**🏠 Enrutamiento en una SPA (Single Page Application)**

**👋 Introducción**

**👋 ¡Hola! ¡Bienvenido a tu próximo paso emocionante en el mundo del desarrollo web!**

En los siguientes pasos, nos sumergiremos en la creación de una aplicación de una sola página (SPA) utilizando una de las librerías más populares y modernas de JavaScript: React. Esta tecnología revolucionaria nos permite construir interfaces de usuario interactivas y dinámicas de manera eficiente.

Una parte clave de cualquier SPA es la gestión de la navegación entre diferentes vistas o páginas, y para ello, vamos a incorporar React Router. React Router es una biblioteca esencial que facilita la creación de rutas en nuestra aplicación React, permitiendo la transición suave entre distintas secciones sin tener que recargar la página completa.

Exploraremos cómo definir rutas, vincularlas a componentes específicos y proporcionar a nuestros usuarios una experiencia de navegación fluida.

**📖 Material Descargable**

Aquí encontrarás el material descargable correspondiente que te dará las bases para tus próximos pasos.

👉[**Teoría Introducción a SPA**](https://drive.google.com/file/d/14YHabPk9vT8jyQ5i3VWBxH8Wfwcqbj9T/view?usp=drive_link)

# 👣 Configuración del enrutador

En este paso, vamos a avanzar en el uso de React Router DOM para construir rutas en aplicaciones de una sola página (SPA). Aprenderás a configurar y administrar rutas de una manera más moderna y eficiente, utilizando createBrowserRouter para manejar las diferentes vistas de la aplicación.

### ✏️ Actividad: Crear la vista Product

1. **Crear la vista Product**:
   * Crear el archivo Product.tsx en la carpeta /src/views
   * Definir la estructura básica de cualquier componente.
   * Busca el archivo product.html del curso JavaScript 1.
   * Copiar todas las etiquetas del contenido del cuerpo <body> de la vista.
   * Pegar el contenido dentro del retorno return de Product.
   * Importar y reutilizar los componentes ya desarrollados (al menos NavBar).
   * Modificar los cierres de etiquetas correspondientes (como img o input), sustituir los atributos for por htmlFor que correspondan y cambiar los atributos class existentes por className.
2. **Definir los estilos de Product**:
   * Crear el archivo Product.module.css en la carpeta /src/components
   * Busca el archivo styles.css del curso JavaScript 1.
   * Copia los estilos necesarios para la vista (Product Page).
   * Pega los estilos copiados en el archivo Product.module.css.
   * Importar los estilos en el componente Product.tsx y modificar las clases correspondientes.

/\* /src/views/Product.tsx \*/

import styles from "./Product.module.css"

function Product() {

  return <> ... </>

}

export default Product;

### ✏️ Actividad: Crear la vista Cart

1. **Crear la vista Cart**:
   * Crear el archivo Cart.tsx en la carpeta /src/views
   * Definir la estructura básica de cualquier componente.
   * Busca el archivo cart.html del curso JavaScript 1.
   * Copiar todas las etiquetas del contenido del cuerpo <body> de la vista.
   * Pegar el contenido dentro del retorno return de Cart.
   * Importar y reutilizar los componentes ya desarrollados (al menos NavBar).
   * Modificar los cierres de etiquetas correspondientes (como img o input), sustituir los atributos for por htmlFor que correspondan y cambiar los atributos class existentes por className.
2. **Definir los estilos de Cart**:
   * Crear el archivo Cart.module.css en la carpeta /src/components
   * Busca el archivo styles.css del curso JavaScript 1.
   * Copia los estilos necesarios para la vista (Cart).
   * Pega los estilos copiados en el archivo Cart.module.css.
   * Importar los estilos en el componente Cart.tsx y modificar las clases correspondientes.

/\* /src/views/Cart.tsx \*/

import styles from "./Cart.module.css"

function Cart() {

  return <> ... </>

}

export default Cart;

### ✏️ Actividad: Instalar el Módulo de React Router DOM

Para integrar rutas en nuestra aplicación React, comenzaremos instalando el módulo React Router DOM. Para eso es necesario abrir una consola y ejecutar el siguiente comando:

npm install react-router-dom

### ✏️ Actividad: Implementar createBrowserRouter

Con el módulo instalado, y los componentes de vistas definidas, el siguiente paso es implementar createBrowserRouter para configurar nuestras rutas.

1. **Configurar el enrutador en App.js**:
   * Importar createBrowserRouter y RouterProvider de react-router-dom en el componente App.
   * Importar los componentes de vista Index, Cart y Product.
   * Crear un objeto browserRouter usando el método createBrowserRouter.
   * Pasar como parámetro del método un array de objetos con las propiedades **path** (¿qué ruta?) y **element** (¿qué elemento debe renderizar?) de la siguiente manera:

/\* /src/App.tsx \*/

const browserRouter = createBrowserRouter([

  { path: "/", element: <Home /> },

  { path: "/cart", element: <Cart /> }

]);

* Utilizar el componente RouterProvider para envolver tu aplicación, pasando browserRouter como atributo del componente.

