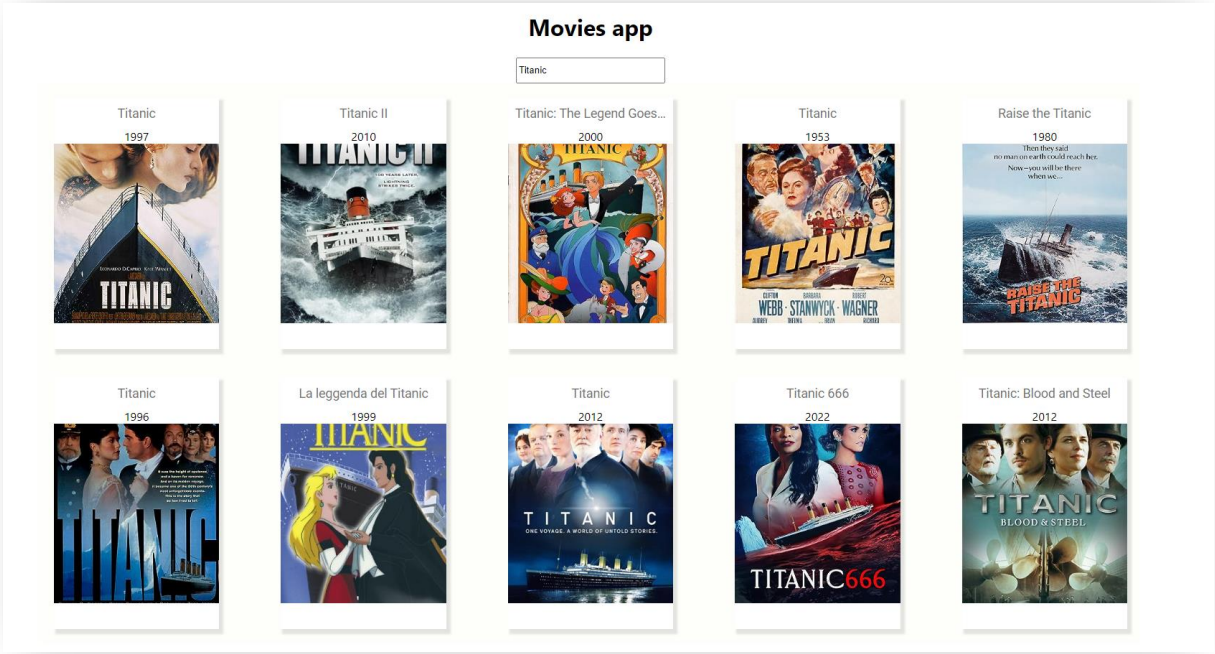
Kata React

Ce Kata s'appuie sur un application créée à partir de Create React App.

L'objectif final est d'afficher une liste de films comme montré ci-dessous :



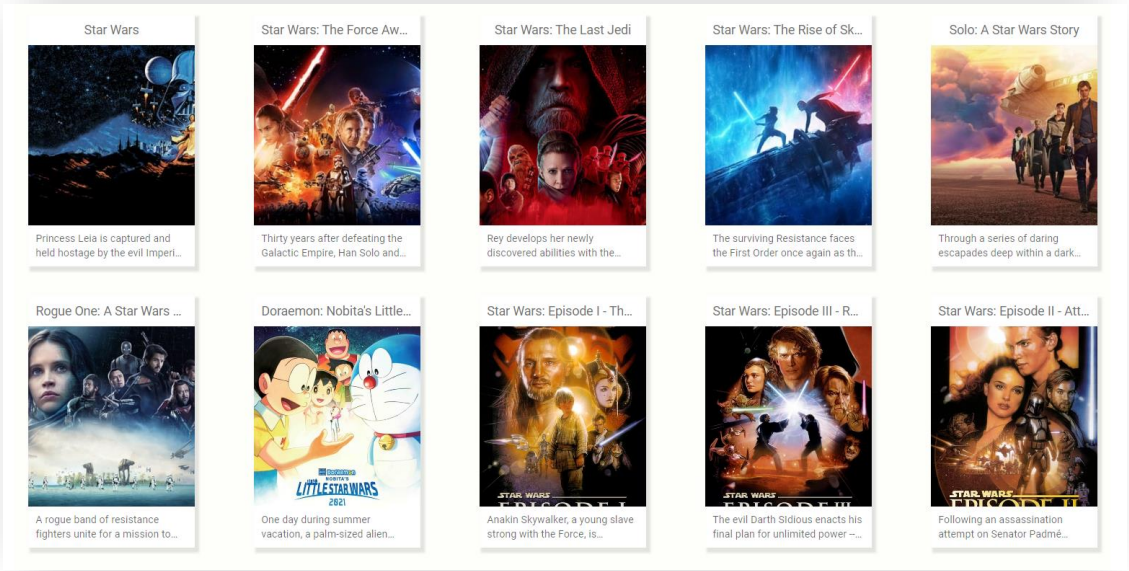
Lors de ce Kata, vous allez d'abord afficher une liste de films à partir d'un fichier json. Ce sera pour vous l'occasion de prendre connaissance des composants de l'application.

