ENRUTAMIENTO EN REACT VITE

Objetivos

- Entender qué es el enrutamiento en una SPA (Single Page Application).
- Configurar react-router-dom en un proyecto creado con Vite.
- Implementar rutas estáticas, dinámicas y anidadas.
- Usar Link, NavLink, useParams, useNavigate y Outlet.
- Crear rutas protegidas simples y usar lazy loading.
- Practicar con ejercicios y una mini-tarea (proyecto).

Estructura propuesta para el proyecto

```
my-router-app/
├─ node_modules/
├─ public/
├─ src/
│  ├─ pages/
│  │  ├─ Home.jsx
│  │  ├─ About.jsx
│  │  ├─ Contact.jsx
│  │  ├─ User.jsx
│  │  ├─ Login.jsx
│  ├─ components/
│  │  ├─ Navbar.jsx
│  │  ├─ ProtectedRoute.jsx
│  ├─ layouts/
│  │  ├─ Dashboard.jsx
│  ├─ App.jsx
│  └─ main.jsx
├─ package.json
└─ vite.config.js
```

¿Qué es el enrutamiento en una SPA?

En una SPA, la aplicación carga una sola página HTML y el JavaScript se encarga de actualizar el contenido visible sin recargar la página. El **enrutamiento** es la forma en que la app decide qué componente mostrar según la URL actual.

Principales conceptos

- **Router**: componente que observa la URL y renderiza las rutas (ej. BrowserRouter).
- **Route**: mapea una ruta (path) a un componente.
- **Link / NavLink**: componentes para navegar sin recargar la página.
- **useParams**: hook para leer parámetros dinámicos de la URL.
- **useNavigate**: hook para navegar programáticamente.
- **Outlet**: placeholder para rutas anidadas.
- **Lazy loading**: carga módulos bajo demanda para mejorar performance.
- **Rutas protegidas**: lógica para impedir acceso si el usuario no está autenticado.

BrowserRouter vs HashRouter

- BrowserRouter usa el history API (URLs limpias). Requiere configuración del servidor en producción para devolver index.html en rutas desconocidas.
- HashRouter usa #/ruta en la URL; útil para hosting estático (GitHub Pages) sin configuración del servidor.

Guía paso a paso

Paso 0 — Crear el proyecto con Vite

```
npm create vite@latest my-router-app -- --template react
cd my-router-app
npm install
npm install react-router-dom
npm run dev
```

Paso 1 — Envolver la app con el Router (main.jsx)

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import { BrowserRouter } from 'react-router-dom'
import App from './App'
import './index.css'

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

Paso 2 — Crear páginas básicas (src/pages)

```jsx
export default function Home() {
  return (
    <div>
      <h2>Home</h2>
      <p>Bienvenidos a la página principal.</p>
    </div>
  )
}

// src/pages/About.jsx
export default function About(){
  return <h2>Acerca de</h2>
}

// src/pages/Contact.jsx
export default function Contact(){
  return <h2>Contacto</h2>
}
```

Paso 3 — Navbar y rutas básicas (App.jsx)

```jsx
import { NavLink } from 'react-router-dom'

export default function Navbar() {
  return (
    <nav>
      <NavLink to="/">Inicio</NavLink> |{' '}
      <NavLink to="/about">Acerca</NavLink> |{' '}
      <NavLink to="/contact">Contacto</NavLink>
    </nav>
  )
}
```

```jsx
import { Routes, Route } from 'react-router-dom'
import Navbar from './components/Navbar'
import Home from './pages/Home'
import About from './pages/About'
import Contact from './pages/Contact'
import User from './pages/User'
import NotFound from './pages/NotFound'

export default function App(){
  return (
    <div>
      <h1>Mi App con Router</h1>
      <Navbar />

      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/contact" element={<Contact />} />
        <Route path="/user/:id" element={<User />} />
        <Route path="*" element={<NotFound />} />
      </Routes>
    </div>
  )
}
```

Paso 4 — Ruta dinámica con useParams (User.jsx)

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

export default function User(){
  const { id } = useParams()
  return (
    <div>
      <h2>Perfil de usuario</h2>
      <p>ID: {id}</p>
    </div>
  )
}
```

Prueba: `http://localhost:5173/user/123` -> mostrará `ID: 123`.

Paso 5 — Navegación programática (`useNavigate`)

```jsx
import { useNavigate } from 'react-router-dom'

export default function Login(){
  const navigate = useNavigate()

  function handleLogin(){
    // simulamos login
    // guardar token en localStorage o usar contexto
    localStorage.setItem('token', 'abc')
    navigate('/dashboard')
  }

  return (
    <div>
      <h2>Login</h2>
      <button onClick={handleLogin}>Iniciar sesión</button>
    </div>
  )
}
```

Paso 6 — Rutas anidadas con `Outlet` (Dashboard ejemplo)

```jsx
import { Outlet, Link } from 'react-router-dom'

export default function Dashboard(){
  return (
    <div>
      <h2>Dashboard</h2>
      <aside>
        <Link to="/dashboard">Inicio</Link> | <Link to="/dashboard/settings">Ajustes</Link>
      </aside>
      <main>
        <Outlet />
      </main>
    </div>
  )
}
```

```jsx
export default function DashboardHome(){
  return <div>Bienvenido al Dashboard</div>
}
```

```jsx
export default function DashboardSettings(){
  return <div>Ajustes del usuario</div>
}
```

```jsx
<Route path="/dashboard" element={<Dashboard />}>
  <Route index element={<DashboardHome />} />
  <Route path="settings" element={<DashboardSettings />} />
</Route>
```

Paso 7 — Rutas protegidas (ProtectedRoute)

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

export default function ProtectedRoute({ children }){
  const token = localStorage.getItem('token')
  if(!token){
    return <Navigate to="/login" replace />
  }
  return children
}
```

```jsx
// Uso en App.jsx
<Route path="/dashboard" element={
  <ProtectedRoute>
    <Dashboard />
  </ProtectedRoute>
}>
  <Route index element={<DashboardHome />} />
  <Route path="settings" element={<DashboardSettings />} />
</Route>
```

Nota: en producción usa un contexto de autenticación más robusto y no guardes tokens sin cifrar.

Paso 8 — Lazy loading de rutas

```jsx
// App.jsx (lazy example)
import React, { Suspense, lazy } from 'react'

const About = lazy(() => import('./pages/About'))

// En Routes:
<Suspense fallback={<div>Cargando...</div>}>
  <Routes>
    <Route path="/about" element={<About />} />
    {/* otras rutas */}
  </Routes>
</Suspense>
```

Paso 9 — Página 404 (NotFound)

```jsx
// src/pages/NotFound.jsx
export default function NotFound(){
  return (
    <div>
      <h2>404 - Página no encontrada</h2>
      <p>La ruta solicitada no existe.</p>
    </div>
  )
}
```

RESOLVER!!!

1. **Que son rutas anidadas y lazyLoading**
2. **¿Qué hook usas para leer parámetros de la ruta?**
3. **¿Cuál es la diferencia entre `Link` y `NavLink`?**
4. **¿Para qué sirve `Outlet`?**
5. **¿Qué componente usas para redirigir si el usuario no está autenticado?**
6. **¿Por qué podrías usar `HashRouter` en lugar de `BrowserRouter`?**