto: module.exports = {}; e trazendo o connection e o crypton**

****

**E chamamos o controller em routes**

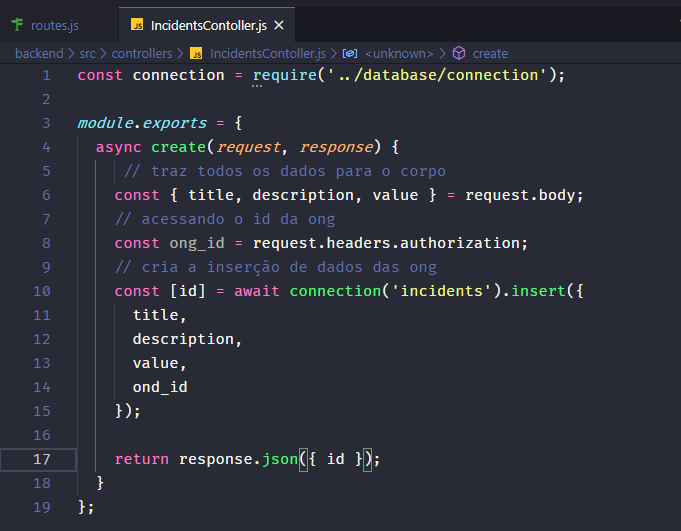
****

**Agora vamos criar um controller para o incidents e logar o id da ong:**

1° criamos um método e colocamos os campos que queremos preencher

2° acessar o id da ong através do cabeçalho: request.headers.authorization.

Authorization porque foi o nome que demos.

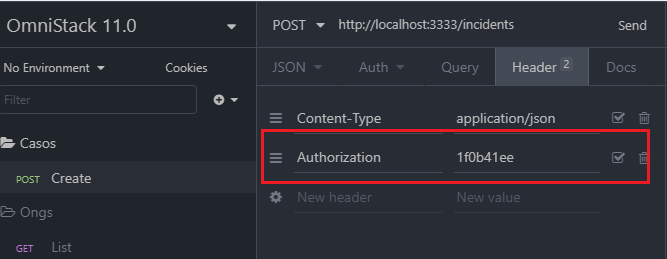
****

**[id] – para armazenar as posições de cada incidents**

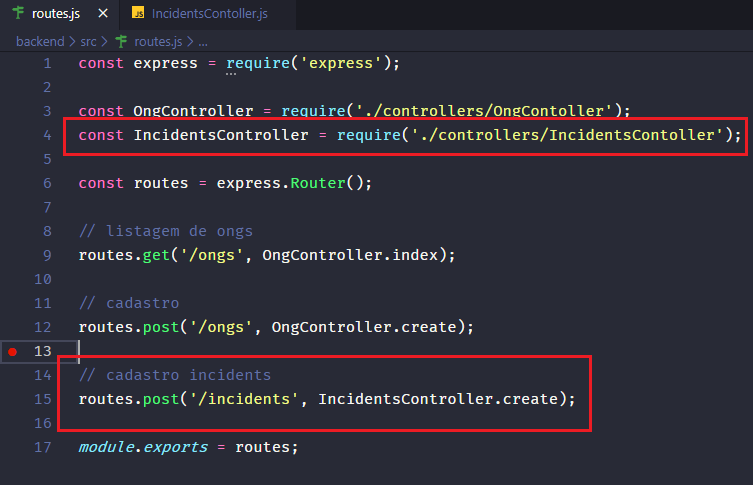
**return – retornando somente o id**

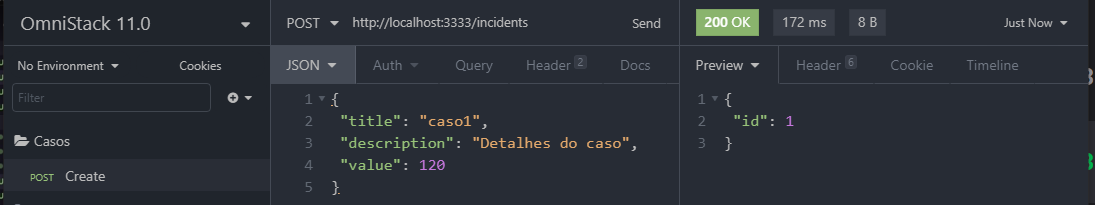
****

**No campo Header criamos a autenticação do id da ong**

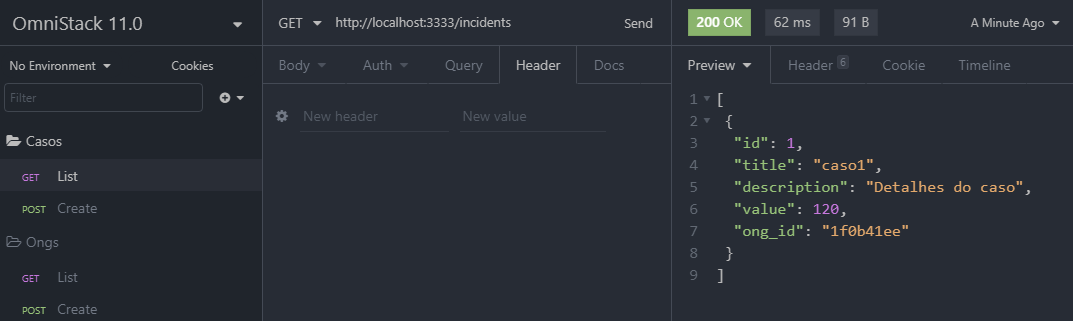
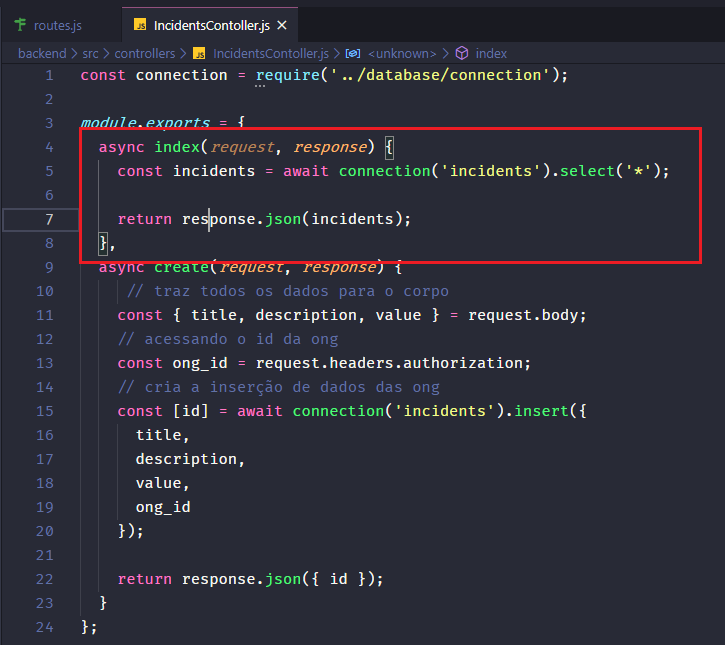
****

**E criamos a rota para incidents**

****

****

**Agora vamos listar os incidents**

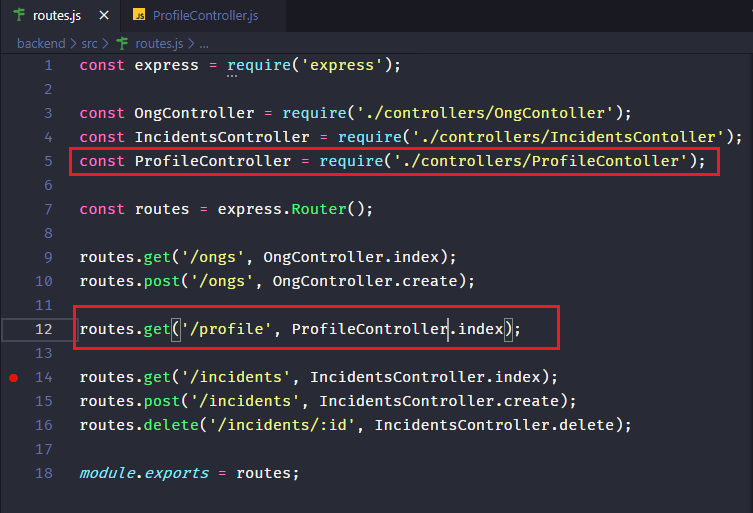
****

**Agora vamos criar uma rota para deletar os incidents**

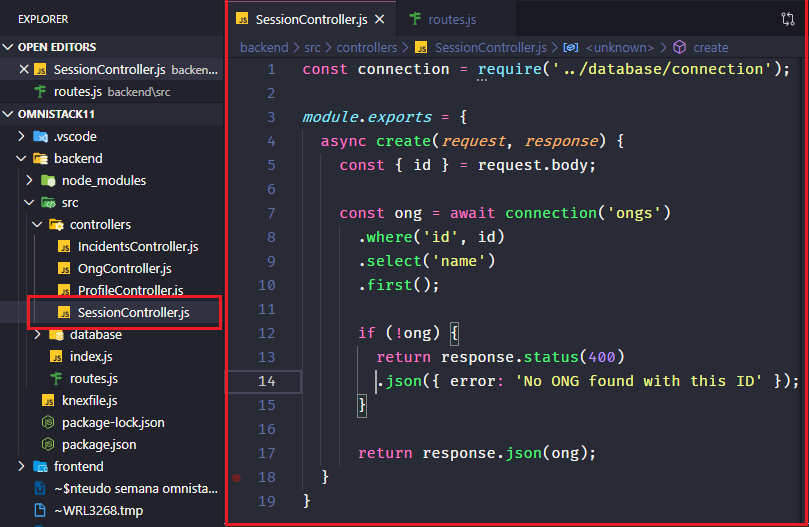
****

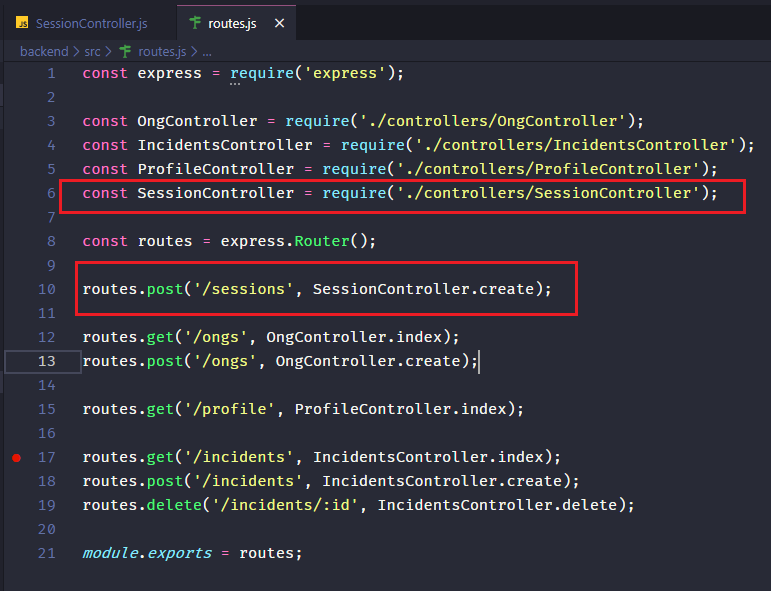
****

**Listando os incidents específico de uma ong**

****

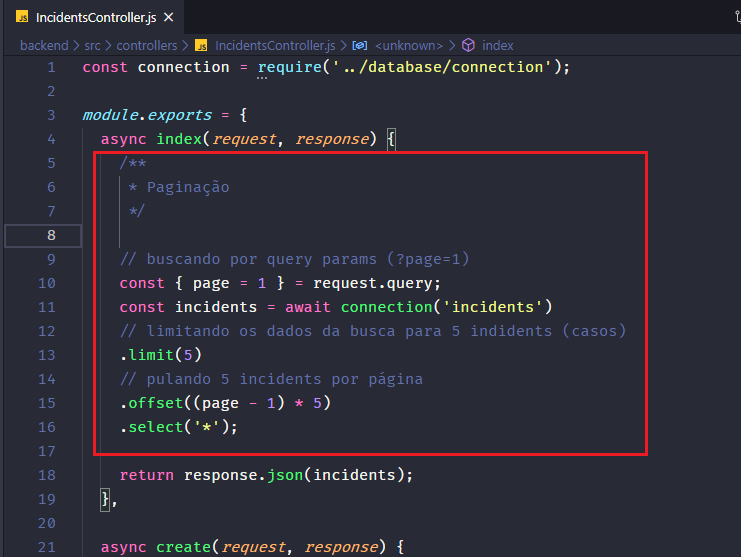
**Agora vamos criar o login para a ong**

