Hooks personalizado - useItemsCart

En esta sección, se creará un hook personalizado denominado **useItemsCart**, con el objetivo de reutilizar y centralizar la lógica del carrito de compras en el componente principal **CartApp**.

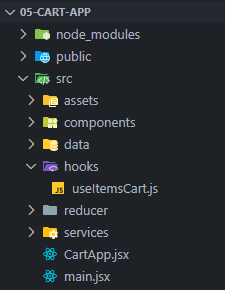
Pasos 74 al 78

Sigue estos pasos:

Paso 74: Crear un hook personalizado

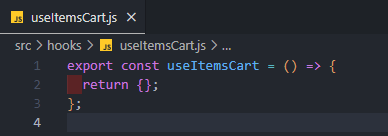
En el componente **CartApp**, actualmente se maneja mucha lógica, incluyendo el uso de **useReducer**, **useEffect** que reacciona a cambios en **cartItems**, y funciones que envían acciones y datos a **itemsReducer** para actualizar el estado. Para mejorar la organización y reutilización del código, se creará un hook personalizado.

Crea una nueva carpeta llamada **hooks** dentro de **src**. Dentro de esta carpeta, crea un archivo llamado **useItemsCart.js**.



Los hooks en React siguen la convención de comenzar con el prefijo **use**.

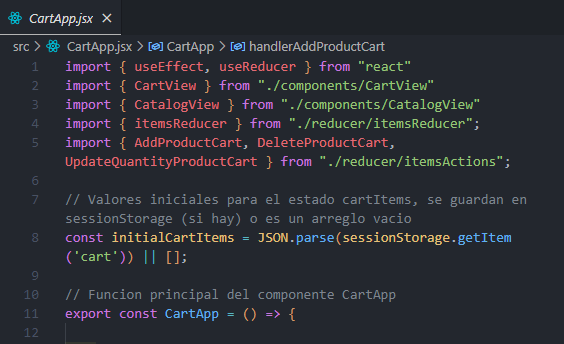
El hook será una función de flecha que exportará un objeto con la lógica del carrito.

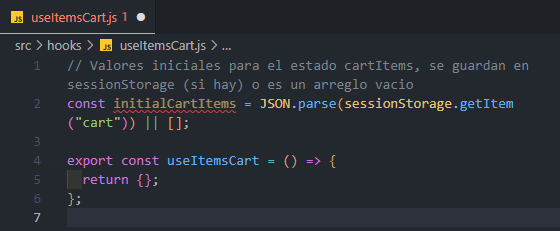


Paso 75: Trasladar la lógica al nuevo hook reutilizable

Traslada la lógica del carrito de compras desde el componente **CartApp** al nuevo hook personalizado llamado **useItemsCart**.

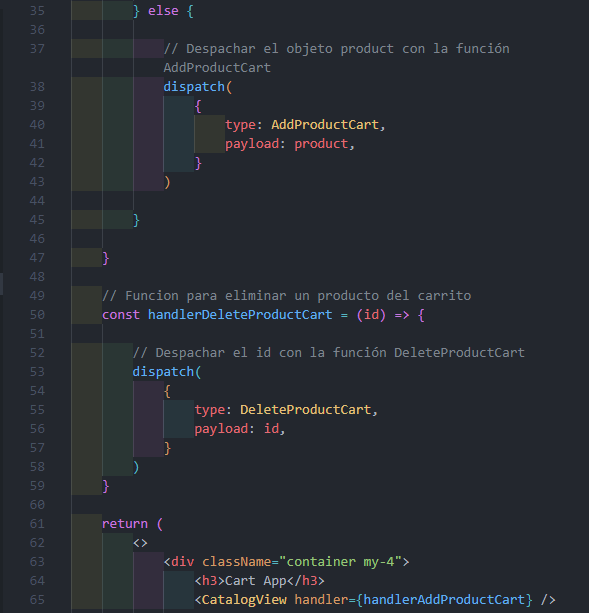
En el componente **CartApp**, corta la definición de la constante **initialCartItems** y pégalo en el archivo **useItemsCart**, antes de la función principal.

