Obteção do API, que traz informações de filmes e videos pra alimentar o site.

* <https://developers.themoviedb.org/3/getting-started/introduction>

Criação do app no react -> npx create-react-app NomeDoApp No cmd executado como adm.

Criação do projeto no firebase, basicamente só seguir os passos sem muito segredo e todar este código dentro da pasta do projeto

* npm install -g firebase-tools
* cd na pasta do projeto react onde está o package.json e npm start

Agora precisamos fazer uma limpeza dos arquivos que o react traz:

App.test.js

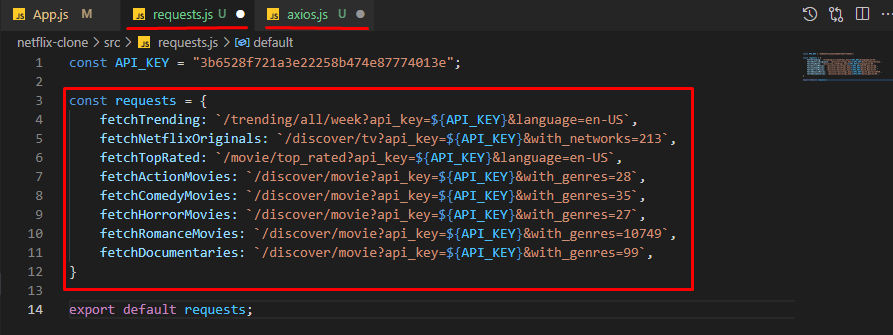
Logo.svg

setupTests.js

Dentro do app.js apagar Header

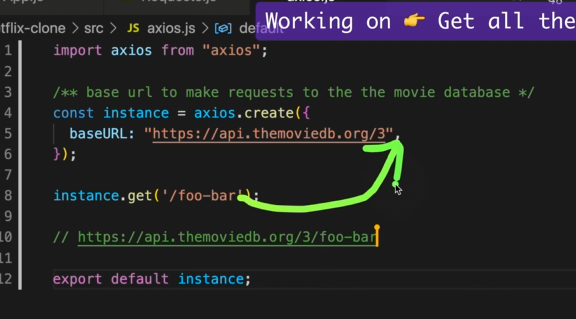
Para puxar todos dados do API é necessário rodar NPM i axios

Para fazer os requests deve-se criar um arquivo chamado requests.js



Obs: para nome de arquivos uteis não usamos primeira letra maiúscula, isso se faz geralmente com componentes.

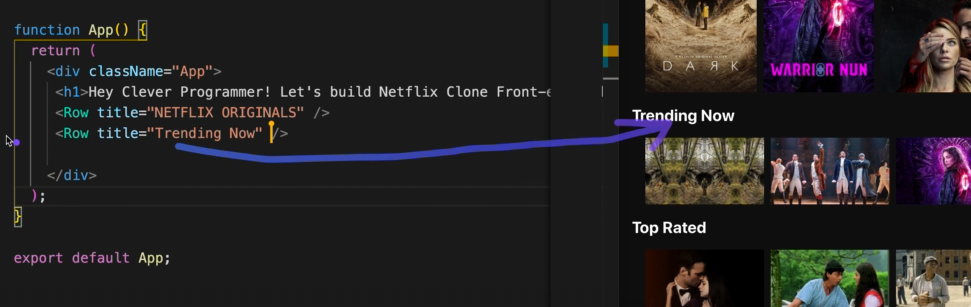
Cria-se o arquivo axios.js com o seguinte codigo



Ele pega o baseURL e faz um append com quaisquer get que seja feito em cima dele.

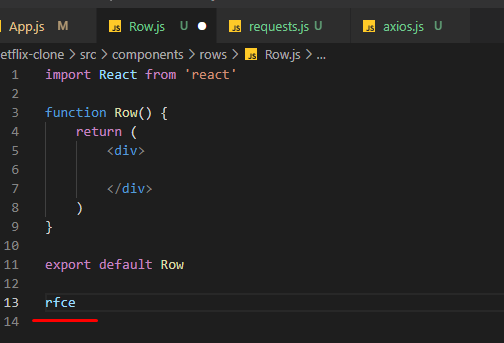
Agora criaremos as linhas de cada tópico de filmes e serie do site, por exemplo **top rated, originals, action movies etc.** Criaremos os componentes pra cada filme/serie/doc passando as informações dele(titulo, foto, duração etc) através de props(Properties).