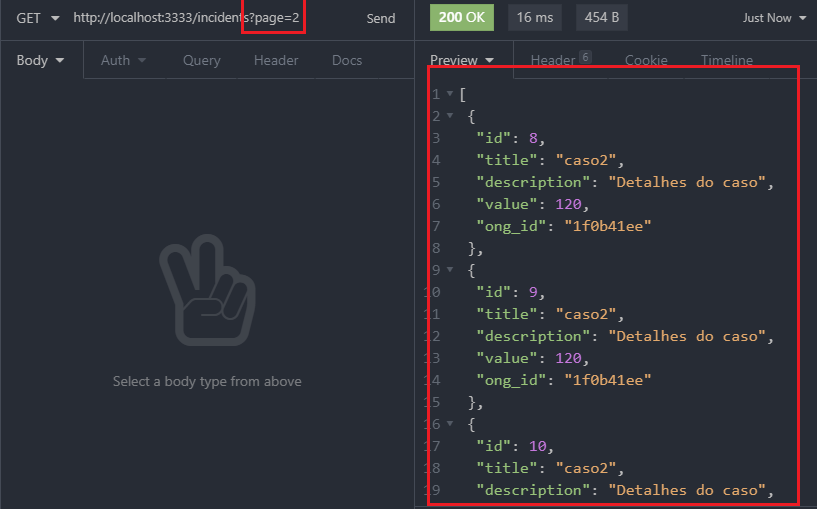
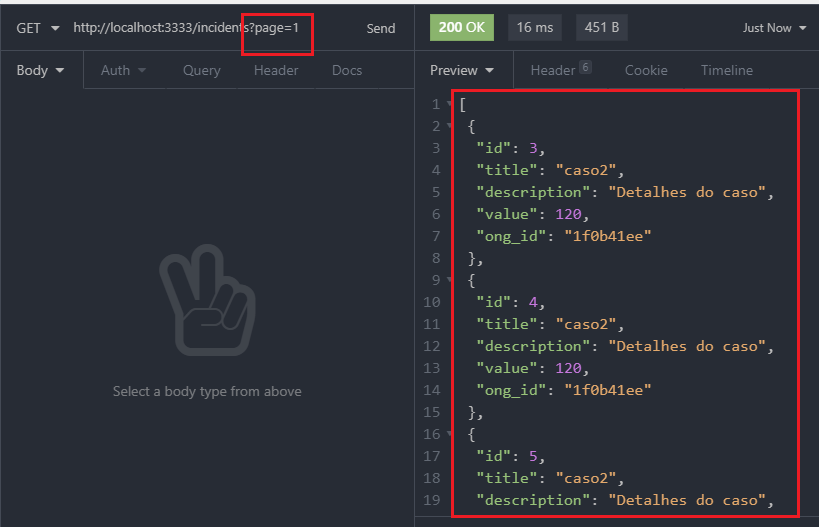
****

****

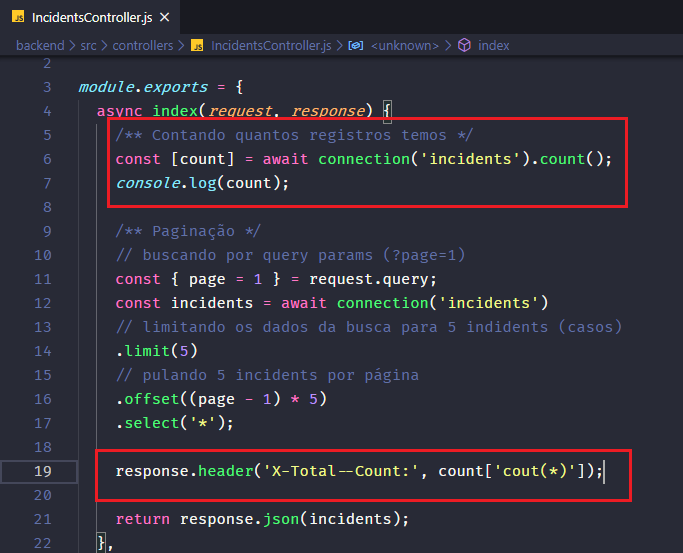
**Agora vamos criar um processo de paginação, limitando 5 incidents por busca.**

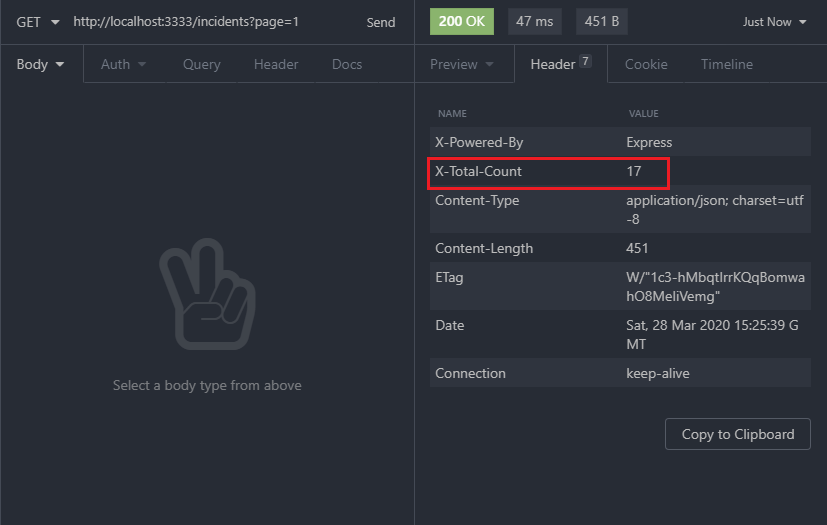
**Obs: Foi feito no método de listagem de todos os incidentes.**

****

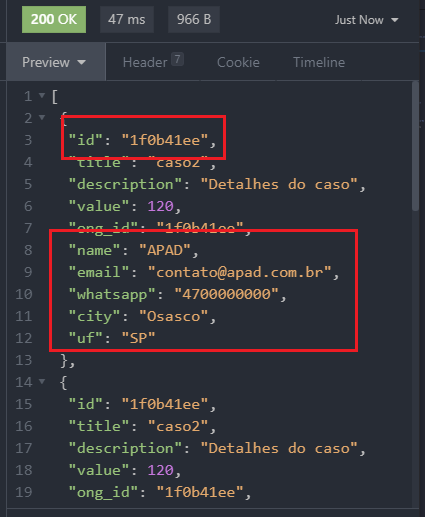
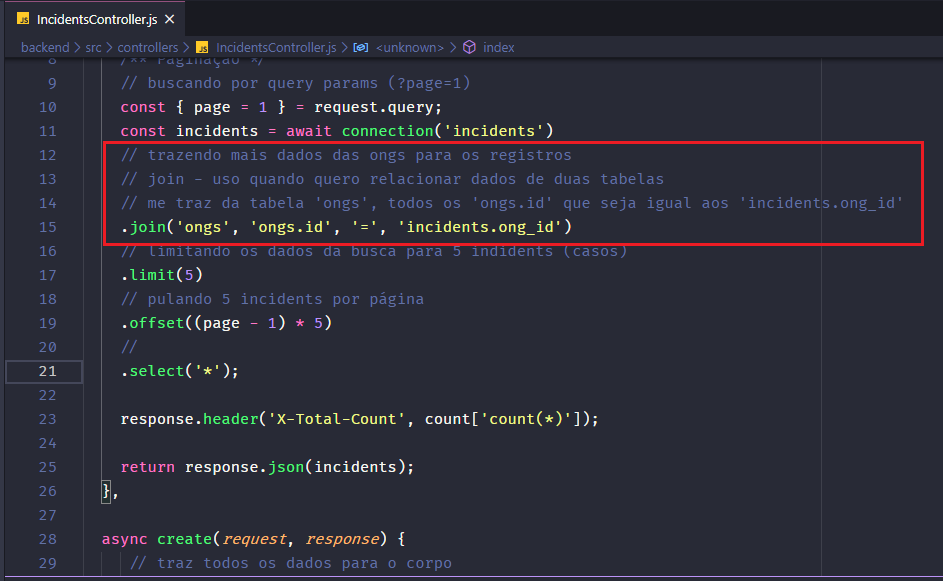
****

**Agora vamos informar o front-end quantos incidents tem no total no banco de dados pelo cabeçalho da resposta**

****

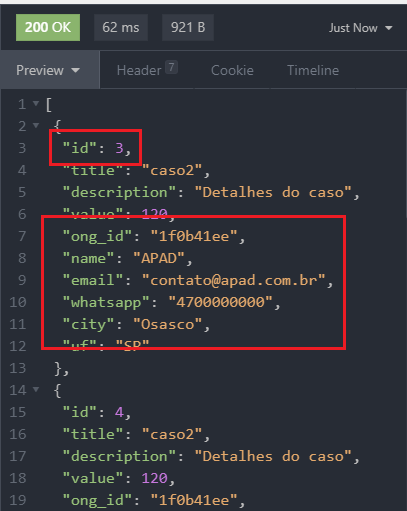
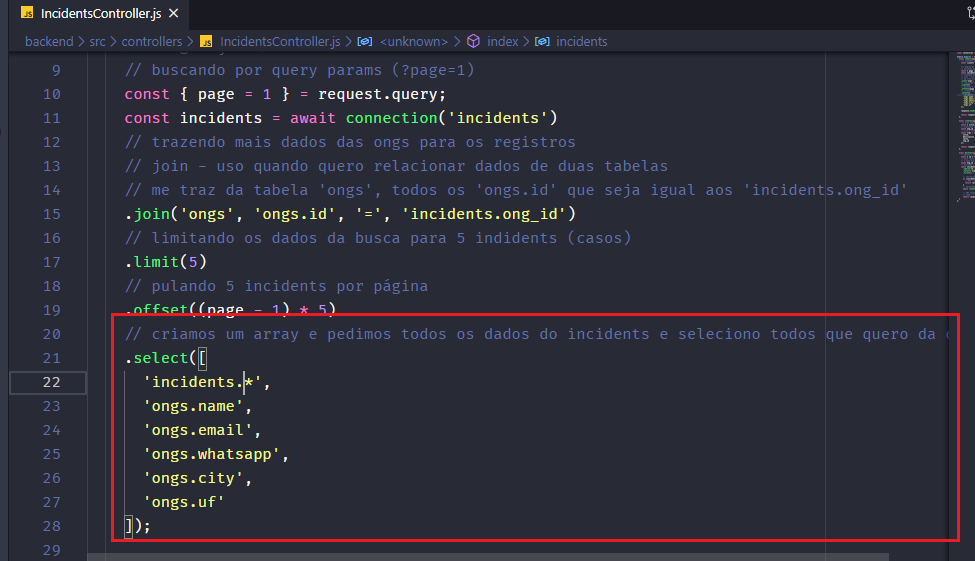
****

**Agora vamos trazer mais dados da tabela de ongs para os registros**

****

**Porem como a ong tbm tem um id, ele sobre pôs no id do incidents, como podemos ver acima**

**Para resolver isso, vamos no select, e criamos um array e pedimos todos os dados do incidents e seleciono todos que quero da ong conforme abaixo:**

****

**Finalizamos nossa base de dados, agora vamos adicionar o módulo de segurança, o CORS, é ele quem determina quem vai poder acessar nossa aplicação**

npm install cors

**vamos começar importando o CORS no index**

****

**Permiti que todas as aplicaçãoes frontends podem acessar**

**Agora vamos enviar nossa aplicação para o github**

**1° criamos o arquivo .gitignore e dizer quais os arquivos não quero enviar para o github:**

**node\_modules – nunca enviamos, porque é muito pesado, a pessoa baixa nosso package.json que já tem a referência do nosso projeto e ai ele vai dar uma “npm install” e automaticamente este comando vai criar o node\_modules.**

****

**Obs: No front-end o .gitignore já vem criado.**

**Depois acompanhar o passo-a-passo**

No terminal do vs code:

git init

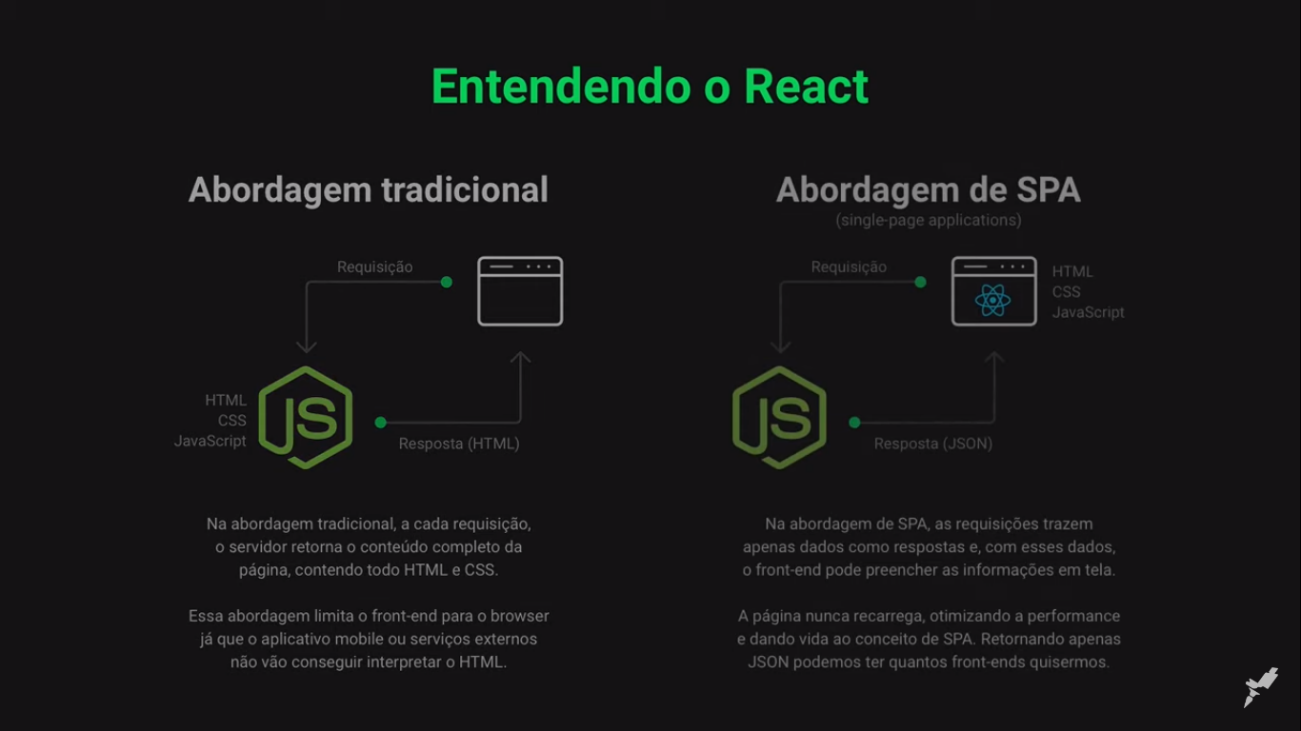
git add .

