Dividiendo en sub-componentes

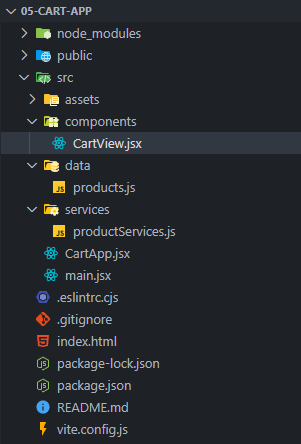
En esta sección, el componente principal se dividirá en subcomponentes. De tal manera que se tenga un subcomponente para el catálogo y otro para el carrito de compras, el cual se subdividirá en componentes individuales para cada fila de un ítem agregado al carrito.

Pasos 21 al 26

Sigue estos pasos

Paso 21: Definir el componente CartView

Crea una carpeta llamada **components** dentro de la carpeta **src**. Dentro de la carpeta **components**, crea un archivo llamado **CartView.jsx**.

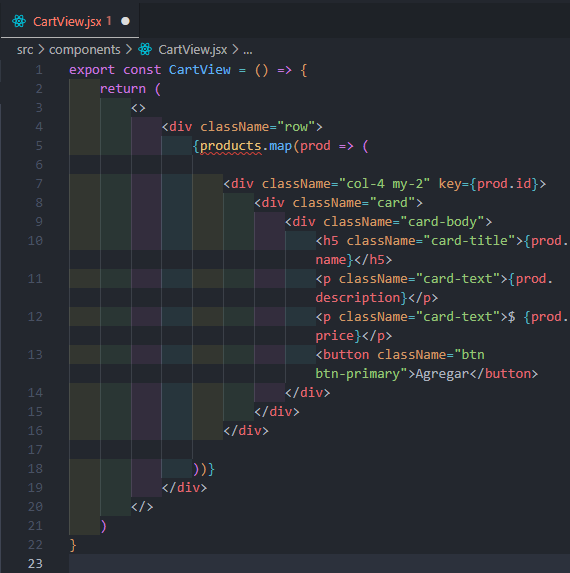


En el archivo **CartView.jsx**, define una función de flecha exportable llamada **CartView** que retorne un fragmento vacío inicialmente.



En el archivo **CartApp.jsx**, corta todo el contenido del **<div>** con la clase **row** y pégalo dentro del subcomponente **CartView** dentro del **return**.



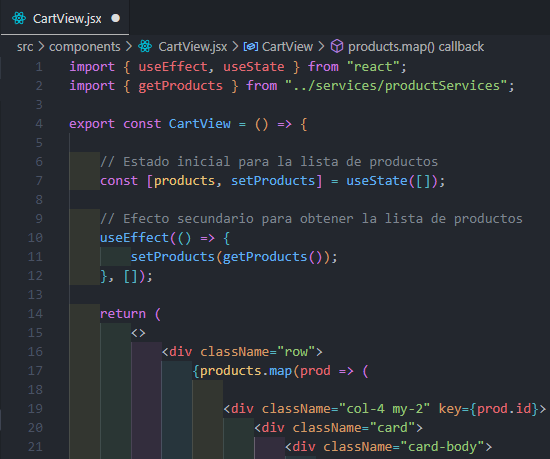


Paso 22: Importar hooks y la función getProducts

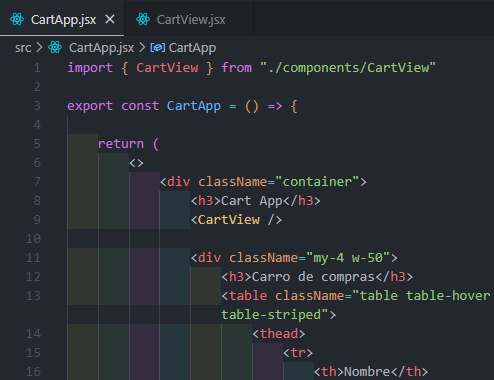
Procede a trasladar (copiar y pegar) el estado **products** y el efecto secundario que contiene el llamado a la función **getProducts** al subcomponente **CartView** para que maneje los productos de manera independiente.

Recuerda importar **useEffect**, **useState** y **getProducts** en el archivo **CartView.jsx**.



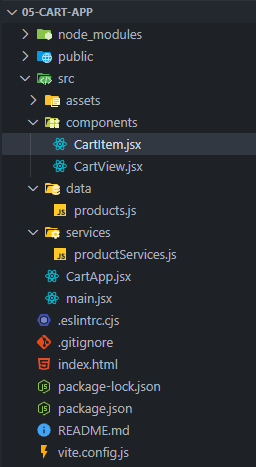


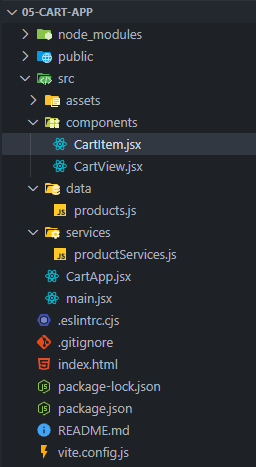
Quita las importaciones (**imports**) de **useEffect**, **useState** y **getProducts** del archivo **CartApp.jsx** y renderiza el componente **CartView** justo debajo del título definido con **<h3>** sin pasarle props, ya que **CartView** ahora maneja su propio estado.



