Separando el componente Form

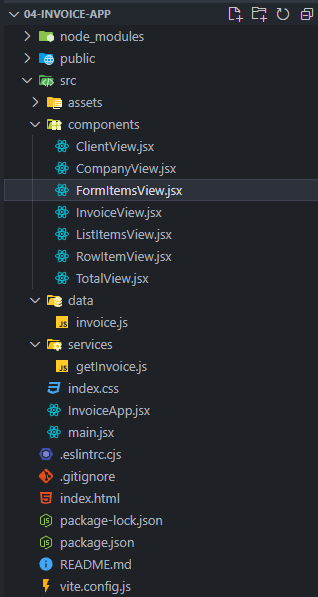
En esta sección, se procederá a desacoplar todo el contenido relacionado con los ítems y el formulario del componente principal **InvoiceApp**. El objetivo es modularizar el código para hacerlo más limpio, manejable y reutilizable.

Pasos 103 al 111

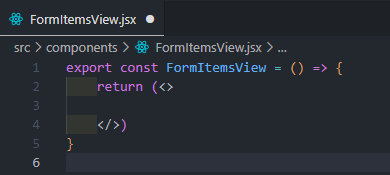
Sigue estos pasos:

Paso 103: Crear el componente FormItemsView

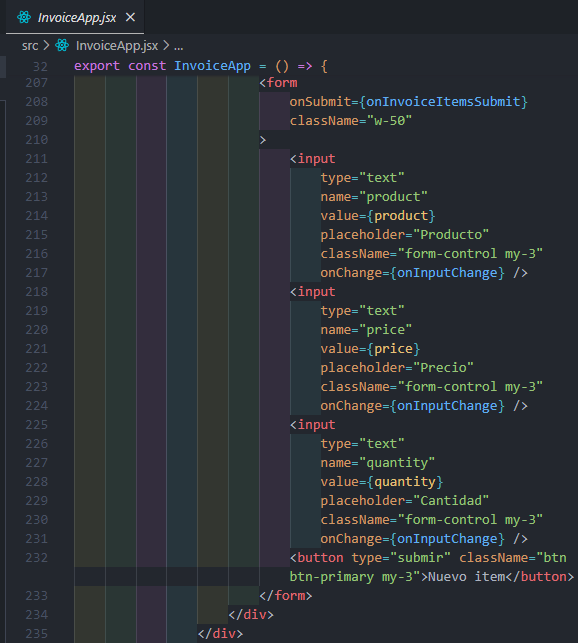
Crea un nuevo archivo llamado **FormItemsView.jsx** dentro de la carpeta **components**, también puede llevar el nombre **FormInvoiceView.jsx**.

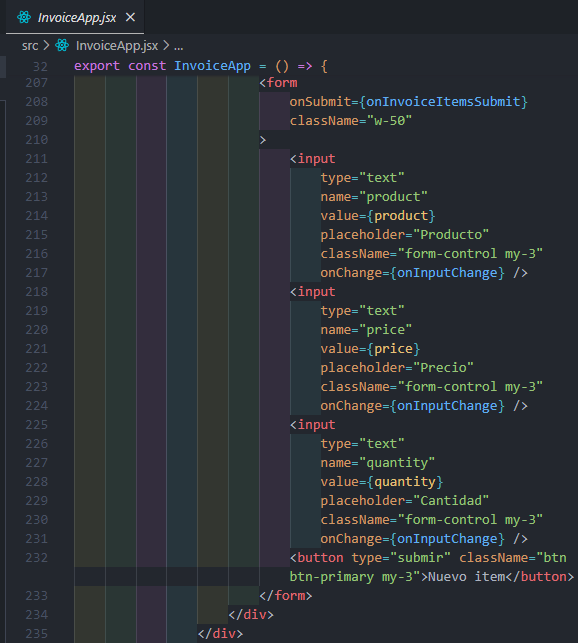


Define una función de flecha exportable con el mismo nombre del archivo, que por el momento retornará un fragmento vacío.



Ve al componente **InvoiceApp** y corta el formulario desde la etiqueta **<form>** hasta la etiqueta de cierre del formulario.





Pega el formulario en el componente **FormItemsView**.