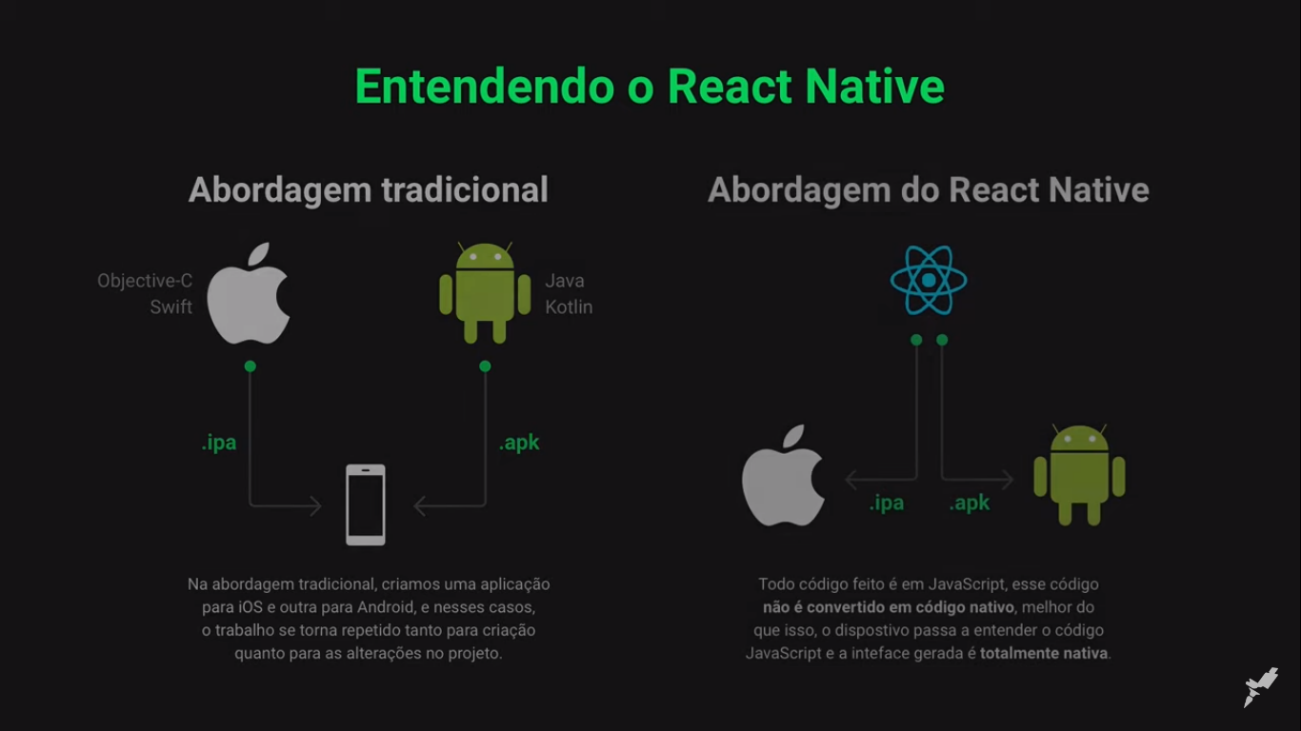
git commit -m “mensagem”

git remote add origin + endereço do repositório

git push -u origin master

**APLICAÇÃO EM REACT**







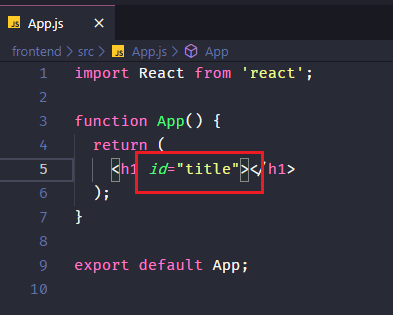


Pasta: Frontend

Criar projeto: npx create-react-app nome\_projeto

**Conceito de propriedade (props)**

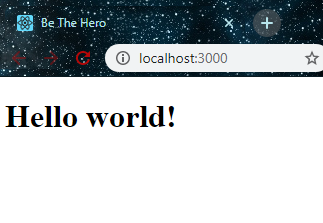
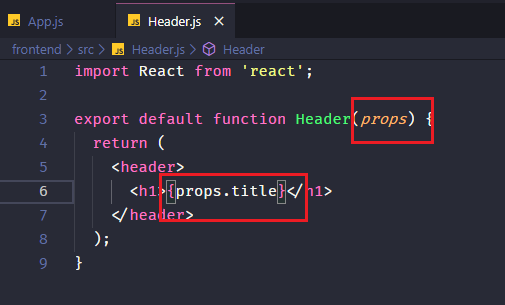
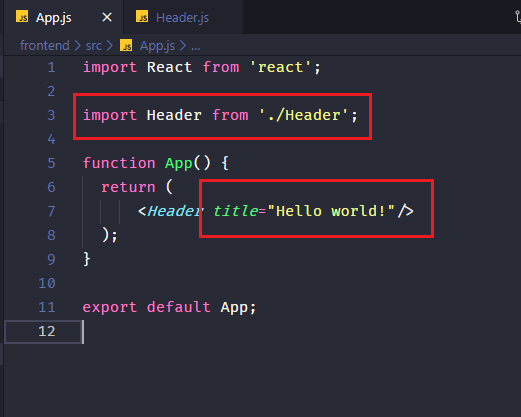
**É como os atributos do HTML, podemos passar os atributos como por exemplo id=”title” conforme abaixo:**

****

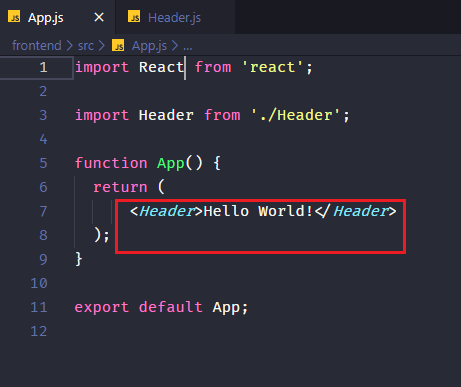
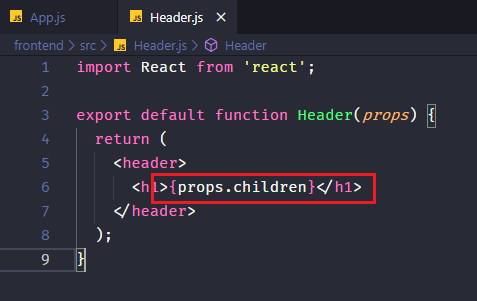
**Mesma sintaxe, porém são atributos passados para componentes ao invés de elementos do HTML.**

**Componentes são partes de uma página que podem ser utilizados várias vezes, são códigos encapsulados, ali em uma parte da página, e que podemos ter uma manutenção simplificada!!!**

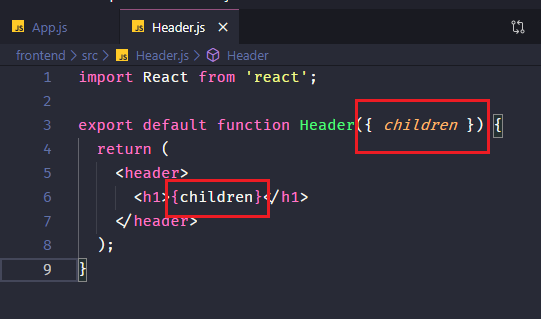
**Agora vamos criar um novo componente, para integrar dentro de outro e usando props**

