Rutas con React Router

En esta sección se implementarán las rutas para la aplicación web. Actualmente, la aplicación cuenta con una sola página que muestra el catálogo sobre el carrito de compras. El objetivo es separar esto en dos páginas independientes: una para el catálogo y otra para el carrito de compras.

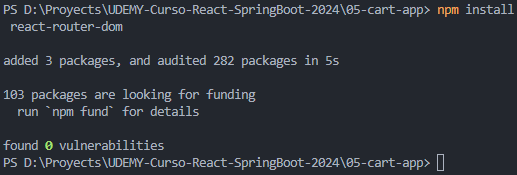
Pasos 79 al 84

Sigue estos pasos:

Paso 79: Instalar la dependencia React Router DOM

React Router DOM es una biblioteca externa que permite gestionar el enrutado en una aplicación React.

Abre la terminal y, en la raíz del proyecto, ejecuta el comando **npm install react-router-dom** para instalar la dependencia:

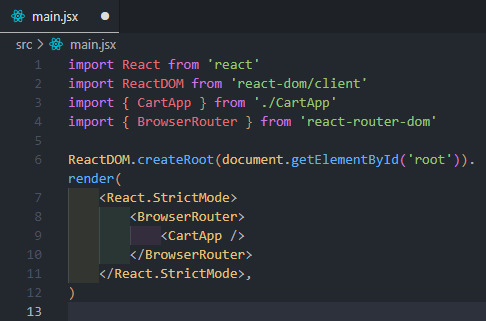


Revisa el archivo **package.json** para asegurar de que **react-router-dom** esté listado en las dependencias.





En el archivo **main.jsx**, debes envolver el componente raíz **<CartApp>** con **<BrowserRouter>** para habilitar el enrutado en la aplicación.



Ahora, **main.jsx** actúa como el componente padre que envuelve toda la aplicación y provee el contexto de enrutado utilizando **BrowserRouter**.

Paso 80: Implementar el sistema de rutas en CartApp

Para mejorar la estructura y navegación de la aplicación, se procede a implementar un sistema de rutas en **CartApp.jsx** utilizando React Router DOM. Esto permitirá tener páginas independientes para el catálogo de productos y el carrito de compras.

Primero se importa **Route** y **Routes** de **react-router-dom** para definir las rutas. Además, es importante asegurarse que el **<div>** con la clase **container** **my-4** y el título **<h3>Cart App</h3>** estén presentes en todas las rutas.

Las rutas se definen utilizando el componente **<Routes>**, dentro del cual se especifican las rutas con el componente **<Route>**.



Cada ruta tendrá una URL a la que se accede, el cual se define con el atributo **path** y un atributo **element** asociado al componente que debe renderizar.

En lugar de usar la forma de apertura y cierre del componente **<Route></Route>**, se puede usar la forma autocontenida: **<Route />**

Por el momento, se definen las rutas **catalog** y **cart**.



Paso 81: Definir los componentes que renderizaran las rutas

Se han definido las rutas **catalog** y **cart**, con el fin de que navegar a **/catalog** va a mostrar el catálogo de productos, y navegar a /**cart** va a mostrar el carrito de compras.

Transfiere la definición del componente **CatalogView**, junto con la propiedad **handler** y su valor **handlerAddProductCart**, al atributo **element** de la ruta definido con **<Route>**, cuyo **path** es **"catalog"**.



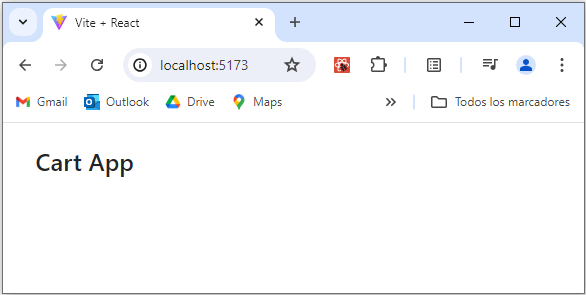


Configura la ruta **cart** para que renderice el componente **CartView** si hay elementos en el carrito (**cartItems**). Si el carrito está vacío, muestra un mensaje indicando que no hay elementos. Para aquello solamente se traslada la condición dentro del atributo **element**, pero contenida en unos paréntesis.

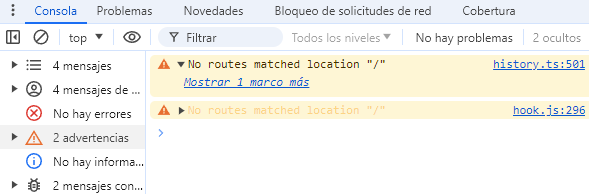


Paso 82: Prueba en el navegador

Guarda todos los cambios y abre el navegador, navega a <http://localhost:5173/>, solamente va a mostrar título de la aplicación.

