Actualizar la cantidad del producto

En esta sección, se abordará la optimización del carro de compras al evitar la duplicación de productos y resolver el problema de las claves duplicadas en la consola. En lugar de duplicar un producto existente al agregarlo nuevamente al carro, se actualizará su cantidad incrementándola en uno cada vez que se agregue.

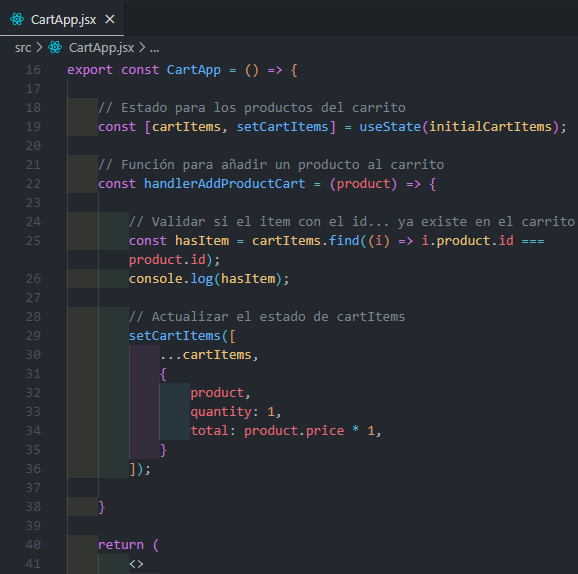
Pasos 39 al 44

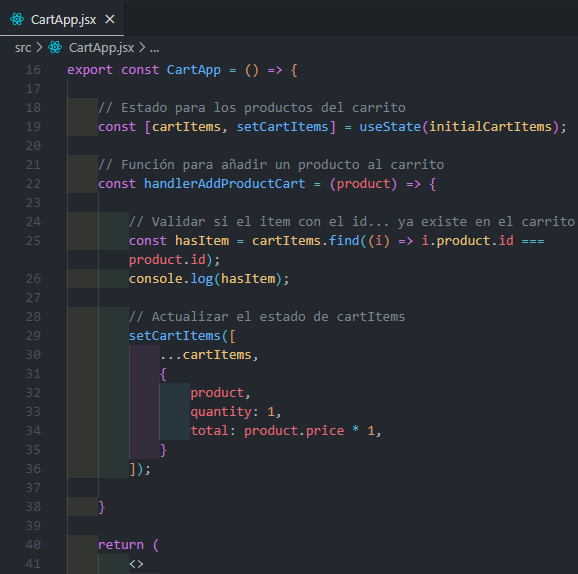
Sigue estos pasos:

Paso 39: Validar si un producto ya existe en el carrito

En el componente **CartApp**, se gestiona el estado del carrito de compras. Lo primero que se tiene que hacer es validar si el producto que se va a agregar ya existe en el carrito (en el estado **cartItems**).

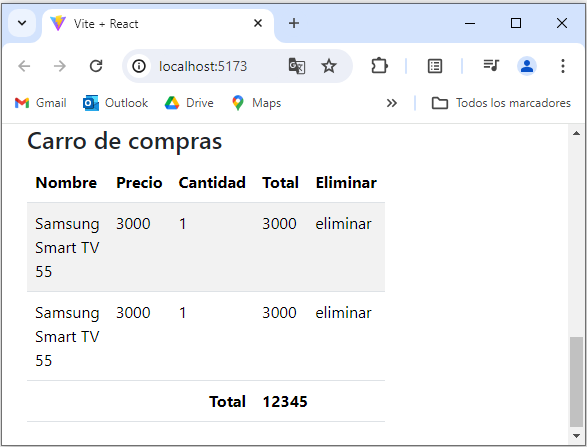
Para esto, se utiliza el método **find**, el cual busca un elemento que cumpla con cierta condición en un arreglo. En este caso, se quiere encontrar un elemento en el carrito cuyo **id** coincida con el **id** del producto que se va a agregar.

