Recalculando el total con el hook useEffect

En esta sección, se implementará una función para calcular automáticamente el total de la factura cada vez que se agregue un nuevo ítem. Esto se realizará de manera dinámica para asegurar que el total refleje inmediatamente cualquier cambio.

Pasos 98 al 102

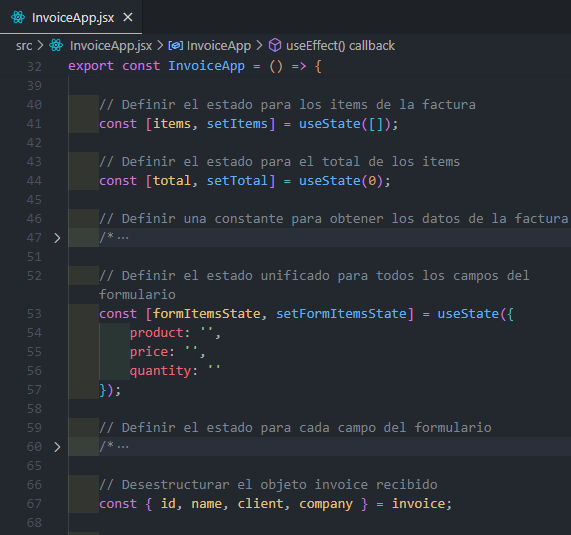
Sigue estos pasos:

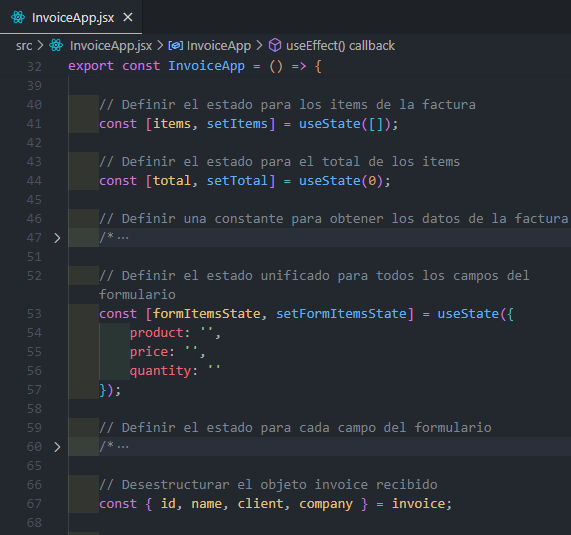
Paso 98: Definir un estado para el total

Crea un estado en React para manejar dinámicamente el **total** de los ítems en la factura. Recuerda que **useEffect** trabaja en conjunto con **useState** para que pueda reaccionar a cambios en el estado.

Define un nuevo estado llamado **total** utilizando la función **useState**. Este estado inicialmente comenzará en **0**.

Luego elimina la desestructuración del atributo **total** del objeto **invoice** (se utiliza para establecer los valores iniciales de la factura), ya que ahora se manejará el **total** directamente a través del estado.

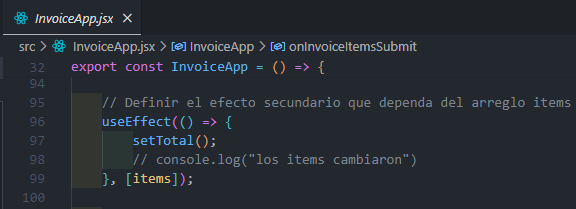




Paso 99: Modificar el efecto secundario de los ítems

Al cambiar los ítems en la factura, el total debe actualizarse en consecuencia. Por lo tanto, dentro del **useEffect** que depende del estado de los ítems, se define una función para actualizar el estado. La función **setTotal** calculará el total y lo establecerá nuevamente en el estado.

Además, se comenta la instrucción **console.log** para evitar la impresión redundante de mensajes.

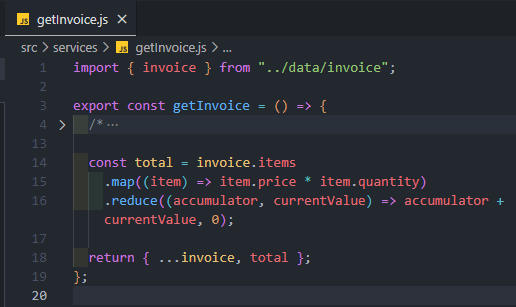


Paso 100: Realizar mejoras en la función

Dentro del archivo **getInvoice.js** (se encuentra dentro de la carpeta **service**), procede a realizar las siguientes mejoras.

Recuerda que el método **map** se utiliza para transformar cada ítem en su valor total (precio por cantidad). Luego, el método **reduce** se utiliza para sumar todos estos valores para obtener el total de la factura.

