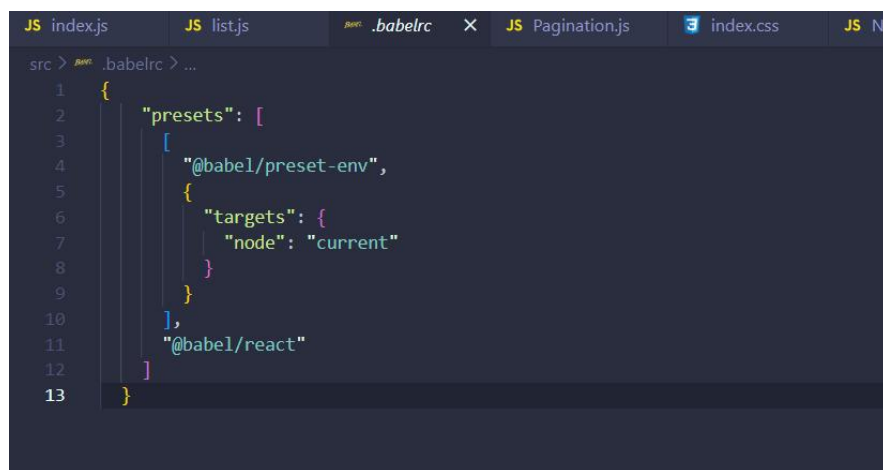


Documentacion App buscador de películas

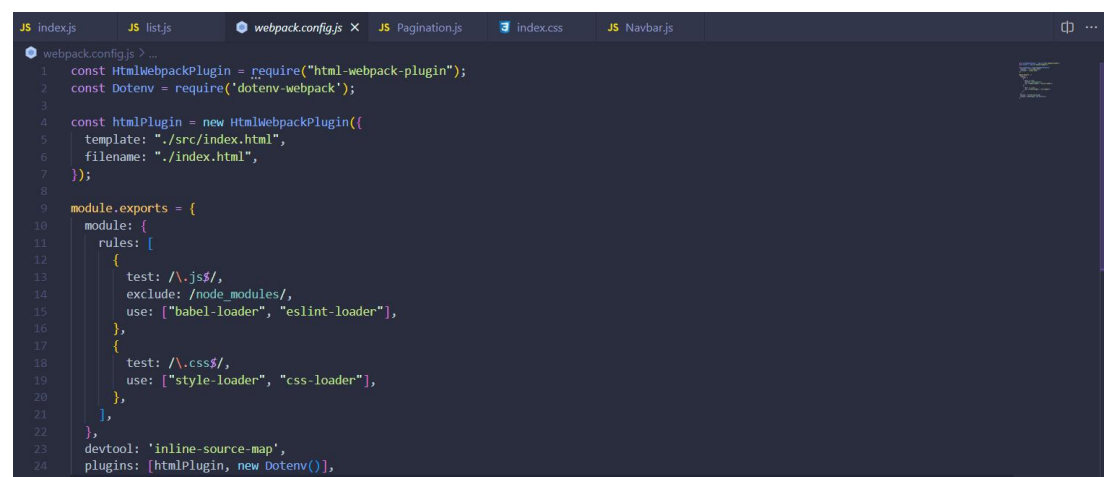
Se utilizo react js para realizar dicha pagina web con una api gratuita de películas llamada OMDB. Se realizo la aplicacion desde cero instalando node modules, webpack, algunas extensiones de babel y npm.

Para poder iniciar la aplicacion principalmente se creo un componente de babel llamado .babelrc que ese es el encargado de traducir el código a un javascript moderno y de esa forma poder escribir syntaxis nueva. Este objeto es para configurar babel y contiene los módulos que instale.



```
src > .babelrc > ...
1  {
2    "presets": [
3      [
4        "@babel/preset-env",
5        {
6          "targets": {
7            "node": "current"
8          }
9        }
10     ],
11     "@babel/react"
12   ]
13 }
```

Despues en el componente webpack se excluyo que cuando corra el proyecto no lo inicie con node modules por que lo estaremos iniciando con webpack



```
JS index.js JS list.js webpack.config.js JS Pagination.js index.css JS Navbars
webpack.config.js > ...
1  const HtmlWebpackPlugin = require("html-webpack-plugin");
2  const Dotenv = require("dotenv-webpack");
3
4  const htmlPlugin = new HtmlWebpackPlugin({
5    template: "./src/index.html",
6    filename: "./index.html",
7  });
8
9  module.exports = {
10   module: {
11     rules: [
12       {
13         test: /\.js$/,
14         exclude: /node_modules/,
15         use: ["babel-loader", "eslint-loader"],
16       },
17       {
18         test: /\.css$/,
19         use: ["style-loader", "css-loader"],
20       },
21     ],
22   },
23   devtool: 'inline-source-map',
24   plugins: [htmlPlugin, new Dotenv()],
25 }
```