**Exemplo:**

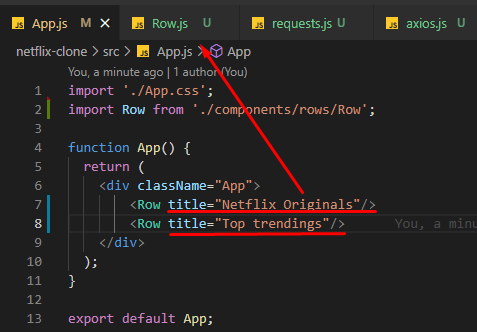


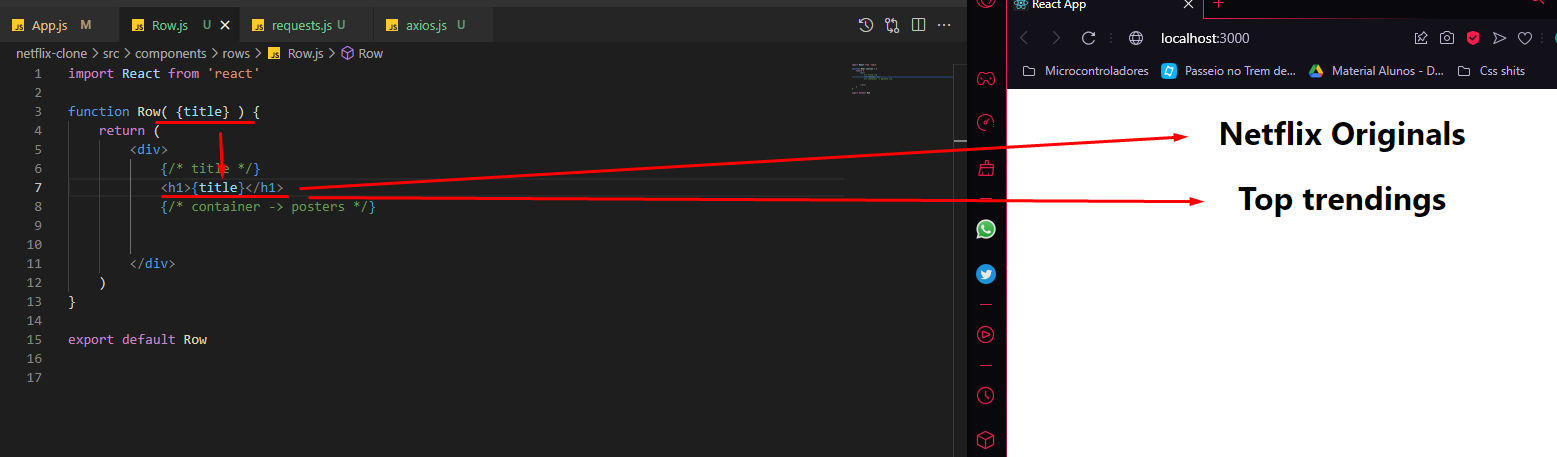
Criamos então o componente “Row.js” dentro de uma pasta components. Primeira letra maiuscula por ser componente

Usando a extensão ES7 para snippet(Atalho pra criar classe) escrevemos rfce:



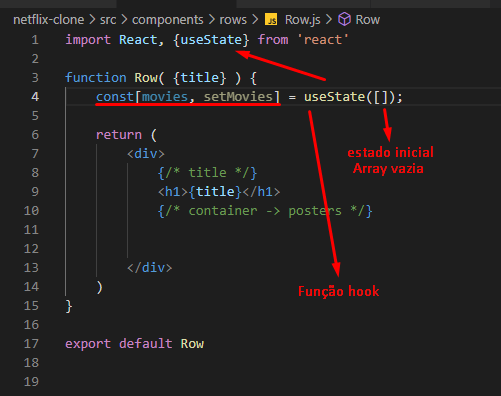
# Passando props para um componente:



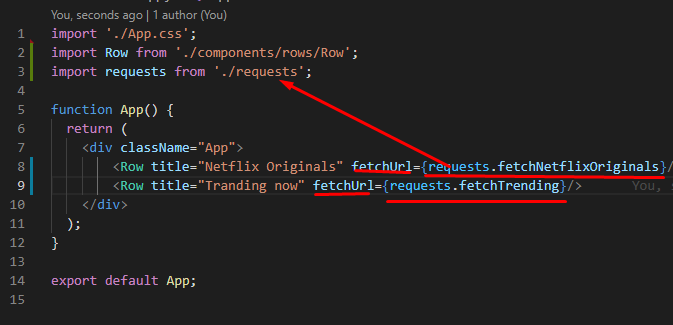


# Criando um state

Um state é feito para guardar memória temporária em uma constante:



# Usando o fetch para puxar dados do request.js



# Usando o useEffect com snippet

useEffect(() =>{

CÓDIGO A SER RODADO QUANDO O COMPONENTE CARREGAR

}, [ ]);

