Desenvolvimento para Dispositivos Móveis

React Native

DS151

UFPR

Para hoje

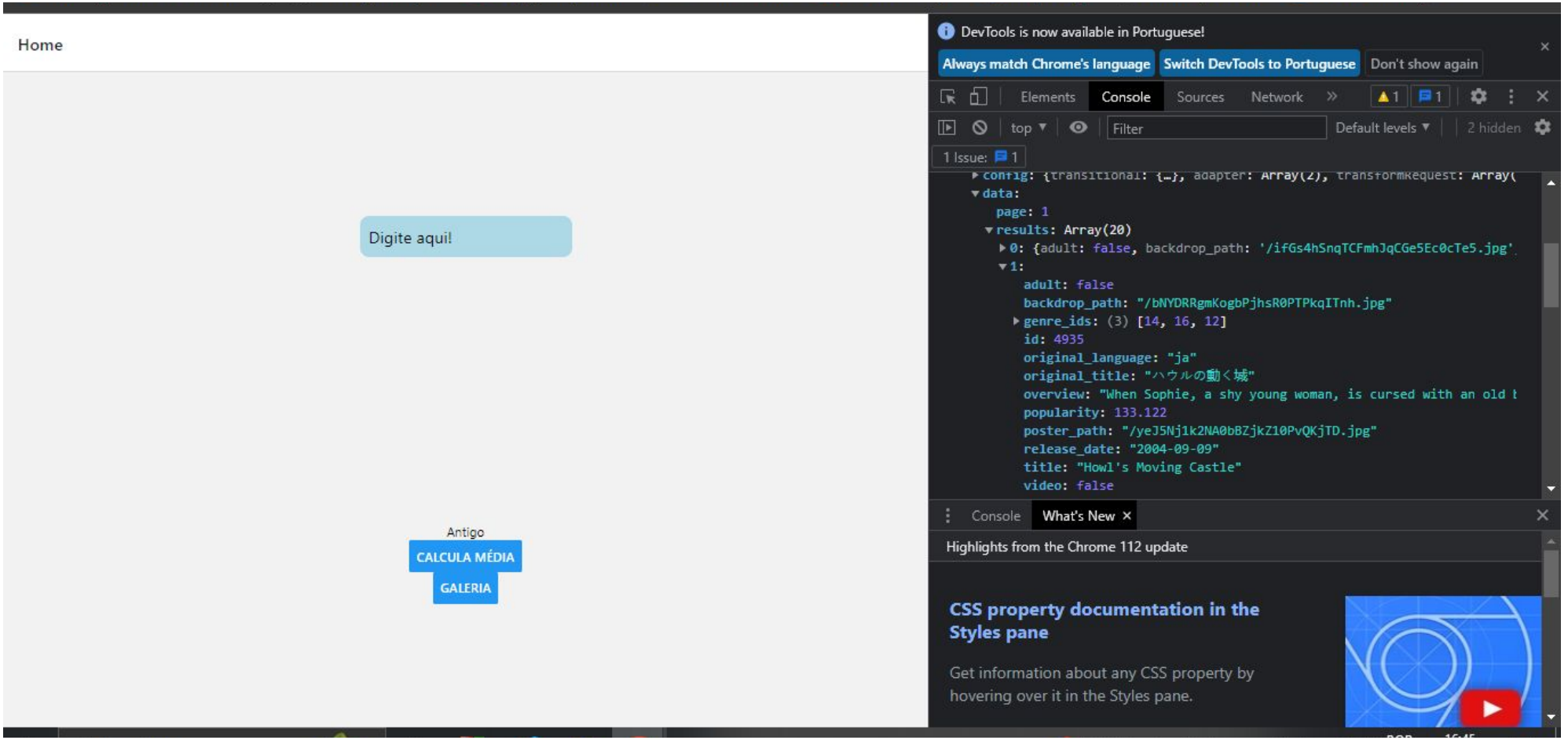
Trabalho 1

APIs

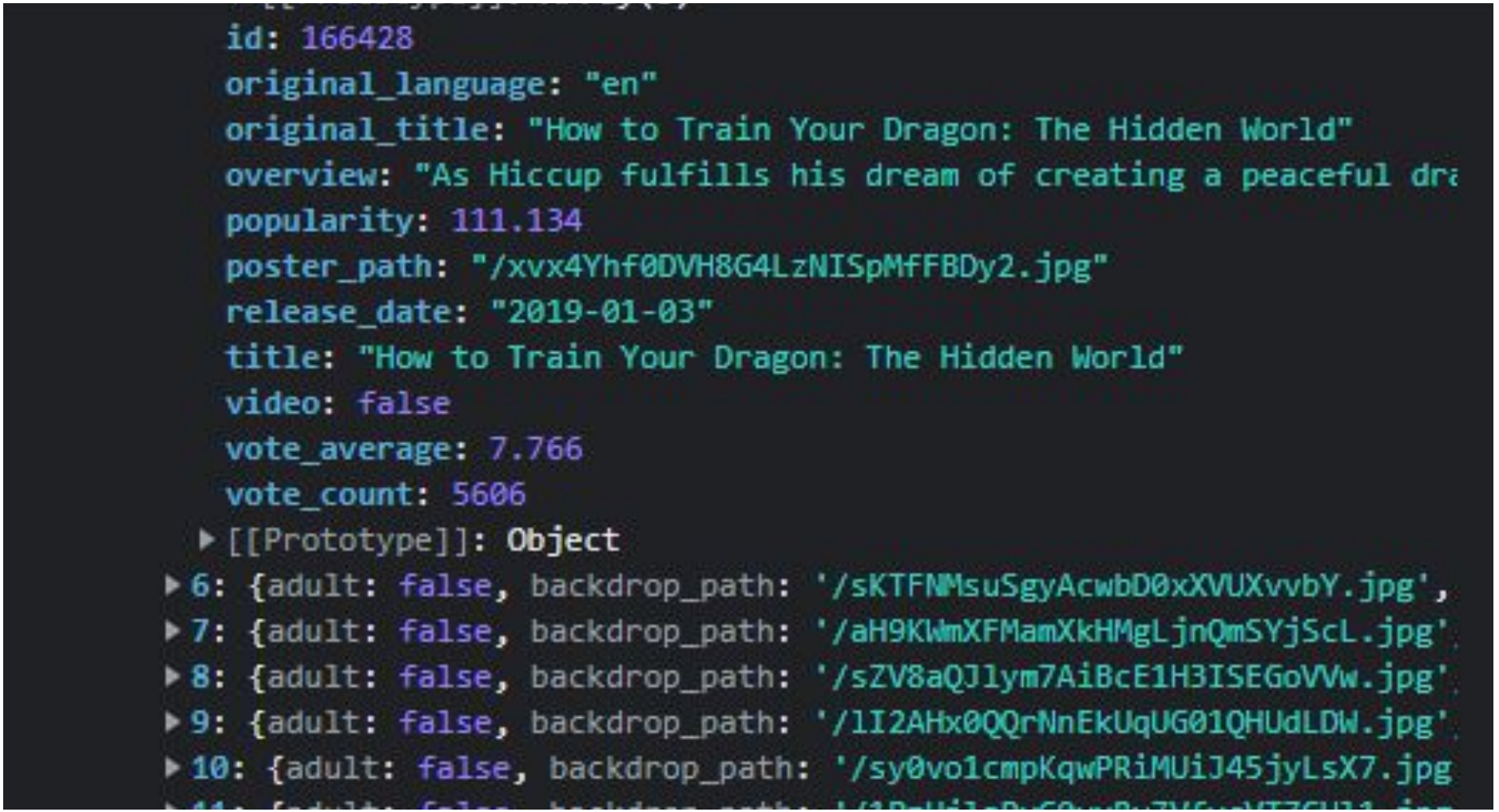
Effect Hooks

Reducer Hooks

AXIOS


Home
Digite aqui!
Antigo
CALCULA MÉDIA
GALERIA
DevTools is now available in Portuguese!
Always match Chrome's language
Switch DevTools to Portuguese
Don't show again
Elements
Console
Sources
Network
top
Filter
Default levels
2 hidden
1 Issue: 1
config: {transitional: {…}, adapter: Array(2), transformRequest: Array(
data:
page: 1
results: Array(20)
0: {adult: false, backdrop_path: '/ifGs4hSnqTCFmhJqCGe5Ec0cTe5.jpg'
1:
adult: false
backdrop_path: "/bNYDRRgmKogbPjhsR0PTPkqITnh.jpg"
genre_ids: (3) [14, 16, 12]
id: 4935
original_language: "ja"
original_title: "ハウルの動く城"
overview: "When Sophie, a shy young woman, is cursed with an old t
popularity: 133.122
poster_path: "/yeJ5Nj1k2NA0bBZjkZ10PvQKjTD.jpg"
release_date: "2004-09-09"
title: "Howl's Moving Castle"
video: false
Console
What's New
Highlights from the Chrome 112 update
CSS property documentation in the Styles pane
Get information about any CSS property by hovering over it in the Styles pane.

