Tecnico em Informatica para Internet

Luis Enrique Chambi Tapia 1021A

Relatório Trabalho Final

Naviraí- MS

2023

MINISTÉRIO DA EDUCAÇÃO
Secretaria de Educação Profissional e Tecnológica
Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso do Sul

INSTITUTO FEDERAL
Mato Grosso do Sul

Tecnico em Informatica para Internet

Luis Enrique Chambi Tapia 1021A

Relatório Trabalho Final

Relatório de Frameworks 2 apresentado como requisito para o encerramento da disciplina.

Professor Guilherme Figueiredo Terenciani como responsável da disciplina Frameworks 2, 6ª semestre do curso Técnico em Informática para Internet .

Naviraí- MS

2023

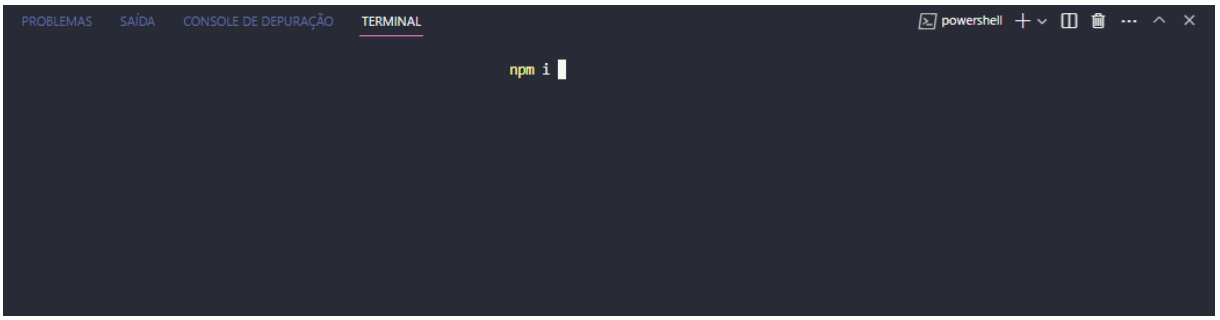
Para a conclusão desta atividade avaliativa foi efetuado uma pesquisa extensa que visou compreender do que foi solicitado no qual resultou em um uso de 70% do prazo estipulado dado para o encerramento da tarefa. Os outros 10% foram destinados para a execução do trabalho final, enquanto os últimos 20% foram alocados para a realização deste relatório.

Entretanto, é essencial concluir que para a execução se é necessário entender os principais aspectos a serem desenvolvidos, nos quais são:

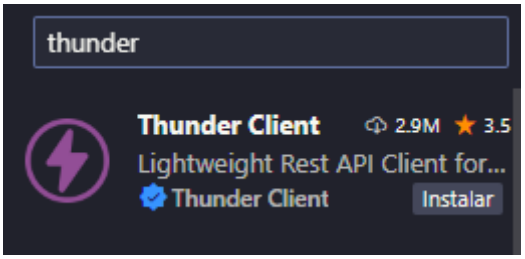
- Estar convertendo a variável "filmes" em um **useState**;
- Estar efetuando uma requisição por meio do Axios, isso adentro de um **useEffect,** neste caso, como observado durante as aulas tal qual seria o **http://localhost:3000/filmes**;
- posteriormente é necessário realizar um map para estar sendo capaz de listar os filmes;

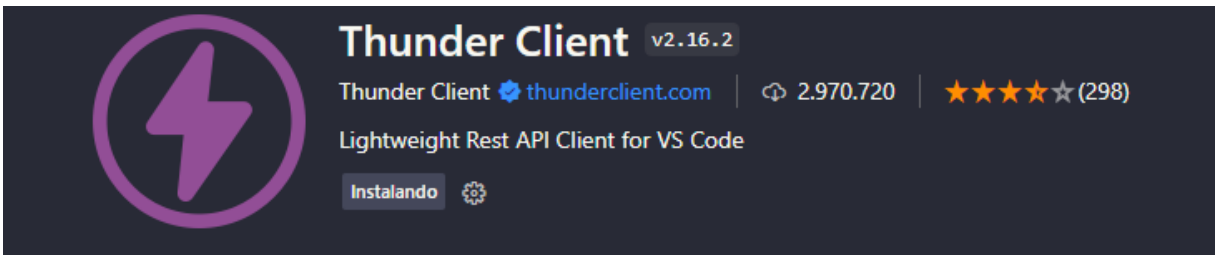
Sendo assim, podemos iniciar realizando a instalação na máquina dos projetos necessários para a execução dos mesmos, os quais são o back-end realizado à parte ao longo do último semestre e o projeto da página da barbie que foi efetuado anteriormente. Este passo não será retratado com imagens devido a ser algo de **senso comum** para ter que ser ensinado novamente aqui (se você não aprendeu isso depois de toda a longa jornada do curso, feche o documento imediatamente porque você não deveria estar aqui).