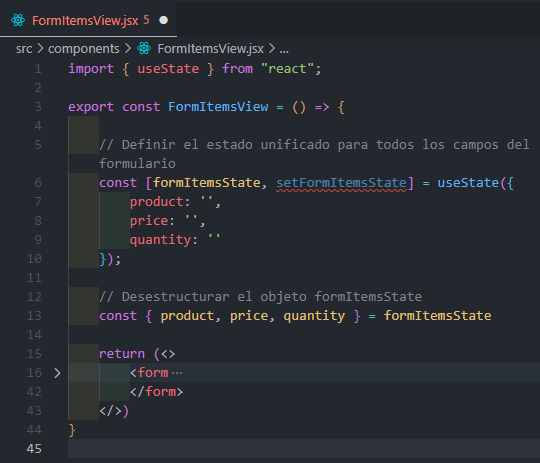


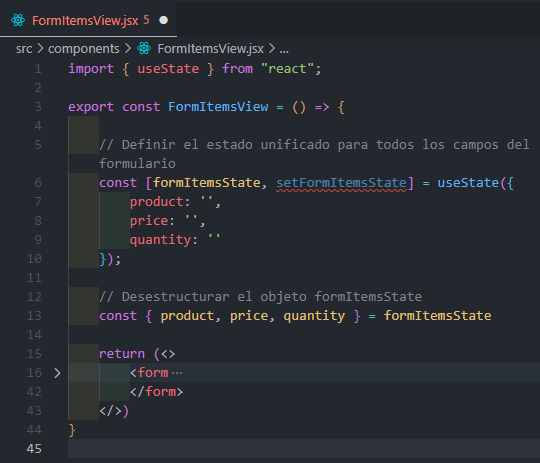
Paso 104: Trasladar los hooks useState y useEffect al componente FormItemsView

Corta el estado unificado **formItemsState** y su desestructuración del componente **InvoiceApp**.

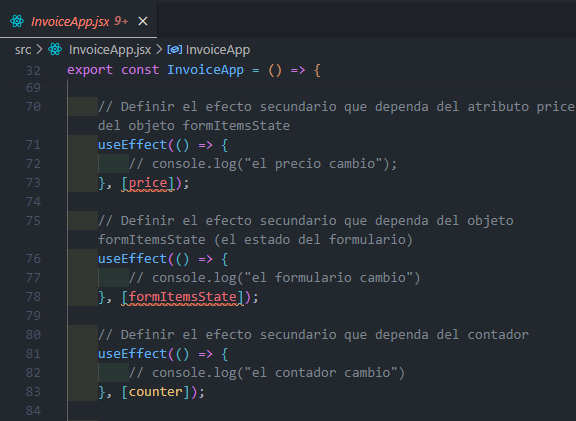


Pega el código anterior en el componente **FormItemsView**





Corta los **useEffect** relacionados con **price** y **formItemsState** del componente **InvoiceApp**.



Pega el código anterior en el componente **FormItemsView**, después de la desestructuración del estado.

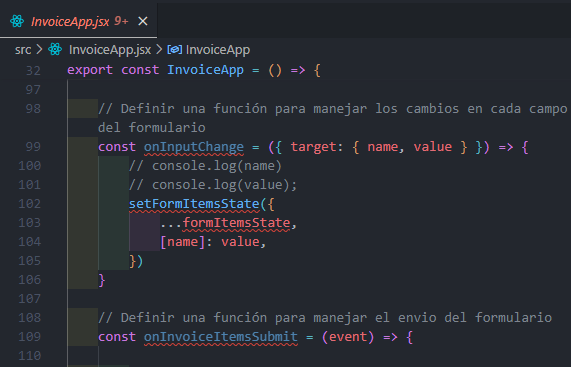


**Nota:** Asegúrate de importar **useState** y **useEffect** de React al inicio del componente **FormItemsView**.

Paso 105: Trasladar las funciones al componente FormItemsView

Traslada las funciones **onInputChange** y **onInvoiceItemsSubmit** del componente principal **InvoiceApp** al nuevo componente **FormItemsView**.