****

**Outra forma de trabalhar com propriedade é o children que retorna todo o conteúdo**

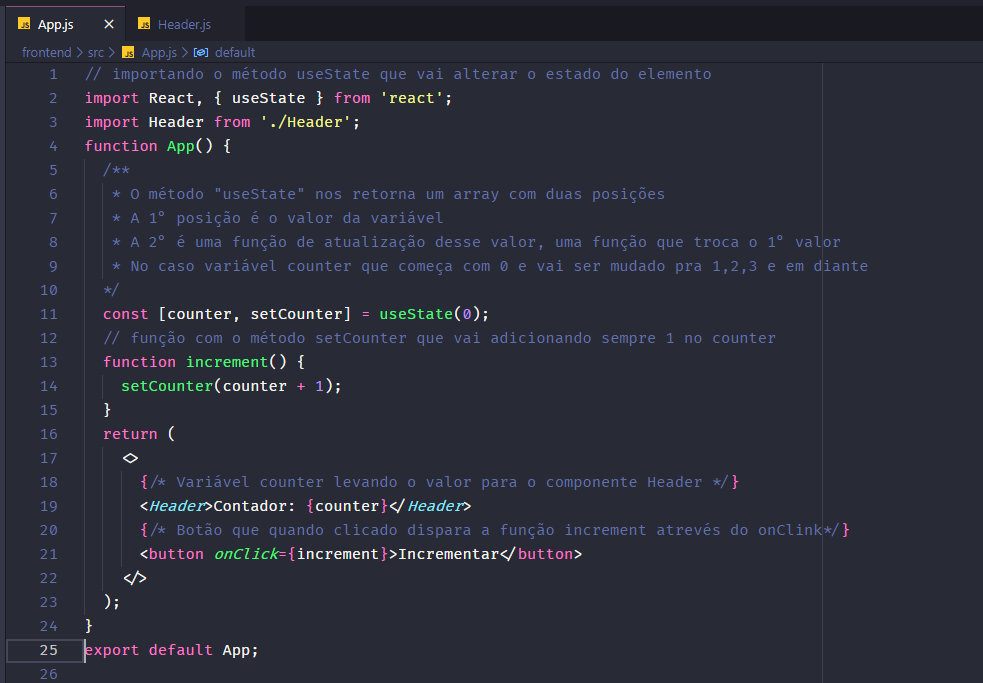
** **

**Ou fazer uma desestruturação, conforme abaixo:**

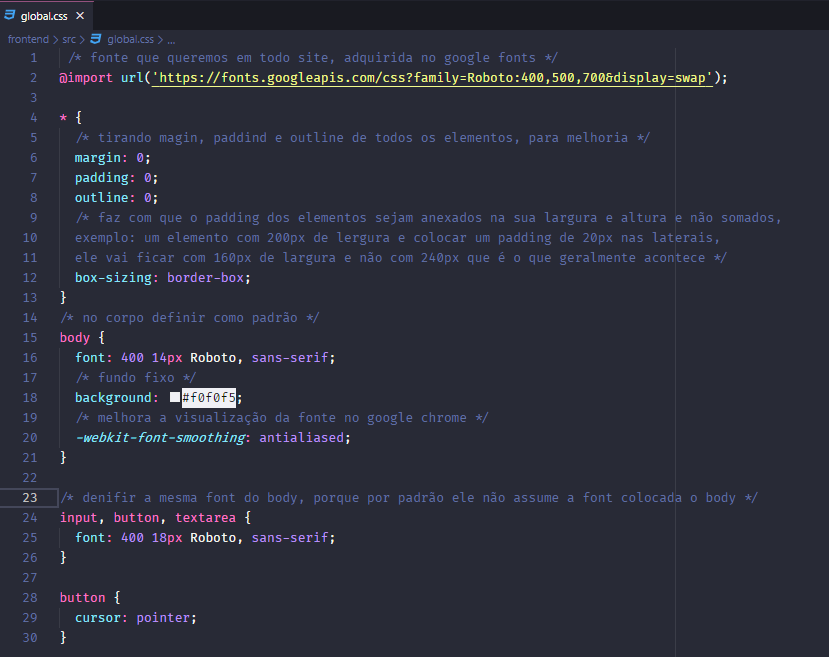
****

**Conceito de estado (state)**

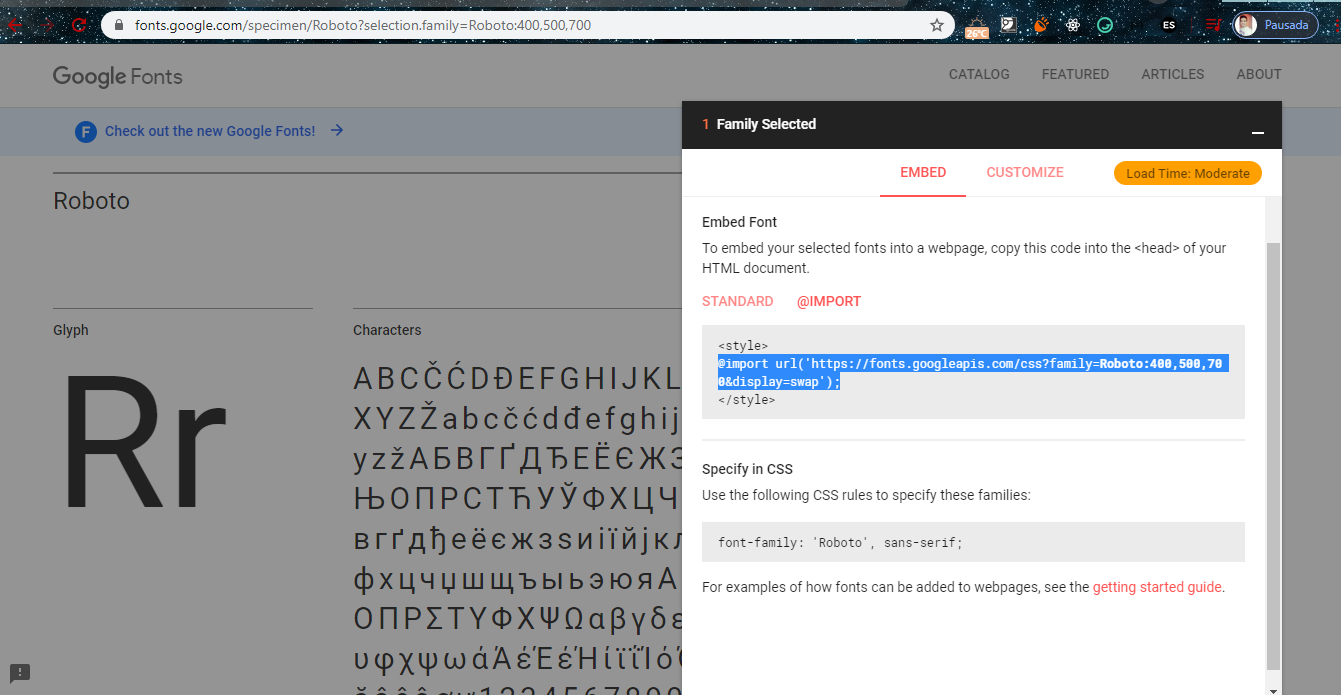
**Podemos entender ele como uma informação que vai ser mantida pelo componente, então imagine que o componente precisa armazenar um tipo de informação, que seja um input do usuário, etc. Ele precisa modificar essa informação, adicionar ou excluir itens e etc. Então no react não podemos usar variáveis convencionais, usamos um conceito chamado estado.**

****

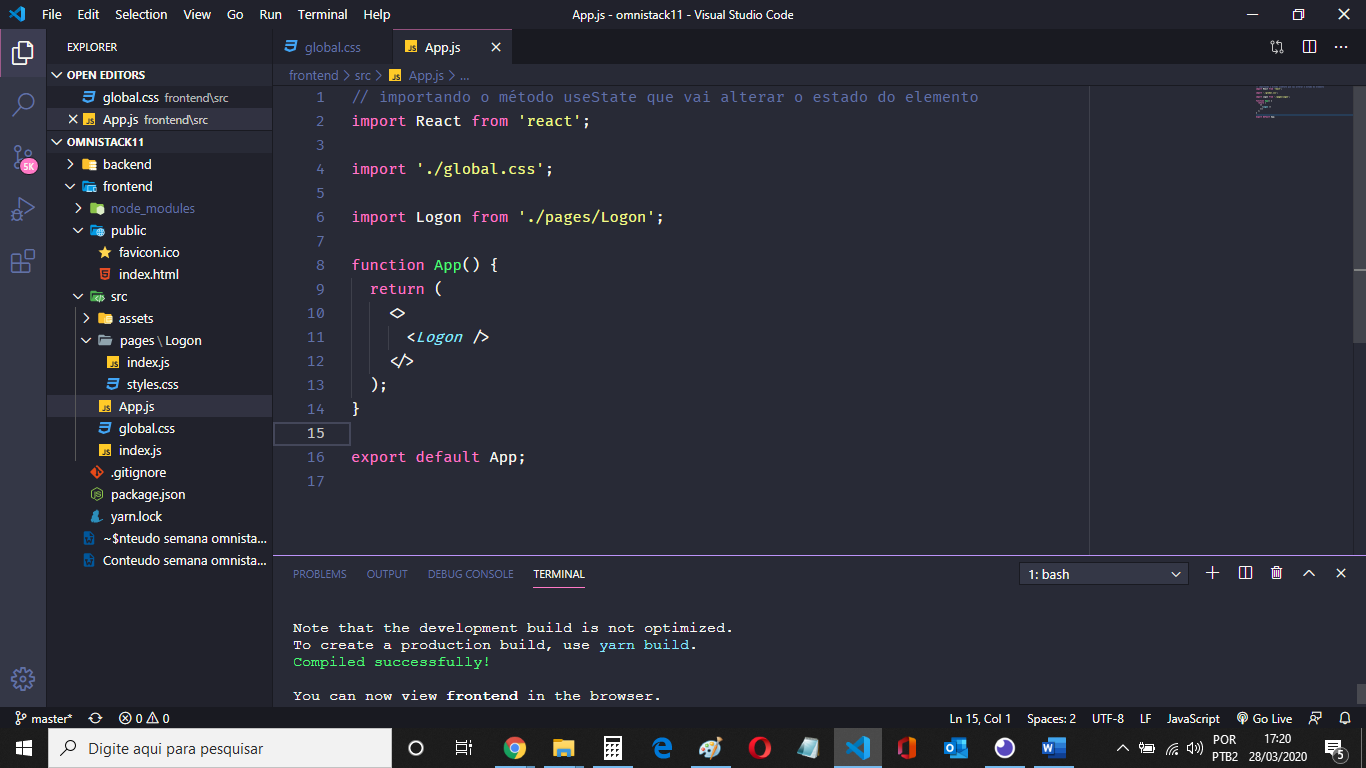
**Primeiro de tudo nesta aplicação vamos estilizar o site todo, global:**

****

**Fonte com três tamanhos, 400, 500, 700, adquiridos no site google fonts:**

****

**E vamos importar esse css global dentro do App**



**Agora vamos para a tela de login**

**Baixar uma dependência para poder usar os icons no react**

npm install react-icons

**Importar no componente, desestruturando o icon**

**Mas infelizmente, não consegui baixar a dependência.**

**Então baixei e inclui o svg do site feathericons**

****

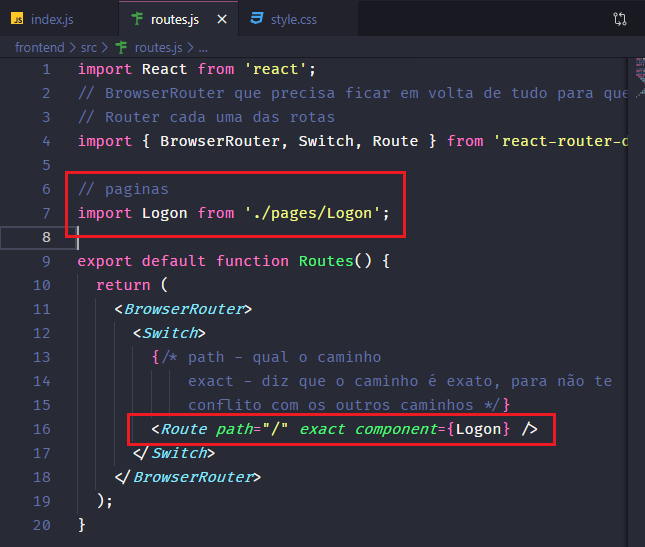
**Agora vamos criar as rotas**

**Pra isso vamos instalar o pacote**

npm install react-router-dom

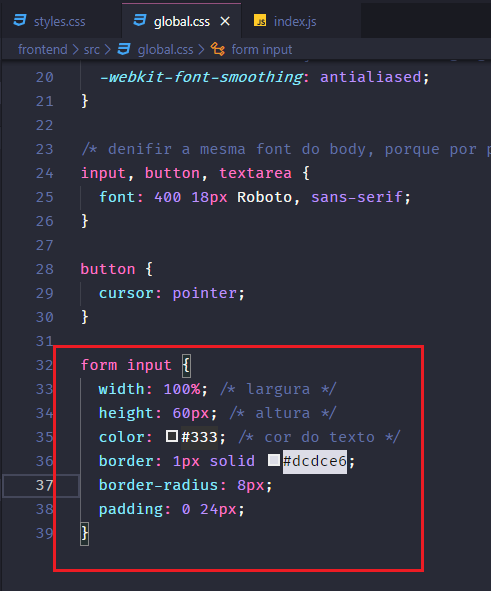
**criamos o arquivo routes.js dentro do src**