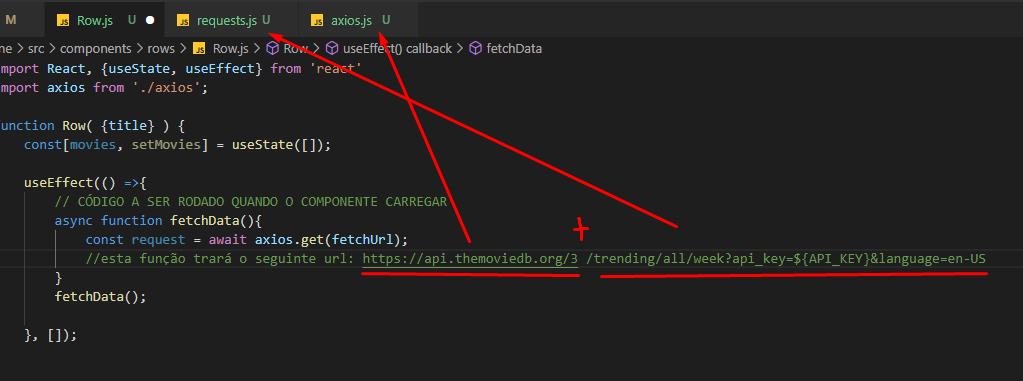
O que o useEffect faz? Dentro de sua arrow function terá um código que deverá ser rodado sempre que o componente carregar no site. Os colchetes vazios servem para dizer para o programa rodar o código apenas uma vez, quando o componente carregar, e não rodar mais. Se eu colocar alguma propriedade dentro dos colchetes ex [movies], o código rodará sempre que a constante movies mudar.

# Importando o axios do axios.js para usar as informações passadas pelo API:

OBS: no arquivo axios.js criamos a const instance e exportamos, mas não precisamos usar ou importar especificamente o instance, pois so temos uma exportação, e ao importar o axios.js ele ja importa e usa o instance como axios.

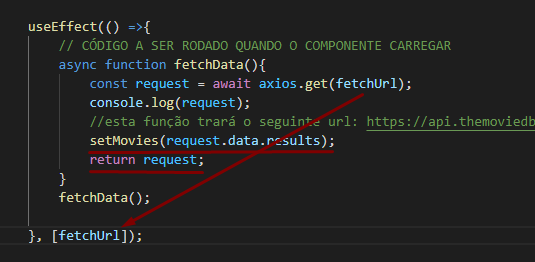
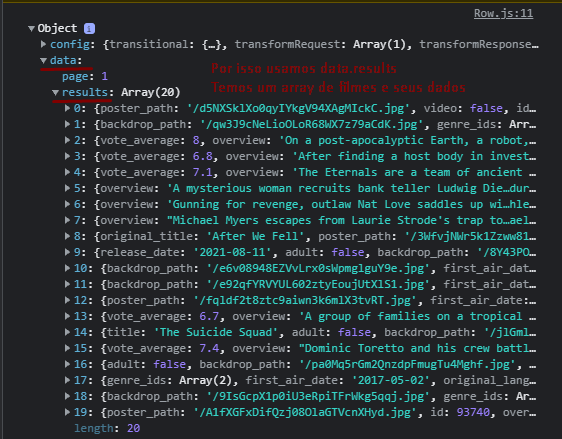
# Criando um async function para puxar dados assíncronos(ao solicitados demoram alguns segundos para chegar, por isso se chama async function):

O await manda o progama esperar o arquivo ou a solicitação chegar para seguir.



Após, podemos dar um console.log(request) e veremos no console que ele pega os dados dos filmes.

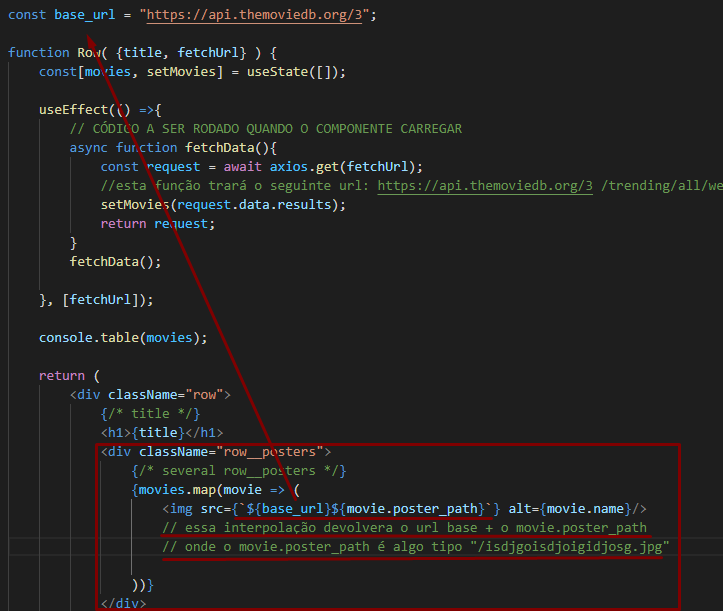
Então podemos setar esses esses resultados, pois no console vimos onde estão os dados dos filmes, no request.data.results. E então retornar requests.