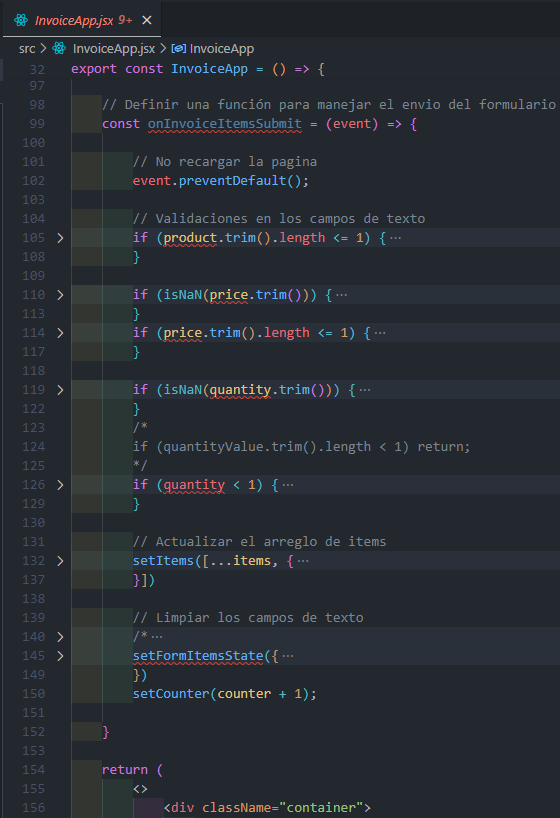
Corta la función **onInputChange** del componente **InvoiceApp** y pégala en el componente **FormItemsView** debajo de los **useEffect**.

