Fernanda Fretes

FrontEnd

FrontEnd - React

Fernanda Fretes

Consumo de API

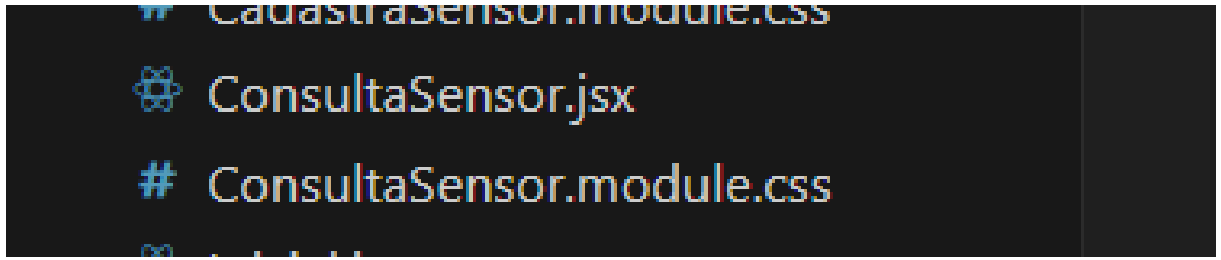
Durante as aulas, nós já fizemos o consumo de API, conectando ao The Movie DB.

Mas para fazer o uso da API criada na aula de Backend para o projeto integrador, já podemos também fazer o uso da mesma lógica.

Vou deixar um exemplo ainda no nosso projeto, para dar ideia no projeto integrador.

FrontEnd - React

Fernanda Fretes



CadastraSensor.jsx | ConsultaSensor.jsx | ConsultaSensor.module.css | index.

src > paginas > ConsultaSensor.jsx > ConsultaSensor > useEffect() callback > fetchSensores

```jsx
import React, { useEffect, useState } from 'react';
import estilos from './ConsultaSensor.module.css';

export function ConsultaSensor() {
    const [sensores, setSensores] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);

```

FrontEnd - React

Fernanda Fretes

Consumo de API

Criando um novo arquivo em páginas, usar para fazer o consumo da API.

- sensores: Armazena a lista de sensores.
- loading: Indica se os dados estão sendo carregados.
- error: Armazena possíveis erros que ocorram durante a busca dos dados, começando com NULL.

FrontEnd - React

Fernanda Fretes

```
useEffect(() => {
    async function fetchSensores() {

        try {
            const response = await fetch('http://localhost:8000/api/sensores/', {
                method: 'GET',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNzE2M
                },
            });

            if (!response.ok) {
                throw new Error('Erro ao buscar os sensores');
            }

            const data = await response.json();
            setSensores(data);
            setLoading(false);
        } catch (error) {
            setError(error.message);
            setLoading(false);
        }
    }

    fetchSensores();
}, []);
```

FrontEnd - React

Fernanda Fretes

Consumo de API

- Obtém um token de acesso do localStorage.
- Faz uma requisição GET para a API de sensores.
- Se a resposta for bem-sucedida, converte os dados para JSON e os armazena no estado sensores.
- Se houver um erro, armazena a mensagem de erro no estado error.
- Em ambos os casos, define loading como false após a tentativa de busca.

Fro

Fernanda Fretes

```
    if (loading) {
        return <p>Carregando...</p>;
    }
    if (error) {
        return <p>{error}</p>;
    }
    return (
        <div className={estilos.conteiner}>
            <h1>Lista de Sensores</h1>
            <ul className={estilos.item}>
                {sensores.map(sensor => (
                    <li key={sensor.id} >
                        <p><strong>Tipo:</strong> {sensor.tipo}</p>
                        <p>MAC Address: {sensor.mac_address}</p>
                        <p>Latitude: {sensor.latitude}</p>
                        <p>Longitude: {sensor.longitude}</p>
                        <p>Localização: {sensor.localizacao}</p>
                        <p>Responsável: {sensor.responsavel}</p>
                        <p>Unidade de Medida: {sensor.unidade_medida}</p>
                        <p>Status Operacional: {sensor.status_operacional ? 'Ativo' : 'Inativo'}</p>
                        <p>Observação: {sensor.observacao}</p>
                    </li>
                ))}
            </ul>
        </div>
    );
```