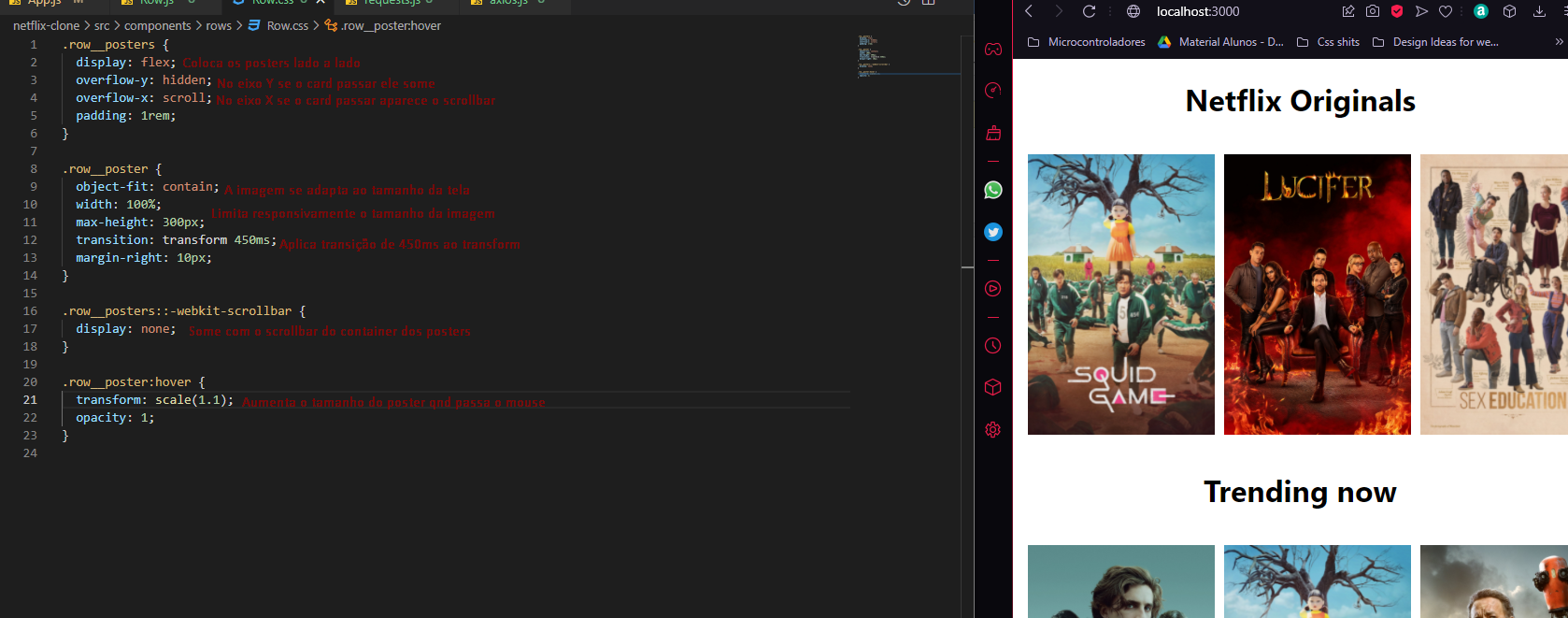
Obs: Muito importante, quando usar um valor de fora do bloco do async function, como no caso o fetchUrl, que vem como parametro da function Row, temos q passar o valor q pegamos para o [ ] do useEffect, se nao n funfará

Agora vamos criar uma div geral, das linhas onde vai os filmes, e uma div dos posters, individual.

A primeira coisa q faremos para testar puxar os dados é pegar a img do poster, então dentro da div de poster criamos uma tag de img e passamos como src uma base\_url interpolado com o movie.poster\_path. Esse movie.poster\_patch é uma propriedade do API, ele traz cada poster em uma string com .jpg no fim. Por isso é necessário o base\_url, q é passado no arquivo axios, somado a url passando pelo api teremos uma image com sucesso:



# Estilizando um cadin



# Otimização do render:

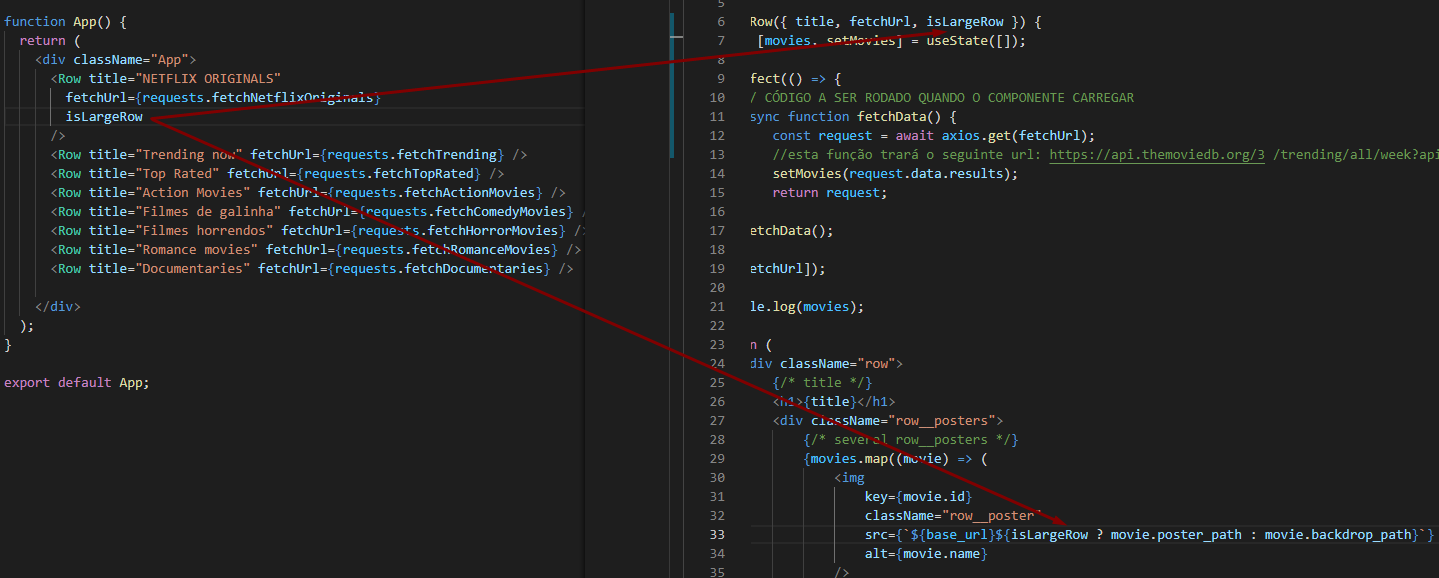
Esta propriedade key identifica cada imagem passada com um id(passado pelo proprio API nesse caso) e ao fazer isso ele força o effect a renderizar apenas certa img de certo id, e não toda página.



