**main.js**

----------

import React from 'react'

import ReactDOM from 'react-dom/client'

import { GifExpertApp } from './GifExpertApp';

import './styles.css'

ReactDOM.createRoot(document.getElementById('root')).render(

<React.StrictMode>

<GifExpertApp />

</React.StrictMode>,

);

GifExpertApp.jsx

import { useState } from 'react';

// Aqui importamos los componentes por medio del index.js

// creado y colocado en la carpeta components. index.js

// contine las exportaciones de componentes como AddCategory.jsx

// forma de agrupar los imports de componentes. Para ello se

// debe importar el index.js pero no es necesario explicitamente

// sino con solo apuntar a la carpeta que lo contiene, javascript

// lo primero que levantara sera el index.js de la siguiente forma.

// y no de la forma: import { AddCategory, GifGrid } from './components/index'

import { AddCategory, GifGrid } from './components'

// Functional Component - usar el snipper: rafc + tab

export const GifExpertApp = () => {

const [categories, setCategories] = useState(['One Punch'])

const onAddCategory = ( newCategory ) => {

// Valida no exista en el arreglo

if ( categories.includes(newCategory) ) return;

// console.log(newCategory);

setCategories( [newCategory, ...categories] )

}

return (

// <> </> se llaman fragmentos: fragments

<>

{/\* titulo \*/}

<h1>Gif Expert App</h1>

{/\* Input para busquedas \*/}

<AddCategory

// setCategories={ setCategories }

onNewCategory={ value => onAddCategory(value) }

/>

{/\* Result o: Grid de tarjetas/Items/Gifs \*/}

{/\* <button onClick={ onAddCategory }>Agregar</button> \*/}

{/\* Eliminamos el <ol></ol> debido a que ya no tiene razon de ser \*/}

{/\* <ol> \*/}

{

categories.map( ( category ) => (

<GifGrid

key={ category }

category={ category }

/>

))

}

{/\* </ol> \*/}

</>

)

}

GifGrid.jsx

// import { useState, useEffect} from "react";

import { GifItem } from "./GifItem";

// import { getGifs } from "../helpers/getGifs";

import { useFetchGifs } from "../hooks/useFetchGifs";

export const GifGrid = ( { category } ) => {

// Creacion de un Custom Hook (Hook Personalizado)

//

const { images, isLoading } = useFetchGifs( category )

// console.log( { images, isLoading } );

// Esta parte fue comentada y pasada debido a la creacion del Custom Hook arriba.

// ==============================================================================

// // Firma de useState

// // const [counter, setCounter] = useState(10);

// // Using useState to manage the state with images.

// const [images, setImages] = useState([]);

// const getImages = async() => {

// // We going to find getGifs in ../helpers/getGifs

// const newImages = await getGifs( category );

// setImages(newImages);

// }

// // Firma de useEffect

// // si dejo la parte de lista dependencias vacia [ ]

// // significa que el Hook useEffect solo se va a disparar

// // la primera vez que se crea y se construye el compomente GifGrid.

// useEffect( () => {

// getImages();

// }, [] )

// ==============================================================================

return (

<>

<h3> { category } </h3>

{/\* 1er forma crear un mensaje del Loading: Usando un ternario \*/}

{/\* {

isLoading ?

( <h2>Loading.....</h2> ) :

null

} \*/}

{/\* 1er forma crear un mensaje del Loading: Usando un && (AND logico) \*/}

{/\* esto es llamado: If corto con una sola condicion \*/}

{/\* Esta fue la forma que dejo el profe \*/}

{

isLoading && ( <h2>Loading.....</h2> )

}

<div className="card-grid">

{

images.map( ( image ) => (

<GifItem

key={ image.id }

{ ...image }

/>

))

}

</div>

{/\* <h5> { counter } </h5> \*/}

{/\* <button onClick={ () => setCounter( counter + 1 ) } >+1</button> \*/}

</>

)

}

GifItem.jsx

export const GifItem = ( { title, url, id } ) => {

// console.log( { title, url } );

return (

<div className="card">

<img src={ url } alt={ title } />

<p> { title } </p>

</div>

)

}

AddCategory.jsx

import { useState } from "react"

// Functional Component - usar el snipper: rafc + tab

export const AddCategory = ( { onNewCategory } ) => {

const [inputValue, setinputValue] = useState('');

const onInputChange = ( { target } ) => {

// console.log( target.value );

setinputValue( target.value );

}

const onSubmit = ( event ) => {

// Evita se haga un REFREESH DEL NAVEGADOR WEB

event.preventDefault();

// console.log( inputValue );

// Valida que tenga valores

if( inputValue.trim().length <= 1 ) return;

// Agrega una nueva categoria.

// setCategories( categories => [ inputValue, ...categories] )

onNewCategory( inputValue.trim() );

// Limpia el componente input

setinputValue('');

}

return (

<form onSubmit={ onSubmit }>

<input

type="text"

placeholder="Buscar Gifs"

value={ inputValue }

onChange={ onInputChange } // Mejor forma se obvia poner el event ya que se le envia lo mismo el event

// onChange={ (event) => onInputChange(event) }

/>

</form>

)

}

getGifs.js

export const getGifs = async( category ) => {

const url = `http://api.giphy.com/v1/gifs/search?api\_key=dloULfTeGAjcZez5ry0M0eRp98YmiQVJ&q=${ category }&limit=10`;

const resp = await fetch( url );

const { data } = await resp.json();

const gifs = data.map( img => ({

id: img.id,

title: img.title,

url: img.images.downsized\_medium.url

}))

// Console Log

// console.log( gifs );

// Return Gifs

return gifs;

}

useFetchGifs.js

import { useEffect, useState } from "react";

import { getGifs } from "../helpers/getGifs";

export const useFetchGifs = ( category ) => {

// Firma de useState

// const [counter, setCounter] = useState(10);

// Using useState to manage the state with images.

const [images, setImages] = useState([]);

const [isLoading, setIsLoading] = useState( true );

const getImages = async() => {

// We going to find getGifs in ../helpers/getGifs

const newImages = await getGifs( category );

setImages(newImages);

setIsLoading(false);

}

// Firma de useEffect

// si dejo la parte de lista dependencias vacia [ ]

// significa que el Hook useEffect solo se va a disparar

// la primera vez que se crea y se construye el compomente GifGrid.

useEffect( () => {

getImages();

}, [] )

return {

// En EMScript 6, cuando encontramos images: images / solo devolvemos images una vez.

images,

isLoading

}

}

**Index.js**

-----------

export \* from './AddCategory';

export \* from './GifGrid';

export \* from './GifItem';