Si abrimos el archivo index.html en nuestro navegador este si va a funcionar por que lo que lo que hace webpack es traducir nuestro código a html. Recordemos que el html tiene la estructura básica de html:5

```
src > index.html > html
1 <!DOCTYPE html>
2 <html Lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>MovieApp</title>
8   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/3.7.2/animate.min.css">
9 </head>
10
11 <body>
12   <id id="root"></id>
13 </body>
14
15 </html>
```

Despues de haber creado esto ya podemos iniciar creando nuestros componentes el primero es list.js que tiene lo mas importante que es la api y el recuadro de búsqueda junto con sus validaciones.

```
JS index.js JS list.js X index.html JS Pagination.js index.css JS Navbar.js
src > containers > JS list.js > list > useEffect() callback
1 import React, { useEffect, useState } from "react";
2 import Card from "../components/Card/Card";
3
4 const API = process.env.API;
5
6 const List = () => {
7   const [state, setState] = useState({
8     data: [],
9     totalResults: 0,
10    loading: true,
11    searchTerm: "",
12    error: "",
13    currentPage: 1,
14  });
15
16  const getMovie = async (page) => {
17    const res = await fetch(`${API}&s=batman&page=${page}`);
18    const resJSON = await res.json();
19
20    if (resJSON) {
21      setState((prevState) => ({
22        ...prevState,
23        data: resJSON.Search,
24        totalResults: parseInt(resJSON.totalResults),
25      }));
26    }
27  };
28
29  useEffect(() => {
30    getMovie(1);
31  }, []);
32
33  return (
34    <div>
35      <input type="text" value={searchTerm} onChange={handleSearch} />
36      <button onClick={handleSearch}>Buscar</button>
37      <div>
38        {data.map((movie) => <Card key={movie.id} movie={movie} />)}
39      </div>
40    </div>
41  );
42}
```

```

src > containers > JS list.js > List > useEffect() callback
14   });
15
16   const getMovie = async (page) => {
17     const res = await fetch(`${API}&s=batman&page=${page}`);
18     const resJSON = await res.json();
19
20     if (resJSON) {
21       setState((prevState) => ({
22         ...prevState,
23         data: resJSON.Search,
24         totalResults: parseInt(resJSON.totalResults),
25         loading: false,
26         error: "",
27       }));
28     }
29   };
30
31   useEffect(() => {
32     getMovie(state.currentPage);
33   }, [state.currentPage]);
34
35   const handleSubmit = async (e) => {
36     e.preventDefault();
37
38     if (state.searchTerm === "") {

```

```

2     useEffect(() => {
3       getMovie(state.currentPage);
4     }, [state.currentPage]);
5
6     const handleSubmit = async (e) => {
7       e.preventDefault();
8
9       if (state.searchTerm === "") {
10        return setState({ ...state, error: "Escribe un texto válido" });
11      }
12
13      const response = await fetch(`${API}&s=${state.searchTerm}`);
14      const data = await response.json();
15
16      if (!data.Search) {
17        return setState({ ...state, error: "No hay resultados" });
18      }
19
20      return setState({
21        data: data.Search,
22        totalResults: parseInt(data.totalResults),
23        searchTerm: "",
24        error: "",
25        currentPage: 1, // Reset current page when performing a new search
26      });
27    };
28  }
29)

```

Y cuando tenemos el primer componente creado lo mando a llamar en el index.js

```

JS index.js > container
import React from "react";
import { createRoot } from 'react-dom/client';
import './index.css'

import List from './containers/list';
import Navbar from './components/Card/Navbar'

const container = document.getElementById("root");
const root = createRoot(container); // createRoot(container!) si usas TypeScript

import "bootstrap/dist/lux/bootstrap.min.css";
import './index.css';

const App = () => {
  return (
    <>
      <Navbar />
      <List />
      <main className="gris-fondo">

```

```
import './index.css';

const App = () => {
  return (
    <>
      <Navbar />

      <main className="gris-fondo">
        <div className="container">
          <List />
        </div>
      </main>

    </>
  );
};
root.render(<App tab="home" />);
```

Despues de crear el componente list cree otro llamado card.js Ahí mando a llamar los campos de la api para que me muestre información básica como año, poster, titulo,etc.

```
1 import React from "react";
2 import PropTypes from "prop-types";
3
4 const Card = ({ movie }) => (
5   <div className="col-md-4">
6     <div className="card animated fadeInUp">
7       <img
8         src={movie.Posters}
9         alt={movie.Title}
10        className="card-img-top"
11        width="100"
12       />
13       <div className="card-body">
14         <h4>{`${movie.Title} (${movie.Year})`}</h4>
15         <p>{`Type: ${movie.Type}`}</p>
16       </div>
17     </div>
18   </div>
19 );
20
21 Card.propTypes = {
22   movie: PropTypes.shape({
23     id: PropTypes.string,
24     Title: PropTypes.string,
```

