Creando componente para ver-cliente

- Puedo usar el useNavigate en el onClick del button con un arrow function para redireccionar
- Lo hago entre backticks para inyectarle el id dinámico que estoy extrayendo de la desestructuración

```
  <button
    className="bg-yellow-600 hover:bg-yellow-700 block
    w-full text-white p-2 uppercase font-bold text-xs"
    type="button"
    onClick={()=>navigate(`/clientes/${id}`)}>Ver</button>
```

- Voy a App.jsx y en el path de un elemento que no he creado todavía llamado VerCLiente, lo dejo sólo con :id
- Al tener esos dos puntos, lo va a tratar como una variable
- Ahora, cómo hago la consulta del id para asociarlo con los id de la API?
- Para eso hay otro hook llamado useParams(). Lo meto en la variable params y le hago un console.log

```
const params = useParams()

console.log(params)
```

- Lee de forma dinámica lo que hay en la url
- Me devuelve en consola un objeto con el id.
- Puedo extraer el id de params con desestructuracion

```
const VerCliente = () => {

    const {id} = useParams()

    return (
    <h1>Ver cliente</h1>
  )
}
```

Mostrando la info de cliente por su id

- Hay que hacer de nuevo una consulta a la APi desde ver cliente para mostrar la info
- Importo useEffect y el useState que usaré después
- Cuando se cargue el componente, cargará una vez y hará un llamado a la API, por eso el arreglo vacío
- Meto la url en un templateString para hacer dinámica la url con el id
- Pongo un console.log dentro del try con resultado, para visualizar el objeto que ya tengo

- Ese objeto es el que debo meter en el setState

```
const [cliente, setCliente] = useState({})
    const {id} = useParams()
    useEffect(()=>{
        const ObtenerClienteApi= async()=>{
            try {
                const url=`http://localhost:4000/clientes/${id}`
                const respuesta = await fetch(url)
                const resultado = await respuesta.json()
                setCliente(resultado)
            } catch (error) {
               console.log(error)
            }

        }
        ObtenerClienteApi()

    },
    [])
```

- Ahora solo tengo que darle formato con el html

```
<p className="text-2xl text-gray-700">
            <span className="uppercase font-bold">Cliente: </span>
            {cliente.nombre}
        </p>
```

- Lo duplico el número de campos a mostrar

Añadiendo componente cuando no hay id

- Si me fijo, si coloco un id que no existe en consola muestra en el state un objeto vacío
- Puedo usar eso como condición en un ternario para mostrar el componente

```
  Object.keys(cliente).length ===0 ? <p>No hay Resultados</p>:(
    //Aqui dentro todo lo que hay en el return de Ver Cliente
```

Edición de clientes

- Voy al botón de editar y en el onClick le coloco un arrow function con un navigate con backticks para inyectarle el id

```
<button
   className="bg-blue-600 hover:bg-blue-700 block
    w-full text-white p-2 uppercase font-bold text-xs mt-3"
   type="button"
    onClick={()=>navigate(`/clientes/editar/${id}`)}>Editar</button>
```

- Copio y pego el contenido de NuevoCliente a EditarCliente
- Necesito importar useEffect para consultar la API, useState para setear el state con el resultado y params para leer el id que hay en la url
- Todo funciona exactamente igual que en VerCliente, porque necesito obtener el cliente que voy a editar. Copio y pego

```
const [cliente, setCliente] = useState({})
 const {id} = useParams()


 useEffect(()=>{
     const ObtenerClienteApi= async()=>{
         try {
             const url=`http://localhost:4000/clientes/${id}`
             const respuesta = await fetch(url)
             const resultado = await respuesta.json()
             setCliente(resultado)
         } catch (error) {
            console.log(error)
         }

     }
     ObtenerClienteApi()

 },
 [])
```

- Le paso por props a Formulario el cliente y lo extraigo en Formulario
- Formik tiene una prop llamada enableReinitialize que sirve para tomar valores iniciales que vienen desde una API. Por default es false
- La coloco en true después de initialValues
- Voy a usar default.props con cliente como objeto vacío para el nuevoCliente
- Funciona como los parámetros por default de las funciones

```
Formulario.defaultProps ={
  cliente: {}
}
export default Formulario
```

- Si no está presente, entran estas defaultprops
- Si ahora voy a NuevoCliente tiene un prop llamado cliente que está con un objeto vacío
- Solucionado esto, ahora si puedo usar initialValues con una condición.
- Si a initialValues le coloco cliente.nombre, lo imprime en el formulario
- Eso es gracias al enableReinitialize

> Si quito enableReinitialize aparece en el objeto en consola pero no lo imprime en el campo. Por eso es importante.

- Ahora, si le doy a nuevoCliente y miro el initialValues en consola lo marca como undefined
- La idea es que inicie como un string vacío
- Para eso le añado un ternario con esta sintaxis
- Si está cliente.nombre añádelo, si no string vacío

```
      <Formik
        initialValues ={{
          nombre: cliente?.nombre ?? "",
          empresa: cliente?.empresa ?? "",
          email:cliente?.email ?? "",
          telefono:cliente?.telefono ?? "",
          notas:cliente?.notas ?? ""
        }}
        enableReinitialize={true}
        onSubmit={async(values, {resetForm})=>{
          await handleSubmit(values)

          resetForm();
        }}
```

- Para mostrar el texto de fomra condicional

```
<h1 className="text-gray-600 font-bold text-3xl uppercase text-center">
      {cliente.nombre ? "Editar Cliente": "Agregar Cliente"}
    </h1>
```

- Hago lo mismo con el botón
- Este parámetro ?? significa si el valor de la izquierda existe asignalo, y si no el de la derecha
- Coloco un ternario en EditarCliente para que no muestre el formulario si no hay cliente
    - en el caso de poner un id que no existe mostrará editar pero no el

```
    return (
      <>
      <h1 className= "font-black text-4xl text-blue-400">Editar Cliente</h1>
         <p className="mt-3">Utiliza este formulario para editar a un cliente</p>

     {cliente.nombre ? (

         <Formulario
              cliente={cliente}/>
     ): "No hay un cliente con ese id"

     }
    </>
    )
```