AXIOS


id: 166428
original_language: "en"
original_title: "How to Train Your Dragon: The Hidden World"
overview: "As Hiccup fulfills his dream of creating a peaceful dra
popularity: 111.134
poster_path: "/xvx4Yhf0DVH8G4LzNISpMfFBDy2.jpg"
release_date: "2019-01-03"
title: "How to Train Your Dragon: The Hidden World"
video: false
vote_average: 7.766
vote_count: 5606
[[Prototype]]: Object
6: {adult: false, backdrop_path: '/sKTFNMsuSgyAcwbD0xXVUXvvbY.jpg',
7: {adult: false, backdrop_path: '/aH9KWmXFMamXkHMgLjnQmSYjScL.jpg'
8: {adult: false, backdrop_path: '/sZV8aQJ1ym7AiBcE1H3ISEGoVVw.jpg'
9: {adult: false, backdrop_path: '/lI2AHx0QQrNnEkUqUG01QHUdLDW.jpg'
10: {adult: false, backdrop_path: '/sy0vo1cmpKqwPRiMUiJ45jyLsX7.jpg

Na última aula

No seu repositório, crie uma View com o consumo de uma API externa (pode ser a TMDB).

Passe o valor de referência através de um inputText interativo

Trabalho 1

Data: 15 de maio

Crie uma aplicação que consuma uma API externa e interaja com seus dados através de navegação.

Seu app deve listar o conteúdo da API através de elementos nativos e navegar através de atributos diferentes.

Consumindo API

Conector

```
import axios from 'axios';

//a5bd835114760bc9b7a834b8762d0644
const apiKey = 'c5568d6e77039a6978ff01e662ba04a1';

export default axios.create({
    baseURL: 'https://api.themoviedb.org/3/',
    params: {
      api_key: apiKey,
    }
  });

```

Consumindo API

Função da query de busca

```
const HomeScreen = ({navigation}) => {
    const [text, setText] = useState('');
    async function searchTmdb(query){
      try{
        const response = await tmdb.get('/search/movie',{
          params: {
            query,
            include_adult: false,
          }});
      }
      catch(err){console.log(err);}
      }
```

Consumindo API

Elemento de entrada

```
const SearchBar = ({onTextChange, onTextSubmit, value}) => {
      return (
        <View style={styles.container}>
            <TextInput
            autoCaptalize="none"
            autoCorrect={false}
            placeholder="Digite aqui!"
            style={styles.textInput}
            value={value}
            onChangeText ={newText => onTextChange(newText)}
            onEndEditing={()=> onTextSubmit(value)}
            ></TextInput>
        </View>
    )
```

Consumindo API

Podemos fazer com que nosso Json seja armazenado e atualizado na aplicação utilizando estados:

```
const [results, setResults] = useState([]);
```

Na API TMDB o Json nos traz os dados em 'data'

```
setResults(response.data.results);
```

Consumindo API

Então podemos exibir os dados com elementos visuais como labels, ou Flatlist (ideal, já que a resposta é dinâmica)

```
<Flatlist
    data={results}
    keyExtractor={item => item.id}
    renderItem={({item})=>{
        return(
         <Text>{item.original_title}</Text>
        );
}}/>
```


```
"id": 8871,
"original_language": "en",
"original_title": "How the Grinch Stole Christmas",
"overview": "Inside a snowflake exists the magical land of
"popularity": 117.351,
"poster_path": "/6B7cxxn1A4zqFFmJC3iT99GDWWm.jpg",
"release_date": "2000-11-17",
"title": "How the Grinch Stole Christmas",
```


UFPR

Consumindo API

Assim já temos associado um props para receber os dados em get para a aplicação.

