Programación Web Full Stack

A fondo con los Hooks

Práctica integradora

Objetivo

Ahora que ya tenemos conocimiento del manejo de APIs y que hemos tocado una y otra vez los estados de componente, ha llegado el momento de trabajar con los Hooks de React. Puntualmente en esta ejercitación vamos a estar trabajando con **useState**, **useEffect** y **useRef**.

¡Así que a "darle átomos"! 😎👍✨

Requisitos:

Vamos a utilizar los mismos archivos base de la clase anterior que podemos **descargar desde este enlace** (recordemos instalar todas las dependencias del proyecto, ejecutando **npm install** 😉). Para activar el servidor debemos ejecutar el comando **npm start** y finalmente en el navegador ejecutar: http://localhost:3000/.

📌 Micro desafío - Paso 1:

Vamos a estar trabajando con la siguiente API. Para usarla, necesitaremos una API Key, la cual podremos crear previamente desde este enlace. Recordemos elegir la opción "**FREE!**" y registrarnos con un **email válido**, ya que allí recibiremos la API Key.

Tengamos presente que dentro del mensaje que recibiremos, **deberemos hacer clic en el enlace de "ACTIVACIÓN".**

Hecho esto, ya podremos acceder a la API, pero antes recomendamos probar que la API Key funciona correctamente. Para ello, debemos ingresar en el navegador a la siguiente dirección: http://www.omdbapi.com/?i=tt3896198&apikey=**ABC**

Recordemos cambiar **ABC** por tu API Key. Si todo va bien, deberíamos ver algo así:

Raw Parsed

```
{
  "Title": "Guardians of the Galaxy Vol. 2",
  "Year": "2017",
  "Rated": "PG-13",
  "Released": "05 May 2017",
  "Runtime": "136 min",
  "Genre": "Action, Adventure, Comedy, Sci-Fi",
  "Director": "James Gunn",
  "Writer": "James Gunn, Dan Abnett (based on the Marvel comics by), Andy
  Lanning (based on the Marvel comics by), Steve Englehart (Star-Lord created
  by), Steve Gan (Star-Lord created by), Jim Starlin (Gamora and Drax created
  by), Stan Lee (Groot created by), Larry Lieber (Groot created by), Jack Kirby
  (Groot created by), Bill Mantlo (Rocket Raccoon created by), Keith Giffen
  (Rocket Raccoon created by), Steve Gerber (Howard the Duck created by), Val
  Mayerik (Howard the Duck created by)",
  "Actors": "Chris Pratt, Zoe Saldana, Dave Bautista, Vin Diesel",
  "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": "USA",
  "Awards": "Nominated for 1 Oscar. Another 15 wins & 57 nominations.",
  "Poster": "https://m.media-
  amazon.com/images/M/MV5BNjM0NTc0NzItM2FlYS00YzEwLWE0YmUtNTA2ZWIzODc2OTgxXkEyX
  FqcGdeQXVyNTgwNzIyNzg@._V1_SX300.jpg",
  "Ratings": [
    {
      "Source": "Internet Movie Database",
      "Value": "7.6/10"
    },
    {
      "Source": "Rotten Tomatoes",
      "Value": "85%"
    },
    {
      "Source": "Metacritic",
      "Value": "67/100"
    }
```

Si podemos ver lo anterior (aunque sea en otro formato), ya tenemos acceso a la API y ahora sí vamos a meternos dentro de React a fondo.

Micro desafío - Paso 2:

Ahora vamos a crear un **Buscador de películas** usando esta API. Pero como somos muy buena gente, dentro de la carpeta de componentes, van a encontrar un componente llamado:

```
<SearchMovies />
```

Este ya tiene toda la estructura necesaria, e incluso van a poder acceder al mismo desde la barra lateral de enlaces de la aplicación.

Este componente de momento tiene un array de películas "hardcodeado", pero nuestro objetivo será:

- Usar **setState** para crear un estado *movies* con un array vacío como valor inicial y también su respectiva función para actualizar: *setMovies*.
- Usar **useEffect** para hacer un llamado asincrónico a la API recién se monte el componente. La URL a donde deberemos hacer el pedido será esta:

  http://www.omdbapi.com/?s=KEYWORD&apikey=APIKEY

  (recordemos poner nuestra API Key y, como keyword, podemos probar con la palabra **action**)

- Actualizar el estado *movies* con las películas que vengan de la API. Tengamos presente ver la estructura de datos que nos da la API para saber cómo vamos a recibir los datos.
- Mostrar y renderizar las películas que vengan de la API en el componente.

Micro desafío - Paso 3:

Ahora vamos a permitir que cualquier persona pueda buscar películas. Para ello, tendremos que:

- Generar una función que se ejecute en el evento **onSubmit** del formulario.
- Capturar el valor del campo de texto. Para esto quizás nos sirva el Hook **useRef**.

Pero, antes de continuar, quizás hayamos notado que la URL puede tener una parte dinámica. Más exactamente donde va el keyword (?s=*keyword*).

Con esto en mente, veamos cómo podemos hacer para que el texto capturado en el campo de texto pueda reemplazar esa parte.

Para ello, recomendamos:

- Usar **useState** para crear un estado que almacene esa keyword.
- Actualizar el estado anterior dentro de la función creada para el **onSubmit**. Pasándole como valor el texto capturado del input.

Con esto estaremos en condiciones de actualizar ese estado, pero ¿cómo podemos hacer para volver a hacer la petición a la API? ¿Será que es necesario volver a hacer un llamado a la API?

Para esto último quizás nos sirva revisar lo que hicimos en el desafío 2, y ver si al **useEffect** le podemos hacer algo para que esté pendiente a este nuevo estado.

Conclusión

Si llegamos hasta aquí resolviendo todos los desafíos, es una excelente oportunidad para felicitarnos por el excelente trabajo.

¡Nos veremos!