React router

Sistemas y Tecnologías Web (CC3062) - 2026

React Router

Semestre 01, 2026

Problemática

Las aplicaciones React son SPA — Single Page Applications

- Solo existe un archivo `index.html`
- JavaScript controla qué se muestra
- El navegador nunca recarga la página

¿Cómo mantenemos la URL sincronizada con lo que se ve?

Sin router: URL estancada

```
Usuario ve: /        → muestra Dashboard
Usuario hace clic   → muestra Perfil
URL sigue siendo: /  ← el navegador no sabe dónde está el usuario
```

Consecuencias:

- No puedes compartir un enlace directo
- El botón "atrás" no funciona como se espera

- El historial del navegador no refleja la navegación

Con React Router: URL sincronizada

```
Usuario ve: /dashboard  → muestra Dashboard
Usuario hace clic       → muestra /perfil
URL cambia a: /perfil   ← el navegador lo sabe
```

- La URL refleja el estado de la UI
- El botón "atrás" funciona
- Los enlaces directos funcionan
- Sin recargar la página

Configuración inicial

Instalación

```
npm install react-router
```

React Router v7 — importaciones desde `"react-router"`

Estructura base

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

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Inicio />} />
        <Route path="/productos" element={<Productos />} />
        <Route path="/contacto" element={<Contacto />} />
```

```
      </Routes>
    </BrowserRouter>
  );
}
```

- `BrowserRouter` — activa el router para toda la app
- `Routes` — contenedor de todas las rutas
- `Route` — asocia una URL con un componente

Funcionamiento

```
URL del navegador: /productos
        ↓
React Router busca en <Routes>
        ↓
Encuentra path="/productos"
        ↓
Renderiza <Productos />
```

- Solo una `<Route>` se activa a la vez
- Si ninguna coincide: no renderiza nada
- Puedes agregar un `path="*"` para manejar 404

Ruta 404

```
<Routes>
  <Route path="/" element={<Inicio />} />
  <Route path="/productos" element={<Productos />} />
  <Route path="*" element={<PaginaNoEncontrada />} />
</Routes>
```

`path="*"` captura cualquier URL que no coincida con las anteriores

Link

El problema con `<a>`

```
// □ MAL — recarga toda la página
<a href="/productos">Ver productos</a>
```

Al usar `<a>` normal:

- El navegador hace una petición HTTP completa
- React se desmonta y vuelve a montar
- Se pierde todo el state de la app

`<Link>`

```
import { Link } from "react-router";

function NavBar() {
  return (
    <nav>
      <Link to="/">Inicio</Link>
      <Link to="/productos">Productos</Link>
      <Link to="/contacto">Contacto</Link>
    </nav>
  );
}
```

- Usa `to=` en lugar de `href=`
- No recarga la página — actualiza la URL y renderiza el componente
- Se ve como un `<a>` en el HTML final

`<NavLink>`

```
import { NavLink } from "react-router";

function NavBar() {
  return (
    <nav>
      <NavLink
        to="/"
        className={({ isActive }) => isActive ? "activo" : ""}
      >
        Inicio
      </NavLink>
      <NavLink
        to="/productos"
        className={({ isActive }) => isActive ? "activo" : ""}
      >
        Productos
      </NavLink>
    </nav>
  );
}
```

- `isActive` es `true` cuando la URL coincide
- Útil para resaltar el ítem de menú actual

useNavigate

useNavigate

A veces la navegación ocurre por lógica, no por un clic directo

- Después de enviar un formulario → redirigir al usuario
- Después de iniciar sesión → ir al dashboard
- Después de un tiempo → redirigir automáticamente

Uso básico

```
import { useNavigate } from "react-router";

function FormularioLogin() {
  const navigate = useNavigate();

  function handleSubmit(e) {
    e.preventDefault();
    // ... validar credenciales ...
    navigate("/dashboard"); // ← redirige al usuario
  }

  return (
    <form onSubmit={handleSubmit}>
      {/* campos del formulario */}
      <button type="submit">Iniciar sesión</button>
    </form>
  );
}
```

Opciones de navegación

```
const navigate = useNavigate();