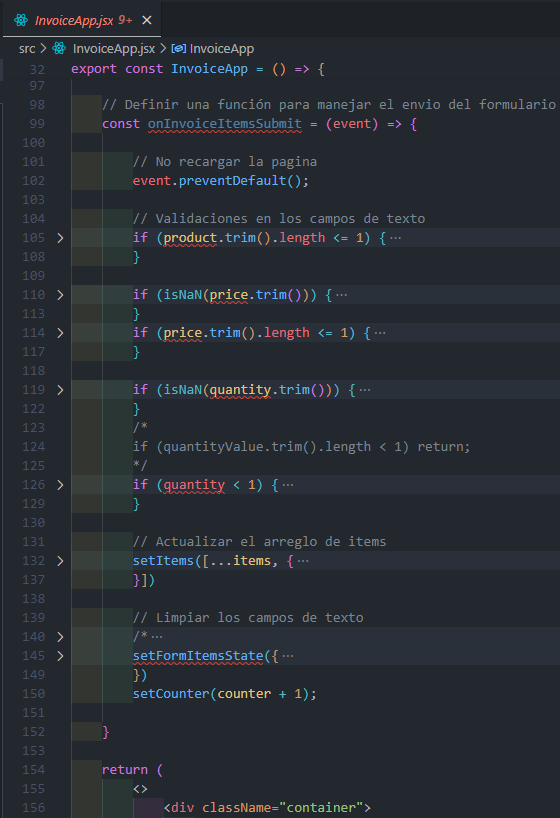


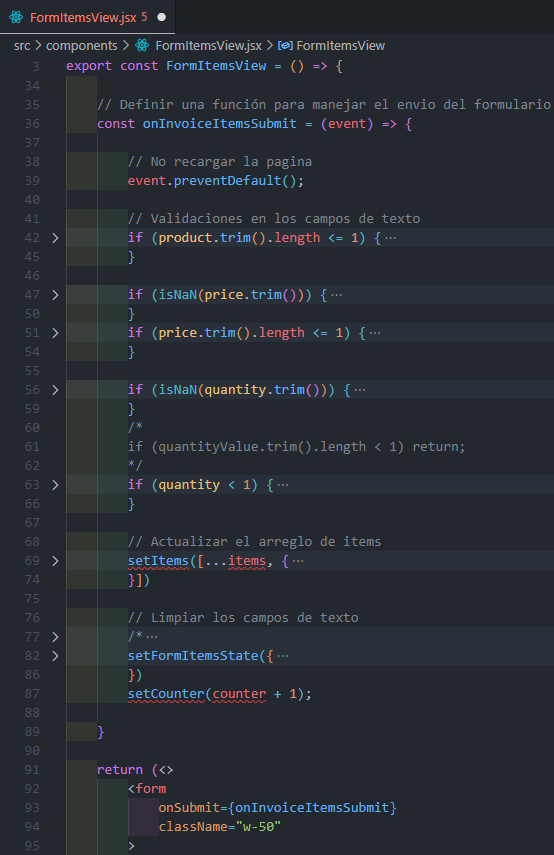


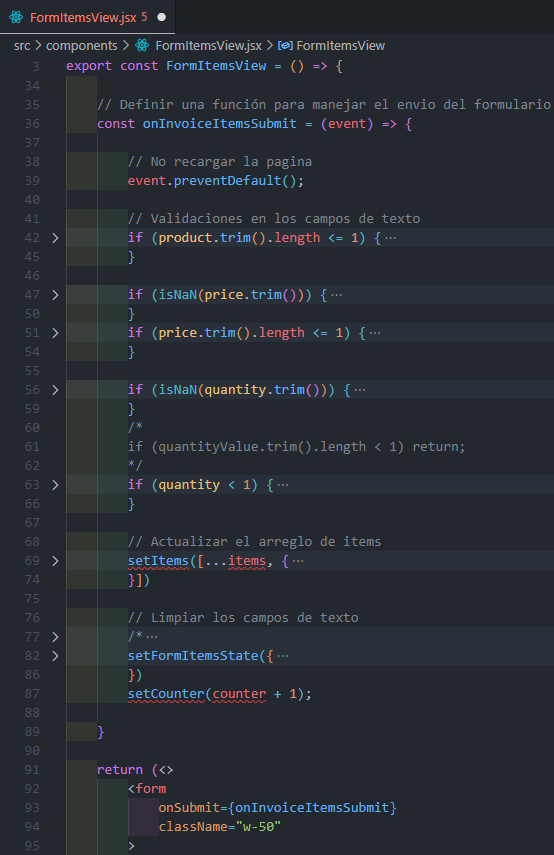


Copia (no cortar) la función **onInvoiceItemsSubmit** del componente **InvoiceApp** y pégala en el componente **FormItemsView**.







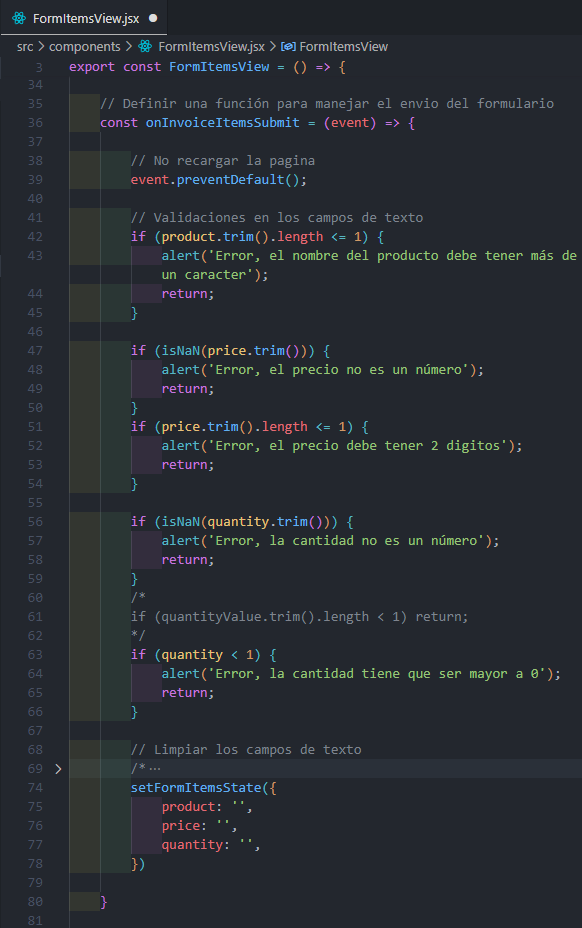


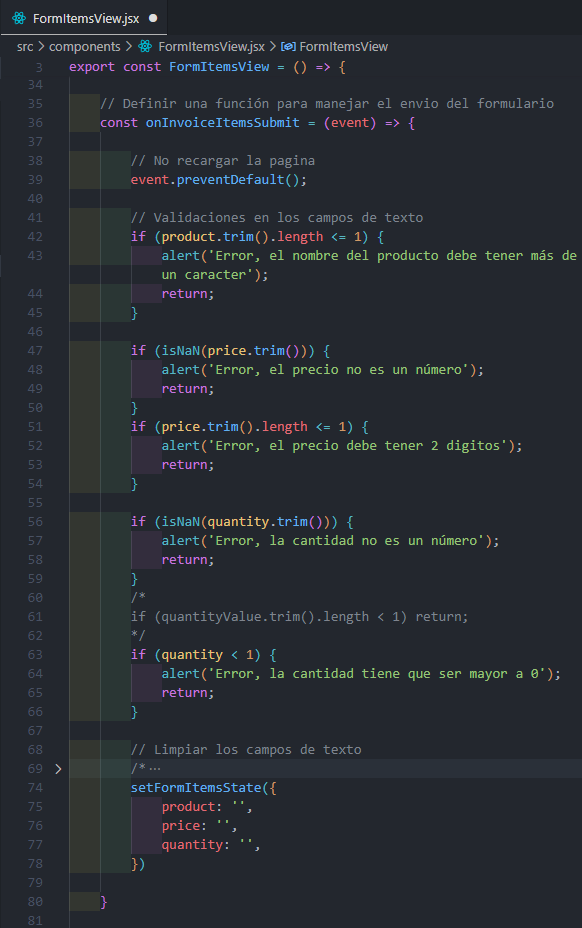
Paso 106: Modificar los procesos definidos en el componente FormItemsView