Luego cree el componente de css y ahi le mando el color de fondo

```
1 * {
2   padding: 0;
3   margin: 0;
4   box-sizing: border-box;
5 }
6
7 body {
8   background: #343A40;
9 }
10
11 .marginTop{
12   margin-top: 50px;
13 }
14
15 .gris-fondo {
16   background-color: #AFAFAF;
17 }
```

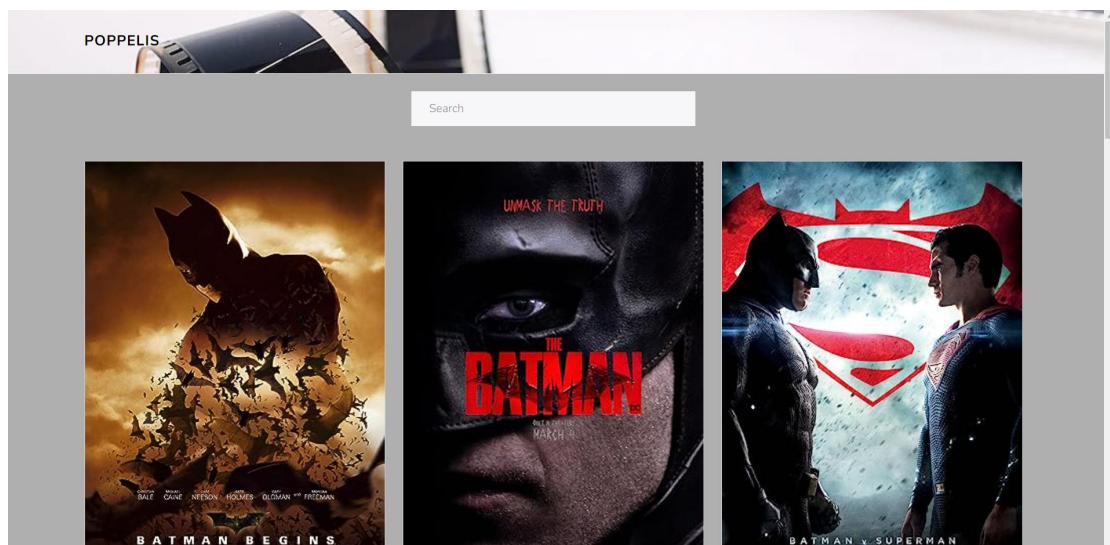
Para mostrar el contenido de las películas en columnas utilice bootstrap en los componentes directamente

```
components > Card > JS Card.js > ...
import React from "react";
import PropTypes from "prop-types";

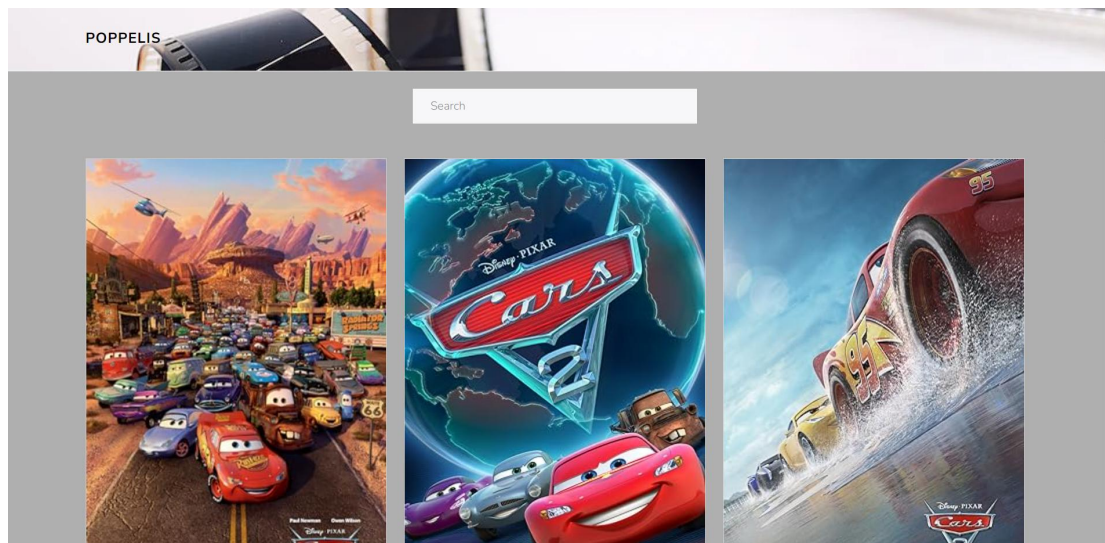
const Card = ({ movie }) => {
  <div className="col-md-4">
    <div className="card animated fadeInUp">
      <img
        src={movie.Poster}
        alt={movie.Title}
        className="card-img-top"
        width="100"
      />
      <div className="card-body">
        <h4>{`${movie.Title} (${movie.Year})`}</h4>
        <p>{`Type: ${movie.Type}`}</p>
      </div>
    </div>
  </div>
);

Card.propTypes = {
  movie: PropTypes.shape({
    id: PropTypes.string,
    Title: PropTypes.string,
```