**e configuramos os caminhos das rotas:**

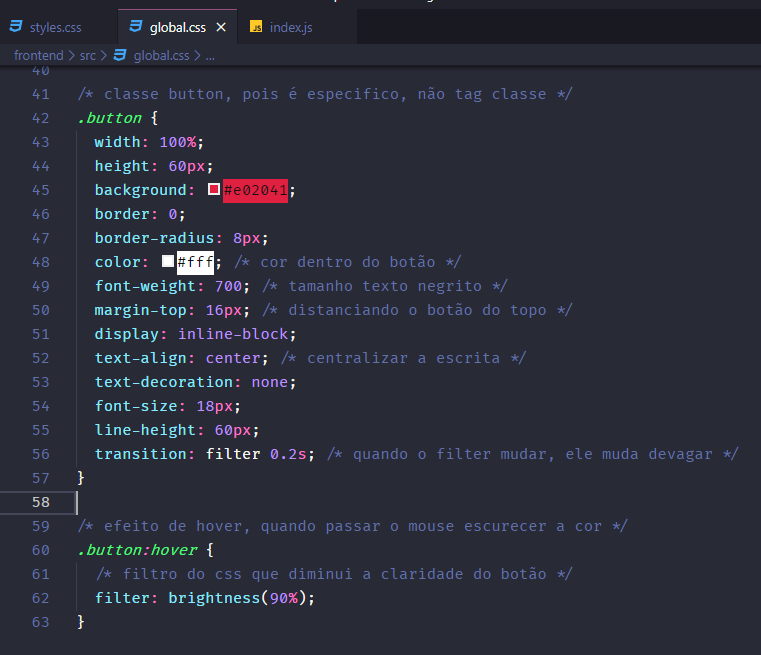
****

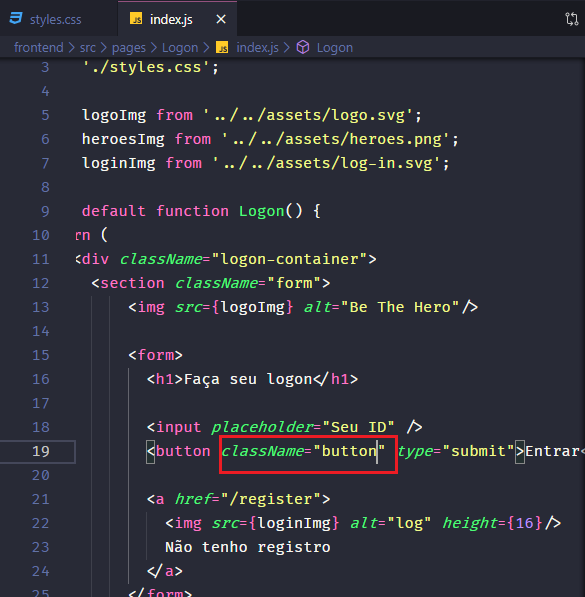
**Agora vamos estilizar**

**Colocamos o input no css global pois será usado em outras páginas com a mesma estilização:**

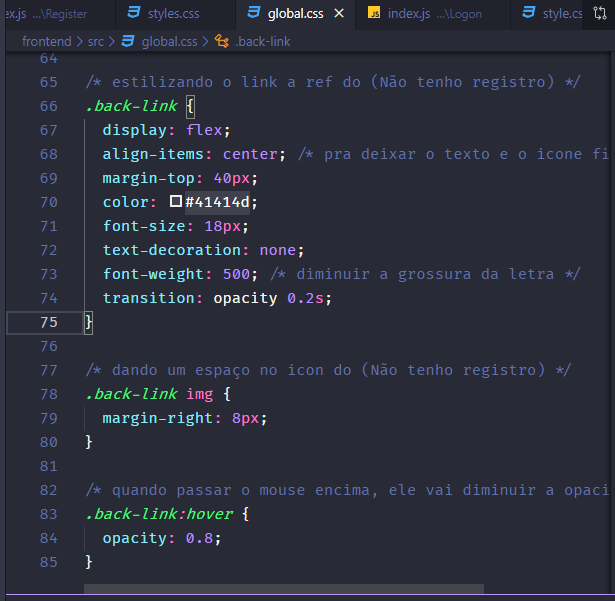
****

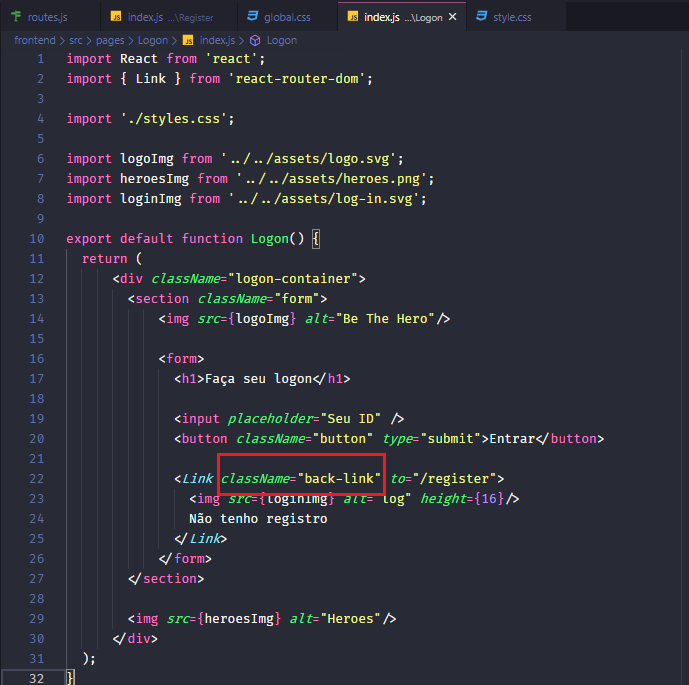
**Estilizar o botão no css global tbm**

****



**Estilizar o (Não tenho registro) no css global**

****

****

**O restante dos elementos foram estilizados no style.css**

/\* div por volta de tudo \*/

*.logon-container* {

  /\* largura \*/

  width: 100%;

  /\* largura maxima \*/

  max-width: 1120px;

  /\* ocupar a altura toda do corpo \*/

  height: 100vh;

  /\* faz com que o conteudo fique sempre centralizado \*/

  margin: 0 auto;

  /\* facilita os alinhamentos e aciona o align e justify

  e tbm faz com que os elementos fique um ao lado do outro

  flex-direction: column - deixa um embaixo do outro, caso queira \*/

  display: flex;

  /\* alinha os itens verticalmente ao centro \*/

  align-items: center;

  /\* alinha os itens horizontalmente ao centro \*/

  justify-content: space-between;

}

/\* estilizando o section.form (section a tag, form a classe) \*/

*.logon-container* section*.form* {

  width: 100%;

  max-width: 350px;

  /\* dar uma distanciada da logo \*/

  margin-right: 30px;

}

/\* estilizando o formulário \*/

*.logon-container* section*.form* form {

  /\* dar uma boa distancia da logo \*/

  margin-top: 100px;

}

/\* estilizando a tag h1 (faça seu logon) \*/

*.logon-container* section*.form* form h1 {

  font-size: 32px;

  /\* distanciando "Faça seu logon" do input \*/

  margin-bottom: 32px;

}

**Agora vamos modificar o link do (Não tenho registro), para ele fazer parte do carregamento do react**

****

**Agora vamos para página de registro dos casos**

import React from 'react';

import { Link } from 'react-router-dom';

import './style.css';

import arrowLeftImg from '../../assets/arrow-left.svg';

import logoImg from '../../assets/logo.svg';

export default function Register() {

  return (

    <div *className*="register-container">

      <div *className*="content">

        <section>

          <img *src*={logoImg} *alt*="Be The Hero" />

          <h1>Cadastro</h1>

          <p>Faça seu cadastro, entre na plataforma e ajude

          pessoas a encontrarem os casos da sua ONG.

          </p>

          <*Link* *className*="back-link" *to*="/">

            <img *src*={arrowLeftImg} *alt*="log" *height*={16} />

            Não tenho registro

          </*Link*>

        </section>

        <form>

          <input *placeholder*="Nome da ONG" />

          <input *type*="email" *placeholder*="E-mail" />

          <input *placeholder*="Whatsapp" />

          <div *className*="input-group">

            <input *placeholder*="Cidade" />

            <input *placeholder*="UF" *style*={{ width: 80 }} />

          </div>

          <button *className*="button" *type*="submit">Cadastrar</button>

        </form>

      </div>

    </div>

  );

}

**Stilização da página register**

*.register-container* {

  /\* largura \*/

  width: 100%;

  /\* largura maxima \*/

  max-width: 1120px;

  /\* ocupar a altura toda do corpo \*/

  height: 100vh;

  /\* faz com que o conteudo fique sempre centralizado \*/

  margin: 0 auto;

  /\* facilita os alinhamentos e aciona o align e justify

  e tbm faz com que os elementos fique um ao lado do outro

  flex-direction: column - deixa um embaixo do outro, caso queira \*/

  display: flex;

  /\* alinha os itens verticalmente ao centro \*/

  align-items: center;

  /\* alinha os itens horizontalmente ao centro \*/

  justify-content: center;

}

/\* div content, pega toda a aplicação  \*/

*.register-container* *.content* {

  width: 100%;

  padding: 96px;

  background: #f0f0f5;

  /\* 0 0 - distancia do eixo x e y, cor preto com 10% de opacidade \*/

  box-shadow: 0 0 100px rgba(0, 0, 0, 0.1);

  border-radius: 8px;

  display: flex;

  /\* space-between - vai jogar a section totalmente para

  esquerda e o formulário totalmente para direita e vai

  por um espaço entre ele \*/

  justify-content: space-between;

  align-items: center;

}

/\* parte da esqueda  \*/

*.register-container* *.content* section {

  width: 100%;

  max-width: 380px;

}

/\* h1 (cadastro)  \*/

*.register-container* *.content* section h1 {

  /\* 64px encima, 0 nas laterais e 32px enbaixo \*/

  margin: 64px 0 32px;

  font-size: 32px;

}

/\* paragrafo (texto) \*/

*.register-container* *.content* section p {

  font-size: 18px;

  color: #737380;

  line-height: 32px;

}

/\* formulario, parte da direita \*/

*.register-container* *.content* form {

  width: 100%;

  max-width: 450px;

}

/\* separar os inputs do formulario \*/

*.register-container* *.content* form input {

  margin-top: 8px;

}

/\* input cidade e uf \*/

*.register-container* *.content* form *.input-group* {

  display: flex;

}

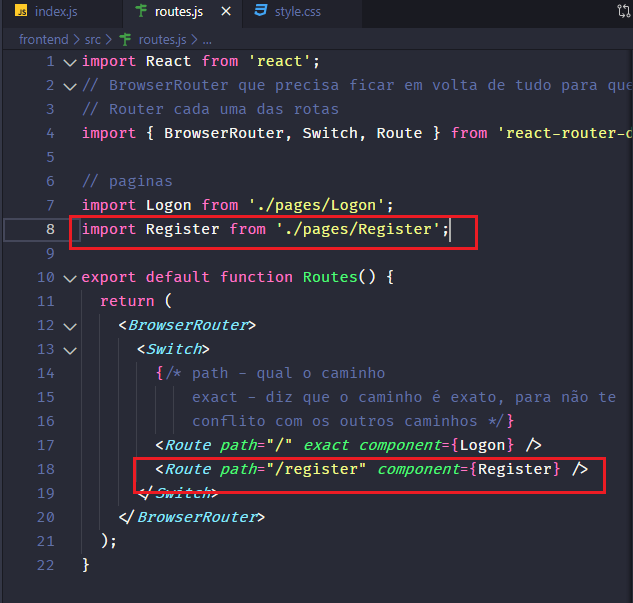
/\* input + input - adicionando style pra todo input a partir do segundo input \*/

*.register-container* *.content* form *.input-group* input + input {

  margin-top: 8px;

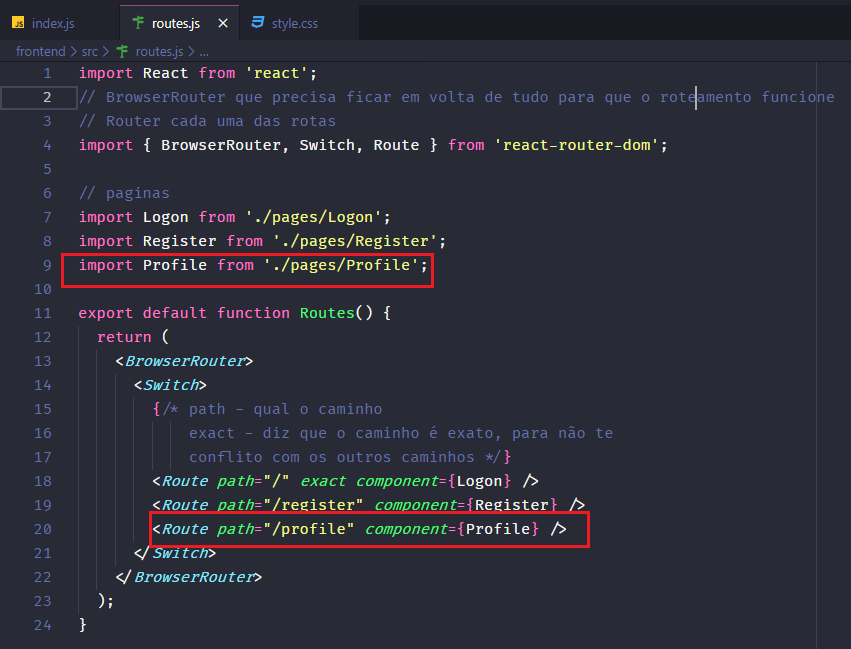
}

**Criando rota register**

****

**Agora vamos criar a página listar os casos**

**Criando rota**

****

**Component Profile**

import React from 'react';

import { Link } from 'react-router-dom';

import './styles.css';

import powerImg from '../../assets/power.svg';

import trash2Img from '../../assets/trash2.svg';

import logoImg from '../../assets/logo.svg';

export default function Profile() {

  return (

    <div *className*="profile-container">

      <header>

      <img *src*={logoImg} *alt*="Be The Hero"/>

      <span>Bem vinda, APAD</span>

      <*Link* *className*="button" *to*="/incidents/new">Cadastrar novo caso</*Link*>

      <button *type*="button">

        <img *className*="imgLogout" *src*={powerImg} *alt*="power" />

      </button>

      </header>

      <h1>Casos cadastrados</h1>

            <ul>

        <li>

          <strong>CASO:</strong>

          <p>Caso teste</p>

          <strong>DESCRIÇÃO:</strong>

          <p>Descrição teste</p>

          <strong>VALOR:</strong>

          <p>R$ 120,00</p>

          <button *type*="button">

            <img *src*={trash2Img} *alt*="trash2" *height*={20} />

          </button>

        </li>

        <li>

          <strong>CASO:</strong>

          <p>Caso teste</p>

          <strong>DESCRIÇÃO:</strong>

          <p>Descrição teste</p>

          <strong>VALOR:</strong>

          <p>R$ 120,00</p>

          <button *type*="button">

            <img *src*={trash2Img} *alt*="trash2" *height*={20} />

          </button>

        </li>

        <li>

          <strong>CASO:</strong>

          <p>Caso teste</p>

          <strong>DESCRIÇÃO:</strong>

          <p>Descrição teste</p>

          <strong>VALOR:</strong>

          <p>R$ 120,00</p>

          <button *type*="button">

            <img *src*={trash2Img} *alt*="trash2" *height*={20} />

          </button>

        </li>

        <li>

          <strong>CASO:</strong>

          <p>Caso teste</p>

          <strong>DESCRIÇÃO:</strong>

          <p>Descrição teste</p>

          <strong>VALOR:</strong>

          <p>R$ 120,00</p>

          <button *type*="button">

            <img *src*={trash2Img} *alt*="trash2" *height*={20} />

          </button>

        </li>

      </ul>

    </div>

  );

