Manejando el estado del carro de compras

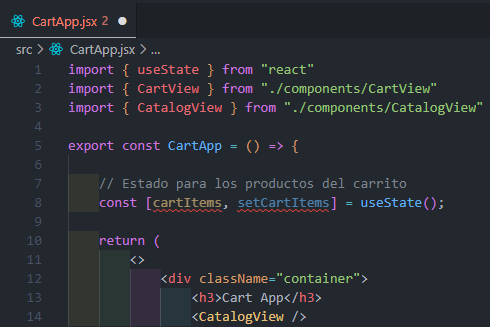
En esta sección, se procederá a administrar el estado del carrito de compras. Se utilizará el hook **useState** de React para gestionar los productos, su cantidad y el subtotal. Cada vez que se agregue un artículo al carrito, se actualizara el estado para reflejar los cambios y mantener actualizado el componente.

Pasos 27 al 30

Sigue estos pasos:

Paso 27: Definir el estado inicial cartItems

Utiliza el hook **useState** de React para establecer el estado inicial del carro de compras. Crea una constante llamada **cartItems** e inicialízala con un valor por defecto, que por el momento se deja vacío.



Luego, define una constante llamada **initialCartItems** fuera de la función principal del componente. Esta constante contendrá datos de ejemplo iniciales para el estado **cartItems**.

El valor inicial de **cartItems** se establece como un arreglo de objetos. Cada objeto contiene información sobre un producto en el carro, representado por un objeto anidado vacío definido con el atributo **product**, su cantidad (**quantity**) y total (**total**). Los valores por defecto para **quantity** y **total** se establecen en 0.

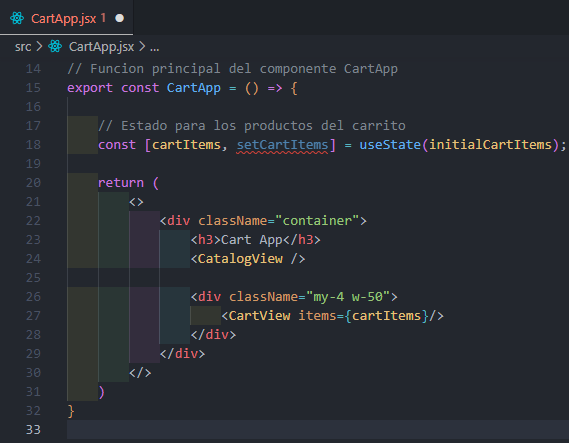
Estos datos de ejemplo son útiles para establecer la estructura básica de los elementos del carro de compras y proporcionar un punto de partida para el estado inicial.

Recuerda que se asigna el valor inicial de **cartItems** utilizando la constante **initialCartItems**. A medida que se van agregando elementos al carro, se agregarán nuevos objetos que contienen el producto, la cantidad y el total. El objeto **product** maneja atributos como precio, id, nombre, descripción, que son los mismos atributos de un objeto en el arreglo de productos definido en **products.js**.



Paso 28: Iterar con los productos en el carrito

En el componente **CartApp**, se trasmite el estado de **cartItems** al componente **CartView** durante su renderizado mediante la propiedad **items**.



Luego, en el componente **CartView**, se utiliza la propiedad **items** que se recibe como argumento en la función principal del componente y se realiza un mapeo sobre ella, dentro del elemento **<tbody>** con el método **map** para renderizar cada elemento del carro de compras.

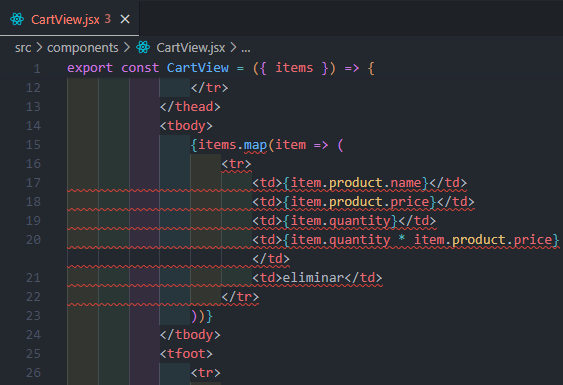
Dentro del cuerpo de la función de flecha definida en el método **map** se traslada la fila de la tabla (definida con el elemento **<tr>** y sus elementos anidados) que contendrá cada ítem del carrito.





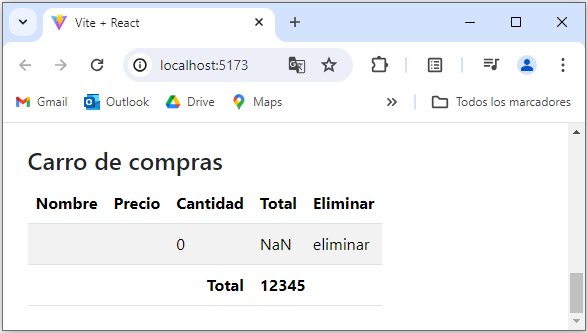
En los elementos **<td>**, se reemplaza el texto estático por los valores de los atributos del objeto **item**. Es importante recordar que **items** es un arreglo de objetos (el valor inicial se encuentra en **initialCartItems**) que contiene los elementos del carrito con los atributos **product**, **quantity** y **total**.

El objeto anidado **product** va a tener los atributos **name** y **price**. Luego, se calcula el total multiplicando la cantidad (**quantity**) por el precio del producto (**product.price**). La opción para eliminar se mantiene por ahora sin cambios.