Podemos fazer um Fetch para buscar parâmetros e valores da API

A API Fetch fornece uma interface JavaScript para acessar e manipular pedidos e respostas.

- Busca recursos de forma assíncrona.

Consumindo API

Solicitações HTTP usando as promises do JavaScript

```
function fetchData(type, id) {
  let params;

  if (type == "discover") {
    setFilmes([]);
    baseUrl += "discover/movie";

    params = new URLSearchParams({
      api_key: token,
      language: "en-EN",
      sort_by: "popularity.desc",
      include_adult: false,
      include_video: false,
      page: 1,
      with_genres: id,
      with_watch_monetization_types: "flatrate",
    });
```

```
fetch(`${baseUrl}?${params}`)
  .then((response) => response.json())
  .then((data) => {
    if (type == "discover") {
      setFilmes(data.results);
      console.log(data.results);
    } else {
      setGeneros(data.genres);
    }
  })
  .catch((error) => {
    console.error(error);
  });
```

Consumindo API

The Super Mario Bros. Movie Info!

Puss in Boots: The Last Wish Info!

Mummies Info!

The Amazing Maurice Info!

Batman: The Doom That Came to Gotham Info!

Marcel the Shell with Shoes On Info!

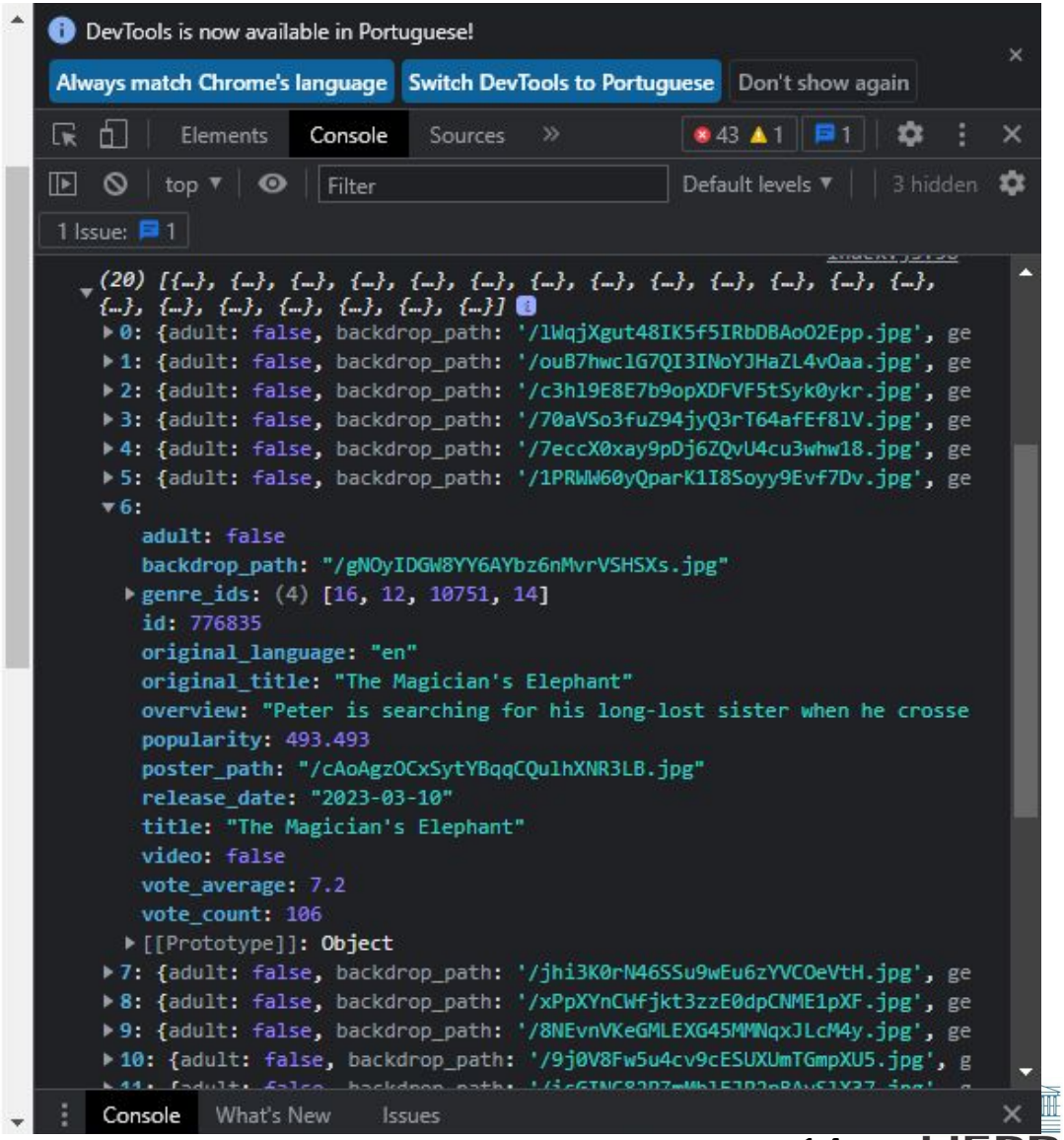
The Magician's Elephant Info!