}

**Style do component Profile**

*.profile-container* {

  width: 100%;

  max-width: 1180px;

  padding: 0 30px;

  margin: 32px auto;

}

*.profile-container* header {

  display: flex;

  align-items: center;

}

*.profile-container* header span {

  font-size: 20px;

  margin-left: 24px;

}

*.profile-container* header img {

  height: 64px;

}

*.profile-container* header *.imgLogout* {

  height: 20px;

}

*.profile-container* header a {

  width: 260px;

  margin-left: auto;

  margin-top: 0;

}

*.profile-container* header button {

  height: 60px;

  width: 60px;

  border-radius: 4px;

  border: 1px solid #dcdce6;

  background: transparent;

  margin-left: 16px;

  transition: border-color 0.2s;

}

*.profile-container* header button*:hover* {

  border-color: #999;

}

*.profile-container* h1 {

  margin-top: 80px;

  margin-bottom: 24px;

}

*.profile-container* ul {

  display: grid;

  grid-template-columns: repeat(2, 1fr);

  grid-gap: 24px;

  list-style: none;

}

*.profile-container* ul li {

  background: #fff;

  padding: 24px;

  border-radius: 8px;

  position: relative;

}

*.profile-container* ul li button {

  position: absolute;

  right: 24px;

  top: 24px;

  border: 0;

}

*.profile-container* ul li button*:hover* {

  opacity: 0.8;

}

*.profile-container* ul li strong {

  display: block;

  margin-bottom: 16px;

  color: #41414d;

}

*.profile-container* ul li p + strong {

  margin-top: 32px;

}

*.profile-container* ul li p {

  color: #737380;

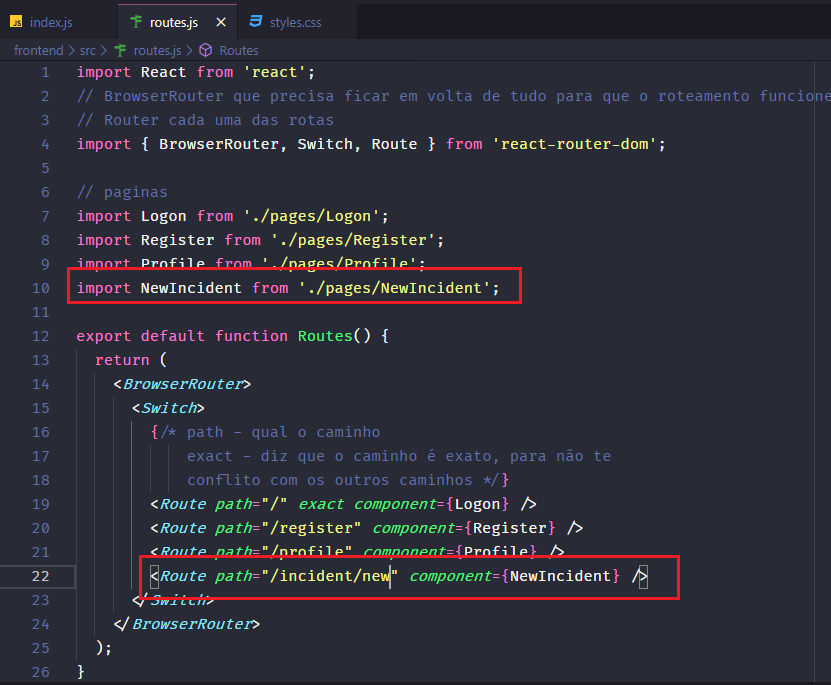
  line-height: 21px;

  font-size: 16px;

}

**Agora vamos criar a página para cadastro dos casos**

**Criando rota incident/new**

****

**Pagina do cadastro dos casos**

import React from 'react';

import { Link } from 'react-router-dom';

import './styles.css';

import arrowLeftImg from '../../assets/arrow-left.svg';

import logoImg from '../../assets/logo.svg';

export default function NewIncident() {

  return (

    <div *className*="new-incident-container">

      <div *className*="content">

        <section>

          <img *src*={logoImg} *alt*="Be The Hero" />

          <h1>Cadastrar novo caso</h1>

          <p>Descreva o caso detalhadamente para encontrar um herói para

            resolver isso.

          </p>

          <*Link* *className*="back-link" *to*="/profile">

            <img *src*={arrowLeftImg} *alt*="arrow-left" *height*={16} />

            Voltar para casos

          </*Link*>

        </section>

        <form>

          <input *placeholder*="Título do caso" />

          <textarea *placeholder*="Descrição" />

          <input *placeholder*="Valor em reais" />

          <button *className*="button" *type*="submit">Cadastrar</button>

        </form>

      </div>

    </div>

  );

}

**Style da página de cadastro dos casos**

*.new-incident-container* {

  /\* largura \*/

  width: 100%;

  /\* largura maxima \*/

  max-width: 1120px;

  /\* ocupar a altura toda do corpo \*/

  height: 100vh;

  /\* faz com que o conteudo fique sempre centralizado \*/

  margin: 0 auto;

  /\* facilita os alinhamentos e aciona o align e justify

  e tbm faz com que os elementos fique um ao lado do outro

  flex-direction: column - deixa um embaixo do outro, caso queira \*/

  display: flex;

  /\* alinha os itens verticalmente ao centro \*/

  align-items: center;

  /\* alinha os itens horizontalmente ao centro \*/

  justify-content: center;

}

/\* div content, pega toda a aplicação  \*/

*.new-incident-container* *.content* {

  width: 100%;

  padding: 96px;

  background: #f0f0f5;

  /\* 0 0 - distancia do eixo x e y, cor preto com 10% de opacidade \*/

  box-shadow: 0 0 100px rgba(0, 0, 0, 0.1);

  border-radius: 8px;

  display: flex;

  /\* space-between - vai jogar a section totalmente para

  esquerda e o formulário totalmente para direita e vai

  por um espaço entre ele \*/

  justify-content: space-between;

  align-items: center;

}

/\* parte da esqueda  \*/

*.new-incident-container* *.content* section {

  width: 100%;

  max-width: 380px;

}

/\* h1 (cadastro)  \*/

*.new-incident-container* *.content* section h1 {

  /\* 64px encima, 0 nas laterais e 32px enbaixo \*/

  margin: 64px 0 32px;

  font-size: 32px;

}

/\* paragrafo (texto) \*/

*.new-incident-container* *.content* section p {

  font-size: 18px;

  color: #737380;

  line-height: 32px;

}

/\* formulario, parte da direita \*/

*.new-incident-container* *.content* form {

  width: 100%;

  max-width: 450px;

}

/\* separar os inputs do formulario \*/

*.new-incident-container* *.content* form input,

*.new-incident-container* *.content* form textarea {

  margin-top: 8px;

}

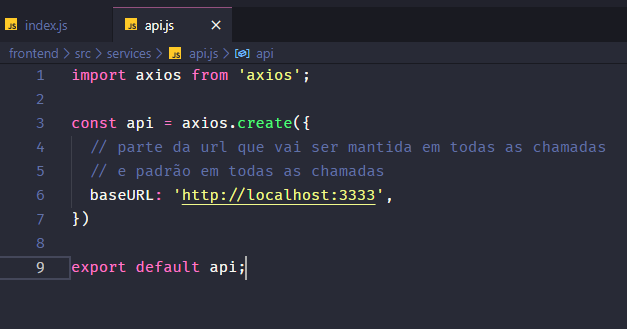
**Agora vamos conectar o backend ao frontend**

**Baixar o pacote cliente HTTP, AXIOS, responsável por fazer as chamadas a api do backend e obter as respostas**

npm install axios

**vamos criar uma pasta chamada services com o arquivo api.js**

**services é todo tipo de arquivo que vai prover algum tipo de integração com algum serviço externo que é esse caso.**

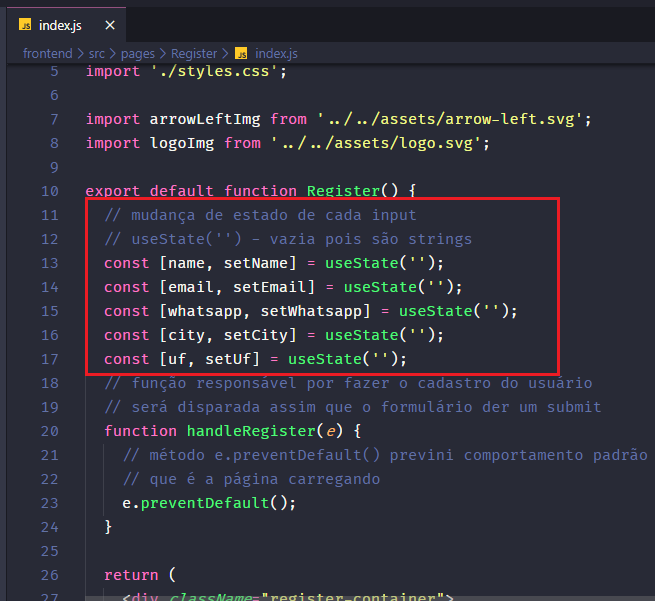
****

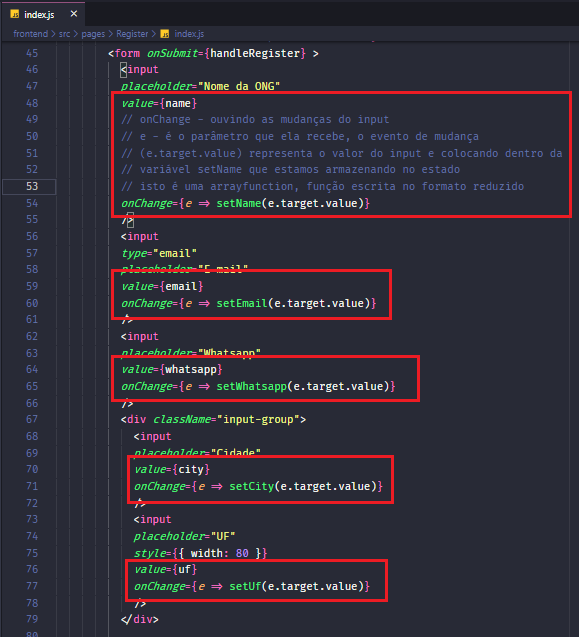
**Vamos começar a primeira integração com a parte de cadastro das ONGs**

**Criando uma função para o formulário enviar os dados através dos inputs para a api criada em services**

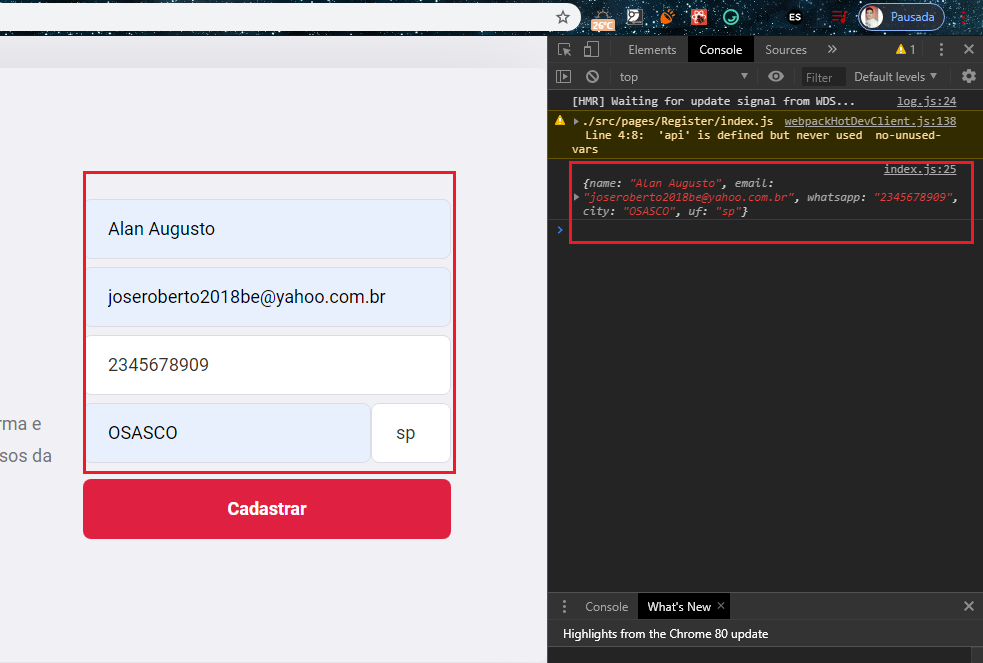
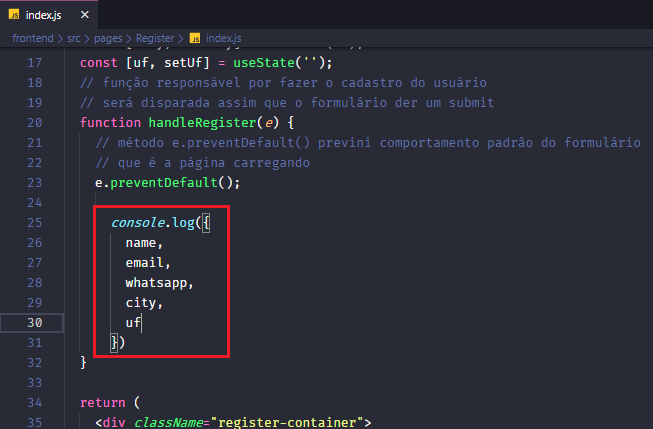
****

