Reiniciando Formulario

- Para limpiar el formulario, voy a Formulario.jsx
- En el onSubmit toma directamente los valores
- Puedo poner una coma y extraer con desestructuración la función resetForm
- Entonces le puedo decir que una vez que envíe los datos los borre del form
- Es un arrow function. Le puedo poner el async y que espere a enviar los datos

```
onSubmit={ async (values, {resetForm})=>{
          await handleSubmit(values)
          resetForm();
        }}
```

- Cuando se agrega un cliente, quiero redireccionar a Clientes para mostrar lo que tengo en la API
- Para redireccionar usaré el hook useNavigate de react-router-dom. Lo importo
- Guardo el hook en la variable navigate

```
const navigate= useNavigate()
```

- Lo coloco en el handleSubmit

```
  const handleSubmit=async(valores)=>{
    try {
      const url="http://localhost:4000/clientes"

      const respuesta = await fetch(url, {
        method: 'POST',
        body: JSON.stringify(valores),
        headers:{
          'Content-Type':'application/json'
        }
      })
        const resultado = await respuesta.json()
        navigate('/clientes')

    } catch (error) {
      console.log(error)
    }
  }
```

- Reiniciar el formulario y redireccionar son buenas prácticas para no duplicar datos

Consultar la APi para obtener resultados

- Quiero mostrar los clientes en la pestaña clientes
- Usaré un useEffect en Inicio porque voy a llamar a la APi cuando el componente esté listo. Lo importo y lo coloco antes del return.
- Le paso el arreglo de dependencias vacío para que se ejecute una sola vez

```
useEffect(()=>{
   const obtenerClientesAPI = async()=>{

    try {
      const url="http://localhost:4000/clientes"

      const respuesta= await fetch(url);

      const resultado= await respuesta.json()
      console.log(resultado)

    } catch (error) {
      console.log(error)
    }
  }
  obtenerClientesAPI();
},[])
```

- Creo un state y lo llamo clientes, setClientes y lo inicio con un arreglo vacío
- En lugar del console.log pongo setClientes(resultado), lo que hace es llenar el state clientes con el resultado, que es el resultado de la promesa con los clientes