Legion of Super-Heroes Info!

Demon Slayer -Kimetsu no Yaiba- The Movie: Mugen Train Info!

Demon Slayer: Kimetsu no Yaiba Sibling's Bond Info!

The Simpsons Meet the Bocellis in Feliz Navidad Info!



Effect Hook

São efeitos que executam sempre que algo é renderizado. Diferente do useState.
Ex. Sempre que o componente é construído.

Podemos definir o comportamento baseado, por exemplo, em alterações de variável.

```
import React, {useState, useEffect} from 'react';
```

Effect Hook

Basta adicionarmos a chamada na função exportada do componente.

```
    useEffect(() =>{
        console.log("Funcionou!")
    });
```

Toda vez que o componente for construído, executa o efeito. Então, podemos passar uma chamada de função, por exemplo:

```
searchTmdb('parametro')
```

Navegação com Parâmetros

Um recurso importante é passar parâmetros durante a navegação de um componente para outro.

Crie uma nova view com o boilerplate padrão (imports, função de exportação e estilo)

Navegação com Parâmetros

Nesta tela, podemos incluir um botão que navega para outra view e carrega um parâmetro (objeto da API, por exemplo)

```
<TouchableOpacity
onPress={()=>navigation.navigate("View2",
id: item.id      )}>
<Text>View 2</Text>
</TouchableOpacity>
```

