Actualizar el total del carrito

En esta sección, se calculará el total del carrito de compras, lo que implica sumar el valor total de cada producto, obtenido al multiplicar su precio por la cantidad correspondiente.

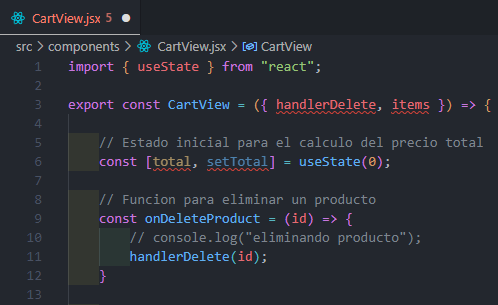
Pasos 50 al 53

Sigue estos pasos:

Paso 50: Definir un estado para el precio total en el componente CartView

Se utiliza el hook **useState** de React para definir el estado **total**, que representará el precio total de los productos en el carrito. Inicialmente, se establece en **0**.

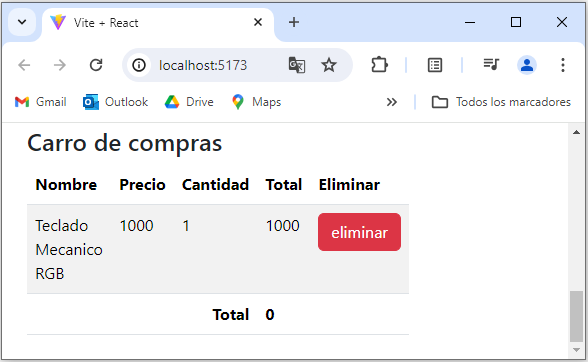
Recuerda que se ha definido una función **onDeleteProduct** que se encarga de eliminar un producto del carrito cuando se hace clic en el botón **"Eliminar"**. Esta función aún no se ha actualizado para reflejar el cambio en el precio total, pero se invocará más adelante cuando se realice esa actualización.



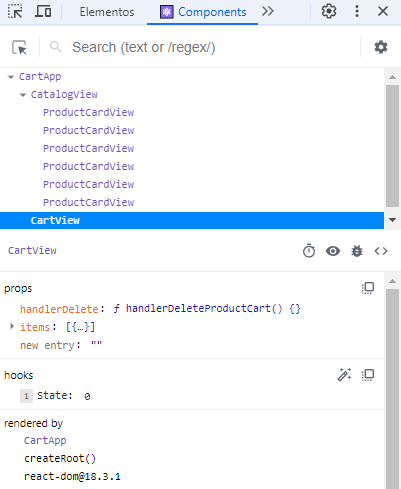
El precio total calculado (**total**) se muestra en la parte inferior de la tabla del carrito dentro de una de las celdas **<td>** contenidas en el elemento **<tfoot>**. Recuerda que la propiedad **colspan** para combinar varias celdas y mostrar el precio total en una única fila.



Al cargar la página en el navegador, el **total** se mostrará inicialmente como **0**, ya que aún no se ha calculado ningún precio total.



Al inspeccionar el componente **CartView** en la pestaña **Components** de Chrome DevTools, se puede observar que se ha creado un estado llamado **total** utilizando el hook **useState**, con un valor inicial de **0**.



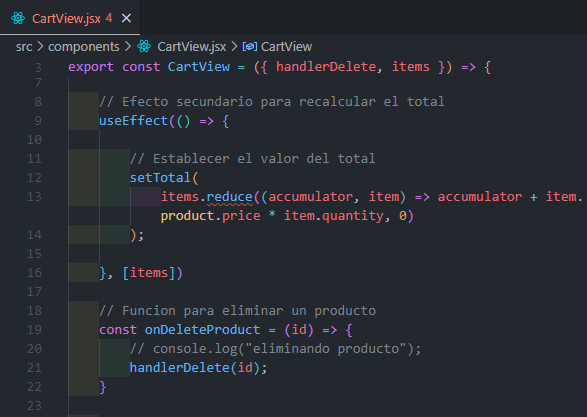
Paso 51: Definir un efecto secundario para recalcular el total

Se utiliza el hook **useEffect** para realizar un efecto secundario cada vez que cambian los elementos del carrito (**items**). Se pasa un array de dependencias como segundo argumento al **useEffect**, lo que significa que este efecto se ejecutará cada vez que **items** cambie.