Realiza las siguientes modificaciones en la función **onInvoiceItemsSubmit** en el componente **FormItemsView**:

Elimina las referencias a **setItems** y **setCounter** de la función **onInvoiceItemsSubmit**, ya que estos estados se manejan en el componente **InvoiceApp**.

Mantén el método **preventDefault**, las validaciones de los campos (**product**, **price** y **quantity**) y la función **setFormItemsState** para limpiar los campos del formulario.





En resumen, la función **onInvoiceItemsSubmit** va a manejar la validación y el reseteo del formulario.

Paso 107: Modificar los procesos definidos en el componente InvoiceApp

Continúa realizando ajustes en la función **onInvoiceItemsSubmit** en el componente **InvoiceApp**.

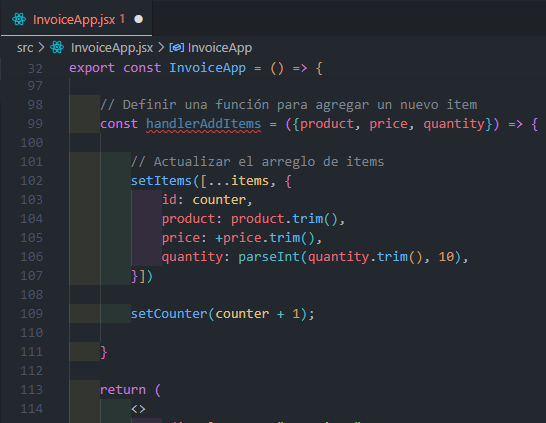
Elimina el manejo de eventos (relacionado con el objeto **event**) y validaciones en esta función, ya que ahora serán responsabilidad del componente **FormItemsView**.

Conserva la función **setItems** para actualizar el estado de los ítems en este componente.

Cambia el nombre de **onInvoiceItemsSubmit** a **handlerAddItems** (o **handlerInvoiceItemsSubmit**), para reflejar que esta función maneja la adición de ítems a la factura.

Desestructura los atributos **product**, **price** y **quantity** del parámetro de la función **handlerAddItems** (se elimina el parámetro **event**), para poder utilizarlos directamente.

Elimina la llamada a la función **setFormItemsState**, ya que no es necesario en esta función.



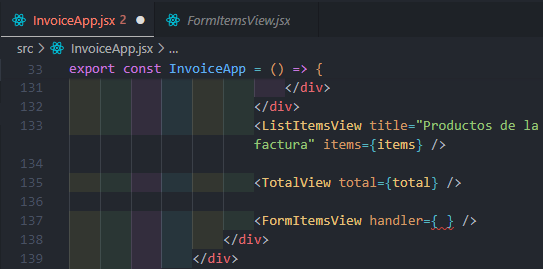
En resumen, la función **handlerAddItems** va a manejar la adición de un nuevo ítem y el contador.

Paso 108: Renderizar el componente FormItemsView

Primero, asegúrate de importar el componente **FormItemsView** en el archivo **InvoiceApp**.

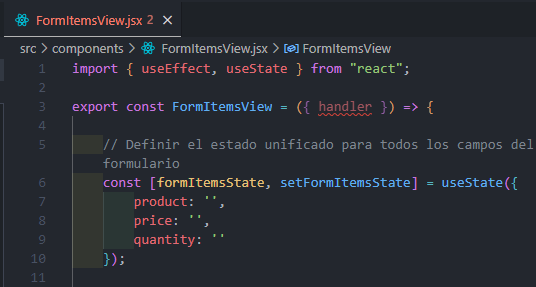


Dentro del componente **InvoiceApp**, renderiza el componente **FormItemsView** debajo de **TotalView**. Este componente va a tener una propiedad (prop) llamada **handler**, por el momento se deja en blanco su valor que se va a pasar.