// Ir a una ruta
navigate("/dashboard");

// Pasar datos con la navegación
navigate("/perfil", { state: { desde: "login" } });

// Volver a la página anterior
navigate(-1);

// Avanzar en el historial
navigate(1);
```

`navigate(-1)` equivale al botón "atrás" del navegador

Rutas anidadas (Nested Routes)

Rutas anidadas

Permiten que un componente padre comparta su layout con componentes hijos

```
/dashboard           → muestra Dashboard + Inicio
/dashboard/ajustes   → muestra Dashboard + Ajustes
/dashboard/perfil    → muestra Dashboard + Perfil
```

- El sidebar y el header del dashboard siempre están visibles
- Solo cambia el contenido central

Configuración en el router

```
<Routes>
  <Route path="/dashboard" element={<Dashboard />}>
    <Route index element={<Inicio />} />
    <Route path="ajustes" element={<Ajustes />} />
    <Route path="perfil" element={<Perfil />} />
  </Route>
</Routes>
```

- Las rutas hijas se anidan dentro de la ruta padre
- `index` = componente por defecto cuando la URL es exactamente `/dashboard`
- Las rutas hijas heredan el path del padre

El componente `<Outlet />`

```
import { Outlet, Link } from "react-router";

function Dashboard() {
  return (
    <div className="dashboard">
      <aside>
        <Link to="/dashboard">Inicio</Link>
        <Link to="/dashboard/ajustes">Ajustes</Link>
        <Link to="/dashboard/perfil">Perfil</Link>
      </aside>

      <main>
        <Outlet /> {/* aquí se renderiza el hijo activo */}
      </main>
    </div>
  );
}
```

Sin `<Outlet />`, los componentes hijos nunca aparecen

Funcionamiento del Outlet

```
URL: /dashboard/ajustes

<Dashboard />          ← siempre visible
  <aside>...</aside>
  <main>
    <Outlet />       ← React Router inserta aquí:
    <Ajustes />      ← el componente que coincide con la URL
  </main>
```

- El padre define la estructura
- El hijo llena el `<Outlet />`

Error común: olvidar el Outlet

```
// ❌ MAL — sin Outlet, los hijos nunca se muestran
function Dashboard() {
  return (
    <div>
      <h1>Dashboard</h1>
      {/* no hay <Outlet /> */}
    </div>
  );
}

// ✅ BIEN — con Outlet, los hijos se renderizan
function Dashboard() {
  return (
    <div>
      <h1>Dashboard</h1>
      <Outlet />
    </div>
  );
}
```

Rutas dinámicas

Rutas dinámicas

Cuando la URL contiene un valor variable

```
/productos/1      → detalles del producto con ID 1
/productos/42     → detalles del producto con ID 42
/usuarios/ana     → perfil de la usuaria "ana"
```

No podemos crear una ruta por cada ID — usamos parámetros

Configuración

```
<Routes>
  <Route path="/productos" element={<ListaProductos />} />
  <Route path="/productos/:id" element={<DetalleProducto />} />
</Routes>
```

`:id` — el `:` indica que es un parámetro de ruta

Coincide con:

- `/productos/1` → `id = "1"`
- `/productos/laptop` → `id = "laptop"`
- `/productos/abc123` → `id = "abc123"`

Leer el parámetro: useParams

```
import { useParams } from "react-router";

function DetalleProducto() {
  const { id } = useParams();

  return (
    <div>
      <h1>Producto #{id}</h1>
      <p>Cargando información del producto {id}...</p>
    </div>
  );
}
```

`useParams()` devuelve un objeto con todos los parámetros de la URL

Navegar a rutas dinámicas

```
// Con Link
<Link to={`/productos/${producto.id}`}>
  Ver {producto.nombre}
```

```
</Link>

// Con useNavigate
const navigate = useNavigate();
navigate(`/productos/${producto.id}`);
```

Se construye la URL con template literals

Principios clave

- Las URLs deben reflejar el estado de la UI
- Usa `<Link>` para navegación visual, `useNavigate` para lógica
- Las rutas anidadas necesitan siempre un `<Outlet />`
- Los parámetros dinámicos se leen con `useParams`