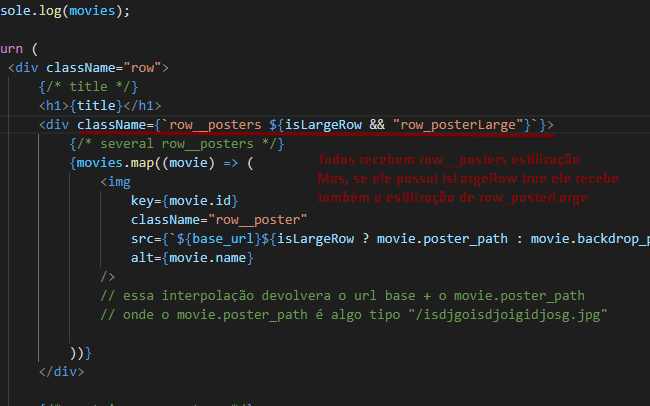
Isto agiliza um pouco o site

# Passando o prop “isLargeRow”

Vai servir para renderizar o estilo baseado nessa propriedade

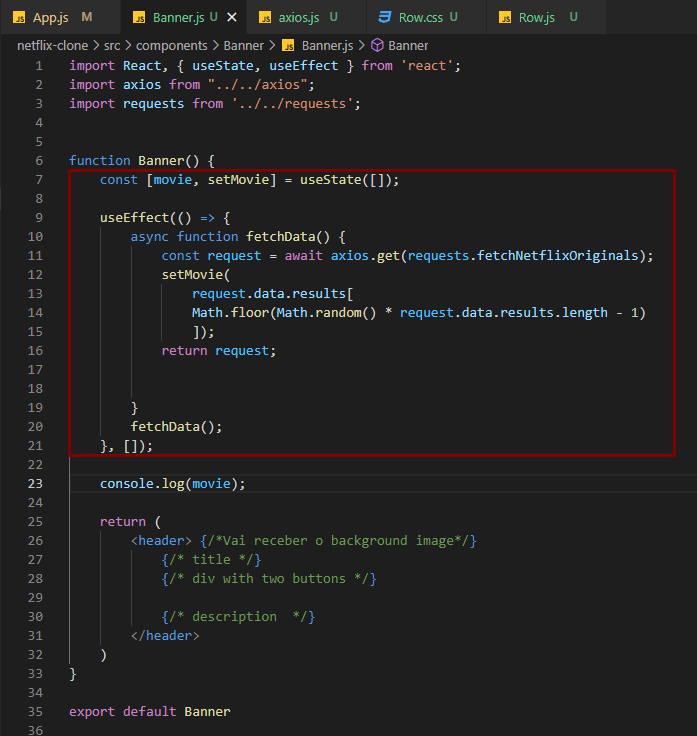


Agora usaremos a seguinte manobra pra aplicar 2 estilização condicionada ao prop isLargeRow