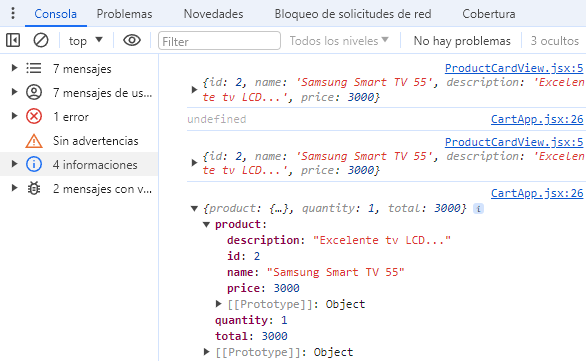




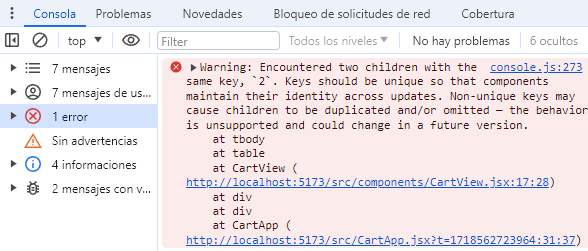
El código dará como resultado **undefined** si el producto no está en el carrito, o el objeto del producto si ya existe. Luego, se actualiza el estado del carrito con **setCartItems**.

Al agregar el mismo producto dos veces en el navegador, primero imprimirá **undefined** y luego el objeto **cartItem**, representando el producto existente en el carrito con sus atributos.





El error en la consola indica que la duplicación de la clave se debe a que aún no se ha implementado la actualización de la cantidad del producto que ya existe en el carrito.



Paso 40: Actualizar la cantidad del producto si ya existe en el carrito

Implementa una condición para actualizar la cantidad de un producto si ya existe en el carrito. Cambia la instrucción **console.log** por una estructura condicional **if**. Si el producto ya está en el carrito (**hasItem** es verdadero), se actualiza la cantidad (**quantity**); de lo contrario, se agrega un nuevo ítem al carrito.

En el bloque **if**, se actualiza el estado de **cartItems** excluyendo el ítem que se va a actualizar. El operador Spread (**...**) se utiliza en **cartItems** seguido de un método **filter** que elimina el ítem con el **id** del producto que se actualizará.

Por otro lado, en el bloque **else**, se actualiza el estado de **cartItems** agregando el nuevo ítem al carrito, manteniendo los ítems existentes y agregando el nuevo con una cantidad inicial de **1** y su precio total (se copia y pega el código que actualizaba el estado de **cartItems**).



Paso 41: Eliminar el atributo total

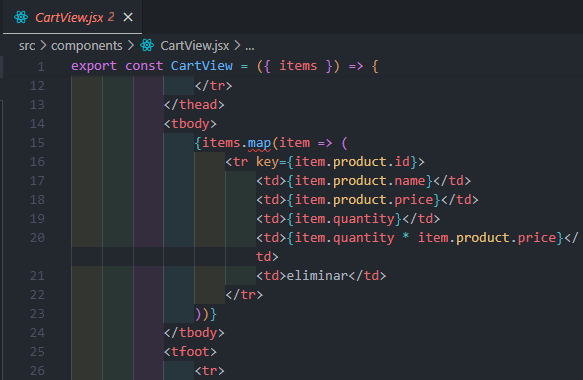
Simplifica la estructura de los ítems del carrito eliminando el atributo **total**. Recuerda que cuando se actualiza la cantidad de un producto existente, simplemente se actualizará la cantidad en lugar de duplicar el ítem en el carrito.

Al actualizar el estado de **cartItems**, ahora simplemente se elimina el ítem existente (si lo hay) con el método **filter**, y luego agregamos el producto nuevamente con la cantidad actualizada (aumenta en **1** la cantidad (**quantity**) del ítem validado con **hasItem**) en el caso de que ya esté en el carrito. Si el producto no está en el carrito, simplemente se agrega con una cantidad inicial de **1**.

Luego, en el componente **CartView**, no se agrega la renderización del atributo **total**, porque se calcula el total para cada ítem en el componente multiplicando la cantidad por el precio del producto.



