Manejando el estado del formulario con useState

En esta sección, se abordará el tema de cómo gestionar el estado para cada uno de los cuadros de texto del formulario en React utilizando el hook useState. Al utilizar el evento **onChange** en cada **<input>** del formulario, se guardará el valor introducido en el estado de React.

Además, es recomendable crear un componente separado que maneje el formulario y la lógica de los eventos para una mejor organización y mantenibilidad del código; esto se aplicará más adelante.

Pasos 62 al 66

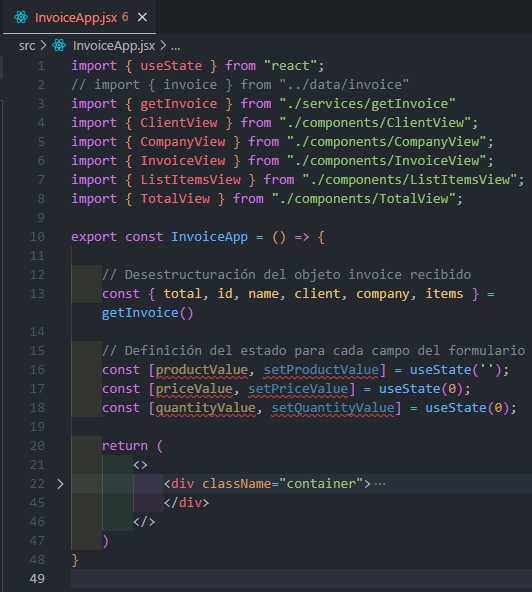
Sigue estos pasos:

Paso 62: Definir el estado para cada campo

Define el estado para cada campo del formulario. Para aquello utiliza el hook **useState**, el cual devuelve un arreglo que contiene el valor del estado y una función para actualizar dicho estado. Importa el hook **useState** de **React** dentro del componente **InvoiceApp.jsx**.

Para cada campo, como **product**, **price** y **quantity** (se utiliza el valor definido en el atributo **name**), se utiliza **useState**. Como valor inicial, se establece un string vacío para el producto y **0** para el precio y la cantidad, ya que estos últimos son valores numéricos.

**Nota:** Para asegurar que la importación de **useState** tenga mayor prioridad en el orden que los componentes importados, se puede colocar al inicio de las importaciones de React.





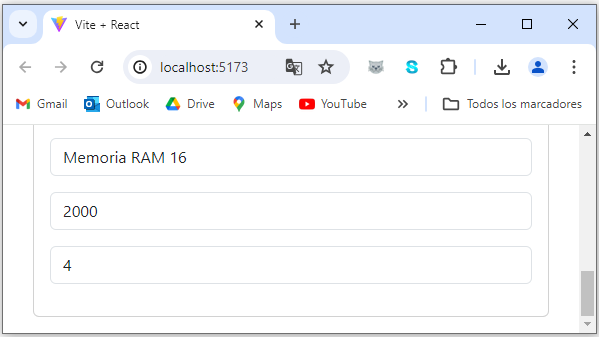
Paso 63: Guardar los valores del formulario en el estado

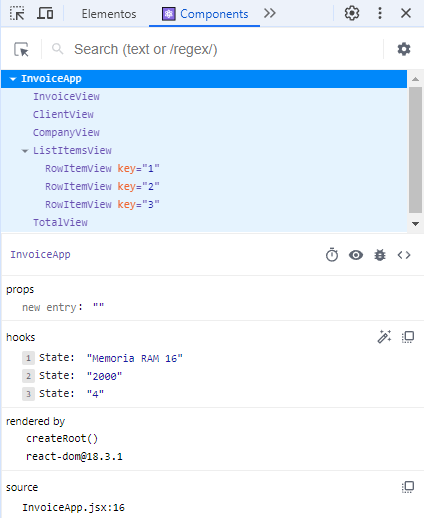
Realiza cambios en el evento **onChange** a cada uno de los campos de entrada para que, cuando el valor de un campo cambie, se actualice el estado correspondiente utilizando las funciones **setProductValue**, **setPriceValue** y **setQuantityValue**.





Con estos cambios se puede visualizar en tiempo real los valores que el usuario introduce en cada campo a través de la consola del navegador en la sección de **Components**.





Con este cambio ya se puede crear un arreglo con los valores asignados y añadir un nuevo ítem a la lista de items de la factura utilizando el operador spread y el hook **useState**.

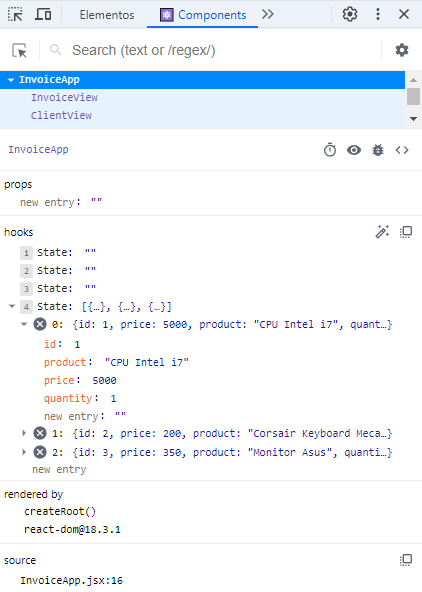
Paso 64: Manejar del estado de items

Para manejar el estado de los ítems de la factura, crea otro **useState** para los ítems e inicializalo con los ítems obtenidos de la factura. Renombra el arreglo de objetos **items** a **itemsInitial** cuando se desestructura de **getInvoice** para evitar conflictos de nombres.