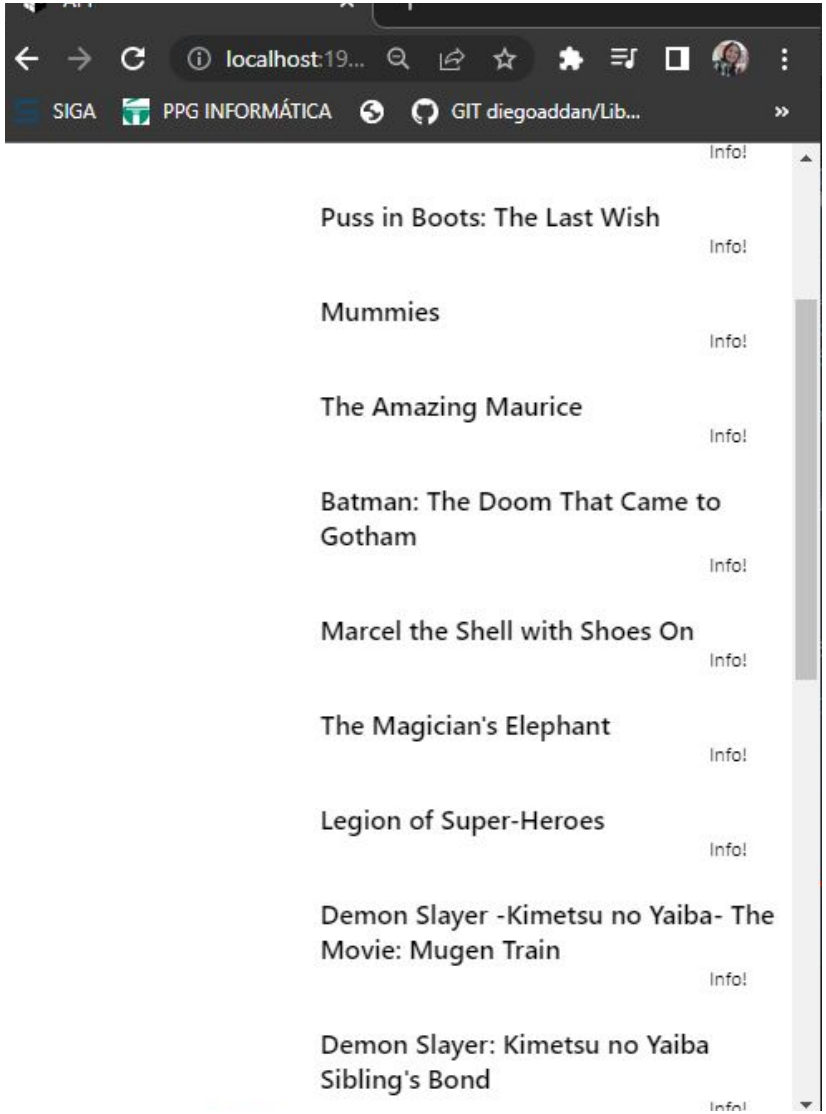
Navegação com Parâmetros

Podemos enviar quantos objetos ou parâmetros quisermos.

Para a outra view receber utilizamos o route, do navigation.

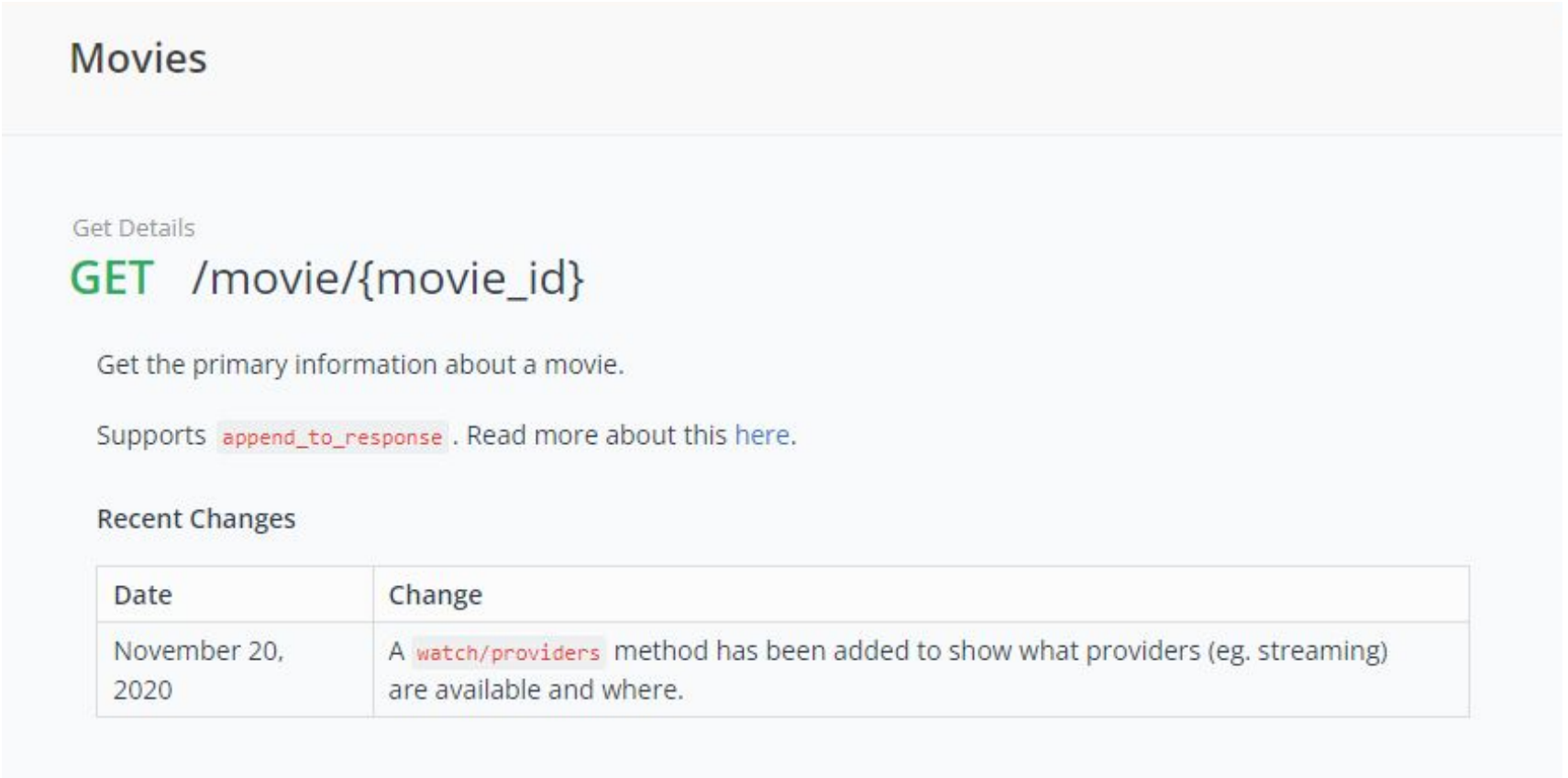
```
const View2 = ({navigation, route}) => {
    return (
        <View>
            <Text>{route.params.id}</Text>
        </View>)}
export default View2;
```

