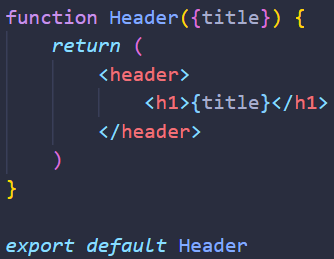
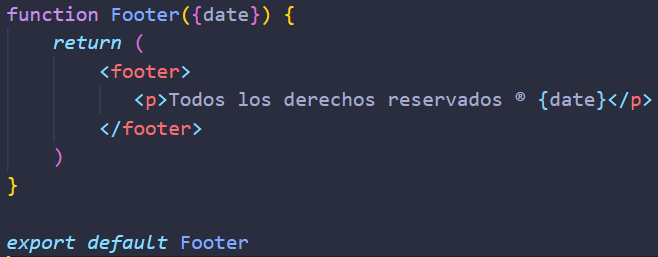
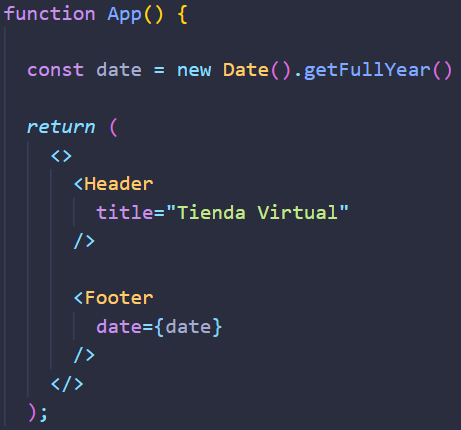
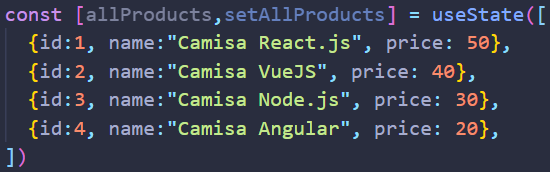
**Proyecto: Lista de productos y carrito de compras**

**Resultado final** sin productos y con productos en el carrito

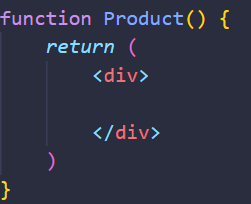
**1**. **Identificamos** los elementos que deben ser **componentes**, en este caso detectamos un **Header**, la lista de **productos**, el **carrito** de compras y el **footer**.

**2**. **Creamos** y exportamos el **componente Header y Footer**. **Desestructuramos las props** que ya sabemos que le vamos a pasar más adelante.

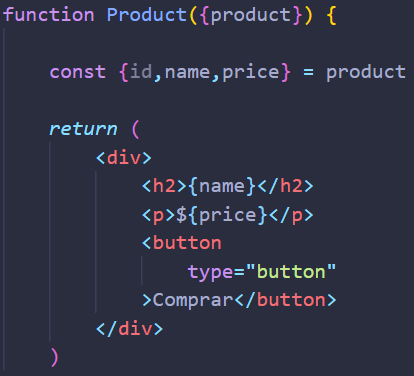
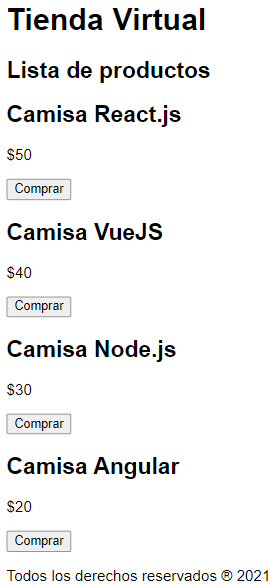
**3**. **Dentro de App.js** importamos estos dos **componentes** y los colocamos **entre dos fragments**. **Creamos** una **constante** para **requerir la fecha** y se la pasamos al Footer como propiedad, además de pasarle la propiedad “title” al Header.

**4.** Ahora vamos a crear la lista de productos. **Dentro de App.js** requerimos el hook “useState” y definimos el estado inicial con un array de objetos literales que contienen información de los productos.

**5.** **Creamos** el **componente** “**Product**”, el cual por el momento lo dejamos vacío. Lo **importamos dentro de App.js** y **hacemos un map del array** de objetos que **contiene nuestra información (allProducts)** y en **cada iteración** imprimimos el **componente “Product”** pasándole como propiedades: **key** (obligatoria en todos los Map) y **product** (el producto iterado)

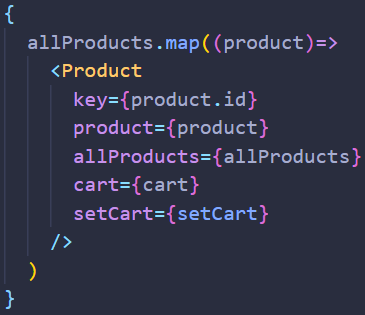
 

**6.** Vamos nuevamente al **componente** “**Product**” y **desestructuramos** la propiedad “**product**” guardando su nombre, precio e ID en una variable. Luego dentro del JSX **imprimimos esos datos y agregamos un button**.