FrontEnd - React

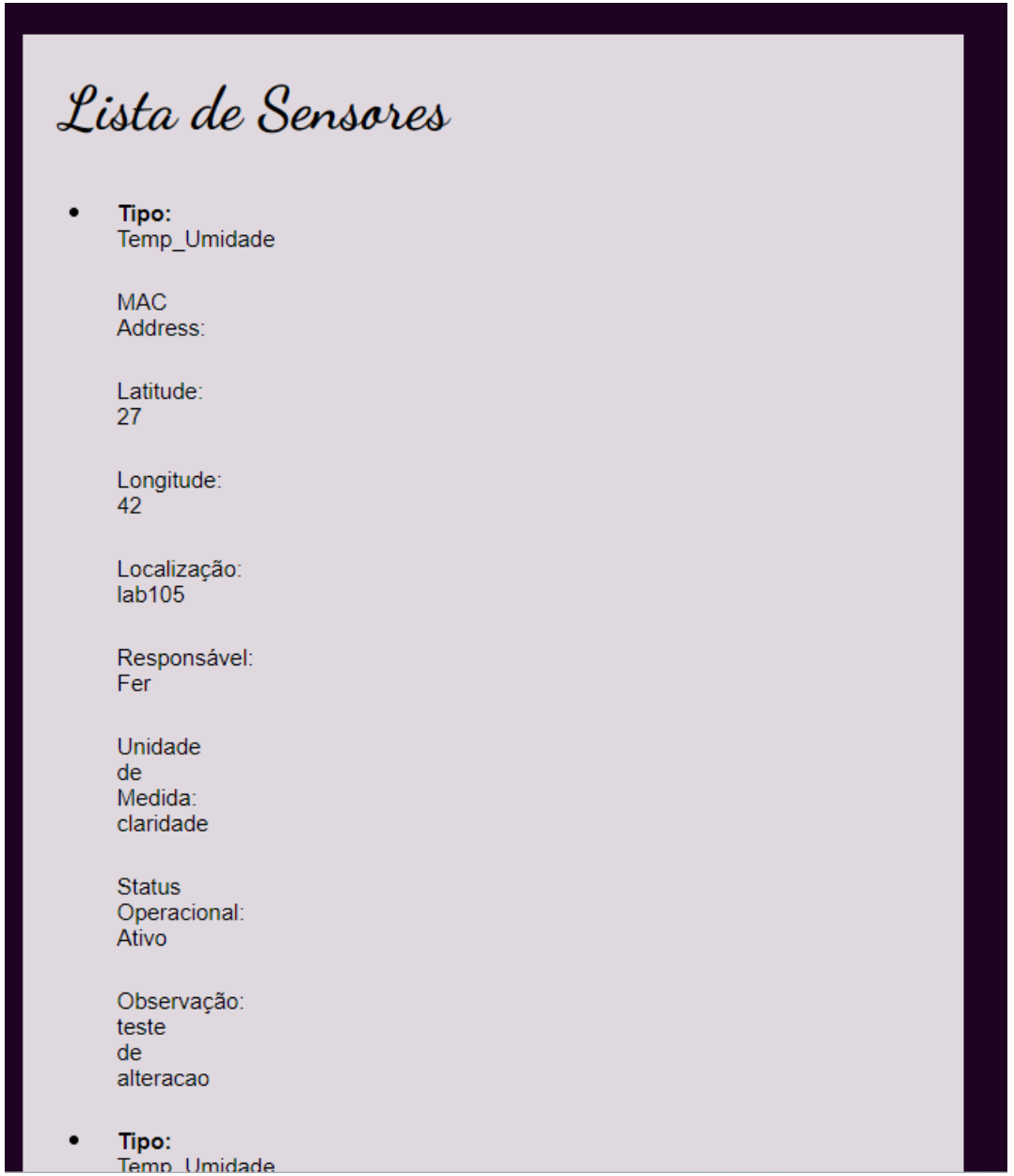
Fernanda Fretes

Consumo de API

- Se os dados ainda estão sendo carregados (loading é true), exibe uma mensagem "Carregando...".
- Se ocorreu um erro, exibe a mensagem de erro.
- Depois exibo a resposta do que estamos recebendo da API.

FrontEnd - React

Fernanda Fretes



FrontEnd - React

Fernanda Fretes

Cadastro via API

- Para fazer o cadastro, podemos contar com o zod para fazer a primeira validação dos dados passados pelo formulário.

```jsx
import React from 'react';
import estilos from './CadastraSensor.module.css';
import { useForm } from 'react-hook-form';
import { z } from 'zod';
import { zodResolver } from '@hookform/resolvers/zod';

const schemaCadastraSensor = z.object({
    tipo: z.string()
            .min(1, 'Informe um nome')
            .max(25, 'Máximo de 25 caracteres'),
    macAddress: z.string()
                .min(1, 'Mínimo de 5 caracteres')
                .max(10, 'Máximo de 10 caracteres'),
    latitude: z.number()
            .min(-90, 'Informe a latitude válida')
            .max(90, 'Informe a latitude válida'),
    longitude: z.number()
            .min(-180, 'Informe a longitude válida')
            .max(180, 'Informe a longitude válida'),
    localizacao: z.string()
            .min(1, 'Informe 10 caracteres')
            .max(50, 'Máximo de 50 caracteres'),
    responsavel: z.string()
            .min(1, 'Informe 8 caracteres')
            .max(15, 'Máximo de 15 caracteres'),
    unidadeMedida: z.string()
            .min(1, 'Informe 8 caracteres')
            .max(8, 'Máximo de 8 caracteres'),
    observacao: z.string()
            .min(1, 'Informe 8 caracteres')
            .max(200, 'Máximo de 200 caracteres'),
});
```