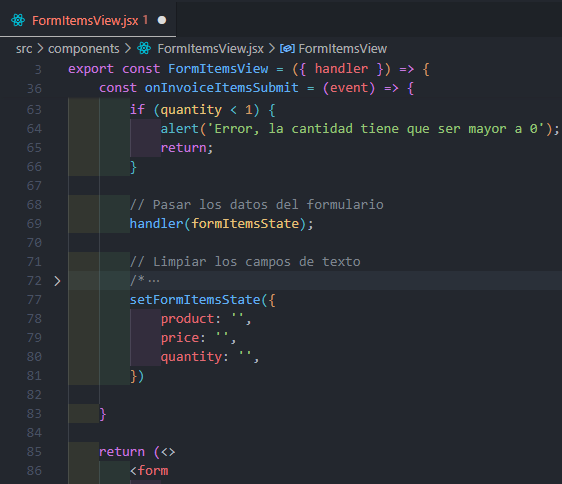


Paso 109: Pasar datos desde el componente hijo al componente padre

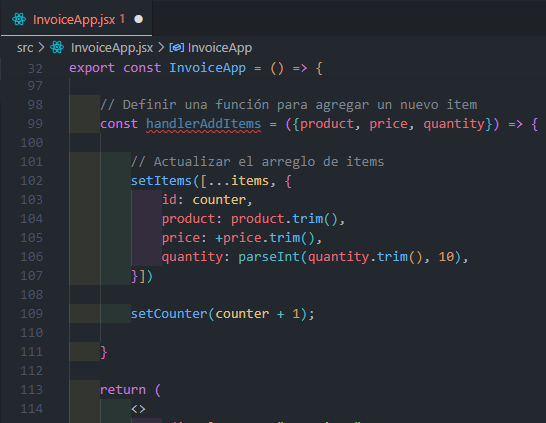
En el componente **FormItemsView**, desestructura la prop **handler** y defínela como una función dentro del componente.



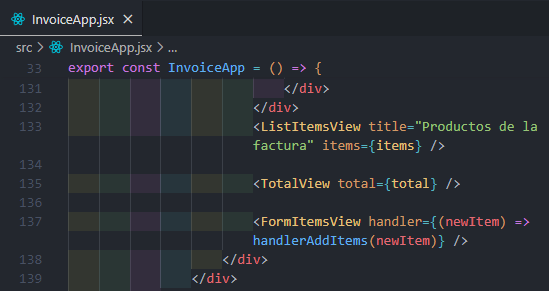
La función **handler** se invoca con el argumento **formItemsState** (el estado de los campos del formulario, que es un objeto que contiene los atributos **product**, **price** y **quantity**) dentro de la función **onInvoiceItemsSubmit**. Esto debe hacerse después de las validaciones de los campos, pero antes de limpiar los campos del formulario.



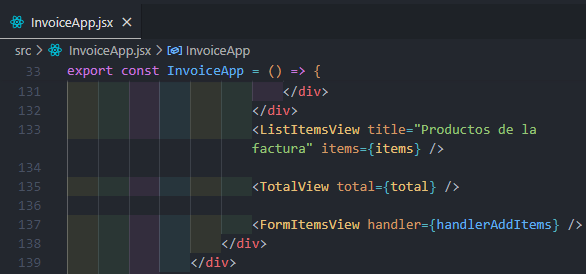
Recuerda que en el componente padre (**InvoiceApp**) se ha definido la función **handlerAddItems**, que recibe los datos introducidos en el formulario desde el componente hijo (**FormItemsView**). Esta función actualiza el estado de los ítems (**items**) y el contador (**counter**).



Al renderizar **FormItemsView** en **InvoiceApp**, pasa la función **handlerAddItems** como la prop **handler**. A su vez se le asigna un parámetro llamado **formItem** o **newItem** y este se le pasa como argumento de la función.



Esta prop puede simplificarse usando una función de flecha si el argumento recibido se pasa directamente como parámetro a otra función. En este caso, se puede utilizar solo el nombre de la función, lo que resulta en una sintaxis más limpia y concisa.