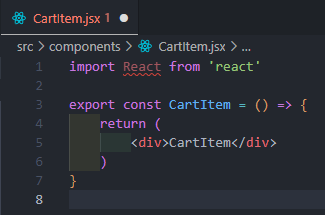
Paso 23: Definir el subcomponente CartItem, para los items,

Dentro de la carpeta **components**, crea el archivo **CartItem.jsx**.

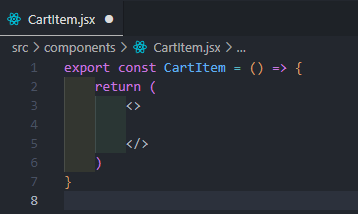




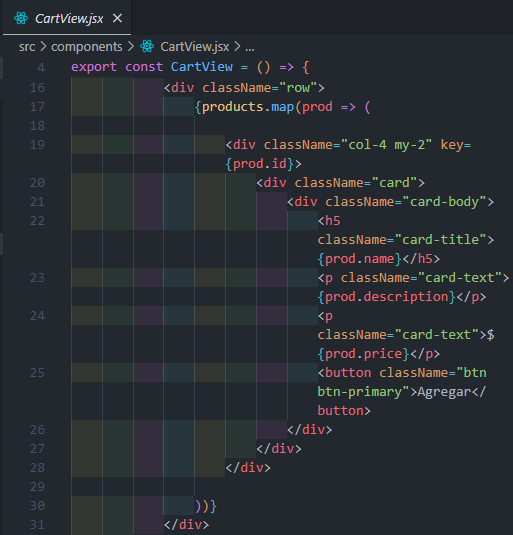
Puedes utilizar el atajo **rafc** para generar rápidamente la estructura básica del componente. Al escribir **"rafc"** en la primera línea de código del nuevo componente y pulsar la tecla Enter para generarlo, se obtiene el siguiente resultado:

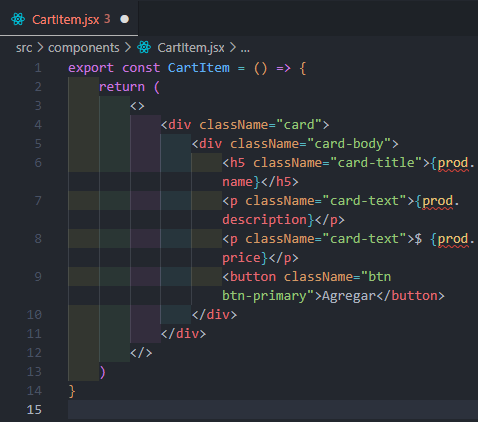


Asimismo, elimina la primera línea que importa React ya que no es necesaria en los proyectos de React 17+.



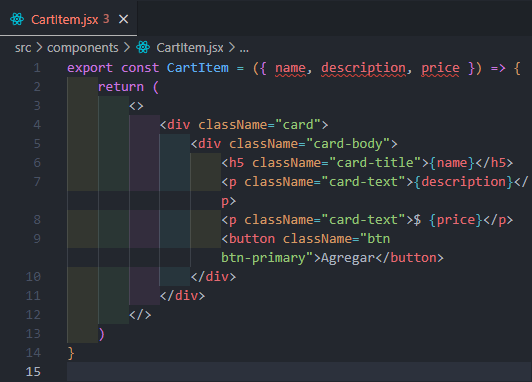
Corta el código del **<div>** con la clase **card** del componente **CartView** y pégalo dentro del componente **CartItem**. Se tiene en cuenta que no se esta incluyendo el atributo **key** para que el componente sea reutilizable.





Paso 24: Definir propiedades y renderizar el componente CartItem

En el componente **CartItem**, desestructura las propiedades **name**, **description** y **price**. Utiliza estas propiedades en lugar de las referencias a **prod**.



En el componente **CartView**, pasa las propiedades **name**, **description** y **price** al renderizar el componente **CartItem** dentro del bucle de mapeo. Ten en cuenta que **CartItem** se renderiza dentro del **<div>** con el atributo **key**.

Recuerda que el objeto **prod** representa cada producto del arreglo de productos obtenidos con el método **map** en el estado de **products**, desde la función **getProducts**.

Para que quede más ordenado se puede dar un salto de línea por cada atributo definido en la parte HTML del componente.





