Front End III

Creamos nuestras peticiones asíncronas

Vamos a construir una estructura para llamar a nuestra API, que se basará en el **useState** para almacenar los datos que recibiremos de nuestra API.

Al principio, crearemos una función y pondremos algunos parámetros dentro para crear el useState. Para iniciar, tenemos que importar el **useState** y el **useEffect**:

```
import React, {useState, useEffect} from 'react';

export default function EffectExample(){
   //useState
   const [vetor, setVetor] = useState([]);
```

Ahora necesitamos la función propiamente dicha que llamará a la API:

```
   //Función para llamar los datos de la API
   const getData = async () => {
       const data = await
fetch('https://jsonplaceholder.typicode.com/comments')
       const convert = await data.json();
       setVetor(convert)
   }
```

Lo que podemos ver en la función anterior es el concepto de **asincronía,** utilizando el famoso **async/await.** Esto nos permite que nuestro estado reciba los datos de la API una

vez que esta nos haya devuelto dicha información, evitando así que colapse el resto de nuestra aplicación.

Contamos con el uso de **fetch** que interceptará parte de la petición, trayendo los datos de la API de forma asíncrona y en formato de texto. Sin embargo, debemos recordar que tenemos un vector que recorre un objeto (usando el estado), así que usaremos para nuestra aplicación el formato **.json()**.

Para facilitarnos la vida, podemos utilizar el método **.json()** para convertir los datos recibidos en la variable date de texto a JSON. Después de crear la variable que se encargará de convertir nuestros datos, debemos utilizarla como parámetro que se pasará a nuestro **setVector**.

Bien, hemos configurado nuestra función que recibirá los datos de la API, ahora necesitamos enviar estos datos de forma organizada al navegador.

Una forma de hacerlo es utilizando **.map()** para recorrer cada elemento de nuestro estado, que será el vector. La estructura es más o menos así:

```
return(
    <div>
        <ul>
            {vetor.map(object => (<li>{object.email}</li>))}
        </ul>
    </div>

)
```

En la estructura anterior, tenemos nuestro vector (estado) utilizando map para recorrer cada objeto (al que llamamos object) y, recorriendo cada uno de ellos, nos devolverá una estructura que definamos, en este caso es un **<li>**, que tendrá como elemento los emails (datos devueltos desde la API) de cada usuario que esté en comentarios:

```
{
    "postId": 1,
```

```
    "id": 1,
    "name": "id labore ex et quam laborum",
    "email": "Eliseo@gardner.biz",
    "body": "laudantium enim quasi est quidem magnam voluptate
ipsam eos\ntempora quo necessitatibus\ndolor quam autem
quasi\nreiciendis et nam sapiente accusantium"
  },
```

¡Pero eso no es todo lo que necesitamos para que nuestra aplicación funcione! Si nos fijamos, podemos observar que no llamamos a la función **getData()**, que hemos definido anteriormente.

Para llamarla, podemos utilizar **useEffect()**, que es un hook especializado en trabajar con el ciclo de vida de nuestro componente y es excelente para trabajar con retornos de la API (como es nuestro caso). La estructura para utilizarlo en nuestra aplicación es la siguiente:

```
    //useEffect
    useEffect(() => {
        getData();
    });
```

¡Ahora sí! Con todo esto, podemos probar nuestra aplicación en el navegador ejecutando un **npm start** para obtener este resultado:

- Eliseo@gardner.biz
- Jayne_Kuhic@sydney.com
- Nikita@garfield.biz
- Lew@alysha.tv
- Hayden@althea.biz
- Presley.Mueller@myrl.com
- Dallas@ole.me
- Mallory_Kunze@marie.org
- Meghan_Littel@rene.us

- Carmen_Keeling@caroline.name
- Veronica_Goodwin@timmothy.net
- Oswald.Vandervort@leanne.org
- Kariane@jadyn.tv
- Nathan@solon.io
- Maynard.Hodkiewicz@roberta.com
- Christine@ayana.info
- Preston_Hudson@blaise.tv
- Vincenza_Klocko@albertha.name
- Madelynn.Gorczany@darion.biz
- Mariana_Orn@preston.org
- Noemie@marques.me
- Khalil@emile.co.uk
- Sophia@arianna.co.uk
- Jeffery@juwan.us
- Isaias_Kuhic@jarrett.net
- Russel.Parker@kameron.io
- Francesco.Gleason@nella.us