Seguimos com a abertura de tais projetos usando o Vs Code e instalando as suas devidas dependências abrindo o terminal em ambos projetos e usando o npm i:

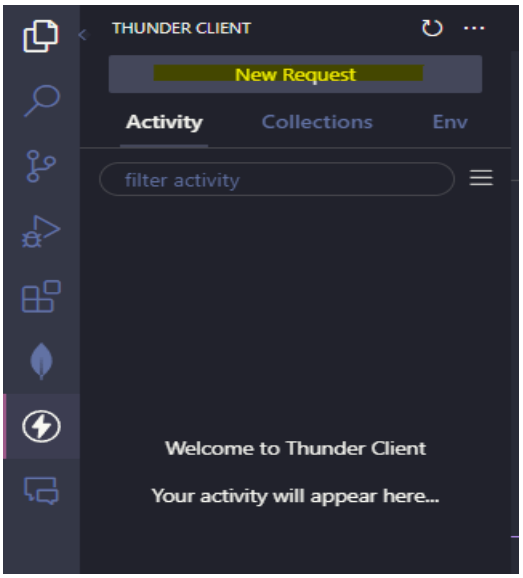


uma vez feito isso continuamos com a implementação da extensão Thunder no vs code:

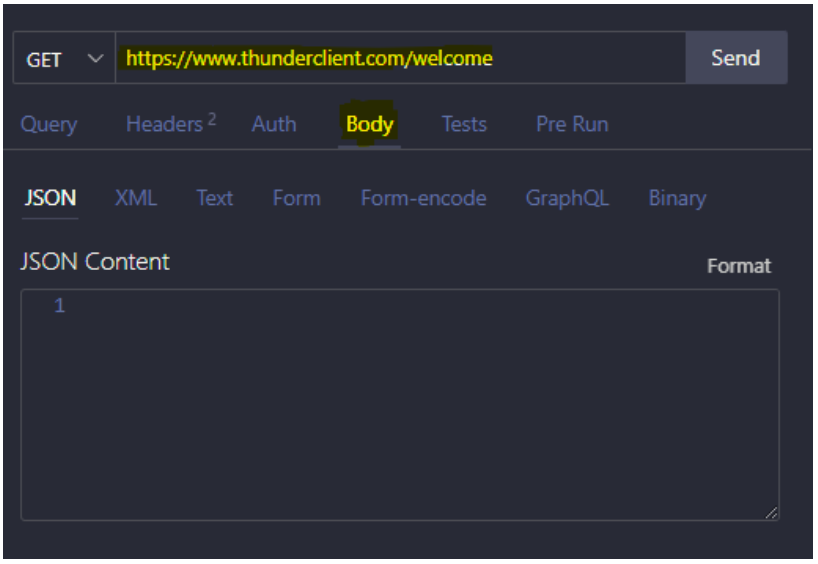




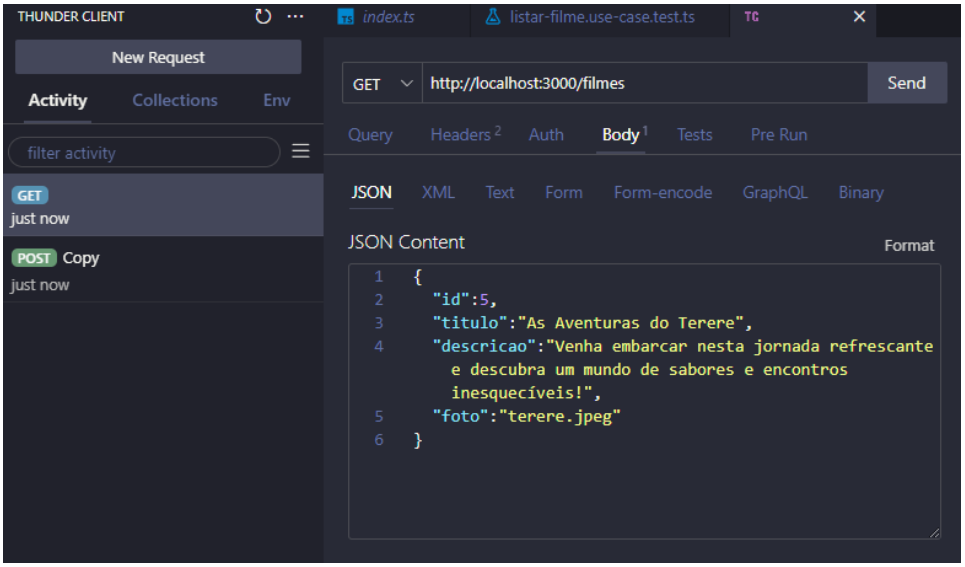
uma vez instalado estaremos selecionando o ícone da extensão e clicando em new request:



o próximo passo agora é estar selecionando **body** e mudando a url pre-inserida:

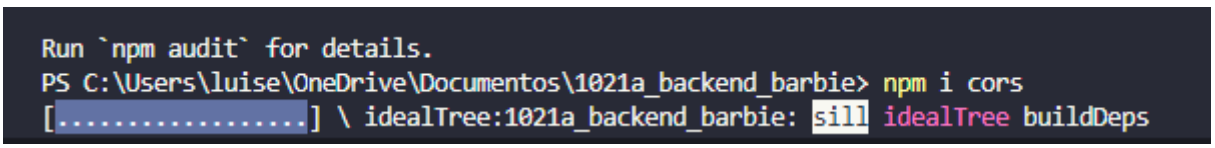


por último, vamos a criar nosso **send** e **post** para estarmos lançando os nossos dados para o banco de dados:



Agora, estaremos deixando isso de lado por ora para continuarmos com os próximos passos.

A seguir vamos a realizar a instalação do **cors** em nosso projeto de banco de dados usando o terminal:



uma vez feito isso vamos a agregar a importação do **cors** no banco de dados dentro do **index.ts**:

```
src > index.ts > ...
1   import express, {Request} from 'express';
2   import BancoMongoDB from './infra/banco/banco-mongodb' //paso 2: banco de dados add
3   import ListarFilme from './aplicacao/listar-filme.use-case' //paso 1: adiconar import listarfilme
4   import cors from 'cors'
5   const bancoMongoDB = new BancoMongoDB();
6   const app = express();
7   app.use(express.json())
8   app.use(cors())
9
10
11  app.get('/filmes', async (req, res) => {
12      const listarFilme = new ListarFilme(bancoMongoDB)
13      const filmes = await listarFilme.execute()
14      // res.send(filmes).status(200)
15      res.status(200).send(filmes) //passar primeiro status seguido do send
16  })
17
18  // // Define uma rota padrão
19  // app.get('/filmes/:id', (req, res) => {
20  //     const id = parseInt(req.params.id)
21  //     const filme = filmes_repositorio.find(filme => filme.id === id)
22  //     if (!filme) res.status(404).send()
```

Tendo realizado isto, podemos deixar avançar para o projeto da página da barbie para continuarmos com a atividade.

Uma vez estando no projeto da página da barbie, realizamos a instalação do axios por meio do terminal:

```
PS C:\Users\luise\OneDrive\Documentos\BarbieDreamhousev2> npm i axios

up to date, audited 226 packages in 2s

41 packages are looking for funding
  run `npm fund` for details
```

seguimos acessando ao **Main.tsx** para a adição da importação do **useEffect** no mesmo, seguido da adição da importação do **axios**:

```
src > componentes > main > Main.tsx > FilmeType
1   import { useEffect, useState } from 'react'
2   import Filme from './../filme/Filme'
3   import './Main.css'
4   import axios from 'axios'
```

O próximo passo a seguir será o de utilizar os hooks useState e useEffect uma vez tendo instalação da biblioteca do axios para fazer a requisição usando o get em nosso banco de dados local.

Neste caso, está sendo usado a variável **filmes** e a função **setFilmes** para tal. já o **FilmeType[]** quer indicar que a variável filmes e um array do tipo **FilmeType**.

Lembrando que era exigido que **filmes** estivessem sendo implementados em um **useState** e que em aulas anteriores já foi realizado a listagem, tendo só que codar o restante faltante.

Também foi usado o **axios.get** para fazer tal requisição por meio do get para o servidor local que seria o **http://localhost:3000/filmes** e o then() para estar lidando com o resultado da requisição dita. Já o **retorno.data** foi usado para este estar contendo os dados retornados pelo **get** e sendo estabelecidos na variável **filmes** através do **setFilmes(retorno.data)**.

Contudo, o programado aqui iria configurar filmes por meio do **useEffect** para usar o **get** em **http://localhost:3000/filmes** e assim que é embutido, através da variável **filmes** e feito essa atualização e listagem dos dados inseridos.

```
useEffect(()=>{
    axios.get('http://localhost:3000/filmes').then(
        (retorno)=>{
            setFilmes(retorno.data)
        }
    )
}, [])
```

por último e não menos importante, o mais provável é que tenha tido sucesso e que na hora de verificar isso no google chrome nao haja aparecido a imagem do mesmo, a imagem não aparecia devido a estar definida no projeto como “imagem”, nesse caso foi realizada alteração para “foto” onde era necessário.

obs: comentei todo o **const filmes:FilmeType[]** que havia no Main.tsx para a execução da atividade (o certo era remover a linha de código mas quis manter ele para me lembrar daquilo que havia antes de substituir pelo solicitado na atividade)

Se você realizou tudo como descrito aqui, meus parabéns… terminou o **trabalho final** do Terere.