Navegação com Parâmetros



Navegação com Parâmetros

Utilizando a API tmdb podemos com este recurso enviar dados mais completos do filme



Movies

Get Details

GET /movie/{movie_id}

Get the primary information about a movie.

Supports `append_to_response`. Read more about this here.

Recent Changes

Date	Change
November 20, 2020	A `watch/providers` method has been added to show what providers (eg. streaming) are available and where.

Navegação com Parâmetros

Criamos uma função para receber os detalhes do filme a partir de um objeto vazio (useState)

```
    const [movie, setMovie] = useState({});
    async function getMovie(id){
        const response = await tmdb.get(`/movie/${id}`);
        setMovie(response.data.results);
    }
    useEffect(()=>{getMovie(route.params.id);})
```

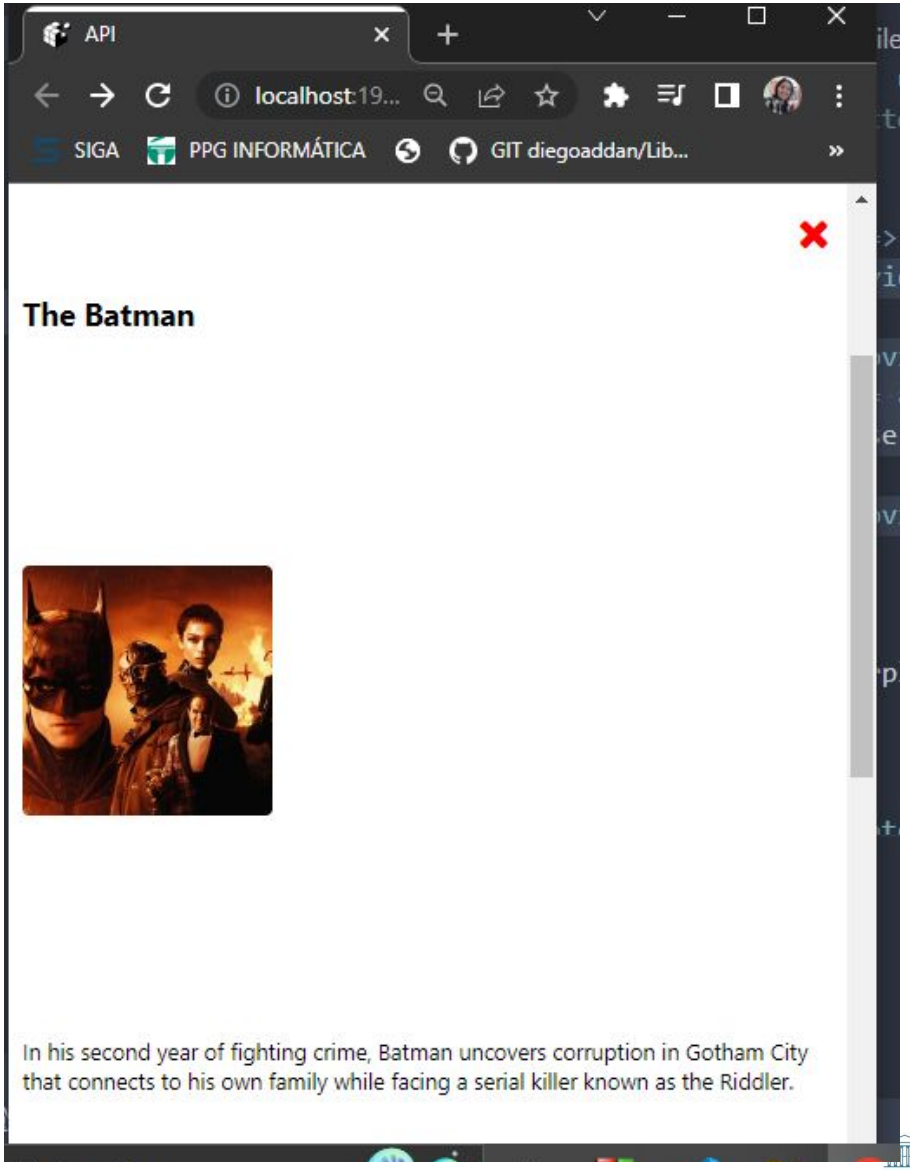
Navegação com Parâmetros

Importante conhecer a API que está utilizando.

Investigue a documentação

Navegação com Parâmetros

Desta forma podemos exibir qualquer parâmetro que a API fornece como título, sinopse, imagem!



Navegação com Parâmetros

Reducer Hook

Alternativa ao useState. Estado + Ação = Novo Estado

Em uma nova view importe o recurso

```
import React, {useState, useReducer} from 'react';
```

E criamos a função de controle

```
const funReducer=(state, action)=>{
    switch(action.type){
        default:return({...state});
 }}
```

Reducer Hook

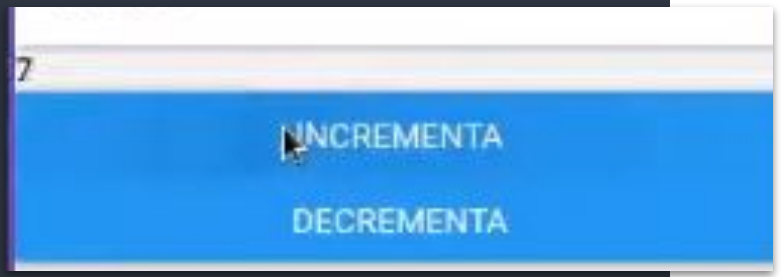
Users > User > Documents > reactN > JS aula7.js > Aula7

```
import React, {useState, useReducer} from 'react';
import { View, Text, Button, StyleSheet } from 'react-native';
const funReducer=(state, action)=>{
    switch(action.type){
        case 'incrementa':
