GIFT EXPERT

Este es el comienzo de un proyecto donde se uso la busqueda de gif, se usaron algunas de las funcionalidades importantes de react, por lo tanto acontinuación veras lo más importante

ClassName

las clases en react son llamadas como className, esto para no confundirse con las class de js

Ejemplo: `<p className="text-center">Hola</p>` los estilos van en forma de objeto **Ejemplo:** `<p style={{marginTop:'15px'}}>Hola</p>`

useEffect

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

Sirve para prevenir ciclos infinitos, un ejemplo a este es cuando hay un cambio de estado en el componente, este vuelve a reenderizar todo de nuevo

Ejemplo:

```
const [state, setState] = useState( 0 );
const function = () => {
    console.log('hola mundo')
}

function();



return (
    <>
        <button onClick={ () => setState( state + 1 ) } > Oprimir </button>
    </>
)
```

En el ejemplo anterior siempre que cambie el setState function se va a llamar de nuevo, ya que el componente se reenderiza de nuevo, con el useEffect podemos prevenir esto, hasta ciclo infinitos, debido a que si hay un cambio de estado dentro de la funcion funtion, este cambio de estado llamara a la funcion de nuevo y se genera otro cambio de estado y asi sucesivamente

```
const [state, setState] = useState( 0 );

//Se llamara una vez, no importa cuantas veces se reenderize el componente
useEffect(() => {
    console.log('hola mundo');
}, []);



return (
    <>
        <button onClick={ () => setState( state + 1 ) } > Oprimir </button>
    </>
)
```

en el siguiente ejemplo se ejecutara el useEffect cada vez que su dependencia cambie

```
const [counter, setCounter] = useState( 0 );

useEffect(() => {
    console.log('hola mundo');
}, [ counter ]);



return (
    <>
        <button onClick={ () => setCounter( counter + 1 ) } >Counter</button>
    </>
)
```

Siempre que se oprima el boton, se mostrara el mensaje de "hola mundo"

Helpers

son archivos js que exportan algunas funciones, arrays importantes que serán consumidas por varios componentes de la pagina.

Ejemplo

```
export const getGifs = async (busqueda = '') => {

    const key = 'lbPAxV2cJQiFDExsZhtZJ6T3F12pZN4l';
    const resp = await fetch(`https://api.giphy.com/v1/gifs/search?q=${ busqueda }&limit=10&api_
    const { data } = await resp.json();

    return data.map( img => {
        return {
            id: img.id,
            title: img.title,
            url: img.images?.downsized_medium.url
        }
    });

};
```

es una funcion que retornara la respuesta de una api, esta respuesta puede ser consumida posteriormente por un customHook o desde un componente, lo tenemos centralizado y no en un solo componente

CustomHooks

Es una forma de extraer la logica de un componente que se quiera reutlizar en cualquier componente

```
import { useEffect, useState } from "react"
import { getGifs } from "../helpers/getGifs";

export const useFetchGif = ( category ) => {

    const [state, setstate] = useState({
        data: [],
        loading: true
    })

    useEffect(async () => {

        const images = await getGifs( category );
        setstate({data: images, loading: false});

    }, [ category ]);



    return state;

}
```

Tenemos a ahora un customHook que podemos llamar desde cualquier componente, la forma de llamarlo seria `const { state } = useFetchGif('one punch man')` siempre que haya un estado de cambio se recibira en el state

por ejemplo cuando el componente que llama al custoHook incie, primero su state será `{data: [], loading: true}`, luego que se ejecute o que se haga la peticion al api el state cambiara a los valores que la api trae `{data: images, loading: false}`