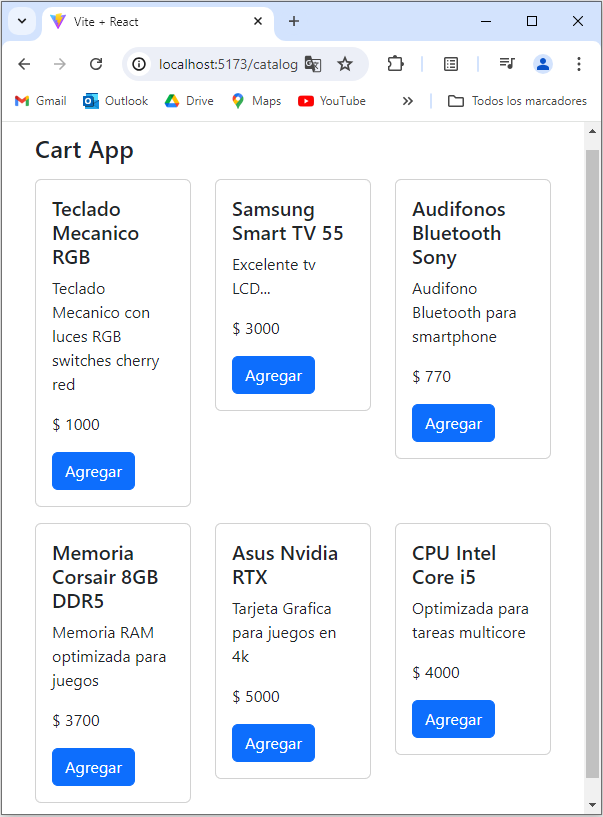


En la consola, se puede notar una advertencia indicando que la ruta " **/** " no está mapeada. Esto ocurre porque no se ha definido un **Route** para la ruta raíz (**/**).

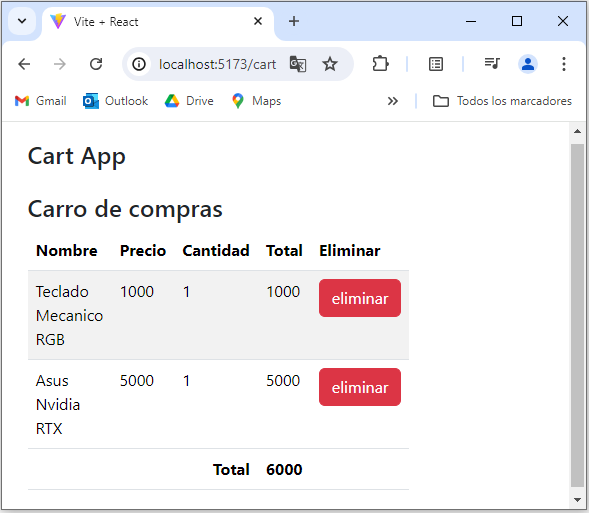


Navega a <http://localhost:5173/catalog>, para ver el catálogo de productos junto con el título, pero sin el carrito de compras.

Puedes agregar algunos productos al carrito pulsando el botón **"Agregar"**.



Navega a <http://localhost:5173/cart>, para visualizar los productos que has añadido al carrito. La funcionalidad de **sessionStorage** sigue funcionando, por lo que los productos se guardan en la memoria y no se pierden al actualizar la página.



Si eliminas todos los productos del carrito, el componente del carrito de compras no se renderizará.

Paso 83: Redirigir al acceder a la ruta principal /

Es importante definir una redirección desde la ruta raíz (**/**) hacia el catálogo de productos. Esto asegura que, al acceder a la página principal, los usuarios sean automáticamente dirigidos al catálogo.

Asegúrate de importar **Navigate** junto con **Route** y **Routes** desde **react-router-dom**.

Dentro del archivo **CartApp.jsx**, define una nueva ruta (**<Route>**) con **path="/"**. Utiliza **<Navigate to="/catalog" />** como el elemento (**element**) para esta ruta, para redirigir automáticamente a los usuarios al catálogo de productos.

Puedes definir el contenido del atributo **to** entre llaves (**{ }**) o sin llaves.



Navega a <http://localhost:5173/> en el navegador y automáticamente serás redirigido a <http://localhost:5173/catalog>.

Paso 84: Mostrar un mensaje cuando no hay productos en el carrito

Se implementará la lógica para mostrar un mensaje en el navegador cuando el carrito de compras esté vacío.

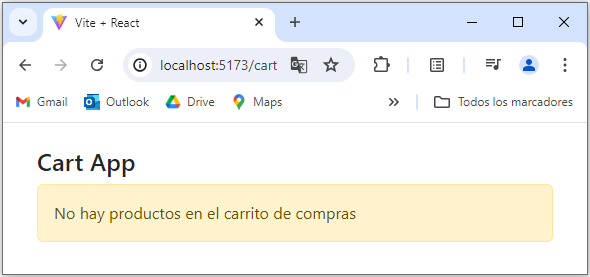
En lugar de usar el operador **||** para manejar la condición de que no hay productos en el carrito, se utilizara un operador ternario.

Si el carrito está vacío, se mostrará un mensaje dentro de un **<div>** con la clase **alert alert-warning**. Si hay productos en el carrito, se mostrará el componente **CartView** con los productos.



Navega a <http://localhost:5173/cart>, si no hay productos en el carrito, va a mostrar un mensaje de advertencia diciendo: **"No hay productos en el carrito de compras!"**.

Puedes agregar productos al carrito y verificarlos que se muestren correctamente en <http://localhost:5173/cart>. Luego elimina todos los productos del carrito y verifica que el mensaje de advertencia se muestre nuevamente.



Enlace de referencia:

El siguiente enlace muestra la documentación de React Router:

* <https://reactrouter.com/en/main>