FrontEnd - React

Fernanda Fretes

Cadastro via API

- Construo meu zodResolver para resolver o schema que montamos
- E montamos a função obterdadosFormulário

```
export function CadastraSensor() {
    const {
        register,
        handleSubmit,
        formState: { errors }
    } = useForm({
        resolver: zodResolver(schemaCadastraSensor)
    });

    async function obterDadosFormulario(data) {
        console.log(`tipo: ${data.tipo}`)
        console.log(`macAddress: ${data.macAddress}`)
        console.log(`latitude: ${data.latitude}`)
        console.log(`longitude: ${data.longitude}`)
        console.log(`localizacao: ${data.localizacao}`)
        console.log(`responsavel: ${data.responsavel}`)
        console.log(`unidadeMedida: ${data.unidadeMedida}`)
        console.log(`statusOperacional: ${data.statusOperacional}`)
        console.log(`observacao: ${data.observacao}`)
```

```
    try {
        const response = await fetch('http://localhost:8000/api/sensores/', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNzE2MTYyNDc5LCJpYXQiOjE3MTYxNjIxNzks
            },
            body: JSON.stringify(data),
        });

        if (response.ok) {
            alert('Sensor criado com sucesso!');
            // Limpar o formulário ou redirecionar conforme necessário
        } else {
            const errorData = await response.json();
            console.error('Erro ao criar sensor:', errorData);
            alert('Erro ao criar sensor.');
        }
    } catch (error) {
        console.error('Erro:', error);
        alert('Erro ao criar sensor.');
    }
}
```

Passo a URL de endereço onde quero fazer o cadastro dos meus sensores, informando que o corpo será feito por Json

Caso a resposta seja OK (200) exibo a mensagem por meio de alerta que o sensor foi cadastrado com sucesso

Caso constrario exibo a mensagem informando que foi dado um erro e reforço isso no uso do Try Catch.

```jsx
export function CadastraSensor() {
    return (
      <div className={estilos.conteiner}>
          <p className={estilos.titulo}>Perfil</p>
          <form
              className={estilos.formulario}
              onSubmit={handleSubmit(obterDadosFormulario)}
          >
              <input
                  {...register('tipo')}
                  className={estilos.campo}
                  placeholder="tipo"
              />
              {errors.tipo && (
                  <p className={estilos.mensagem}>{errors.tipo.message}</p>
              )}
              <input
                  {...register('macAddress')}
                  className={estilos.campo}
                  placeholder="macAddress"
              />
              {errors.macAddress && (
                  <p className={estilos.mensagem}>{errors.macAddress.message}</p>
              )}
              <input
                  {...register('latitude', { valueAsNumber: true })}
                  className={estilos.campo}
                  placeholder="latitude"
                  type="number"

              />
              {errors.latitude && (
                  <p className={estilos.mensagem}>{errors.latitude.message}</p>
              )}
```

```
<input
    {...register('longitude', { valueAsNumber: true })}
    className={estilos.campo}
    placeholder="longitude"
    type="number"

/>
{errors.longitude && (
    <p className={estilos.mensagem}>{errors.longitude.message}</p>
)}
<input
    {...register('localizacao')}
    className={estilos.campo}
    placeholder="localizacao"
/>
{errors.localizacao && (
    <p className={estilos.mensagem}>{errors.localizacao.message}</p>
)}
<input
    {...register('responsavel')}
    className={estilos.campo}
    placeholder="responsavel"
/>
{errors.responsavel && (
    <p className={estilos.mensagem}>{errors.responsavel.message}</p>
)}
<input
    {...register('unidadeMedida')}
    className={estilos.campo}
    placeholder="unidadeMedida"
/>
{errors.unidadeMedida && (
    <p className={estilos.mensagem}>{errors.unidadeMedida.message}</p>
)}
```

```
                    <input
                        {...register('observacao')}
                        className={estilos.campo}
                        placeholder="observacao"
                    />
                    {errors.observacao && (
                        <p className={estilos.mensagem}>{errors.observacao.message}</p>
                    )}
                    <button
                        className={estilos.botao}
                        type="submit"
                    >Confirmar</button>
                </form>
            </div>
        );
    }
```