Desafio PA - API
Davi Dias Monsores dos Santos

Neste documento será explicada a abordagem utilizada para implementação dos 3 scripts do desafio 1, buscando mostrar como foi pensada a execução da solução, desde a compreensão dos requisitos do desafio até os tratamentos de erro.

Compreensão dos Requisitos

Para começar, como foi informado na explicação do desafio, foi necessário pegar uma key para utilizar na header das chamadas para a API do TMDb. Após isso, realizei a leitura de todos os scripts para entender o que estava sendo pedido e tive a ideia de implementar de forma que fosse visualmente agradável e de fácil entendimento, criando um pequeno site para mostrar o retorno de tudo o que era pedido nos scripts.

Sendo assim, após rodar a aplicação com “npm run dev”, é possível navegar pelas páginas dos 3 scripts, onde os retornos pedidos estão sendo mostrados. Para que fosse possível essa navegação entre rotas, utilizei a biblioteca “react-dom/client”. Para estilização básica realizada nas páginas, foi utilizado tailwind. Além disso, o projeto foi desenvolvido em TypeScript, por conta de eu já ter alguma experiência recente em implementações de API utilizando-o.

Implementação dos 3 Scripts

Nesta seção, toda a parte de implementação dos códigos para os 3 scripts será explicada. Inicialmente, como eu havia percebido que diversas requisições de diferentes modos seriam feitas para a API, decidi montar um arquivo chamado “api.ts”, onde a requisição é montada quase que por completo, criando uma instância personalizada do axios com a url base da API e seus headers. Dessa forma, apenas é necessário enviar o endpoint da url na hora de realizar as requisições.

O axios foi utilizado na implementação da API por conta de eu já possuir experiência utilizando-o, o que facilita o processo de desenvolvimento. Além disso, foi utilizado um arquivo chamado “requestContent.ts” para guardar as requisições para todos os scripts em um só local, que é chamado pelas páginas quando se faz necessária a requisição de algum filme.

Por fim, para realizar as requisições a chave utilizada foi guardada num “.env” que foi ocultado do GitHub por ser uma medida básica de segurança. Caso queiram utilizá-lo, aqui está o que deve ficar no “.env”:

VITE_TMDB_TOKEN=Bearer eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI4MmUwNzAwOTBiNTllMjIyMWNiMTdhMDlhZGQ4N2MwNyIsIm5iZiI6MTc0Nzc3NzAzMC41MzUsInN1YiI6IjY4MmNmNjA2NjI5OWI2MWU3MzY0OTRkNyIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ2ZXJzaW9uIjoxfQ.gph3VtyHrLyZHaywo8awjy3b8-f_-QqqqJzrOZ6yRcI

Script 1:

Para o Script 1, duas requisições para a API eram necessárias. A primeira foi para pegar os filmes populares e, a partir deles, extrair id, título, data de lançamento e descrição. Como o retorno da API é um array com os filmes populares, na função de requisição apenas apliquei uma formatação das datas de lançamento de cada filme para que mostrasse DD/MM/AAAA em vez de AAAA/MM/DD e retornei para a página de exibição o array formatado.

Na página de exibição, a chamada das funções do requestContent é realizada com Promise.all, para que ela chame ao mesmo tempo as duas funções, otimizando o tempo de processamento, já que para a segunda parte do Script 1 utiliza-se outro retorno de dados como será explicado daqui a pouco.

Após isso, os arrays retornados são armazenados por meio de um useState, utilizado para que a atualização da interface ocorra quando ele for alterado, e os filmes são exibidos pela realização de um map no novo estado com os filmes populares. Dessa forma, apenas foi necessário olhar com um "console.log" quais eram os campos id, título, data de lançamento e descrição e exibir tudo na tela.

Já a segunda requisição, onde foi necessário buscar pelo id dos 5 filmes populares mais recentes para pegar os trailers e montar a url dos trailers para exibição em um player do YouTube, eu utilizei um "sort" para comparar as datas de lançamento dos filmes no array de populares entre si e um "slice" para pegar apenas os 5 mais recentes.

Após isso, realizei um map nesse novo array com os 5 filmes mais recentes e verifiquei se o tipo do vídeo era "Trailer" e o site do vídeo era "YouTube". Caso a verificação fosse positiva, eu modificaria o campo "trailerUrl" pela url do YouTube com o endpoint sendo a chave do trailer encontrado, retornando o array dos 5 mais recentes com o "trailerUrl" modificado para a página de exibição.

