

Clase 21

Rutas y navegación (*)

Cuando definimos una página web, la misma puede estar conformada por un menú principal, desde el cual se acceden a distintos componentes o funcionalidades de la aplicación. Cuando el usuario realiza click en un enlace o ítem del menú, se requiere de un proceso de enrutamiento para la carga del nuevo componente o página. En las páginas web tradicionales el ruteo se realizaba en el servidor. Los ruteos del lado del cliente (browser) permiten que la aplicación actualice la URL sin necesidad de hacer un request o pedido al servidor, sino que inmediatamente renderiza el componente de la interfaz del usuario.

El ruteo del lado del cliente se habilita en React, agregando una librería de ruteo, y luego crear un "Router" y enlazando las páginas con 'Link' y 'Form'.

En nuestro caso vamos a utilizar la librería <https://reactrouter.com/>

Para instalar la librería utilizar el siguiente comando:

```
npm install react-router-dom
```

Ventajas del ruteo del lado del cliente (*)

- Permite experiencias de usuario más rápidas porque el navegador no necesita solicitar un documento completamente nuevo o volver a evaluar los activos.
- Permite experiencias de usuario más dinámicas porque es posible aplicar animaciones.

Agregar una Ruta (*)

Lo primero que necesitamos agregar a nuestro código para agregar una Ruta es crear un objeto llamado **BrowserRouter** y configurar allí nuestra primer ruta. Esto va habilitar en el browser el ruteo del lado del cliente para nuestra aplicación web. En el objeto **BrowserRouter** vamos a configurar las diferentes rutas de nuestra aplicación. Además, podemos agregar rutas hijas para hacer una navegación dentro de una página en particular. Lo único que debemos agregar extra es el componente **<Outlet/>**.

Una vez que haya instalado la biblioteca, podemos comenzar a utilizarla en nuestro proyecto. Lo primero que debemos hacer es importar algunos componentes de "React Router" en nuestro archivo de "App.js". Agregue las siguientes líneas de código al principio de su archivo:

```
import { BrowserRouter, Route, Routes, Navigate } from "react-router-dom";
```

Ahora, debemos **envolver toda nuestra aplicación en un componente "BrowserRouter"**. Este componente es el encargado de manejar todas las rutas de nuestra aplicación. Para hacer esto, agregue el siguiente código alrededor de todo el contenido en su archivo "App.js":

```
function App() {  
  return (  
    <BrowserRouter>  
      <div className="App">  
        // Todo el contenido de su aplicación aquí  
      </div>  
    </BrowserRouter>  
  );  
}  
  
export default App;
```

Ahora podemos comenzar a definir las rutas de nuestra aplicación. En "React Router", **podemos hacer esto utilizando el componente "Route"**. El componente "Route" toma dos propiedades: "path" y "component". La propiedad "path" define la URL que activará la ruta, y la propiedad "component" define el componente de React que se renderizará cuando se visite la ruta. Por ejemplo, si tenemos un componente llamado "Home" que queremos mostrar en la ruta raíz de nuestra aplicación, podemos agregar el siguiente código debajo del componente "Router":

```
function App() {  
  return (  
    <BrowserRouter>  
      <Router>  
        <div className="App">  
          <Routes>  
            <Route path="/" element={Home} />  
          </Routes>  
        </div>  
      </Router>  
    </BrowserRouter>  
  );  
}  
  
export default App;
```

El componente "Routes" es un componente que forma parte de la biblioteca "React Router" y se utiliza para envolver varias rutas. **Su función principal es renderizar solamente la primera ruta que coincida con la URL actual.**

Cuando se utiliza "Routes", React Router recorre cada ruta en el orden en que aparecen dentro del componente y compara la URL actual con la propiedad "path" de cada ruta. La primera ruta que coincida con la URL se renderizará, y las demás rutas no se renderizarán.

Routes utiliza una sintaxis de **árbol de rutas anidadas**. Esto **significa que podemos definir rutas anidadas dentro de otras rutas, lo que hace que** sea más fácil y flexible definir rutas complejas en la aplicación.

Esto es útil cuando se tienen varias rutas que comparten una parte de la URL, ya que garantiza que solo se renderice una ruta a la vez.

Por ejemplo se podrían definir cuatro componentes: Inicio, Acerca, Detalle y Error404:

```
// Inicio.js
export default function Inicio() {
  return <h2>Esta es la página de inicio</h2>;
}

// Acerca.js
export default function Acerca() {
  return <h2>Esta es la página de Acerca</h2>;
}

// Detalle.js
export default function Detalle({ id }) {
  return <h2>Detalle del elemento con ID: {id}</h2>;
}

// Error404.js
export default function Error404() {
  return <h2>La página que buscas no existe (Error 404)</h2>;
}
```

y se podrían definir las siguientes rutas:

```
import { BrowserRouter, Route, Routes, Switch } from "react-router-dom";
import Acerca from "../components/Acerca";
import Inicio from "../components/Inicio";
import Detalle from "../components/Detalle";
import Error404 from "../components/Error404"

function App() {
  return (
    <div>
      <BrowserRouter>
        <Routes>
          <Route path="/" element={<Inicio />} index />
          <Route path="/acerca" element={<Acerca />} />
          <Route path="/detalle/:id" element={<Detalle />} />
          <Route path="*" element={<Error404 />} />
        </Routes>
      </BrowserRouter>
    </div>
  );
}
```

```
export default App;
```

La penúltima ruta ("/detalle/:id") tiene un parámetro dinámico, lo que significa que cualquier URL que comience con "/" y tenga un segmento adicional se considerará una coincidencia

La propiedad "index" en la ruta indica que solo se renderizará el componente "Inicio" cuando la URL sea exactamente igual a "/" .