Procede a cortar todo el contenido definido en la constante **total**



Crea una nueva función exportable llamada **calculateTotal**. Esta función se va a encargar de calcular el **total** de la factura a partir de los ítems proporcionados como argumento. Toma un parámetro llamado **items**, que por defecto es un arreglo vacío en caso de que no se proporcione ningún valor.

Luego pega el contenido cortado en la parte de retorno (**return**) de la función. Este contenido representa el cálculo del total de la factura a partir de los ítems proporcionados.

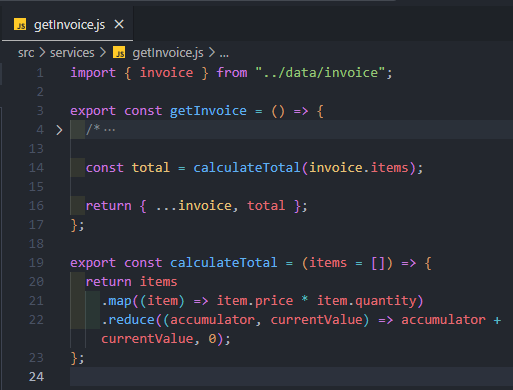




Dentro de la función **getInvoice**, se modifica el cálculo del **total**. En lugar de realizar el cálculo directamente dentro de la función, se utiliza la nueva función **calculateTotal**.

Se llama a **calculateTotal** pasando **invoice.items** como argumento. Esto permite que la lógica de cálculo del total esté encapsulada en la función **calculateTotal**, lo que hace que **getInvoice** sea más limpio y legible.

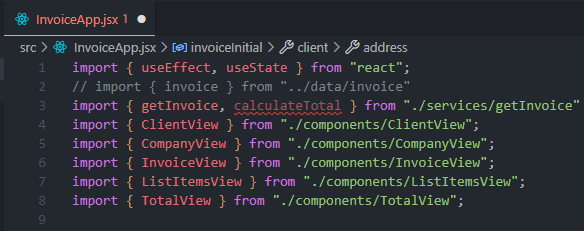
El **total** calculado se asigna a una variable llamada **total**, que luego se agrega al objeto de la factura utilizando el operador Spread (**...invoice**).



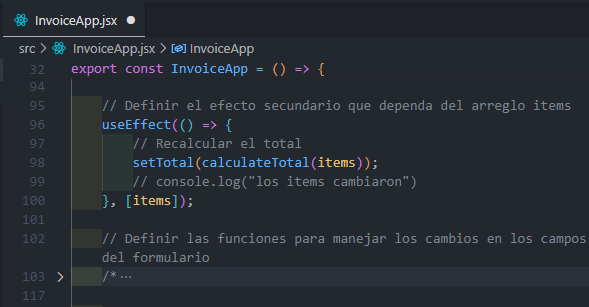
Con estas modificaciones, la lógica para calcular el **total** se ha encapsulado en la función **calculateTotal**, lo que hace que sea más fácil de entender y mantener. La función **getInvoice** ahora se centra en la obtención de la factura en sí, delegando el cálculo del total a otra función.

Paso 101: Importar la función en el componente InvoiceApp

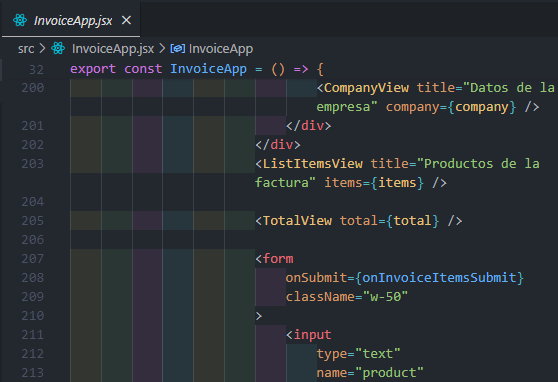
Primero, importa la función **calculateTotal** en el componente principal **InvoiceApp.jsx** desde el servicio **getInvoice**.

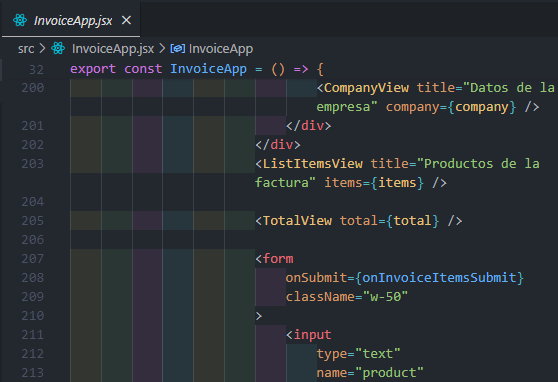


Luego, dentro de la función del **useEffect** del **total**, se calcula el total de los ítems utilizando **calculateTotal(items)**. Se pasa el arreglo de ítems (la misma dependencia) como argumento para que el **total** se recalcule cada vez que haya un cambio en los ítems, generando así un nuevo total:



Este total recalculado se utiliza más abajo en el renderizado del componente. Se pasa como propiedad al componente **TotalView**, donde se actualiza de forma automática. La prop **total** del componente **TotalView** es igual al estado de **total**.