**Agora vamos preparar os inputs para enviar os dados**

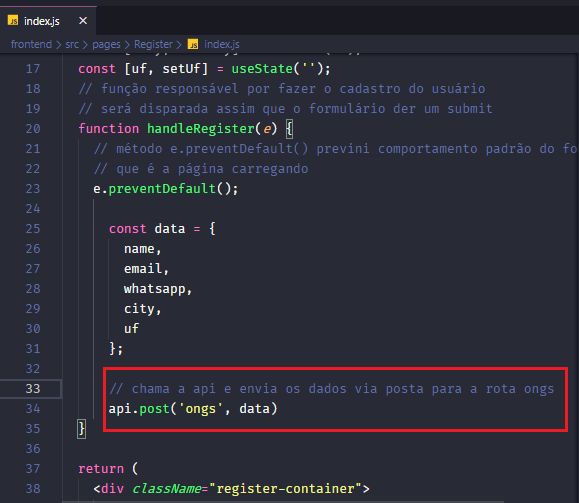
**Existem várias estratégias e vamos usar um “state” pra isso:**

****

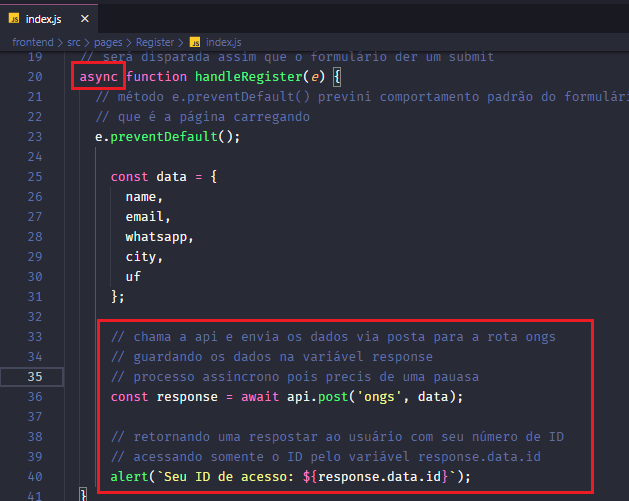
**Agora os dados já estão sendo guardados, confira abaixo pelo console:**

****

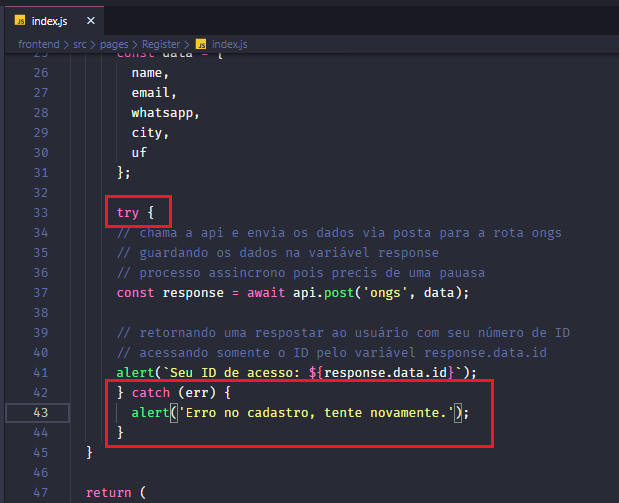
**Dando continuidade, vamos guardar estes dados em uma variável e enviar para a api enviar ao banco**

****

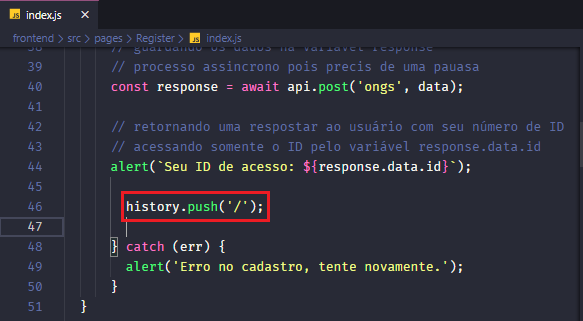
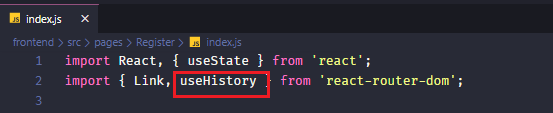
**Em seguida colocamos uma resposta ao usuário com o número de seu ID**

****

**Se isso falhar, mostramos um erro ao usuário:**

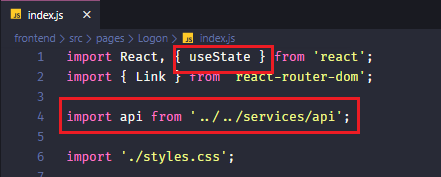
****

**Depois do cadastro, vamos enviar o usuário para página de login, importando a função useHistory, que serve para fazermos a navegação através de uma função javascript quando não podemos colocar o “Link” do react-router-dom, então depois o “alert” colocamos history.push(‘/’) para enviar para rota raiz.**

****

**Agora vamos para página de Logon para deixar o login funcional**

**Importo o useState e api**

****

**Criamos um “estado” id troca para setId.**

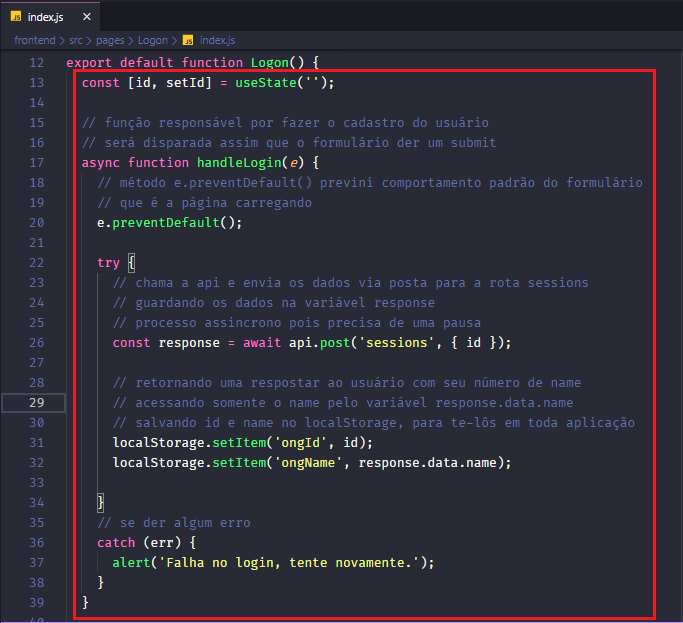
**Passamos a função preventDefault para não carregar a página.**

**Try – para retornar um erro.**

**Chamo api via post e guardo os dados do input na rota sessions.**

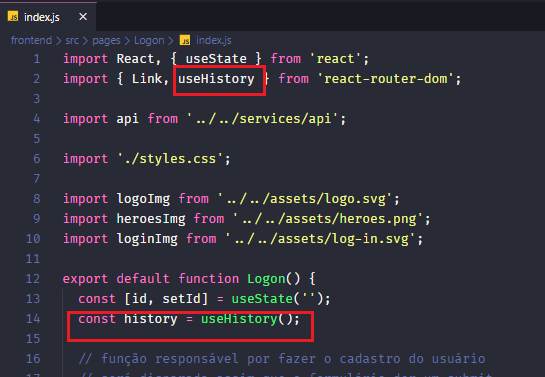
**Passamos o id da ong que esta fazendo a operação.**

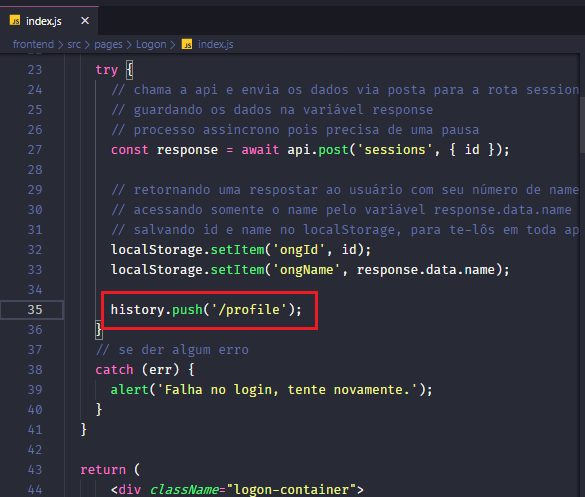
**Guardamos id e name no localStore para te-lôs em toda a aplicação.**

****

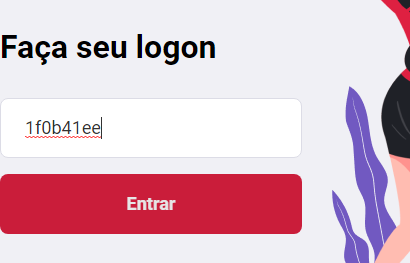
**Depois do login, vamos enviar o usuário para página profile, onde temos a lista dos casos**

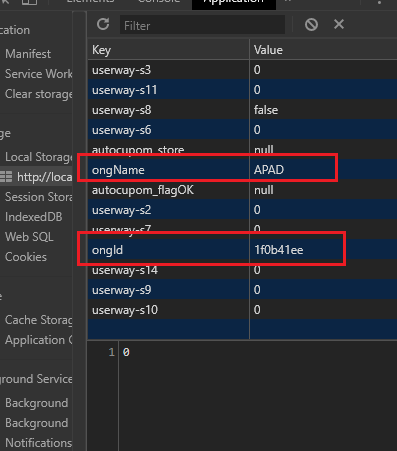
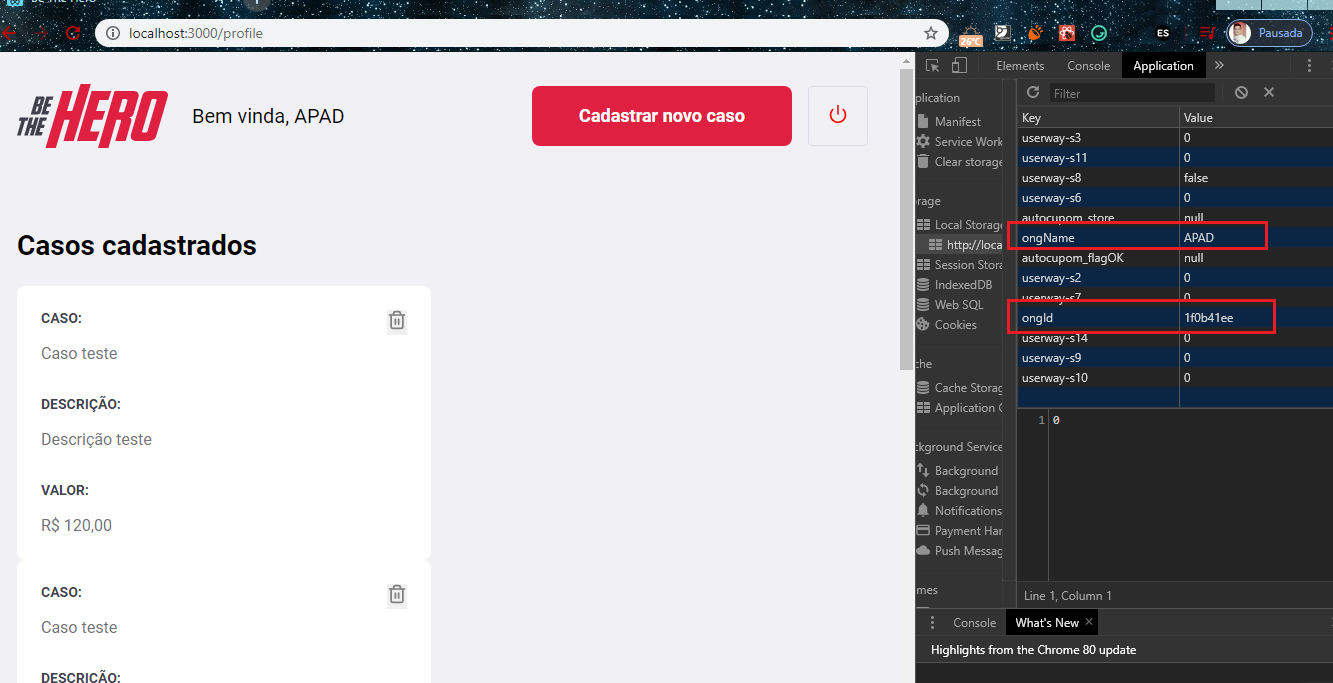
**Vamos importar o history e configurá-lo:**