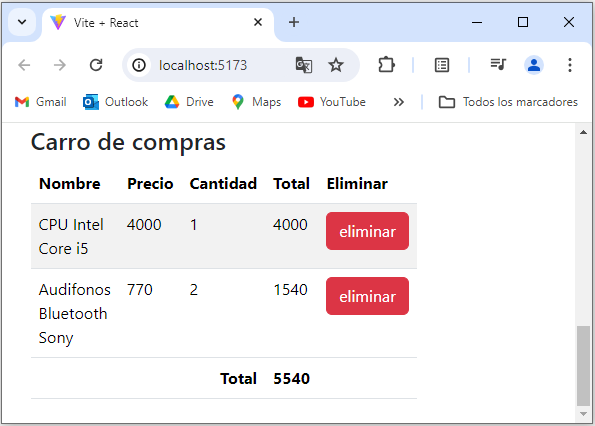
Dentro del cuerpo del **useEffect**, se calcula el precio total sumando el precio de cada producto multiplicado por su cantidad. Esto se realiza utilizando el método **reduce** en el array **items**.

El método **reduce** recorre cada elemento (**item**) del array **items** y acumula el resultado de la multiplicación del precio (**item.product.price**) por la cantidad (**item.quantity**) en una variable llamada **accumulator**. El segundo argumento de **reduce** es el valor inicial del acumulador, que se establece en **0**.

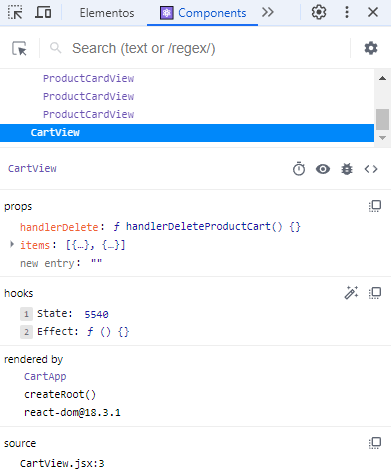
Una vez calculado el precio **total**, se actualiza el estado total utilizando la función **setTotal**.



En el navegador, se observa que el precio total se actualiza correctamente cada vez que se agrega, elimina o actualiza un producto en el carrito.



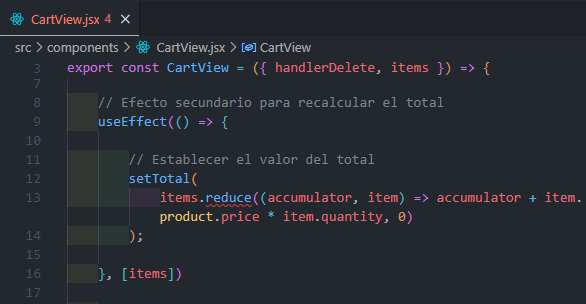
Al inspeccionar el componente **CartView** en la pestaña **Components** de Chrome DevTools, se puede verificar que se ha creado un estado llamado **total** y un efecto secundario que se dispara cuando cambian los elementos del carrito.

