React Router

Objetivos

- ¿Por qué debemos crear rutas?
- Conocer que es React Router
- Instalar el paquete y crear nuestras primeras rutas
- Crear una ruta not found
- Navegando entre rutas
- Buenas prácticas al momento de crear rutas

¿ Por qué debemos crear rutas?

Las aplicaciones web modernas suelen ser complejas y constan de múltiples vistas o páginas. Para que los usuarios puedan navegar de manera efectiva y acceder a diferentes partes de la aplicación, es necesario implementar un sistema de enrutamiento. En React, esto se logra utilizando bibliotecas como React Router.

Las razones por las cuales debemos crear rutas es la siguiente:

Organización de la Aplicación: Las rutas permiten dividir la aplicación en múltiples componentes o vistas, lo que facilita la organización y el mantenimiento del código. Cada ruta puede representar una sección lógica de la aplicación.

Navegación Eficiente: Las rutas proporcionan un sistema de navegación eficiente para los usuarios. Los enlaces y botones pueden llevar a los usuarios a diferentes vistas sin tener que recargar toda la página.

URLs Amigables: El uso de rutas permite tener URLs amigables y descriptivas. Esto no solo mejora la experiencia del usuario, sino que también es beneficioso para la optimización de motores de búsqueda (SEO).

Experiencia de Usuario Mejorada: La navegación basada en rutas proporciona una experiencia de usuario más coherente y agradable, ya que los usuarios pueden comprender intuitivamente cómo moverse por la aplicación.

¿Qué es React Router?

React Router es una librería de enrutamiento para aplicaciones web construidas con React. React Router se utiliza para gestionar la navegación en una aplicación de una sola página (Single Page Application o SPA), lo que significa que permite a los desarrolladores crear aplicaciones web que cambian de contenido y de URL sin tener que cargar una página completamente nueva desde el servidor.

Algunas características de React Router son:

Enrutamiento Declarativo: React Router permite definir rutas y componentes asociados con esas rutas de manera declarativa, lo que facilita la configuración y comprensión de la estructura de navegación de la aplicación.

Rutas Anidadas: Puedes definir rutas anidadas, lo que significa que puedes tener una jerarquía de rutas dentro de tu aplicación. Esto es útil para crear diseños complejos y componentes reutilizables.

Manejo de Parámetros: React Router permite la captura de parámetros desde la URL, lo que facilita la creación de rutas dinámicas que pueden mostrar contenido basado en los valores de la URL.

Historial de Navegación: La librería proporciona un historial de navegación que permite al usuario retroceder y avanzar en la navegación de la aplicación, todo sin recargar la página.

Integración con React: React Router se integra perfectamente con React, lo que significa que puedes usar componentes React para definir las vistas de tus rutas.

Manejo de Redirecciones: Puedes configurar redirecciones fácilmente en tu aplicación utilizando React Router.

Instalando React Router y crear nuestra primera ruta

1. Instalamos el paquete con el siguiente comando

```
npm i react-router-dom@6
```

2. Importar el componente BrowserRouter

   Una vez que hayas instalado el paquete React Router, necesitas importar el componente BrowserRouter en tu componente principal. Este componente es el encargado de proporcionar el enrutamiento para tu aplicación.

En tu componente principal, importa el componente BrowserRouter de la siguiente manera como estamos usando vite lo debes hacer en main.jsx:

```jsx
import React from "react";  6.9k (gzipped: 2.7k)
import ReactDOM from "react-dom/client";  513 (gzipped: 319)
import App from "./App.jsx";
import "./index.css";
import { BrowserRouter } from "react-router-dom";  5k (gzipped: 2.2k)

ReactDOM.createRoot(document.getElementById("root")).render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

3. Crear tu primera ruta

Ahora que has importado el componente BrowserRouter, puedes empezar a crear tus rutas. Una ruta es una definición de un segmento de la URL que se corresponde con un componente específico.

Para crear tu primera ruta, importa el componente Route y Routes de la siguiente manera en App.jsx:

```jsx
1   import "./App.css";
2   import { Route, Routes } from "react-router-dom";  10.5k (gzipped: 4.1k)
3   import HomePage from "./components/HomePage";
4   function App() {
5     return (
6       <>
7         <Routes>
8           <Route path="/" element={<HomePage />} />
9         </Routes>
10      </>
11    );
12  }
13
14  export default App;
15
```

Donde:

Routes: Se refiere a las definiciones de las rutas que se utilizan para establecer la navegación en una aplicación web basada en React, es decir funciona como un envoltorio de rutas.

Route: Se utiliza para definir una ruta específica dentro de una aplicación web y especificar qué componente debe renderizarse cuando la URL coincida con esa ruta. Es uno de los componentes esenciales para la configuración del enrutamiento en una aplicación React.

El componente <Route> toma dos propiedades principales:

path: Esta propiedad especifica la URL que debe coincidir con la ruta definida para que el componente sea renderizado. Puede ser una cadena que represente la ruta o una expresión regular para rutas más avanzadas. Por ejemplo, **path="/acerca-de"** coincidirá con la URL cuando esta sea **"http://tu-sitio.com/acerca-de"**.

element: Esta propiedad especifica que componente se debe mostrar en esa ruta.

4. Creamos el componente HomePage para poder mostrarlo en nuestra app

5. Prueba tu app y veras que al iniciar el proyecto ya nos muestra el componente HomePage esto se debe a que declaramos en el path solo un **/** esto indica que esta será la ruta raíz de nuestro proyecto

Creando una ruta not found

En el desarrollo de aplicaciones web con React, es fundamental proporcionar una experiencia de usuario fluida y amigable. Parte de esta experiencia implica manejar de manera adecuada las rutas y las URL en la aplicación. Sin embargo, a veces los usuarios pueden intentar acceder a URL que no existen o escribir direcciones incorrectas en la barra de direcciones del navegador. En lugar de mostrar un error genérico, es más conveniente guiar al usuario hacia una página personalizada que indique que la ruta solicitada no está disponible, lo que se conoce como una página "Not Found" o "404".

¿Por qué crear una ruta "Not Found"?

1. **Mejora la experiencia del usuario**: Proporcionar una página de "Not Found" mejora la experiencia del usuario al darle retroalimentación clara sobre el error y

mantenerlo dentro de tu aplicación en lugar de redirigirlo a una página de error genérica del navegador.

2. **Mantiene a los usuarios en tu aplicación**: En lugar de perder a los usuarios debido a errores de navegación, una página "Not Found" les permite explorar otras partes de tu aplicación.

3. **Profesionalismo y personalización**: Una página de "Not Found" personalizada muestra profesionalismo y atención al detalle en tu aplicación. Puedes personalizar el diseño y el contenido para que se ajuste al estilo de tu aplicación.

A continuación agregaremos una ruta Not Found" a nuestro proyecto

1. Crea un componente para la página not found. Este componente puede ser tan simple o complejo como quieras. Por ejemplo, puedes crear un componente que solo muestre un mensaje de error, o puedes crear un componente que incluya más información, como una lista de posibles rutas que el usuario podría estar buscando.

2. Crea una ruta para el componente not found. Esta ruta debe tener un path que coincida con todas las rutas que no se encuentren en tu aplicación. En la mayoría de los casos, esta ruta tendrá un path de *.

3. Importa el componente not found en tu componente principal.

4. Envuelve el componente not found en un componente Route con un path de *.

Siguiendo los pasos anteriores nuestra nueva ruta debería quedar de la siguiente forma:

```
import "./App.css";
import { Route, Routes } from "react-router-dom";  10.5k (gzipped: 4.1k)
import HomePage from "./components/HomePage";
import NotFound from "./components/NotFound";
function App() {
  return (
    <>
      <Routes>
        <Route path="/" element={<HomePage />} />
        <Route path="*" element={<NotFound />} />
      </Routes>
    </>
  );
}

export default App;
```

Navegando entre rutas

Para navegar entra rutas crearemos un navbar con React Bootstrap de esta manera podremos navegar de manera simple entre componentes usando el componente Link de React Router.

Antes de construir nuestro navbar vamos a conocer que es y cómo funciona Link.

En React Router, el componente Link se utiliza para crear enlaces de navegación. Este componente acepta una propiedad to que especifica la ruta a la que se llevará al usuario cuando haga clic en el enlace.

Este componente es similar al elemento <a> HTML, pero tiene algunas diferencias importantes. En primer lugar, el componente Link no recarga la página cuando el usuario hace clic en él. En cambio, usa la API de historia del navegador para navegar a la nueva ruta.

En segundo lugar, el componente Link puede aceptar un componente como valor de la propiedad to. Esto permite crear enlaces que renderizan componentes personalizados en lugar de simplemente navegar a una nueva ruta.

A continuación podemos ver un ejemplo de como usar Link:

```
<Link to="/about">Ir a Acerca de</Link>
```

Ahora que ya sabemos usar Link vamos a crear nuestro menú de navegación

1. Instala la librería React Bootstrap con el comando npm install react-bootstrap bootstrap

2. Importa los estilos en main.jsx con import 'bootstrap/dist/css/bootstrap.min.css';

3. Crea un componente para la barra de navegación

4. Importa los componentes Container, Nav, NavBar, NavDropdown y Link de React Bootstrap y React Router.

5. Agrega los enlaces a la barra de navegación usando el componente Link

Nuestro componente se debería ver como en la siguiente imagen:

```jsx
import Container from "react-bootstrap/Container";  2.4k (gzipped: 1.2k)
import Nav from "react-bootstrap/Nav";  21.6k (gzipped: 6.4k)
import Navbar from "react-bootstrap/Navbar";  35.8k (gzipped: 12k)
import NavDropdown from "react-bootstrap/NavDropdown";  64.3k (gzipped: 20.5k)
import { Link } from "react-router-dom";  7.6k (gzipped: 3k)

const NavBar = () => {
  return (
    <Navbar expand="lg" className="bg-body-tertiary">
      <Container>
        <Navbar.Brand href="#home">React-Bootstrap</Navbar.Brand>
        <Navbar.Toggle aria-controls="basic-navbar-nav" />
        <Navbar.Collapse id="basic-navbar-nav">
          <Nav className="me-auto">
            <Link to="/" className="mt-2 pe-2 text-decoration-none">
              Home
            </Link>
            <Link to="/register" className="mt-2 text-decoration-none">
              registro
            </Link>
            <NavDropdown title="Dropdown" id="basic-nav-dropdown">
              <NavDropdown.Item href="#action/3.1">Action</NavDropdown.Item>
              <NavDropdown.Item href="#action/3.2">
                Another action
              </NavDropdown.Item>
              <NavDropdown.Item href="#action/3.3">Something</NavDropdown.Item>
              <NavDropdown.Divider />
              <NavDropdown.Item href="#action/3.4">
                Separated link
              </NavDropdown.Item>
            </NavDropdown>
          </Nav>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
```

Finalmente importamos el componente de navegación en App.jsx para que esté disponible en todas las vistas.

src > App.jsx > ...

```jsx
import "./App.css";
import { Route, Routes } from "react-router-dom";  10.5k (gzipped: 4.1k)
import HomePage from "./components/HomePage";
import NotFound from "./components/NotFound";
import Register from "./components/Register";
import NavBarMenu from "./components/NavBarMenu";
function App() {
  return (
    <>
      <NavBarMenu />
      <Routes>
        <Route path="/" element={<HomePage />} />
        <Route path="/register" element={<Register />} />
        <Route path="*" element={<NotFound />} />
      </Routes>
    </>
  );
}

export default App;

```

Buenas prácticas para la creación de rutas

1. **Usar nombres legibles**: Los nombres de las rutas deben ser descriptivos y legibles. Utiliza palabras en minúsculas separadas por guiones para que las rutas sean fácilmente comprensibles.

2. **Singular vs. Plural**: Decide si prefieres utilizar nombres en singular o en plural para las rutas que representan recursos. Mantén la coherencia en toda la aplicación. Por ejemplo, puedes optar por /producto en lugar de /productos si prefieres el singular.

3. **Evitar caracteres especiales**: Evita el uso de caracteres especiales, a menos que sean necesarios para representar información en la ruta. Los guiones (-) y barras (/) son generalmente suficientes.

4. **Parámetros descriptivos**: Si utilizas parámetros en las rutas, dales nombres descriptivos que indiquen su propósito. Por ejemplo, en lugar de /usuarios/:id, podrías usar /usuario/:userId para que sea más claro.

5. **Nombres lógicos**: Los nombres de las rutas deben reflejar la jerarquía y la lógica de tu aplicación. Por ejemplo, si tienes una sección de administración, podrías tener rutas como /admin/usuarios o /admin/configuracion.

6. Evitar nombres genéricos: Evita nombres de rutas demasiado genéricos como /pantalla o /pagina. Los nombres deben ser lo suficientemente específicos como para identificar claramente el propósito de la ruta.

7. **Nombres en inglés**: Siempre que sea posible, utiliza nombres en inglés para las rutas y otros elementos de la aplicación, ya que esto es más común en el desarrollo web y puede facilitar la colaboración con otros desarrolladores.

8. Evitar abreviaciones confusas: Evita abreviaciones que puedan ser confusas o ambiguas. Si decides usar abreviaciones, asegúrate de que sean ampliamente comprensibles.