            return({...state, count:state.count + 1});
        case 'decrementa':
            return({...state, count:state.count - 1});
        default:
            return({...state});}}
const Aula7 = () => {
    const [state, dispatch] = useReducer(counterReducer, {count:0});
    return (
        <View >
            <Text>Reducer Hook</Text>
            <Button title='Incrementa' onPress={() =>{dispatch({type:'incrementa'})}} />
            <Button title='Decrementa' onPress={() =>{dispatch({type:'decrementa'})}} />
        </View>
    )
```

Reducer Hook

Users > User > Documents > reactN > JS aula7.js > Aula7

```
import React, {useState, useReducer} from 'react';
import { View, Text, Button, StyleSheet } from 'react-native';
const funReducer=(state, action)=>{
    switch(action.type){
        case 'incrementa':
            return({...state, count:state.count + 1});
        case 'decrementa':
            return({...state, count:state.count - 1});
        default:
            return({...state});}}
const Aula7 = () => {
    const [state, dispatch] = useReducer(counterReducer, {count:0});
    return (
        <View >
            <Text>Reducer Hook</Text>
            <Button title='Incrementa' onPress={() =>{dispatch({type:'incrementa'})}} />
            <Button title='Decrementa' onPress={() =>{dispatch({type:'decrementa'})}} />
        </View>
    )
```



Vimos

- Effect Hook
- Reducer Hook
- Navegação com Parâmetros

Navegação com Parâmetros

E basicamente é isso que devem construir na app para o T1.

- Utilize API externa,
- Consuma esta API e navegue pelos parâmetros com elementos que aprendemos em aula.
- Exiba uma navegação e exibição destes dados em 3 camadas.