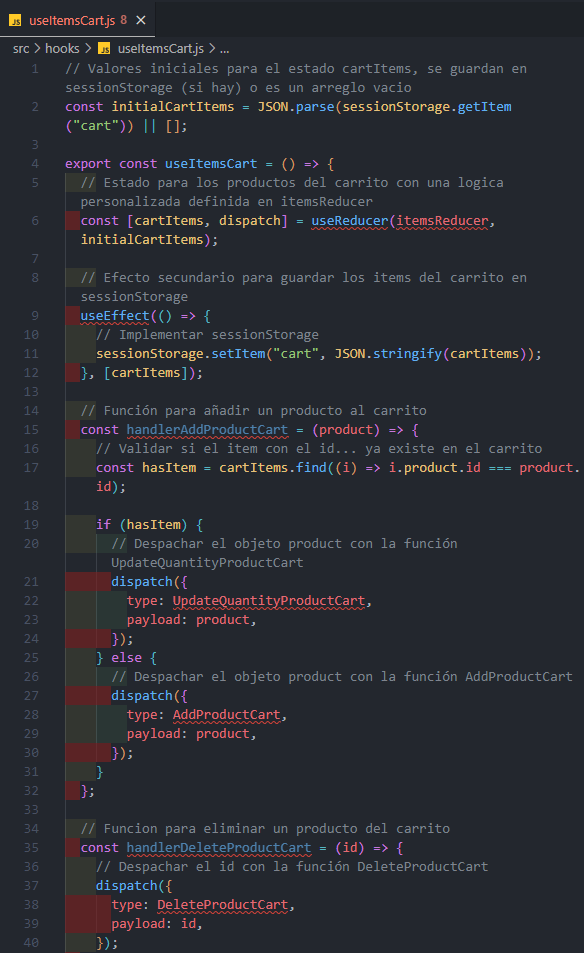




Seguidamente corta todo el contenido en la parte JavaScript del componente **CartApp** y pégalo en el nuevo hook **useItemsCart**, antes del **return**.



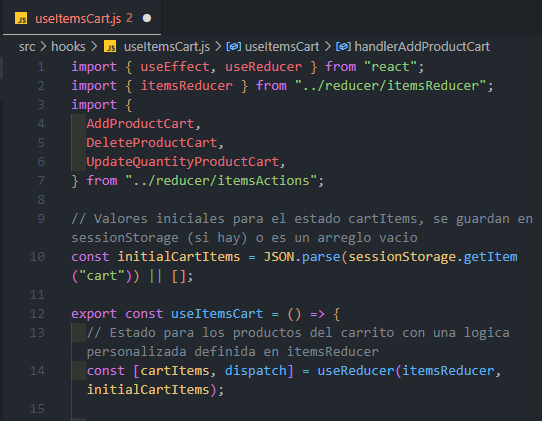




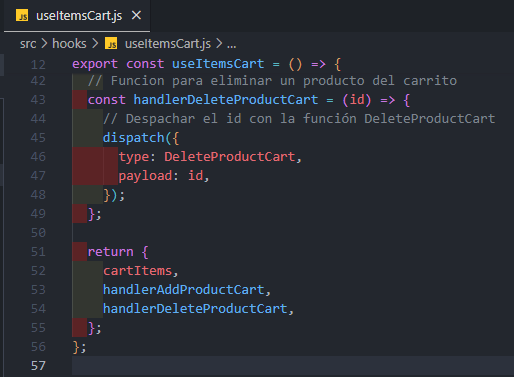


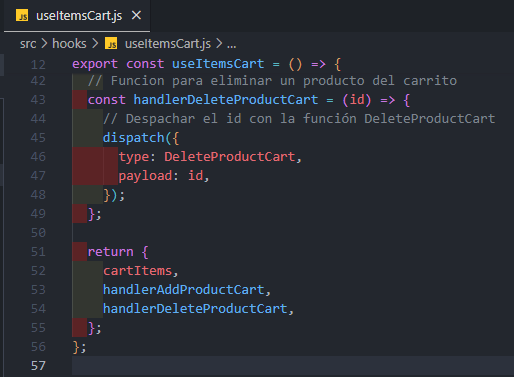
Paso 76: Importar los módulos necesarios y retornar un objeto con los datos y funciones