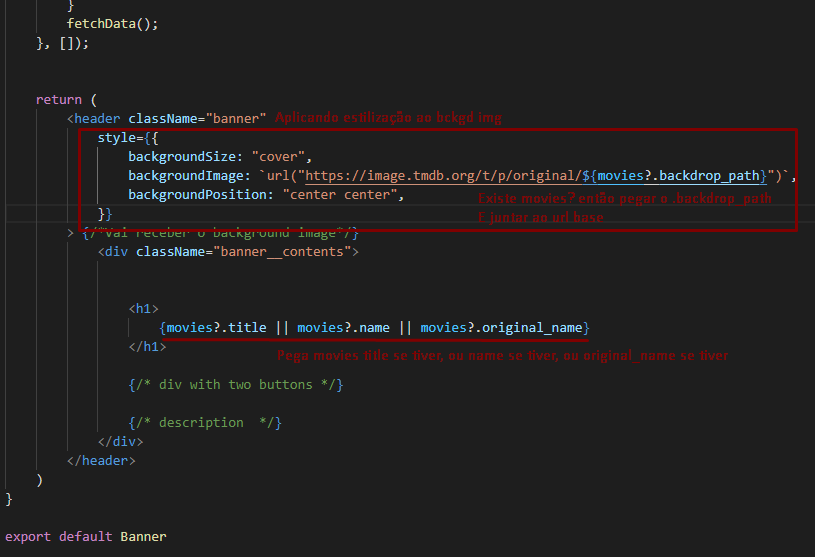
So q no caso printei errado, esse codigo sublinhado tem q estar no classname do img e n do div

# Criando o banner



Agora vamos criar o bannerzao que aparece em cima, pra isso vamos criar uma const movie para puxar os dados de algum filme, aleatoriamente, para puxar aleatoriamente vamos meter no setMovie direto algum valor do array resquest.data.results usando a formula Math.floor(Math.random() \* resquest.data.results.length -1) isso vai pegar qlqr valor entre 0 e o ultimo valor desse array, sorteando assim qlqr filme do array NetflixOriginals

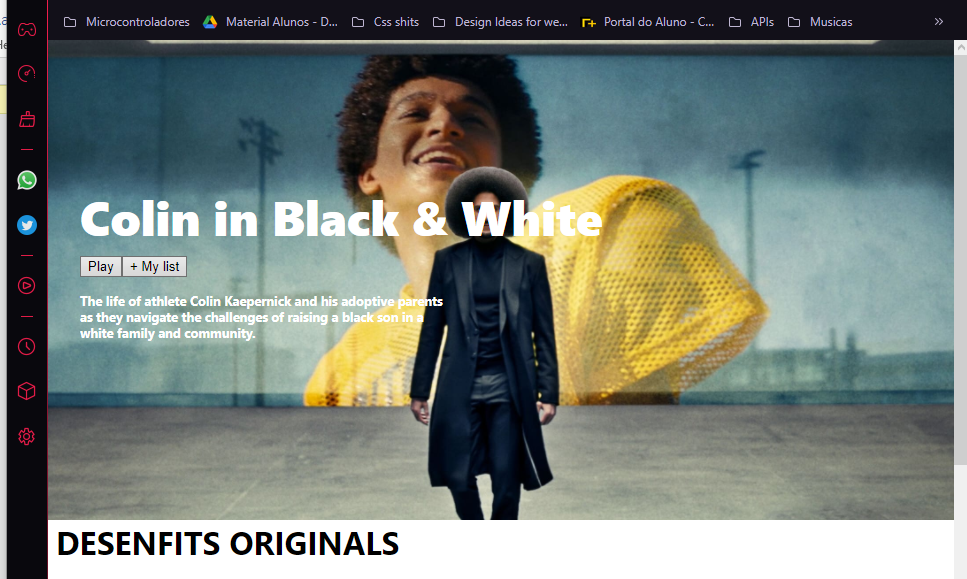
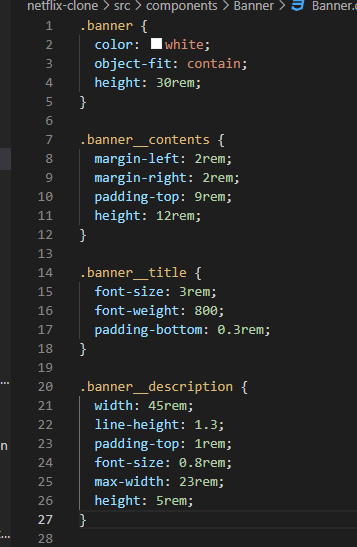
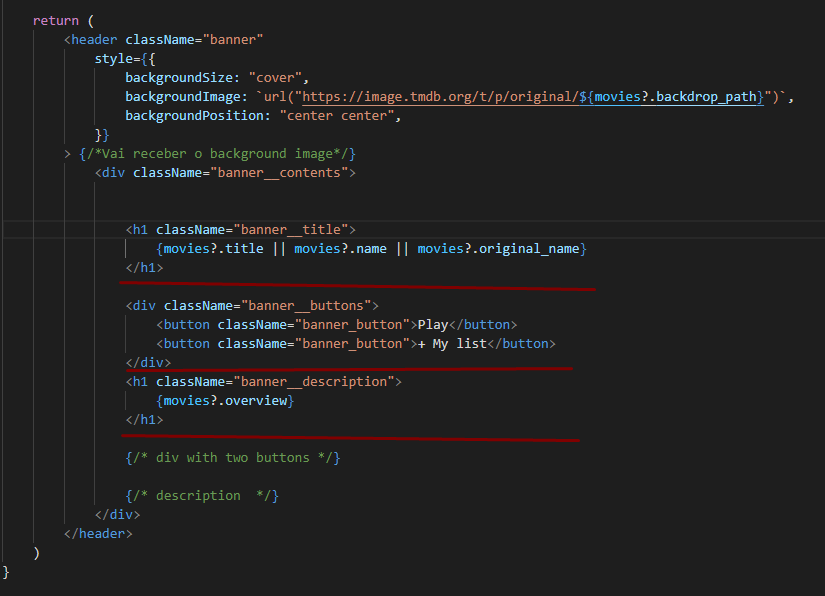
# Setando e estilizando o background image do banner