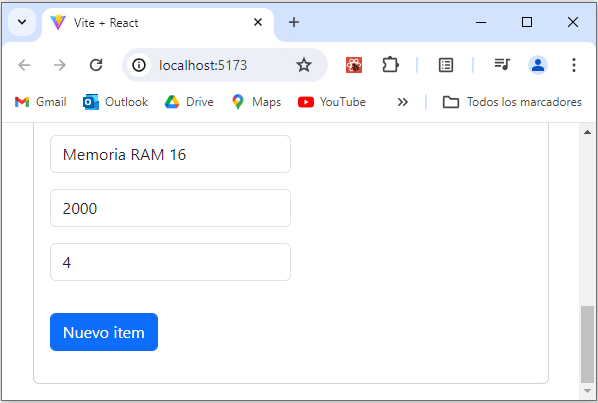
Al seguir estos pasos, se logra que los datos del formulario en **FormItemsView** se envíen al componente padre **InvoiceApp** a través de la función **handlerAddItems**. Este proceso se conoce como pasar datos desde un componente hijo a un componente padre en React utilizando funciones.

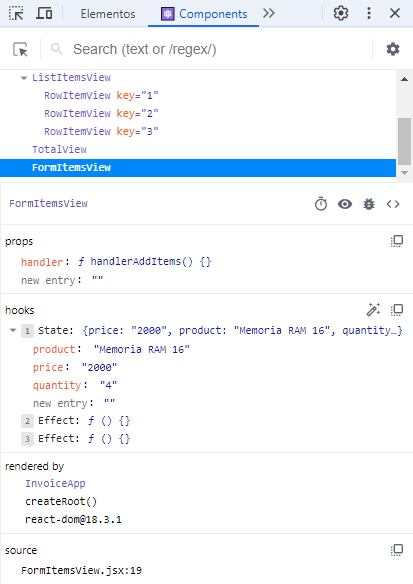
Paso 110: Realizar una prueba en el navegador

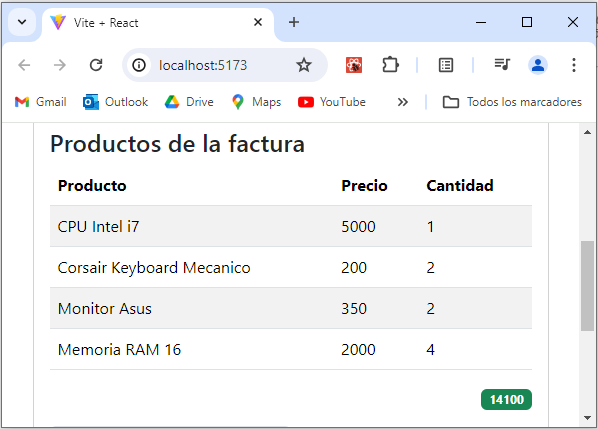
Procede a realizar una prueba en el navegador. Recuerda que el formulario cuenta con validaciones y observa cómo se pasan los datos desde el componente hijo al componente padre a través de la función **handlerAddItems**.

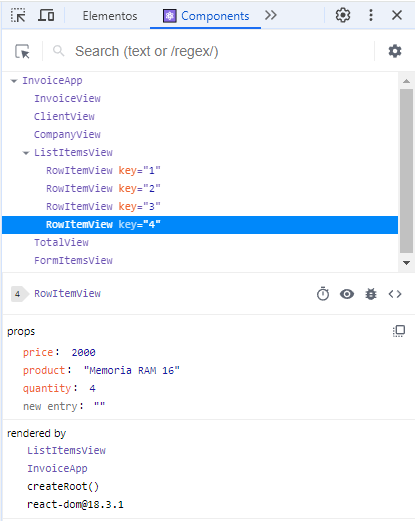
El componente padre (**InvoiceApp**) le pasa la función **handler** al componente hijo (**FormItemsView**); y este último, mediante una función, pasa estos datos que se escriben en el formulario al **handler** para que los reciba nuevamente el componente padre y guarde los datos en el estado de **items**.

Adicionalmente en la consola de Chrome, sección **Components**, el estado de los campos del formulario se encuentra en el componente **FormItemsViews** y no en el componente principal **InvoiceApp**. Luego al enviar el formulario, el nuevo ítem se crea en el componente **RowItemView** que posee un valor **4** en su propiedad **key**.