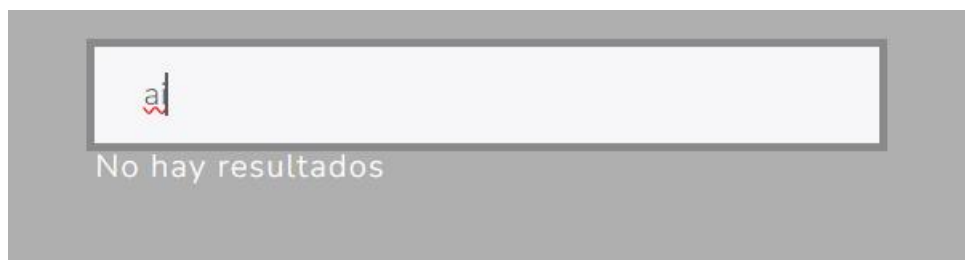
Si abrimos la app en nuestro local host con npm start nos va dar el siguiente resultado



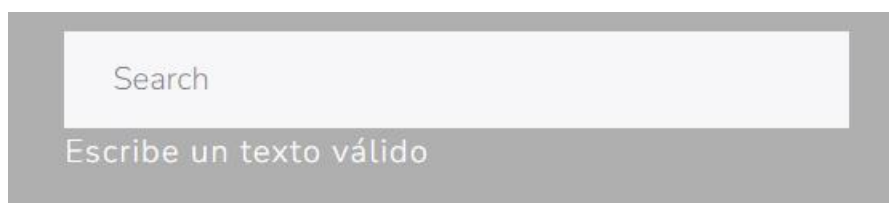
Podemos buscar las películas que queramos aquí yo busque cars y nos arroja el resultado de la api



Pero si busco algo como ai que es un resultado que no existe esta validado para que nos muestre un texto diciendo que ese resultado no existe



O si no escribimos nada y damos enter también esta validado



También se agrego una paginación y los resultados que encontró



Esto se hizo en un componente llamado pagination.js

```
import React from "react";

const Pagination = () => {
  return (
    <nav aria-label="Page navigation example" className="marginTop">
      <ul className="pagination justify-content-center">
        <li className="page-item disabled">
          <a className="page-link">Previous</a>
        </li>
        <li className="page-item"><a className="page-link" href="#">1</a></li>
        <li className="page-item"><a className="page-link" href="#">2</a></li>
        <li className="page-item"><a className="page-link" href="#">3</a></li>
        <li className="page-item">
          <a className="page-link" href="#">Next</a>
        </li>
      </ul>
    </nav>
  );
};

export default Pagination;
```

Para ver los parámetros de la api me guie con postman

The screenshot shows the Postman interface with a POST request to `http://www.omdbapi.com/?i=tt3896198&apikey=9e13ae90&type=movie&page=10`. The request parameters are:

Key	Value	Description
i	tt3896198	
apikey	9e13ae90	
type	movie	
page	10	

The response status is 200 OK, with a time of 155 ms and a size of 1.5 KB. The response body is in JSON format, showing details for "Guardians of the Galaxy Vol. 2".

```
{
  "Title": "Guardians of the Galaxy Vol. 2",
  "Year": "2017",
  "Rated": "PG-13",
  "Released": "05 May 2017",
  "Runtime": "136 min",
  "Genre": "Action, Adventure, Comedy",
  "Director": "James Gunn",
  "Writer": "James Gunn, Dan Abnett, Andy Lanning",
  "Actors": "Chris Pratt, Zoe Saldana, Dave Bautista",
  "Plot": "The Guardians struggle to keep together as a team while dealing with their personal family issues, notably Star-Lord's encounter with his father, the ambitious celestial being Ego.",
  "Language": "English",
  "Country": "United States",
  "Awards": "Nominated for 1 Oscar. 15 wins & 60 nominations total",
  "Poster": "https://m.media-amazon.com/images/M/MV5BNjM2NTc5MTI2M2YlYS80YzEwLWE8YmUtNTA2ZWZlZDc2OTRkXkE5XkF6cGdeOXVvNTgwNzIvNzQ@."
}
```


Y esta es la app ya funcionando haciendo una petición a una api externa

