GifExpertApp - Aplicación

¿Qué veremos en esta sección?

Custom Hooks

Fetch hacia un API

Comunicación entre componentes

Clases de CSS

Animaciones

Enviar métodos como argumentos

Crear listados

keys

Giphy

Esta es una aplicación pequeña pero muy ilustrativa que explica cómo utilizar React + customHooks para poder resolver necesidades en específico que podremos re-utilizar después.

1. Documentación para estructurar archivos:

https://hackernoon.com/structuring-projects-and-naming-components-in-react-1261b6e18d76

https://es.reactjs.org/docs/faq-structure.html

2. Listas en React

¿Porque las listas necesitan una key?: https://www.geeksforgeeks.org/reactjs-keys/#:~:text=A%20%E2%80%9Ckey%E2%80%9D%20is%20a%20special,the%20elements%20in%20the%20lists.

Para un list item deberiamos usar como key el value del elemento y no el index (en caso de utilizar un array). Ya que si reformulo el array al utilizar un index es muy posible que generemos conflictos al re renderizar. Con una key asociativa nos ahorramos este problema. Las keys numericas suelen ser de bases de datos.

Manera de mainpular un array con useState: setCategories([...categories, 'Invincible']); De esta manera mantengo mi arreglo anterior y le agrego un nuevo elemento. Si tengo un array vacio en la primera renderizacion del useState simplemente le paso la referencia del array.

3. Comunicación entre componentes

Para comunicarnos entre componentes podemos pasarle las funciones del hook como prop a otro componente.

https://stackoverflow.com/questions/59555534/why-is-json-asynchronous#:~:text=json()%20operation%20ends%20up,incoming%20stream%20that%20is%20asynchronous.

4. useEffect

Documentación: https://es.reactjs.org/docs/hooks-effect.html

useState va a renderizar TODO el componente independimente si en el fragmento utilizamos una funcion o no, actualizando todas las referencias del component. Por ejemplo si tenemos una variable que almacena imagenes que traemos desde una API y si bien estas imagenes no las utilizamos en si en un fragmento, al ser parte del componente va a volver a realizar la peticion y el guardado.

useEffect nos permite ejecutar un codigo de manera condicional. useEffect se compone de un callback y un arreglo de dependencias.

Ejemplo de useEffect:

```
useEffect(() => {
  getGifs();
}, [])
```

Si en el arreglo no enviamos nada por defecto se ejecuta una unica vez.

En el arreglo se puede ejecutar una evaluacion. Por ejemplo el valor que le pasemos al useEffect si cambia que vuelva a enviarse. Puedo tener algo como [value] y si ese value !== value se vuelve a lanzar el useEffect, como se ve se compara con si mismo para saber si cambio y se tiene que ejecutar.

4. Clases CSS en React

Como class es una palabra reservada de JavaScript vamos a utilizar className para colocar CSS en el JSX.

5. Encoding para peticiones

Con el metodo encodeURI(STRING) JS se va a encargar de formatear un string con espacios o caracteres especiales y así incustrarlo en una URL.

6. helpers

Para no tener un codigo muy cargado en un componente, todo lo que sea procesar informacion no crucial, pedir informacion, etc. Lo vamos a ubicar en una carpeta helpers para esta app.

Un ejemplo puede ser un getter de informacion.

7. Custom Hooks

Los custom hooks es una forma de extraer logica de algun componente o logica que yo quiero reutilizar y extraerla de tal manera que sea sencillo utilizarla nuevamente. Pueden funcionar como Functional componentes, teniendo estado, efectos, contexto, etc.

Los custom hooks normalmente van en un directorio llamado hooks.

Los custom hooks llevan use al principio, por ejemplo: useFetchGifs.js

8. Animaciones para CSS

https://animate.style/