Paso 111: Borrar comentarios innecesarios en el código fuente

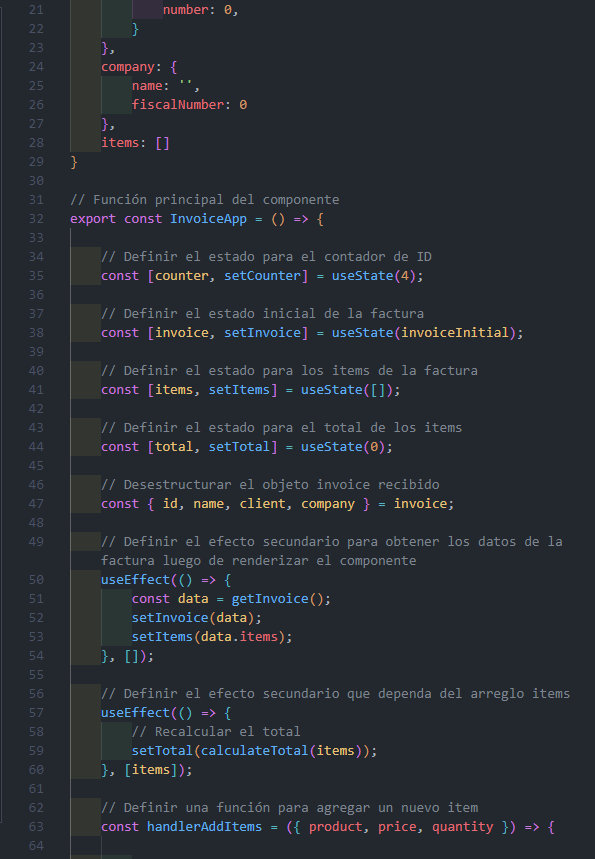
Opcionalmente puedes eliminar los comentarios que no son necesarios en el código fuente, así como también los hooks **useEffect** que no realizan ninguna acción.

En el componente principal **InvoiceApp** se borra lo siguiente:

* La importación del objeto **invoice** desde la carpeta **data**.
* La constante **invoice** para obtener la función **getInvoice**
* Los estados para cada campo del formulario (**productValue**, **priceValue** y **quantityValue**)
* El efecto secundario (**useEffect**) que depende del contador (**counter**).
* Las funciones individuales para manejar los cambios en los campos del formulario (**onProductChange**, **onPriceChange** y **onQuantityChange**).
* Todos los mensajes de consola definidos con **console.log**.

En conclusión, el código fuente del archivo **InvoiceApp** debería quedar de la siguiente manera (los recuadros rojos indican que algo se ha borrado):







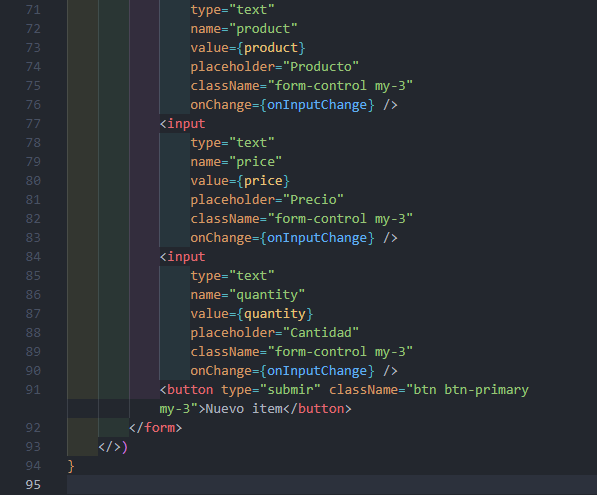
En el componente **FormItemsView** se borra lo siguiente:

* La importación del hook **useEffect** de React.
* Los efectos secundarios que dependen del atributo **price** del objeto **formItemsState** y del mismo objeto **formItemsState**.
* La condición innecesaria del campo **quantityValue** en las validaciones del formulario.
* Las funciones para establecer los estados **setProductValue**, **setPriceValue** y **setQuantityValue**
* Todos los mensajes de consola definidos con **console.log**.

En conclusión, el código fuente del archivo **FormItemsView** debería quedar de la siguiente manera (los recuadros rojos indican que algo se ha borrado):







Al revisar la parte HTML de ambos componentes, se nota que queda más limpio. En la parte JavaScript del componente **InvoiceApp**, se tiene la función **handlerAddItems**, que permite recibir los datos de **FormItemsView**, guardarlos y anexarlos al estado **items**.

La función **handler** se invoca para pasar los datos al componente padre (**InvoiceApp**). Esta función se recibe en el componente hijo (**FormItemsView**) desde el componente padre (**InvoiceApp**) como una propiedad.