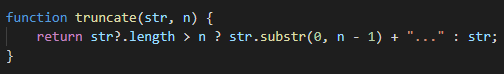
Aplicado style direto no arquivo JS pra poder puxar o movies.backdrop\_path (Gambiarra)

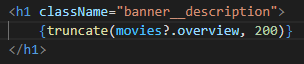
No h1 passa movies title ou name ou original\_name, dependendo de qual existir, depende do API, pois as vzs um API pode ter passado o titulo do filme de maneiras diferentes em campos diferentes.

# Estilizando o banner

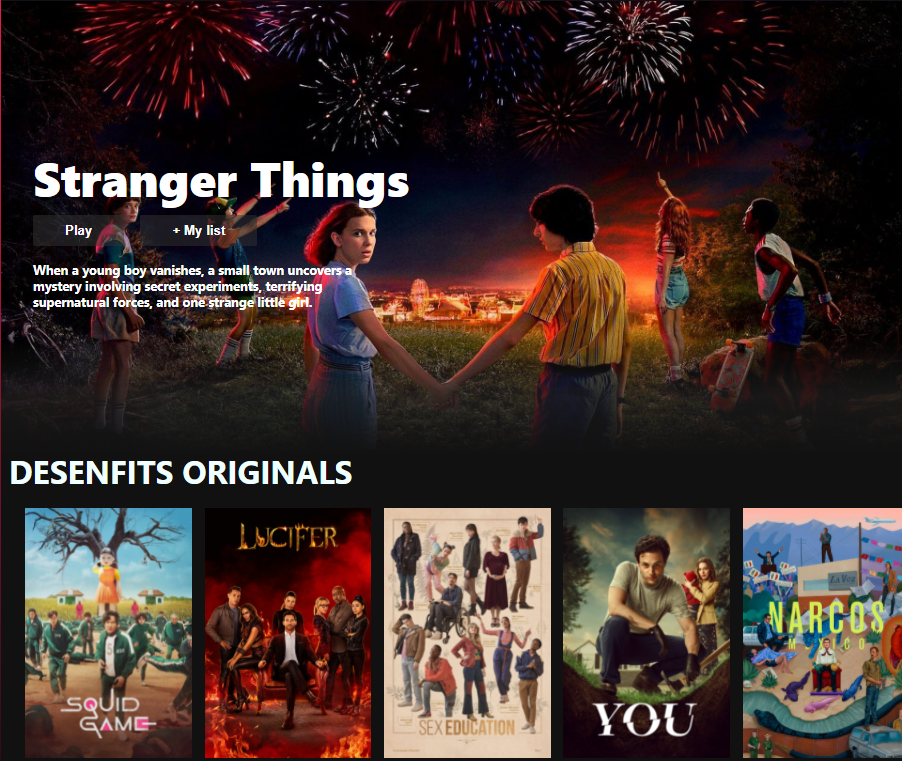
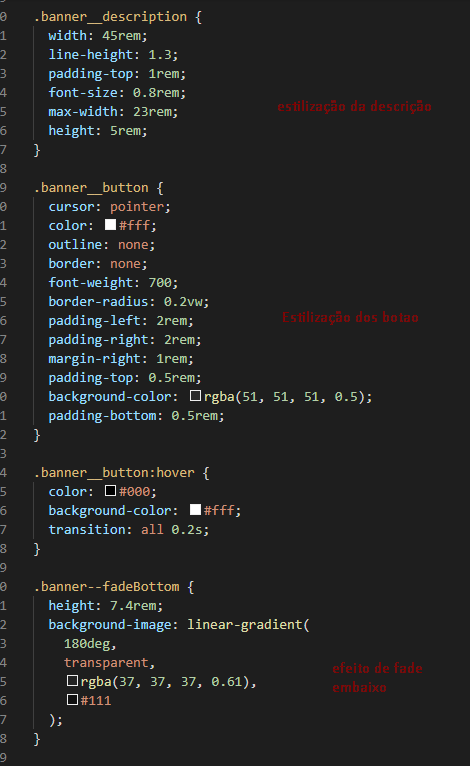


Apliquei truncate pra por “...” qnd tiver mais de 200 caracteres na descrição :