En el archivo **useItemsCart**, Importa **useReducer** y **useEffect** de React, además de **itemsReducer** del archivo **itemsReducer.js** y las constantes de acción del archivo **itemsAction.js**, ya que estas librerías y módulos se utilizaban previamente en el componente **CartApp**, antes de cortar el código.



El hook **useItemsCart** va a devolver un objeto que contiene el estado **cartItems** (como un objeto anidado) y las funciones **handlerAddProductCart** y **handlerDeleteProductCart**.

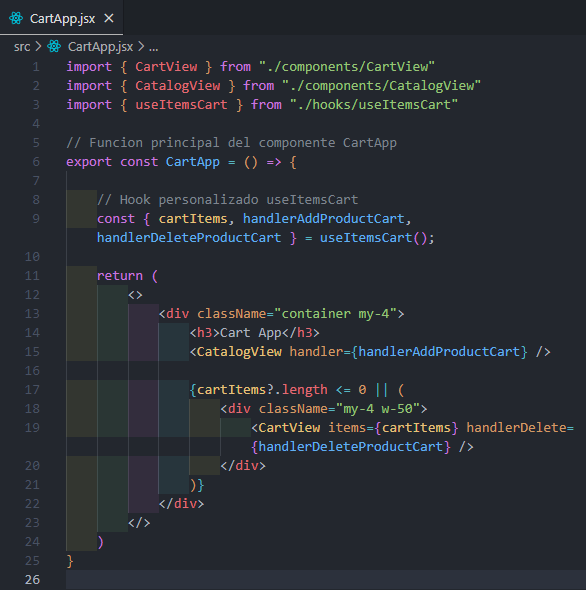




Paso 77: Importar el hook useItemsCart en CartApp

Importa el hook **useItemsCart** junto con los componentes **CartView** y **CatalogView**. Desestructura **cartItems**, **handlerAddProductCart**, y **handlerDeleteProductCart** del hook **useItemsCart**.

También elimina las importaciones que no se están utilizando.

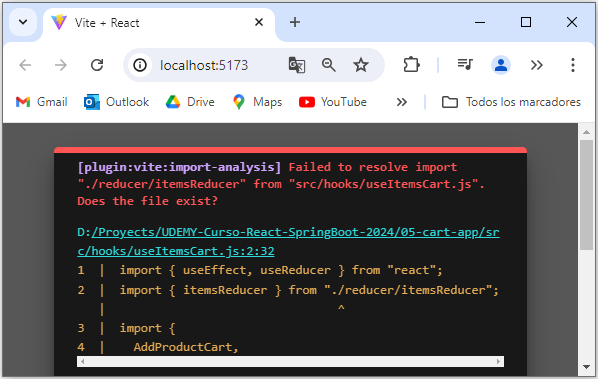


Se tiene en cuenta los elementos desestructurados:

* **cartItems:** Contiene los ítems del carrito, utilizados en **CartView** para mostrar los productos del carrito.
* **handlerAddProductCart:** Función para añadir un producto al carrito, pasada como propiedad a **CatalogView**.
* **handlerDeleteProductCart:** Función para eliminar un producto del carrito, pasada como propiedad a **CartView**.

En resumen, el hook **useItemsCart** encapsula la lógica de gestión del carrito de compras, mejorando la organización y reutilización del código en el componente **CartApp**. Con esta integración, **CartApp** se convierte en un componente más limpio y fácil de mantener.

Si aparece el siguiente error en el navegador, es posible que algo no se esté importando correctamente.