Par la suite, vous devrez ajouter le champ de recherche et un appel à une api pour récupérer la liste des films. Cela sera ici l'occasion de développer un composant (la textbox de recherche), d'utiliser des hooks et de faire un appel d'api dans un custom hook.

Nous allons procéder par étape.

Etape 1 – un peu de map, un peu de css :

Compléter le composant SearchPage.js pour afficher les cards (composant Card.js) comme montré ci-dessous :



Vous aurez certainement une petite correction css à faire afin d'avoir le résultat voulu.

Etape 2 - création d'un component :

Rendez-vous dans le composant SearchBar pour compléter l'implémentation :

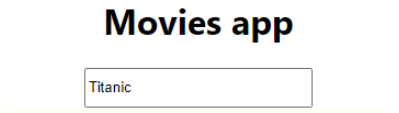
```
import React from 'react';
import './SearchBar.css'

const SearchBar = ()=>{

    return (<div></div>);
}

export default SearchBar;
```

Vous obtiendrez un simple Input de type text qui informera le composant de votre choix de sa valeur.



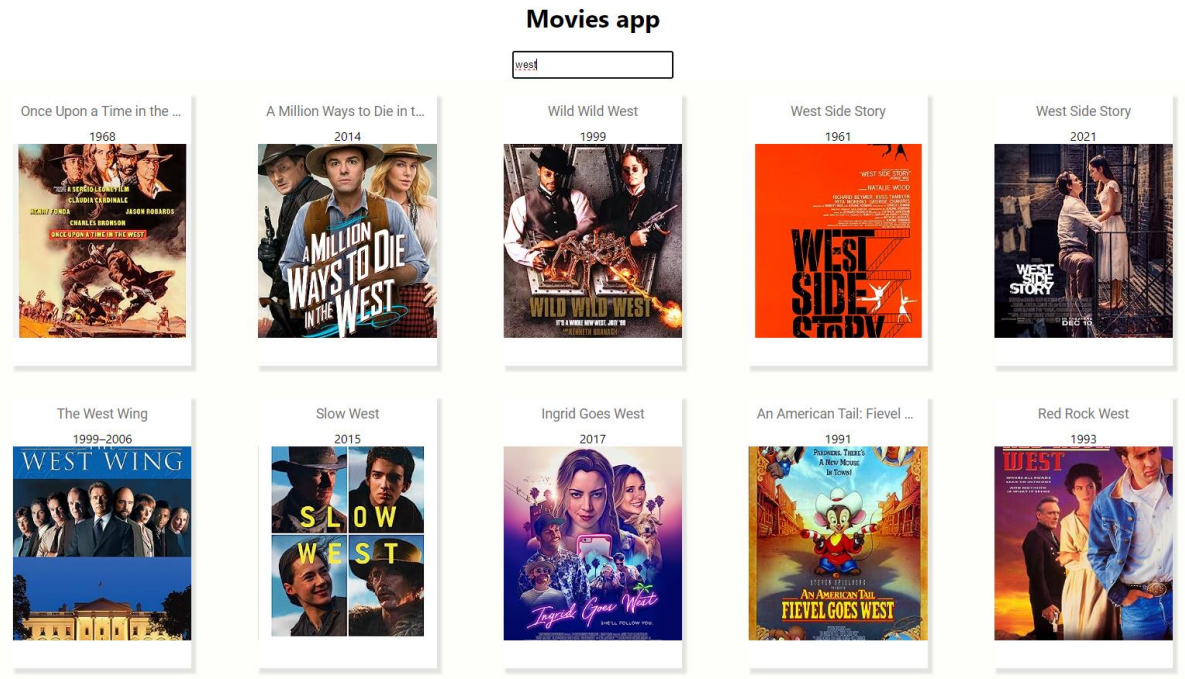
Etape 3 – appel d'api et utilisation de hook :

Dans un premier temps, vous pouvez faire un appel de l'api des films dans le composant SearchPage.js.

Vous allez retrouver l'url en question dans le code :

```
http://www.omdbapi.com/?apikey=417978c1&s=${searchTerm}
```

Vous devrez modifier le code de ce composant pour que la mise à jour de la recherche s'accompagne de la mise à jour de la liste des films :



Etape 4 – implémentation d'un custom hook :

Pour faire des appels d'apis, nous utilisons Axios. Vous allez devoir réaliser un custom hook, pour factoriser les appels d'apis. Vous ne devrez pas faire deux fois le même appel (l'url d'appel doit être différente à chaque fois).

AIDES

Pour l'étape 2 :

```
import React from 'react';
import './SearchBar.css'

const SearchBar = ({onSearchChange, searchValue})=>{
   return (<div><input className="searchinput" onChange={(e)=>onSearchChange(e.target.value)}
value={searchValue}></input></div>);
}

export default SearchBar;
```

Pour l'étape 4, le custom hook peut s'écrire ainsi :

```
const useAxios= (url)=>
{

        const [data, setData] = useState(null);

        useEffect(()=>{
                axios.get(url).then(
                 rep=>{console.log(rep.data.Search);
                setData(rep.data.Search);
        }
        );
 },[url])

  return data;

}
```