\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Aula Página Login**

Criar as páginas Gênero, Filmes, Perfil

No arquivo routers.tsx acrescentar as rotas para estas páginas criadas

Alterar o componente Button para Input do tipo Submit para uso mais adequado do Form:

import React from 'react';

import './style.css';

interface ButtonProps {

value: string;

}

const Button: React.FC<ButtonProps> = ({ value }) => {

return (

<input className="button" type="submit" value={value} />

);

}

export default Button;

CSS - Button

.button{

width: 180px;

height: 58px;

background-color: #F6511D;

border: 0;

color: #ffff;

}

Na página Login:

Acrescentar useState no import

import React, { useState } from 'react';

Acessando a API para realizar o login

function Login() {

//O **useHistory** é a um componente que é usado dentro de uma página para levar o usuário para outra rota do site. Para usá-la, deve-se primeiro instancia-la em uma variável, depois usar a função push e colocar a rota que você quer.

**let history = useHistory();**

//**UseState** é uma forma de definir e atualizar o estado “interno ” de um componente

const [email, setEmail] = useState('');

const [senha, setSenha] = useState('')

//criando função de acesso aos dados da API para realização do login

const login = () => {

const login = {

email: email,

senha: senha

}

//utilizando fetch para acesso à API, determinando o método, body e headers

fetch('http://localhost:5000/api/conta/login', {

method: 'POST',

body: JSON.stringify(login),

headers: {

'content-type': 'application/json'

}

})

//promises

.then(response => response.json())

.then(dados => {

.then (dados => {

//Verificação do token

if(dados.token != undefined)

{

//A funcionalidade do localStorage consiste em salvar, adicionar, recuperar ou excluir dados localmente em um navegador Web.

//Esta informação é guardada na forma de pares de chave-valor e os valores podem ser apenas strings.

//métodos do armazenamento local: setItem(chave, valor)-armazena, getItem(chave)-Recupera, removeItem(chave)-Remove

localStorage.setItem('token-filmes',dados.token)

**//levando para rota (desejada) de quando o usuário estiver logado**

history.push('/');

}else{

alert('Senha ou Email incorretos! Tente novamente! :(')

}

})

.catch(err=>console.error(err))

}

Trabalhando o retorno da Página Login

return (

<div className="principal">

<Header description="Faça o login e acesse a Coletânea" />

<main>

<div className="centro">

<div>

<h1>Login</h1>

//No form , o evento **onsubmit** é um evento que acontece quando se submete o formulário, ao clicar num botão submit . Pode-se então executar uma ou mais functions,e realizar as validações que deseja.

<form onSubmit={event => {

// preventDefault() impede que o evento padrão ocorra, seguir um link, por exemplo

event.preventDefault();

//executa a função login

login();

}}

//O evento onChange é utilizado para que seja realizada a ação após alguma mudança no input.O **e** é o evento change, **target** é o elemento que acionou o evento, que neste caso é o input, e **value** é o valor do inputElement

<Input name="email" label="E-mail" onChange={e => setEmail(e.target.value)} /><br/>

<Input type="password" name="senha" label="Senha" onChange={e => setSenha(e.target.value)} /><br/>

<div className="btn">

<Button value="Enviar" />

</div>

</form>

</div>

</div>

</main>

<Footer />

</div>

);

}