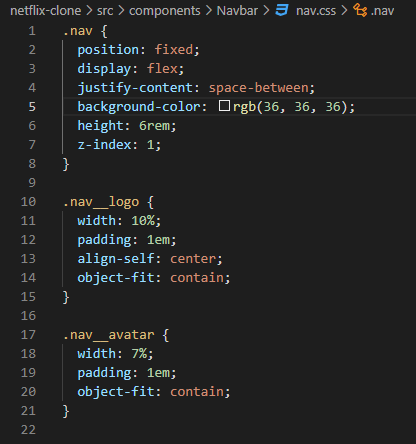


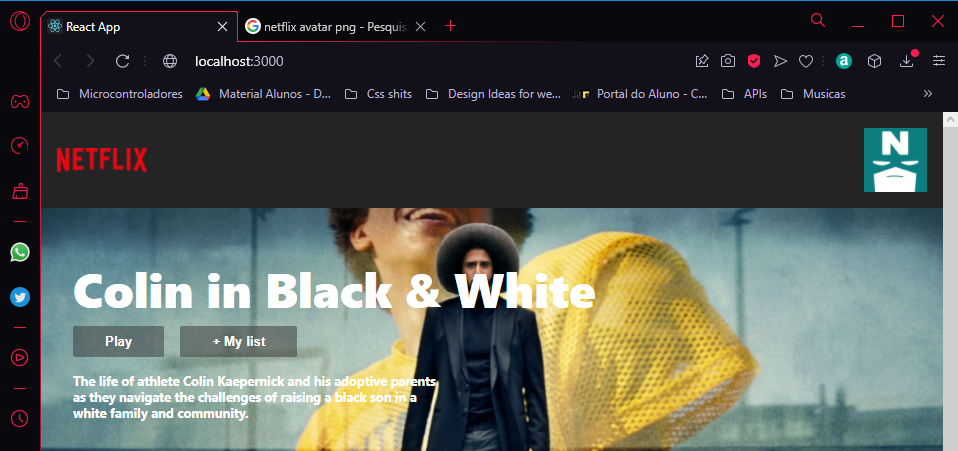


# Mais estilização



Criando o Navbar. É simples, estilização tranquila, eu fiz sem seguir o tutorial e ficou melhor e mais prático que o deles:

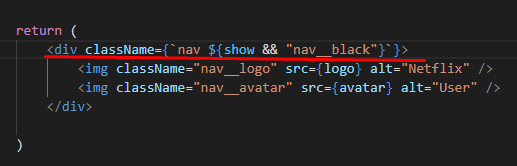




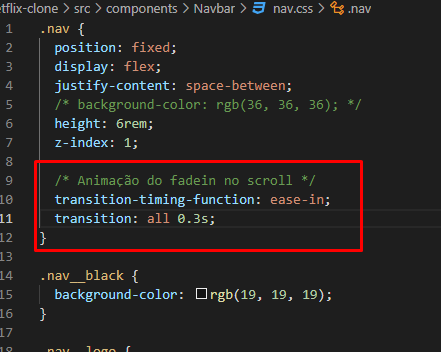
Agora vamos criar um scroll listener que vai servir para nao ter o nav no topo, e qnd scrollar pra baixo ele aparecer em efeito fade-in. Usaremos useEffect, uma função EventListener, consegue “ouvir” no parametro windows o scroll, logo, o que essa função basicamente diz é: Quando scrollar no eixo Y por 100 pixels faça tal coisa .



Este return serve para ele remover o listener após feito o que deve ser feito, para que o site n fique aplicando listener o tempo todo.



Agora, se show for true, ou seja, o usuario ter scrolado mais de 100px no site, aplique, além da estilização nav, a estilização nav\_\_black.



E por fim aplicar animação para gerar esse fadein, este código diz para qualquer mudança feita aplicar animação

# Inserindo os trailers dos programas, usando as bibliotecas REACT YOUTUBE e MOVIE TRAILER

Para instalar: npm i react-youtube e npm i movie-trailer

import YouTube from 'react-youtube';

Criar a const opts:

const opts = {

        height: '390',

        width: '640',

        playerVars: {

            // https://developers.google.com/youtube/player\_parameters

            autoplay: 1,

        },

    };

E chamar o component YouTube:

<YouTube videoId={trailerUrl} opts={opts} />

Lembrando que é YouTube com Y e T maiusculo kkkkk

    {/\* Essa parte diz, quando tiver um trailerUrl dai mostrar o COMPONENTE \*/}

                    {/\* YouTube, se n n mostra \*/}

            {trailerUrl && <YouTube videoId={trailerUrl} opts={opts} />}

Agora vamos fazer com que apareca o trailer qnd clicar na imagem do filme e desapareça qnd clicar de novo:

Cria-se essa função:

const handleClick = (movie) =>{

        if(trailerUrl){

            setTrailerUrl('');

        } else{

            movieTrailer(movie?.name || "")

            .then((url)=>{

                // https://www.youtube.com/watch?v=MZb2uCFNjwQ

                const UrlParams = new URLSearchParams(new URL(url).search);

                setTrailerUrl(UrlParams.get('v'));

            }).catch(error => console.log(error));

        }

    }

Ele vai pegar a parte do URL do youtube a partir do ‘v’ usando essa funçãoURLSearchParams e essa gambiarra toda ai em cima.