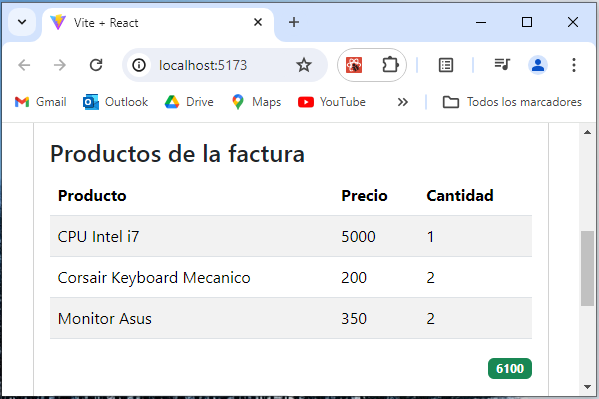




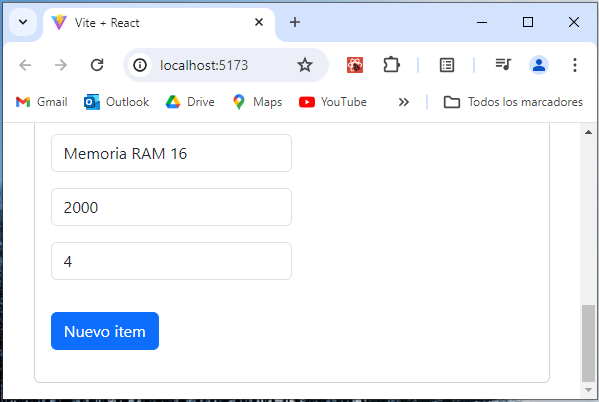
Paso 102: Realizar una prueba en el navegador

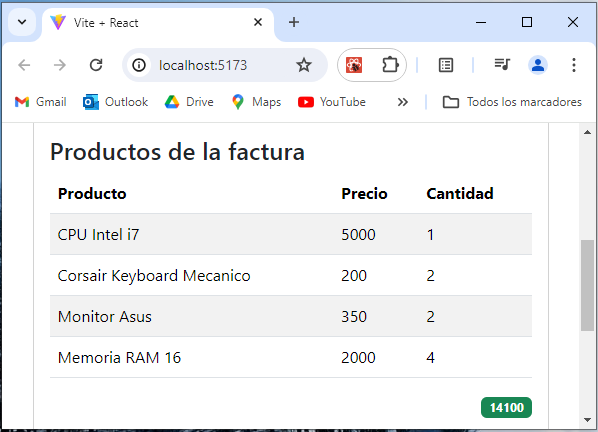
Para verificar que todo funcione correctamente, realiza una prueba en el navegador. Inicialmente el total calculado es **6100**.

**Nota:** Ten en cuenta que, al haber quitado los comentarios de las líneas de código de **React.StrictMode**, los mensajes en la consola se mostrarán dos veces.



Procede a añadir un nuevo producto desde el formulario.





Al añadir el nuevo producto, observa que el total se actualiza automáticamente. El cálculo manual del nuevo total sería:

* **Total inicial:** 6100
* **Nuevo producto:** 2000 \* 4 = 8000
* **Nuevo total:** 6100 + 8000 = 14100

Puedes seguir añadiendo otros productos y el **total** se va a volver a recalcular; esto es posible gracias al manejo de hooks como **useEffect**, la función **setTotal** y la función **calculateTotal**.

La estructura del código es muy desacoplada y modular. Recuerda lo siguiente:

* Cuando la página se carga por primera vez, los ítems se cargan mediante un **useEffect**.
* Inicialmente, el estado de **items** es un arreglo vacío. Al cargar los ítems, se obtiene la factura usando el método **getInvoice**, que devuelve un objeto con los detalles de la factura.
* El API o la fuente de datos asigna la factura y los ítems. Esto provoca un cambio en el estado de **items** y dispara el evento (**useEffect**) que calcula el **total** de la factura inicial (cuando se carga la página web).
* Cada vez que los ítems cambian, ya sea al añadir un nuevo ítem o al modificar uno existente, el **useEffect** que depende del arreglo de **items** se vuelve a ejecutar y recalcula el **total**.