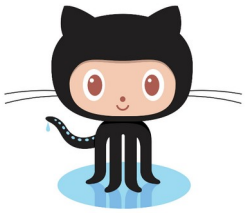
Desarrollo web en entorno cliente

UD – 5

React Router

Contenidos



ÍNDICE

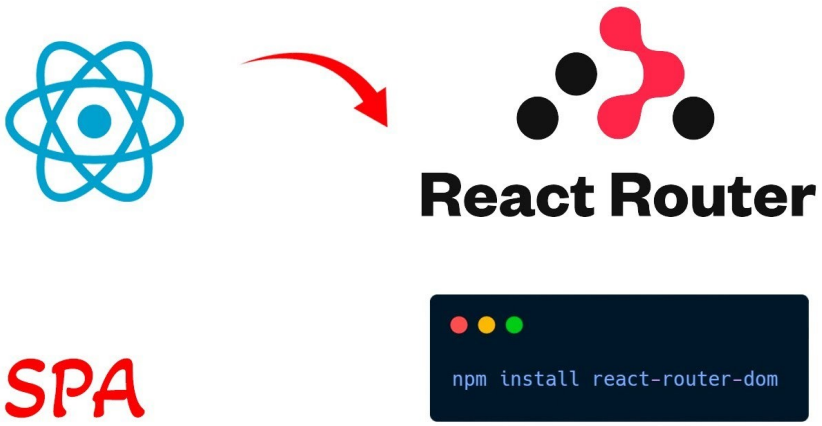
1. INTRODUCCIÓN

React Router es una biblioteca de enrutamiento para aplicaciones React que permite la navegación entre diferentes vistas o componentes sin recargar la página. Proporciona una forma eficiente de gestionar rutas en aplicaciones de una sola **página (SPA)**, mejorando la experiencia del usuario.

Con React Router, puedes definir rutas **dinámicas** y **anidadas**, lo que facilita la estructuración de aplicaciones complejas. Utiliza el concepto de ***history*** para manejar la navegación y permite la sincronización con la URL del navegador.

Algunas de sus características clave incluyen la navegación declarativa con <Routes> y <Route>, la gestión de parámetros con useParams, la navegación programática con useNavigate y la protección de rutas mediante autenticación.

Además, React Router admite ***lazy loading*** para optimizar el rendimiento y mejorar la carga de la aplicación. Gracias a su integración con React, facilita la creación de experiencias interactivas y fluidas sin comprometer la accesibilidad ni la estructura de la aplicación.

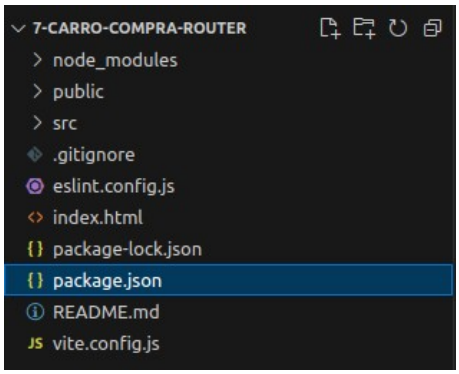


2. ¿INSTALAR?

Para instalar react, vamos a utilizar npm:

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

Adaptará los archivos de tu proyecto: package.json y package.json.lock. Además de la carpeta node_módules.




```
"dependencies": {
  "react": "^18.3.1",
  "react-dom": "^18.3.1",
  "react-router-dom": "^7.1.1"
},
```

Una vez instalado, puedes importarlo en tu código para empezar a usarlo:

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

3. COMPONENTE BROWSERROUTER

Es el **contenedor principal** que debe envolver toda la aplicación para habilitar la funcionalidad de enrutamiento. Define el contexto de enrutamiento para toda la aplicación.

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

function App() {
  return (
    <BrowserRouter>
      {/* El resto de tu aplicación va aquí */}
    </BrowserRouter>
  );
}
```

4. COMPONENTES ROUTEES Y ROUTE

- **Routes** → Un contenedor que define las rutas de la aplicación.
- **Route** → Define una ruta dentro de la aplicación. Cada ruta tiene un path y un element.
  - **path** → Define la URL de la ruta.
  - **Element** → Es el componente que se renderiza cuando la ruta es coincidente.

```jsx
import { Routes, Route } from 'react-router-dom';

function App() {
  return (
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/about" element={<About />} />
    </Routes>
  );
}
```

5. RUTAS DINÁMICAS Y CON PARÁMETROS

Las rutas pueden ser **dinámicas**, lo que significa que puedes pasar parámetros a través de la URL. Usamos **":"** para definir un parámetro en la ruta.

A continuación, un ejemplo de como especificar un parámetro en la ruta:

```jsx
<Route path="/producto/:id" element={<Producto />} />
```

Como obtener el valor de id con el hook useParams, dentro de un componente:

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

function Producto() {
  const { id } = useParams(); // Accede al parámetro 'id' de la URL

  return <h1>Detalle del Producto {id}</h1>;
}
```

6. ENLACES DE NAVEGACION CON LINK

El componente **<Link>** reemplaza el comportamiento de **<a>** para permitir la navegación sin recargar la página. La propiedad to define la ruta a la que navegar.

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

function Menu() {
  return (
    <nav>
      <ul>
        <li><Link to="/">Inicio</Link></li>
        <li><Link to="/about">Acerca de</Link></li>
      </ul>
    </nav>
  );
}
```

7. RUTAS ANIDADAS

React Router permite **anidar rutas** dentro de otras rutas. Esto es útil cuando tienes subcomponentes o vistas dentro de una ruta principal.

```
<Route path="/producto/:id" element={<Producto />}>
  <Route path="detalle" element={<Detalle />} />
</Route>
```

Para renderizar las rutas anidadas, debes utilizar el componente <Outlet> dentro del componente principal.

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

function Producto() {
  return (
    <div>
      <h1>Producto</h1>
      <Outlet />  {/* Renderiza las rutas hijas aquí */}
    </div>
  );
}
```

8. REDIRECCIÓN CON NAVIGATE

Puedes redirigir a los usuarios a otra ruta utilizando el componente <Navigate>. Esto es útil para redirigir después de una acción o si un usuario no tiene acceso a una ruta.

```jsx
import { Navigate } from 'react-router-dom';

function Login() {
  const isLoggedIn = false;

  if (!isLoggedIn) {
    return <Navigate to="/login" />;
  }

  return <h1>Bienvenido</h1>;
}
```

9. PROTECCIÓN DE RUTAS PRIVADAS

Si deseas proteger ciertas rutas (por ejemplo, solo accesibles para usuarios autenticados), puedes crear un **componente de ruta privada**.

```jsx
function ProtectedRoute({ children }) {
  const isAuthenticated = false;  // Aquí compruebas si el usuario está autentica

  if (!isAuthenticated) {
    return <Navigate to="/login" />;
  }

  return children;
}

<Route path="/profile" element={<ProtectedRoute><Profile /></ProtectedRoute>} />
```