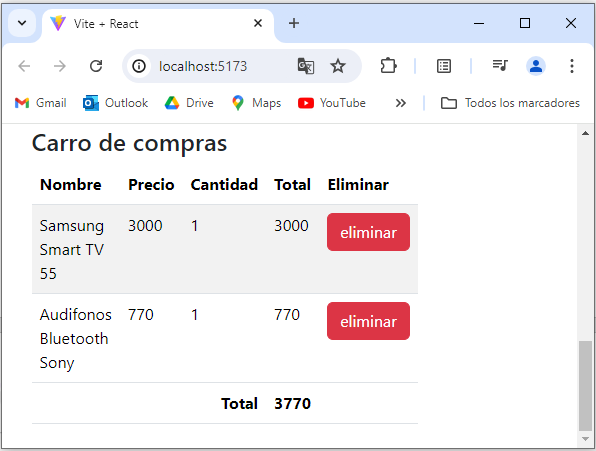
Para solucionar esto, revisa las rutas especificadas en los **imports**, es por ello que se recomienda importar manualmente que copiar y pegar las rutas. Recuerda que:

* **../** se utiliza para referenciar la carpeta raíz **src**.
* **./** se utiliza para referenciar la carpeta actual.

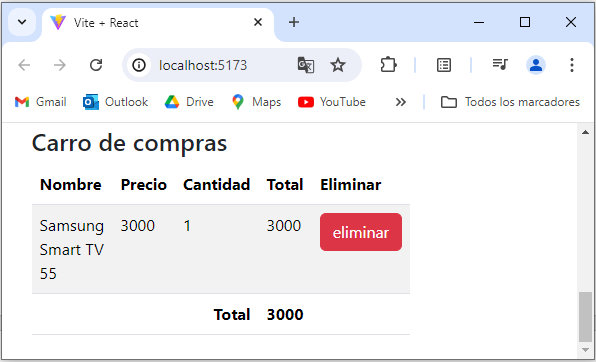
Asegúrate de que todas las rutas estén correctamente especificadas para evitar errores de importación.

Paso 78: Realizar una prueba en el navegador

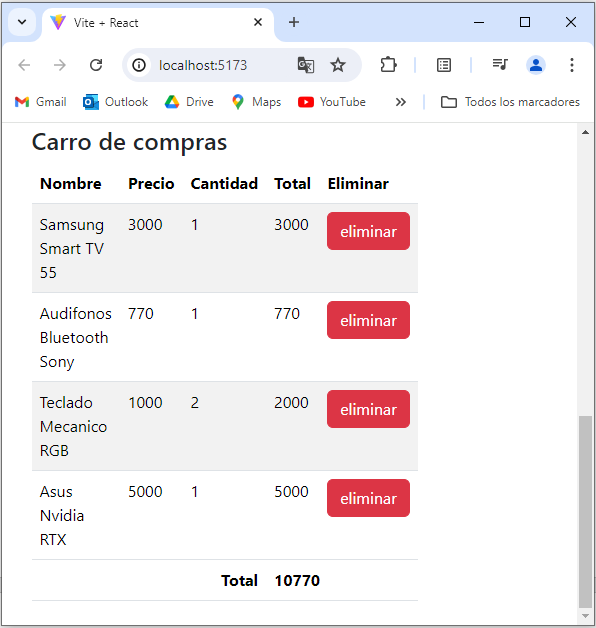
Procede a realizar una prueba en el navegador, de tal manera que permita agregar ítems al carrito al pulsar el botón **"agregar"**.



También asegúrate de eliminar unos ítems.



Por último, procede a agregar el mismo producto más de una vez y luego actualiza la página web para que no se pierdan los datos.



El total cambia y se mantiene correctamente al actualizar los ítems. La lógica personalizada ahora reside en el hook **useItemsCart**, lo que resulta en menos líneas de código en el componente principal y delega la lógica de negocio a los hooks y otros componentes. El hook **useItemsCart** maneja los ítems del carrito de compras utilizando **useReducer**.

Esto cumple con los estándares de React y mejora la organización y reutilización del código.