Tenemos una ruta dinámica que utiliza un parámetro de URL ":id". Cuando se accede a una URL como "/detalle/1", React Router coincidirá con la ruta "/detalle/:id" y pasará el valor "1" como parámetro al componente "Detalle".

Si deseamos ir a otro componente desde un enlace podemos especificar de la siguiente manera:

```
<Link to="/contacto">Contacto</Link>
```

Hook useParams

`useParams` es un hook proporcionado por React Router que te permite acceder a los parámetros de la ruta actual. Los parámetros de ruta son segmentos dinámicos de la URL que pueden cambiar según la navegación del usuario. Los parámetros de ruta se definen en la configuración de la ruta utilizando dos puntos : seguidos del nombre del parámetro, como **/ruta/:parametro**.

Por ejemplo dentro del componente funcional Detalle, si se posee una ruta `/detalles/:id`, se utiliza el hook `useParams` para obtener un objeto que contiene todos los parámetros de la ruta actual. En este caso, sólo tendrás un parámetro, `id`.

```
// Detalle.js
import { useParams } from 'react-router-dom';

export default function Detalle() {
  const { id } = useParams();

  return (
    <div>
      <h2>Detalle del elemento</h2>
      <p>ID: {id}</p>
    </div>
  );
}
```

En este ejemplo, si el usuario navega a una ruta como **/detalles/42**, el componente Detalle extraerá el parámetro `id` usando `useParams`. En este caso, `id` será 42.

Usar `useParams` con un único parámetro es similar a usarlo con múltiples parámetros. La única diferencia es que el objeto devuelto por `useParams` tendrá una única clave que corresponde al nombre del parámetro

definido en la ruta.

Rutas con múltiples parámetros

En React, es posible definir rutas con múltiples parámetros para crear rutas más complejas y dinámicas. La sintaxis para definir rutas con múltiples parámetros es similar a la de las rutas con un solo parámetro, pero se utilizan múltiples segmentos de ruta que se corresponden con los diferentes parámetros.

Aquí hay una explicación detallada de cómo usar useParams en una aplicación React:

Importa useParams desde react-router-dom.

```
import { useParams } from 'react-router-dom';
```

Dentro de tu componente funcional, utiliza el hook useParams para obtener un objeto que contiene todos los parámetros de la ruta actual. Las claves de este objeto corresponden a los nombres de los parámetros definidos en la ruta.

```
const params = useParams();
```

Ahora puedes acceder a los parámetros de ruta usando el objeto params y sus claves. Supongamos que tienes una ruta como /productos/:categoria/:id y un componente Producto. Aquí hay un ejemplo de cómo usar useParams para acceder a los parámetros categoria e id en el componente Producto:

```
// Producto.js
import { useParams } from 'react-router-dom';

export default function Producto() {
  const { categoria, id } = useParams();

  return (
    <div>
      <h2>Producto</h2>
      <p>Categoría: {categoria}</p>
      <p>ID: {id}</p>
    </div>
  );
}
```

Si el usuario navega a una ruta como /productos/electronica/1, el componente Producto extraerá los parámetros categoria e id usando useParams. En este caso, categoria será electronica e id será 1.

UseNavigate

UseNavigate es un hook que devuelve una función y nos va a permitir manipular la navegación similar a <a href...>. Admite como argumento la url a navegar y un segundo argumento opcional para que reemplace en

el stack de memoria donde react router va guardando las urls de las páginas.

```
import { useNavigate } from 'react-router-dom';  
const navigate = useNavigate();
```

```
<button onClick={()=>navigate("contactos")}>Contactos</button>  
<button onClick={()=>navigate("acerca de")}>Acerca De</button>
```

```
navigate("/404", {replace: true});
```

Navigate

Navigate es un componente en React Router que te permite realizar una navegación programática en tu aplicación React. Aunque el hook useNavigate es el enfoque más común para realizar la navegación programática, el componente Navigate también puede ser útil en ciertos escenarios.

El componente Navigate recibe una prop to que indica la ruta a la que deseas navegar, y puede recibir una prop replace que indica si se debe reemplazar la entrada actual en el historial de navegación (por defecto es false, lo que significa que se agregará una nueva entrada al historial).

Aquí tienes un ejemplo de cómo usar el componente Navigate:

```
import { Route, Routes, Navigate } from "react-router-dom";  
import Inicio from "./Inicio";  
import Acerca from "./Acerca";  
  
function App() {  
  const [redirectToAcerca, setRedirectToAcerca] = useState(false);  
  
  const handleClick = () => {  
    setRedirectToAcerca(true);  
  };  
  
  return (  
    <div>  
      {redirectToAcerca && <Navigate to="/acerca" />}  
      <button onClick={handleClick}>Ir a Acerca</button>  
      <Routes>  
        <Route path="/" element={<Inicio />} />  
        <Route path="/acerca" element={<Acerca />} />  
      </Routes>  
    </div>  
  );  
}
```

En este ejemplo, al hacer clic en el botón "Ir a Acerca", se actualiza el estado `redirectToAcerca` a `true`. Cuando `redirectToAcerca` es `true`, el componente `Navigate` se monta y redirige al usuario a la ruta `/acerca`.

Sin embargo, es importante mencionar que el enfoque recomendado para la navegación programática en `React Router v6` es usar el hook `useNavigate`, ya que es más idiomático en aplicaciones `React` y proporciona un mejor control sobre la navegación y el flujo de la aplicación. El componente `Navigate` puede ser útil en casos específicos donde necesitas realizar una redirección condicional basada en el estado de un componente, pero no es la solución preferida en la mayoría de los casos.

Ejercitación

[Realizar la ejercitación guiada de ruteo](#)