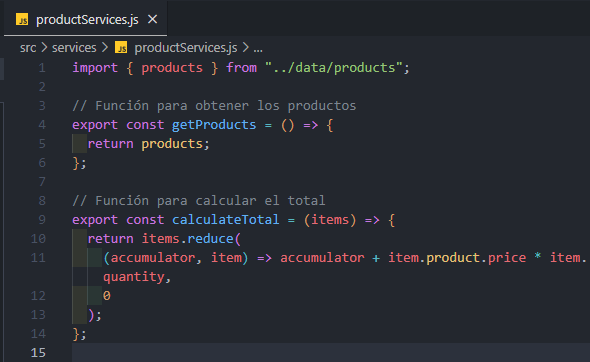


Paso 52: Trasladar la lógica del cálculo del total al servicio

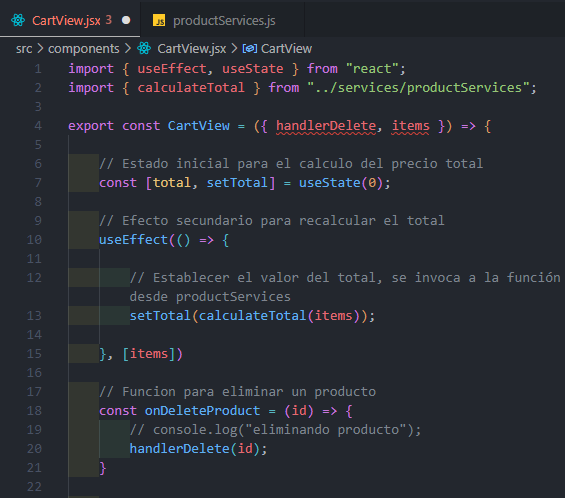
Se crea una nueva función llamada **calculateTotal** en el servicio **productService.js**. Esta función toma como argumento **items**, que representa los elementos del carrito, y utiliza el método **reduce** para calcular el precio total de todos los productos en el carrito.

Este método suma el resultado de multiplicar el precio de cada producto por su cantidad, comenzando desde un acumulador inicializado en 0 (se traslada la función definida en **setTotal** al servicio).



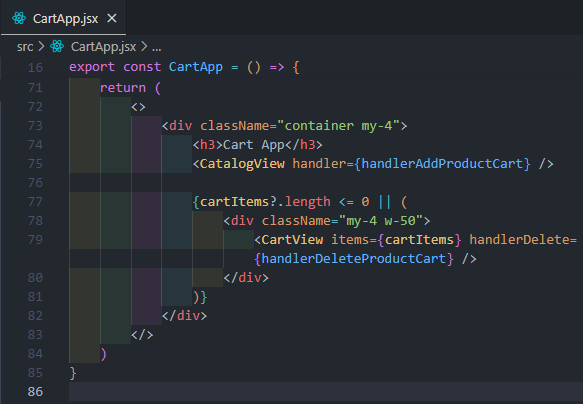


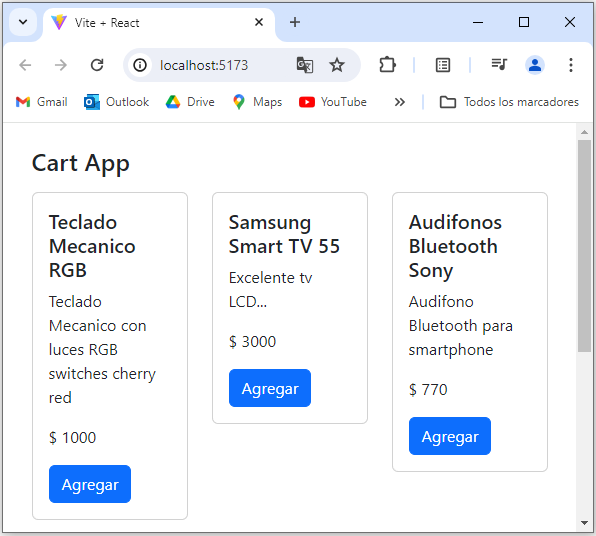
En el componente **CartView**, se importa la función **calculateTotal** desde **productService.js**. Dentro del efecto secundario **useEffect**, se llama a **calculateTotal** y se pasa **items** como argumento para calcular el nuevo total cada vez que cambian los elementos del carrito.



Paso 53: Realizar una prueba en el navegador

Antes de realizar una prueba, se realiza un ajuste de estilos en el componente **CartApp** para agregar un margen vertical al contenedor principal. Se agrega la clase **my-4** al **<div>** contenedor con la clase **container**, lo que proporciona un margen vertical de 4 unidades.





Se realiza una prueba en el navegador para verificar que las actualizaciones implementadas funcionen correctamente. Durante las pruebas, se pueden realizar diversas acciones, como agregar productos al carrito, eliminar productos del carrito y observar cómo se actualizan dinámicamente los totales y la visualización del carrito.