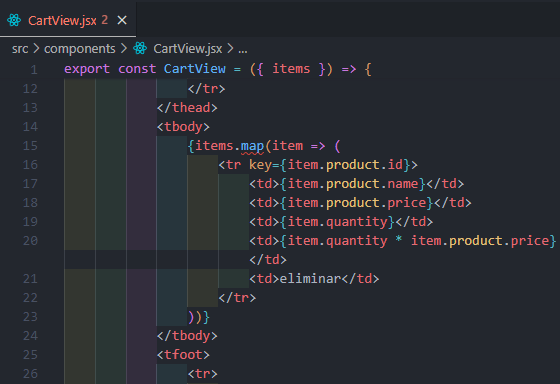
En el navegador, los campos **name** y **price** no se muestran porque no se ha proporcionado un valor por defecto para ellos en el código. La cantidad muestra **0** porque así se inicializó el estado **quantity** en el objeto **initialCartItems**.

El cálculo del total muestra **"NaN"** (Not a Number) porque, al multiplicar la cantidad por el precio, el resultado es indeterminado debido a la falta de valores definidos.

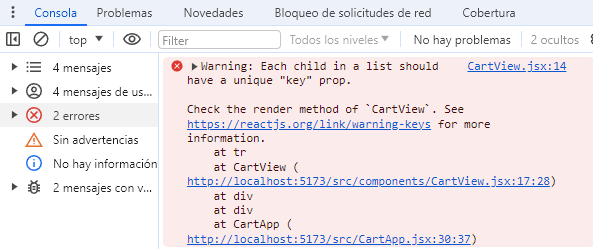


Paso 29: Solucionar el error del atributo key, y NaN con una prueba

Añade el atributo key al elemento padre **<tr>** dentro del mapeo de los elementos del carrito (**items**). Esto asegura que cada elemento tenga un identificador único y resuelve el error relacionado con la falta de un atributo **key**.



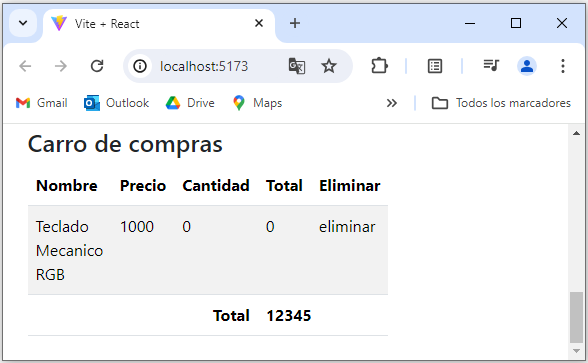
Aparte del error en la consola de **NaN**, porque el cálculo del total no es un número, el error de la consola respecto al atributo **key** persistirá:



Para solucionar ambos errores de la consola, en el cálculo del **total**, asegúrate de que el objeto **initialCardItems** tenga un valor para el atributo **id** (este no se ha definido). Puedes utilizar datos de ejemplo de **products.js** para definir un producto dentro de **initialCartItems** en el atributo **product**, el cual contiene un objeto.

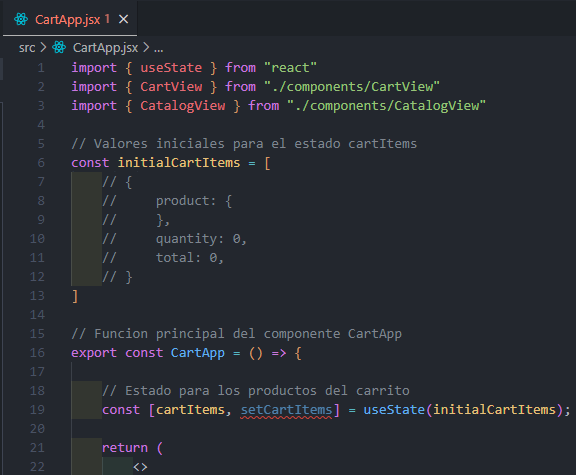


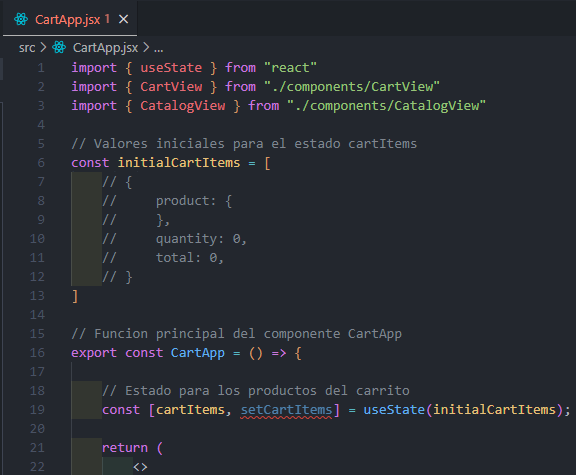
Con estos ajustes, en el navegador se mostrará correctamente el nombre y precio del producto, mientras que la cantidad y el total se mostrarán con un valor de **0**, ya que se inicializan con ese valor en **initialCartItems**.



Paso 30: Cambiar el estado inicial de cartItems

Para permitir agregar productos dinámicamente al carrito al hacer clic en el botón, modifica el estado inicial **initialCartItems** para que esté vacío, eliminando cualquier producto predefinido. Comenta el código relacionado con **initialCartItems**, manteniendo los corchetes, pero sin contenido (arreglo vacío).





**Nota:** En Visual Studio Code, puedes usar **CTRL + K + C** para comentar varias líneas de código seleccionadas y **CTRL + K + U** para descomentarlas.

Con estos cambios, el carrito estará inicialmente vacío, lo que permitirá agregar productos dinámicamente, sin mostrar errores en la consola del navegador.

