React Nivel 8: Rutas y Navegación con React Router 📝
En este nivel, aprenderás a implementar rutas en tu aplicación React usando **React Router**. Esta herramienta es fundamental cuando tu aplicación necesita tener múltiples vistas o páginas y deseas permitir la navegación entre ellas sin recargar la página. Veremos cómo configurar rutas, utilizar parámetros dinámicos y proteger ciertas rutas, todo con React Router.

Temas que Cubriremos 📚

1. ¿Qué es React Router y cuándo usarlo?
2. Configuración básica de rutas con React Router.
3. Uso de Links para navegar entre páginas.
4. Rutas dinámicas con parámetros.
5. Redirección de páginas con `Navigate`.
6. Protección de rutas (autenticación y autorización).

---

1. **¿Qué es React Router y Cuándo Usarlo?** 🤔
   React Router es una librería que permite agregar navegación y rutas a una aplicación React. Usarlo es esencial cuando tu aplicación tiene varias páginas o vistas, y deseas que los usuarios puedan navegar entre ellas sin recargar la página. Es especialmente útil para aplicaciones de una sola página (SPA, Single Page Applications).

Cuándo usar React Router:

- Cuando necesitas manejar múltiples vistas o páginas en tu aplicación.
- Para evitar la recarga completa de la página, mejorando la experiencia del usuario.
- Cuando tu aplicación necesita rutas dinámicas basadas en parámetros de la URL, como IDs de productos o usuarios.

---

2. **Configuración Básica de Rutas con React Router** ⚙️
   Para empezar con React Router, necesitas instalarlo y luego configurar tus rutas usando el componente `BrowserRouter` y `Route`. Aquí está la estructura básica:

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';

// Páginas
const Inicio = () => <h1>Bienvenido a mi página</h1>;
const SobreMi = () => <h1>Sobre mí</h1>;
const Contacto = () => <h1>Contacto</h1>;

function App() {
  return (
    <Router>
      <div>
        <Routes>
```

```
          <Route path="/" element={<Inicio />} />
          <Route path="/sobre-mi" element={<SobreMi />} />
          <Route path="/contacto" element={<Contacto />} />
        </Routes>
      </div>
    </Router>
  );
}

export default App;
```

Explicación:

- `BrowserRouter`: Envuelve toda la aplicación y permite usar las rutas.
- `Routes`: Contenedor para las rutas.
- `Route`: Define las rutas individuales con su respectivo componente asociado a la URL.

---

3. **Uso de Links para Navegar entre Páginas** 🖇️
   Para navegar entre las páginas, utilizamos el componente `Link` de React Router. Este reemplaza a las etiquetas `<a>` tradicionales, evitando la recarga de la página. Ejemplo:

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

function Navegacion() {
  return (
    <nav>
      <Link to="/">Inicio</Link>
      <Link to="/sobre-mi">Sobre mí</Link>
      <Link to="/contacto">Contacto</Link>
    </nav>
  );
}
```

Explicación:

- `Link`: Permite la navegación entre las rutas definidas sin recargar la página.

---

4. **Rutas Dinámicas con Parámetros** 📡
   Las rutas dinámicas permiten que los componentes se rendericen de acuerdo con un parámetro de la URL. Esto es útil cuando, por ejemplo, quieres mostrar detalles de un producto basado en su ID.

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

const ProductoDetalle = () => {
  const { id } = useParams();
  return <h1>Detalles del Producto {id}</h1>;
};
```

Configuración de ruta dinámica:

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

Explicación:

- `useParams`: Hook para obtener los parámetros de la URL, en este caso, el `id` del producto.
- `/producto/:id`: Define la ruta dinámica donde `:id` es un parámetro de la URL.

---

5. **Redirección con `Navigate`** 🔄
   En ciertas situaciones, puede ser necesario redirigir al usuario a otra página. Para esto, usamos el componente `Navigate`.

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

function Redirigir() {
  const redirigir = true;

  if (redirigir) {
    return <Navigate to="/contacto" />;
  }

  return <h1>Bienvenido a la página de inicio</h1>;
}
```

Explicación:

- `Navigate`: Permite redirigir al usuario a otra ruta de manera programática.

---

6. **Protección de Rutas (Autenticación y Autorización)** 🔐
   A veces necesitamos proteger ciertas rutas para que solo puedan ser accedidas si el usuario está autenticado. Esto se puede hacer utilizando un estado de autenticación y condicionalmente renderizando las rutas.

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

function RutaProtegida() {
  const estaAutenticado = false; // Simulando la autenticación

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

  return <h1>Ruta protegida</h1>;
}
```

Explicación:

- Usamos un estado de `estaAutenticado` para verificar si el usuario tiene acceso a la ruta protegida.
- Si el usuario no está autenticado, se redirige automáticamente a la página de login.

Consejos Adicionales para este Nivel:

- **Recuerda usar rutas anidadas** cuando tengas subcomponentes dentro de una misma ruta.
- **Protege rutas sensibles** como el perfil del usuario o la página de administración para garantizar que solo los usuarios autenticados puedan acceder.
- **Prueba rutas dinámicas** en aplicaciones que manejen datos dinámicos, como productos o perfiles de usuario, para comprender mejor cómo funcionan los parámetros.

¡Felicidades por completar el Nivel 8! 🎉 Ahora sabes cómo manejar la navegación y las rutas en React, lo que te permitirá construir aplicaciones más complejas y ricas en funcionalidad.