Interacción entre los componentes de la App:  
  
A continuación, se describe cómo interactúan estos componentes y el servidor:  
  
  
Main.jsx:

* Este archivo es el punto de entrada de la aplicación. Importa los módulos necesarios, incluyendo React, ReactDOM y el componente principal App.
* Renderiza el componente App dentro del elemento con el id root en el DOM.
* Importa los estilos globales y de Bootstrap para proporcionar un diseño coherente.

App.jsx:

* App es el componente principal de la aplicación que organiza la estructura y las rutas de la aplicación.
* Utiliza BrowserRouter de react-router-dom para definir las rutas de la aplicación.
* Incluye Navbar para la navegación y Routes para definir las rutas a ItemListContainer y ItemDetailContainer.
* Envuelve toda la aplicación en CartProvider para proporcionar el contexto del carrito a todos los componentes hijos.

CartContext.jsx:

* Define el contexto del carrito utilizando React.createContext.
* CartProvider es el componente proveedor que envuelve la aplicación y proporciona el estado del carrito y las funciones para agregar y quitar productos.
* useEffect se utiliza para cargar los productos desde un archivo JSON simulado cuando el componente se monta.
* Proporciona funciones addToCart y removeFromCart para gestionar la adición y eliminación de productos en el carrito.
* Calcula el total de productos en el carrito y el monto total del carrito.

Navbar.jsx:

* Componente de navegación que incluye enlaces a las diferentes categorías de productos.
* Utiliza CartWidget para mostrar el número total de productos en el carrito y el monto total del carrito.
* Utiliza useContext para acceder al contexto del carrito y mostrar la información relevante.

CartWidget.jsx:

* Componente que muestra la cantidad total de productos en el carrito.
* Utiliza useContext para acceder al contexto del carrito y mostrar la cantidad total de productos.

ItemListContainer.jsx:

* Componente que muestra la lista de productos de una categoría específica.
* Utiliza useParams de react-router-dom para obtener el parámetro de la categoría de la URL.
* Hace una solicitud a la API para obtener los productos de la categoría seleccionada y los muestra utilizando tarjetas de Bootstrap.
* Utiliza ItemCount para permitir a los usuarios seleccionar la cantidad de un producto y agregarlo al carrito.

ItemDetailContainer.jsx:

* Componente que muestra los detalles de un producto específico.
* Utiliza useParams de react-router-dom para obtener el parámetro del ID del producto de la URL.
* Hace una solicitud a la API para obtener los detalles del producto y los muestra.

ItemCount.jsx:

* Componente que permite a los usuarios seleccionar la cantidad de un producto y agregarlo al carrito.
* Utiliza useState para manejar el estado de la cantidad seleccionada.
* Utiliza useContext para acceder al contexto del carrito y agregar productos al carrito.

ProductList.jsx:

* Componente que muestra una lista de productos con botones para agregar y quitar productos del carrito.
* Utiliza useContext para acceder al contexto del carrito y manejar la adición y eliminación de productos.

## Configuración del servidor

server.js:

* Configura un servidor Express que sirve como API para la aplicación.
* Define endpoints para obtener todos los productos, obtener un producto por ID y obtener productos por categoría.
* Utiliza “fs” para leer los datos del archivo products.json.
* Configura CORS y parseo de JSON para manejar solicitudes adecuadamente.

## Configuración de Vite

vite.config.js:

* Configura Vite, el empaquetador de la aplicación.
* Incluye el plugin de React para Vite.
* Establece la base URL de la aplicación para /PreEntrega1Frutos/.

## Interacción entre el frontend y el backend

El frontend (React) hace solicitudes HTTP al backend (Express) para obtener los datos de los productos.

Los componentes del frontend utilizan los datos obtenidos para renderizar la información de los productos y manejar la lógica del carrito.

El contexto del carrito (CartContext.jsx) centraliza el estado del carrito y proporciona funciones para modificar dicho estado.

Los componentes de la aplicación se comunican entre sí a través de props y el contexto del carrito para mantener el estado sincronizado y ofrecer una experiencia de usuario coherente.

Con esta estructura, la aplicación proporciona una navegación fluida y una gestión eficiente del carrito de compras, permitiendo a los usuarios explorar productos, agregar productos al carrito y ver detalles específicos de cada producto.