/\* /src/App.tsx \*/

function App() {

  return <RouterProvider router={ browserRouter } />

}

1. **Verificar el correcto renderizado de las vistas**:
   * Acceder a / para verificar que se renderiza el componente de vista Home.
   * Acceder a /cart para verificar que se renderiza el componente de vista Cart.

### ✏️ Actividad: Definir una Ruta Dinámica

Las rutas dinámicas desempeñan un papel fundamental en la presentación de contenido dinámico en aplicaciones y páginas web.

Un ejemplo ilustrativo de su utilidad se encuentra en la programación de la vista de detalle, donde se requiere renderizar diferentes productos según la necesidad. Para lograr esto, es esencial incorporar un parámetro de ruta dinámico, como el identificador único (id), que actuará como un marcador distintivo para identificar y recuperar la información específica del producto que se desea mostrar.

1. **Añadir una ruta dinámica en la configuración del browserRouter**:
   * Es necesario agregar un objeto con las propiedades path y element al array de configuración del createBrowserRouter.
   * El parámetro puede tener cualquier nombre y se define con **":"**.

{ path: "/product/:id", element: <Product /> },

1. **Verificar el correcto renderizado de las vistas**:
   * Acceder a /product/id1 para verificar que se renderiza el componente de vista Product.

**👣 Creación de la navegación de la tienda**

En el contexto de una Single Page Application (SPA), se utiliza un único archivo HTML, lo que implica que la navegación no se realiza entre archivos, sino entre componentes de vistas.

En este paso, enlazaremos correctamente las diversas interfaces de la aplicación, al mismo tiempo que repasaremos conceptos previamente abordados.

**✏️ Actividad: Generar los enlaces de la barra de navegación**

El componente <Link> de React Router DOM facilita la navegación entre las vistas de una aplicación de una sola página (SPA), de manera similar a la etiqueta <a> entre las distintas páginas HTML.

1. **Crear los enlaces de NavBar**:
   * En el archivo /src/components/NavButton importar el componente Link del módulo de RRD.
   * Cambiar la etiqueta a por el componente Link.
   * A diferencia del anchor, el componente Link direcciona mediante el atributo to, en lugar de href, para especificar hacia qué componente de vista debe dirigirse.

/\* /src/components/NavButton.tsx \*/

import styles from "./NavButton.module.css";

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

function NavButton({ title, href }) {

   return <Link className={...} to={href}>{title}</Link>;

}

export default NavButton;

1. **Enlace hacia Home**:
   * Al hacer clic en el logo de la tienda, el usuario debe navegar a la página principal /.
2. **Enlace hacia Cart**:
   * Luego de los íconos de las redes sociales agregar la imagen del carrito /cart.png para navegar hasta la página del carrito /cart.



1. **Verificar el correcto renderizado de la vista y links.**

**✏️ Actividad: Configurar componente de vista NotFound**

Cuando una vista no existe, React Router DOM define por defecto una vista poco amigable para el usuario. En esta actividad, te encargarás de desarrollar una vista y de programarla para que sea compatible con todas las vistas que no forman parte de la aplicación.

1. **Crear el componente de vista NotFound**:
   * Crear el archivo NotFound.tsx en la carpeta /src/views
   * Definir la estructura básica de cualquier componente.
   * Copiar de alguna otra vista los elementos comunes de la vista (NavBar, section/Hero, main y footer/Footer).
   * Pegar el contenido dentro del retorno return de NotFound.
   * Maquetar el mensaje "NOT FOUND" de la vista dentro de la etiqueta main.
2. **Definir los estilos de NotFound**:
   * Crear el archivo NotFound.module.css en la carpeta /src/views
   * Copiar y pegar los estilos relevantes de la vista.
   * Importar los estilos en el componente NotFound.tsx y crear las clases correspondientes.

/\* /src/views/NotFound.tsx \*/

import styles from "./NotFound.css";

function NotFound() {

   return <> ... </>;

}

export default NotFound;

1. **Añadir una ruta en la configuración del browserRouter**:
   * Es necesario agregar un objeto con las propiedades path y element al array de configuración del createBrowserRouter de App.
   * Este objeto debe ser **siempre** el último objeto de la configuración.
   * Cualquier ruta que no sea programada renderizará NotFound, definiendo la ruta **"\*"**.

{ path: "/\*", element: <NotFound /> },

1. **Verificar el correcto renderizado de la vista y links.**
   * Acceder a /profile para verificar que se renderiza el componente de vista NotFound.