El estado de **items** se utiliza en el componente **ListItemsView** para iterar y mostrar los items. Esto permite modificar, guardar y agregar nuevos items al estado.



Al abrir la consola del navegador, en la sección de **Components**, se puede visualizar un estado que se encarga de mostrar los items.



Paso 65: Implementar la lógica para agregar nuevos elementos a la lista

El evento **onSubmit** en un formulario de React se utiliza para manejar lo que ocurre cuando el formulario es enviado, y se define en la etiqueta de apertura de **<form>**.

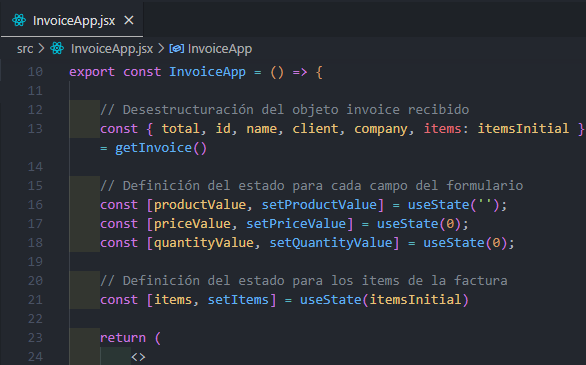
Cuando se activa este evento, se llama a una función que recibe el objeto **event** como parámetro, permitiendo acceder a la información del formulario y realizar acciones específicas, como enviar datos al servidor o realizar validaciones.

En el contexto del componente **InvoiceApp**, los datos introducidos en los campos de texto (**product**, **price** y **quantity**) se van a agregar a la lista de ítems de la factura al momento de enviar el formulario.

Sigue estos pasos para definir y utilizar el manejador del evento **onSubmit**:

1. Cuando un formulario HTML es enviado, por defecto la página se recarga. Para evitar esto en una aplicación React, se utiliza **event.preventDefault** en el manejador del evento **onSubmit**.
2. Se utiliza el hook **useState** para mantener y actualizar el estado de los **items**. El estado inicial es **itemsInitial**, que proviene de la función **getInvoice**.
3. Cuando el formulario es enviado, se crea un nuevo objeto que representa el nuevo ítem. Este objeto contiene los mismos atributos que los items existentes: **id**, **product**, **price**, y **quantity**. Los valores para este nuevo ítem se obtienen de los estados **productValue**, **priceValue**, y **quantityValue**. El **id** se asigna un valor estático de **4** solamente para esta prueba.
4. Para agregar el nuevo ítem a la lista de items sin mutar el arreglo original, se utiliza el operador spread (**...**). Esto crea una nueva lista que contiene todos los elementos del arreglo original más el nuevo ítem.
5. Finalmente, se utiliza la función **setItems** para actualizar el estado de **items** con la nueva lista. Ten en cuenta que todo el contenido del argumento de la función **setItems** se define entre corchetes (**[ ]**) porque se trata de un arreglo.

Todo esto se puede integrar de la siguiente manera:

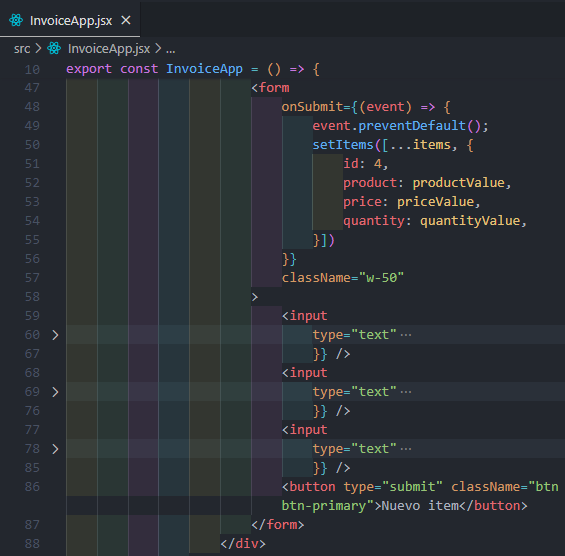




Paso 66: Agregar el botón de envío del formulario

Agrega un botón de tipo **submit** (el tipo se define con el atributo **type**) dentro del formulario, debajo del último **<input>**. Este botón va a tener la clase de bootstrap **btn btn-primary**, el cual sirve para aplicarle el estilo de un botón.

Por otro lado, se utiliza la clase de Bootstrap **w-50** en la etiqueta **<form>** para que el formulario ocupe el 50% del contenedor.



En el navegador, se tiene en cuenta lo siguiente:

* Los campos de texto se pueden probar para asegurar que el estado se actualice correctamente.
* Al enviar el formulario, el nuevo item se agrega a la lista de items (en la consola, sección **Components**).
* En la página web, la tabla de ítems se actualiza cada vez que se añade un nuevo ítem.