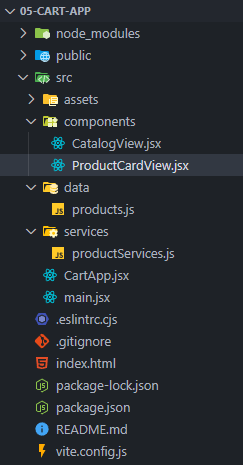
Al pasar las propiedades **name**, **description** y **price** al componente **CartItem**, se utiliza la reutilización de componentes para renderizar cada elemento del carrito de compras de manera eficiente y organizada.

Recuerda que **CartView** tiene un componente principal que es el elemento **<div>** con la clase **"row"**, el cual itera a través de los productos y contiene un contenedor común para cada ítem, renderizando cada tarjeta de producto individualmente.

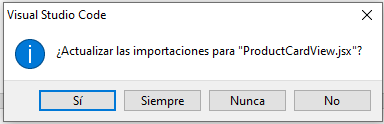
En el navegador todo funciona igual que antes, pero subdividido en componentes.

Paso 25: Renombrar los componentes

El objetivo de este paso es renombrar los componentes para reflejar mejor su propósito. En lugar de **"CartView"** y **"CartItem"**, se utilizará nombres más descriptivos: **"CatalogView"** y **"ProductCardView"**. Así, el catálogo será más claro y los nombres de los componentes representarán con mayor precisión sus funciones.



En el caso de que aparezca el cuadro de dialogo para actualizar las importaciones, haz clic en **"Sí"**.

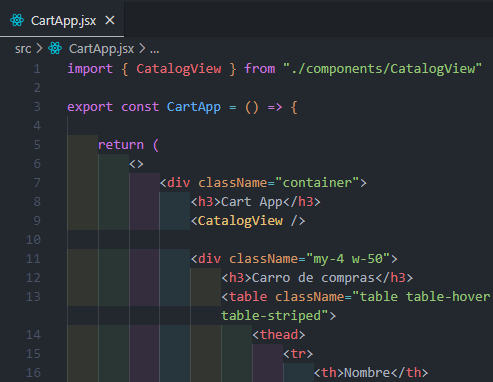


Realiza los cambios de nombre en el código fuente de los componentes de forma manual o pulsando la tecla **CTRL + H** en Visual Studio Code para buscar y reemplazar, principalmente se reemplaza las importaciones y los nombres de los componentes al momento de renderizar.

El componente **CatalogView** debe quedar de la siguiente manera:



También se realiza los reemplazos en el componente principal **CartApp**:



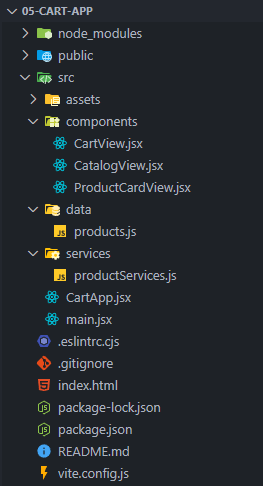
Finalmente, en el componente **ProductCardView**:



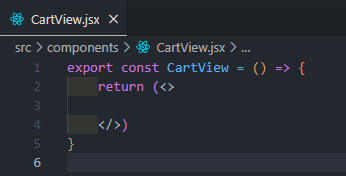
Con estos cambios, los componentes ahora tienen nombres que reflejan mejor su propósito. En el navegador, la aplicación debería funcionar como antes, mostrando el catálogo y las tarjetas de productos adecuadamente.

Paso 26: Definir el componente CartView

Dentro de la carpeta **components**, crea el archivo **CartView.jsx**.



Usa el atajo **rafc** (solamente escríbelo y pulsa **Enter**) para generar la estructura básica del componente, asimismo elimina la importación de React y el <div> de retorno reemplázalo por un fragmento vacío.

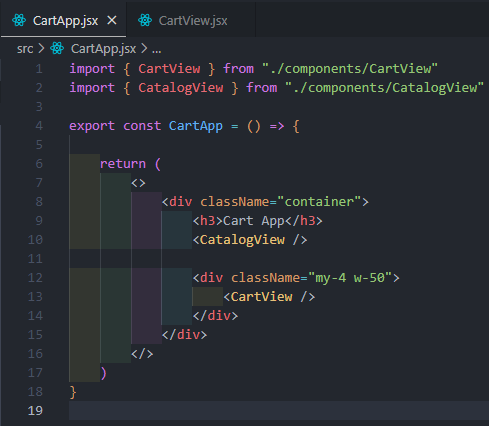


Corta el código del carrito de compras en **CartApp** (el título y la tabla) y pégalo en **CartView**.





Importa **CartView** en el componente principal **CartApp**, luego renderiza el componente **CartView** en el lugar apropiado dentro de **CartApp**.



Con estos cambios, **CartApp** ahora renderiza **CatalogView** y **CartView** como subcomponentes, haciendo que el código sea más modular y fácil de mantener. En el navegador, la aplicación funciona igual que antes, mostrando el catálogo de productos y el carrito de compras de manera organizada.