****

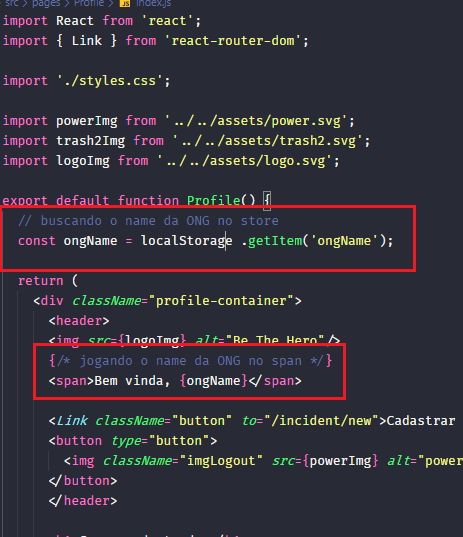
****

**Vamos fazer um teste de login ? Bora!**

****

****

**Agora vamos colocar o nome da ong em profile, onde estão os casos dela**

****

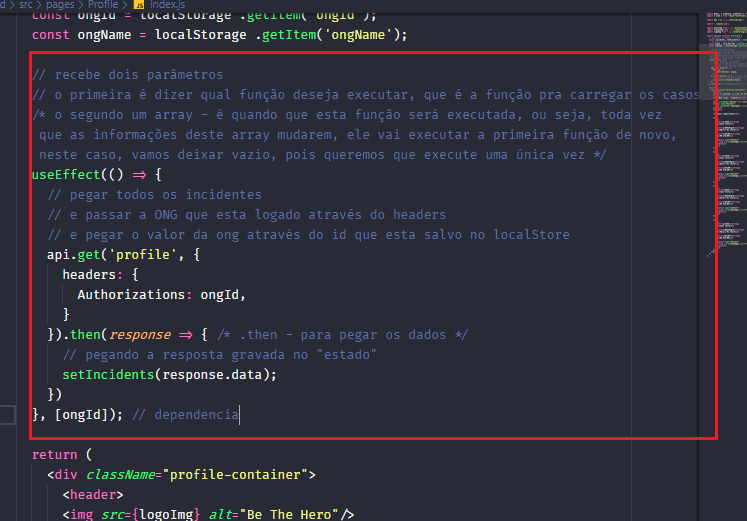
**Agora vamos listar os casos cadastrados pela ONG**

**Usando a função “useEffect”, que serve para disparar uma função em determinado momento do component, exemplo: assim que ele mostrado em tela:**

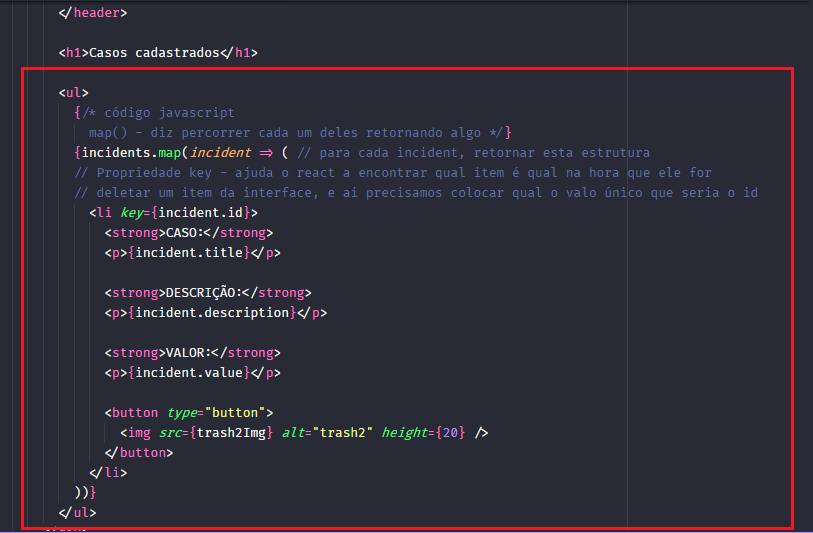
**Usando o “useState” para guardar os dados e realizar a troca.**

**Usando a “api” para trazer a listagem de casos através do “id” da ONG**

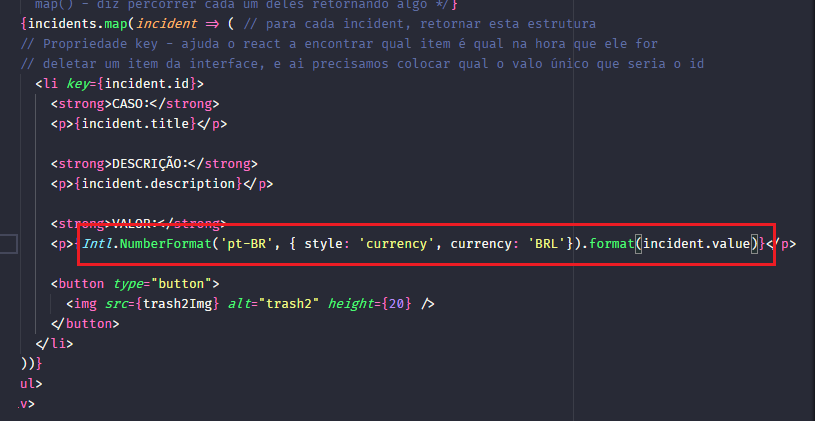
****

****

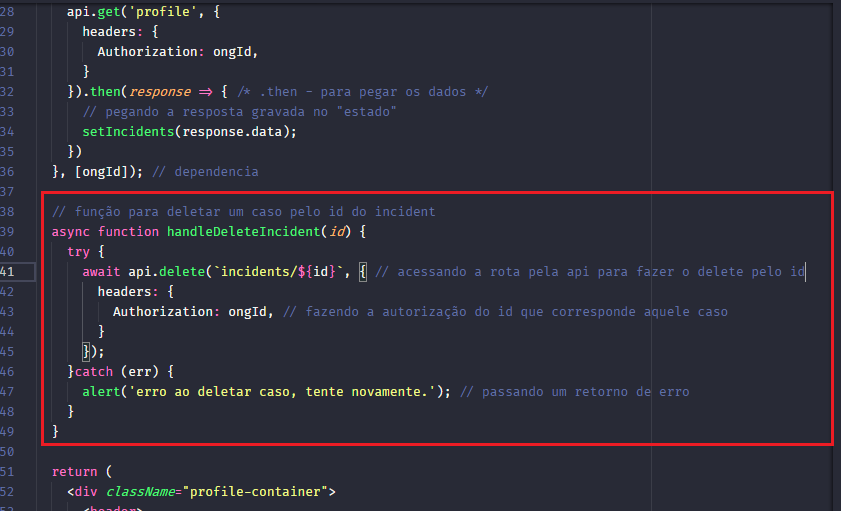
**Agora vamos preparar os casos para receber os dados:**

****

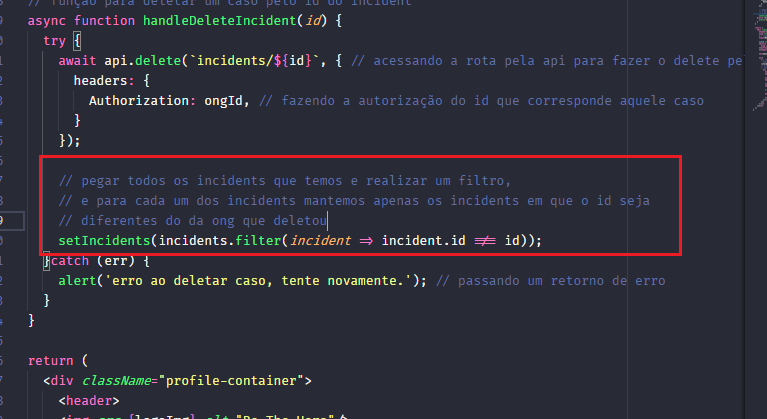
**Agora vamos formatar valor em real**

****

**Agora vamos deletar um caso**

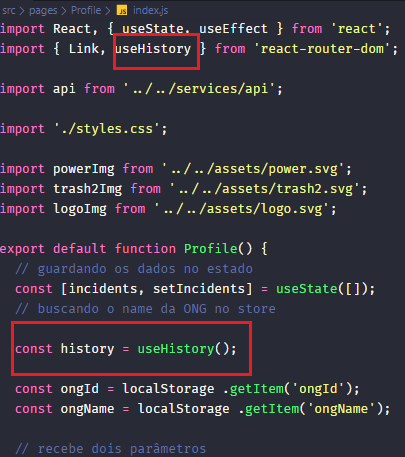
****

**Precisando chamar uma função para deletar o caso em tempo real**

****

**Agora vamos fazer o logout**

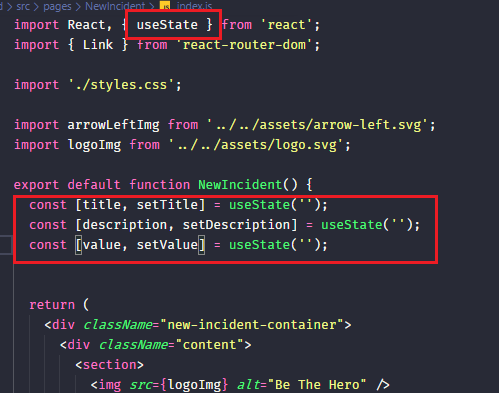
**Importando e configuramos o useHistory**

****

**Criamos uma função e passamos no botão do logout**

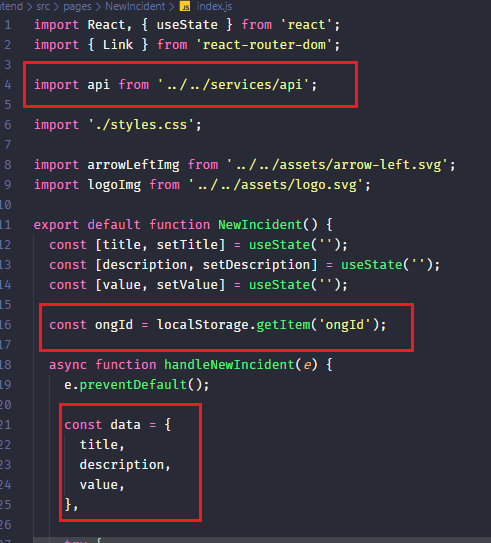
**Agora vamos para página de cadastro de uma caso**

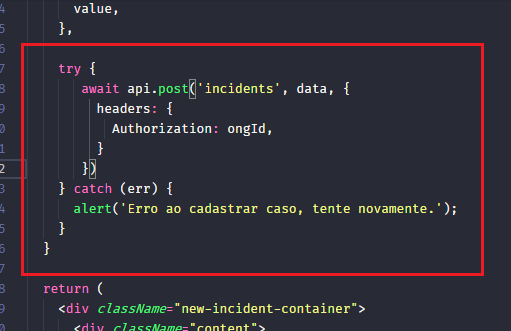
**Importamos o useState e criamos os estados de cada input**

****

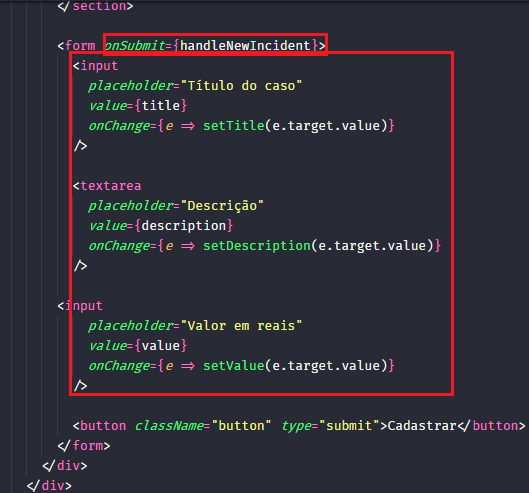
**Importar o api.**

**Buscar o id no storage.**

****

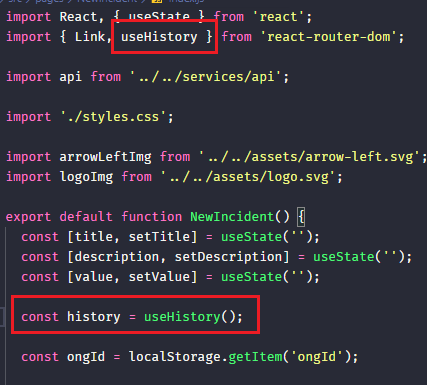
****

**Configuramos o formulário para receber a função e os inputs para receber e enviar os dados via api para o banco**

****

**Depois de cadastrar vamos retorná-lo para página do perfil**

**Importamos e configuramos o useHistory**

****

**E navegamos o usuário para página profile**

****