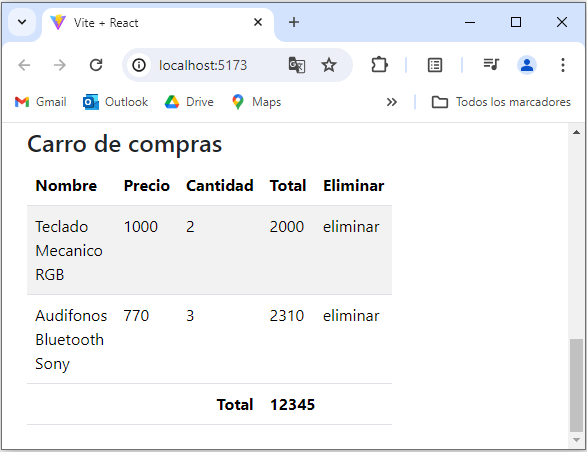




Paso 42: Agregar el producto más de una vez en el navegador

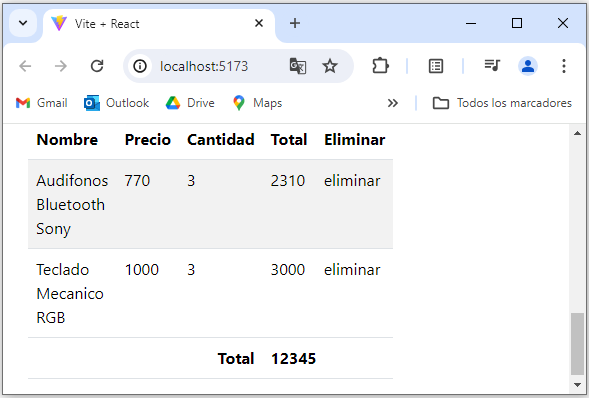
En el navegador, agrega varias veces el mismo producto, por ejemplo, **"Teclado mecánico RGB"** y **"Audífonos Bluetooth Sony"**.

Observa que la cantidad del producto seleccionado se incrementa correctamente cada vez que se agrega y el total correspondiente a ese producto se recalcula automáticamente.



Dado que el producto se elimina y se vuelve a agregar con la cantidad actualizada, aparecerá como el último elemento en la tabla del carrito.

Por ejemplo, si inicialmente se tiene los productos **"Teclado mecánico RGB"** y **"Audífonos Bluetooth Sony"** en el carrito, al agregar otro **"Teclado mecánico RGB"**, este se moverá al final de la lista en la tabla del carrito con la cantidad actualizada.



Paso 43: Utilizar el método map para agregar un ítem al carrito

Se puede simplificar la lógica de actualización del carrito utilizando el método **map** en lugar de **filter** para manejar la cantidad de un producto existente.

Comenta el bloque **if** que utiliza **filter** para eliminar y volver a agregar el producto con la cantidad actualizada.

Utiliza el método **map** para recorrer **cartItems**. Este método devuelve un nuevo arreglo modificado, lo que va a permitir actualizar la cantidad del producto existente sin necesidad de usar el operador spread.

Se sigue la siguiente lógica: **map** recorre cada ítem del arreglo **cartItems**. Si el **id** del producto coincide con el **id** del producto que se está agregando, actualiza la cantidad sumando **1**. Si no coincide, devuelve el ítem sin cambios.

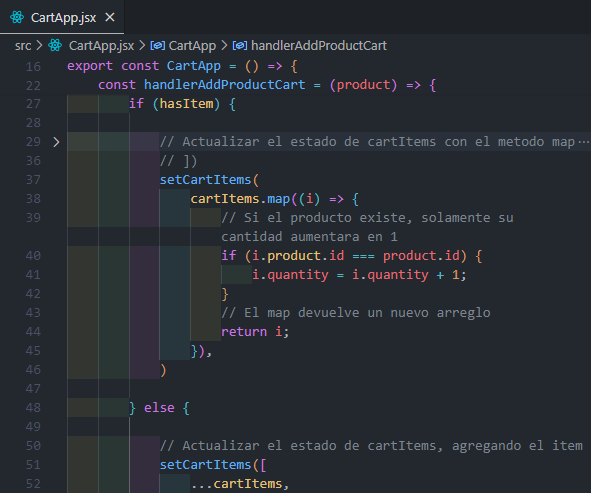


Paso 44: Simplificar el código para agregar un ítem y mantener el orden de los elementos

No es necesario utilizar el operador **spread** para clonar el arreglo, ya que **map** devuelve un nuevo arreglo modificado.

Se elimina la función de flecha que excluía el producto por su **id** y se reemplazará con una estructura más extensa utilizando llaves. Dentro de esta estructura, se añade una condición que verificará si el producto con el **id** especificado ya existe en **cartItems**. Si existe, se actualizará la cantidad de ese producto incrementándola en **1**.

Si el producto ya existe en el carrito, se incrementará su cantidad utilizando **i.quantity + 1**. Es importante recordar que map devuelve un nuevo arreglo con las modificaciones aplicadas. Al final, la función **map** retornará el ítem modificado (**i**).



En el navegador, al agregar varias veces el mismo producto, la cantidad y el total se actualizarán correctamente sin duplicar los ítems y sin cambiar el orden de los elementos en el carrito. Ahora se puede agregar tantos ítems como sea necesario, y la cantidad se actualizará adecuadamente.

Este método asegura que el carrito mantenga el orden original de los elementos y solo actualice la cantidad del ítem correspondiente. Es una forma eficiente y clara de manejar la actualización de cantidades en el carrito de compras.