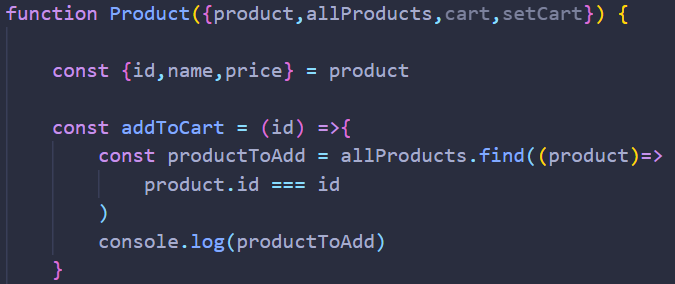


**7.** Ahora vamos a crear la funcionalidad de **agregar productos al carrito**, para esto primero tenemos que crear un estado para el mismo. Dentro de **App.js** **creamos** “**cart**” y “**setCart**” y le damos un valor **array vacío** como valor inicial.



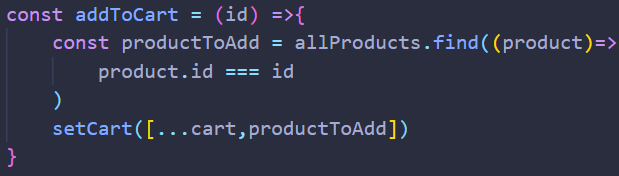
**8.** **Al map de productos** que hicimos anteriormente le **pasamos 3 nuevas propiedades**: “**allProducts**” (son todos los productos, estos nos va a servir para filtrar dentro del componente “Product”), “**cart**” y “**setCart**”.

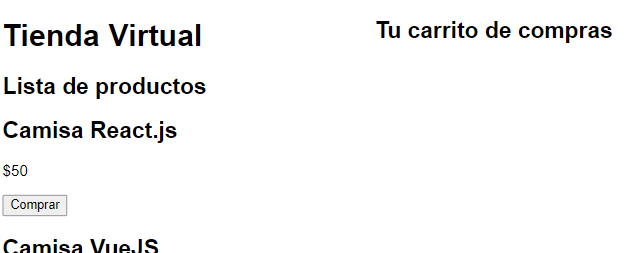
**9.** Dentro del componente “**Product**” **desestructuramos las nuevas props** y **creamos** una **función** que se va a encargar de **identificar** el **producto** **que estamos comprando (addToCart)**. Recibe como **parámetro un id**, y a partir del mismo **busca** en la lista de todos los **productos** **aquel** que **coincida** con el mismo, luego lo imprimimos por consola para probar que funcione.

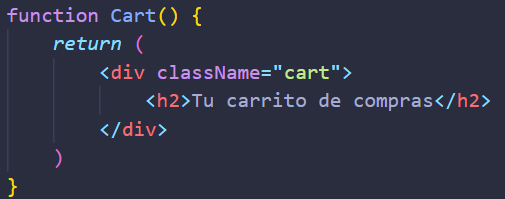


**10**. **Agregamos** **esta función en el <button>** para que se ejecute cuando se hace click sobre el mismo.

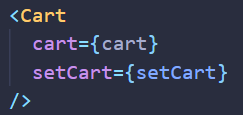


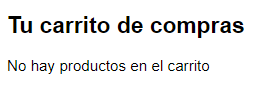
**11.** Una vez que vemos que funciona, **debemos modificar el estado del carrito con el producto que estamos comprando**. Para esto reemplazamos el console.log por la variable de escritura del carrito “**setCart**” y le **pasamos un array con 2 datos**: el primero es el **contenido actual del carrito descomprimido(…)** **y** el segundo es **el producto que filtramos anteriormente**. De esta forma cada producto que agreguemos se sumara al anterior en lugar de reemplazarlo.

**12.** Para hacer visible estos productos agregados dentro del estado del carrito debemos **crear** un nuevo **componente** llamado “**Cart**”. Le agregamos una clase al <div> para poder posicionar el contenido del lado derecho del documento. **Finalmente lo incluimos dentro del archivo App.js para que se renderize.**



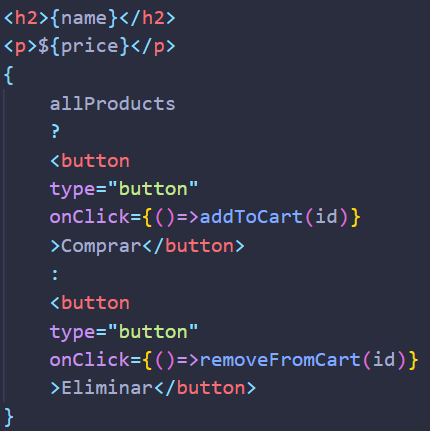
**13.** Dentro de **App.js** le **pasamos la propiedad** “**cart**” y “**setCart**” al componente Cart. Dentro del mismo **desestructuramos la prop**, y hacemos un **if ternario**. Si la propiedad **cart es igual a 0** significa que **no hay productos** por lo que **renderizamos un mensaje** informándolo, en caso que se **si haya productos** vamos a hacer un m**ap del carrito imprimiendo el componente “Product” en cada iteración** (lo estamos reutilizando ya que va bien para el carrito) y le **pasamos las propiedades**: **key**, **product**, **cart** y **setCart**.







**14.** **Creando la opción de eliminar**: para esto debemos distinguir de qué lado nos encontramos (si en el listado de productos o en el carrito). Viendo las props que se le pasan al componente “Product” vemos que “allProducts” únicamente se lo pasamos del lado del listado por dicho motivo vamos a **hacer un ternario en los <button>** y dar la **opción de eliminar en caso de que estemos en el carrito** (y le agregamos una función para efectivamente eliminarlo)



**15.** **Creamos** la **función** para **eliminar** el **producto** **del estado del carrito**. Su lógica es la siguiente: **filtra a todos los productos cuyos IDs no sean iguales al del producto a eliminar**, filter crea un nuevo array sin ese producto. Luego **updateamos el estado del carrito con esa nueva lista.**