Na página de exibição, um useState também é utilizado para armazenar o array dos 5 filmes e o player de exibição dos trailers é montado com o iframe.

Script 2:

Para o Script 2, o objetivo era requisitar da API a lista de gêneros, filtrar os que começassem com a letra "A" e extrair o id desses gêneros. A partir disso, foi pedida a exibição de id, título e pôster dos filmes para cada id de gênero extraído.

Sendo assim, primeiro requisitei da API a lista de gêneros e foi retornado um array com os gêneros. Utilizando o console.log, verifiquei qual o campo de nome dos gêneros e para cada gênero verifiquei se a primeira letra era "A" utilizando "genre.name[0]".

Após isso, utilizei um map para requisitar os filmes de cada gênero da lista de gêneros que começavam com a letra "A" e guardei em uma espécie de array de arrays, onde, para cada gênero, uma string com o nome do gênero está guardando os arrays com os filmes desses gêneros em específico. Então, retornei essa lista de arrays encadeados para a página de exibição.

Na página de exibição, um useState foi utilizado para armazenar os arrays e, utilizando um map, para cada gênero, todos os filmes presentes no array tem seu id, título e pôster exibidos.

Script 3:

Para o Script 3, o objetivo era buscar na API os filmes “Em Alta” e exibir id, título, data de lançamento, média de votos e descrição. Além disso, também foi pedido que eu escolhesse um filme da lista anterior para exibir apenas a contagem de votos desse filme. Sendo assim, após realizar a requisição dos filmes now playing para a API, retornei o arrays com eles para a solicitação da página de exibição.

Na página de exibição, um useState armazenou os dados dos filmes retornados para que fossem exibidas as informações solicitadas por meio de um map para cada filme, Além disso, outro useState foi utilizado para pegar apenas a contagem de votos (campo vote_count encontrado no console.log dos filmes) do primeiro filme da lista de now playing retornada anteriormente e exibir essa contagem após os filmes now playing.

Desafio Extra Script 3:

No desafio extra do Script 3, um menu interativo foi pedido para que o usuário pesquise o título de um filme e seja retornada a contagem de votos desse filme. Dessa forma, coloquei uma lupa no topo direito da página do Script 3, onde, após apertar nela, o usuário é redirecionado para uma página com um input text.

Na página da lupa, o usuário pode pesquisar por um título de filme, mesmo que incompleto, e, ao clicar no botão “Buscar”, receber como resposta os filmes associados ao título com a sua contagem de votos. Caso exista mais de um filme com o mesmo título buscado, os filmes serão retornados em ordem decrescente de votos totais, para garantir que os de maior relevância estejam no topo.

Para que isso funcione, na página de busca, ao clicar no botão “Buscar” o programa verifique se o input text não está vazio e, caso não esteja, envia a string do título digitado para a função searchContents em requestContent.

Na função searchContents, o valor da string é recebido e a seguinte requisição é feita para a API, para que sejam retornados os filmes com aquele título:

api.get(`3/search/movie?query=${searchMedia}&language=pt-BR`)

Dessa forma, um array com os filmes que possuem a string enviada na requisição como parte do título serão retornados. Assim, envio esses filmes para a página da lupa novamente e faço a exibição dos filmes com título e contagem de votos.

Tratamento de Erros

Durante todo o código, após uma requisição verificou-se se a resposta não foi status 200. Caso retornasse com um erro de requisição, um console.error mostraria a mensagem “Resposta inválida da API”. Além disso, como os códigos são executados com axios para as chamadas da API, ou seja, são assíncronos, todas as áreas de requisição possuem um “try - catch (error)” para possíveis erros diversos, como DNS, CORS, API fora do ar, etc.

Uso de Ferramentas de IA

Durante a implementação do código, o ChatGPT foi usado quase que unicamente para estilização com tailwind, pois eu não me recordava de todos os comandos para estilizar da forma que eu estava querendo. Além da estilização, o ChatGPT também me ajudou com a formatação das datas de inglês para português, pois eu não lembrava como era feita a

chamada da função Intl.DateTimeFormat. Por fim, ele me ajudou a guardar os dados no formato <Record<string, any[]> no Script 2, pois eu estava querendo mostrar o nome do gênero com os filmes abaixo na hora de exibir, não apenas os filmes.

Minha avaliação sobre o uso é de que foi muito benéfica, pois, a partir de uma ideia já formulada, o ChatGPT consegue encontrar uma solução de forma bem eficaz, diferente de quando se pede para ele pensar na ideia e implementá-la.