Guía Técnica del Proyecto

Esta guía técnica describe la interacción entre los componentes de la aplicación, la configuración del servidor, la configuración de Vite, y la interacción entre el frontend y el backend. Los archivos principales del proyecto se detallan a continuación:

Interacción entre los Componentes de la App

Main.jsx

- **Punto de entrada** de la aplicación. Importa módulos clave como React, ReactDOM, y el componente principal `App`.
- **Renderiza** el componente `App` dentro del elemento con el id `root` en el DOM.
- Importa estilos globales y de Bootstrap para mantener una **consistencia en el diseño**.

App.jsx

- `App` es el **componente principal** que organiza la estructura y define 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 rutas hacia `ItemListContainer` y `ItemDetailContainer`.
- **Envuelve la aplicación** en `CartProvider` para proporcionar el contexto del carrito a todos los componentes hijos.
- Incluye `ToastContainer` para mostrar notificaciones en la interfaz.

CartContext.jsx

- Define el **contexto del carrito** utilizando `React.createContext`.
- `CartProvider` es el componente proveedor que envuelve la aplicación, proporcionando el estado del carrito y las funciones para agregar y quitar productos.
- Utiliza `useEffect` para **cargar productos desde Firebase Firestore** 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**.

Navbar.jsx

- Componente de **navegación** que incluye enlaces a diferentes categorías de productos.

- Utiliza `CartWidget` para mostrar la **cantidad total de productos** en el carrito y el monto total.
- Utiliza `useContext` para acceder al contexto del carrito y mostrar la información relevante.
- Implementa un **carrusel de imágenes** que cambia según la sección (inicio, hombre, mujer) en la que se encuentra el usuario.
- Los filtros de subcategorías se despliegan cuando el usuario pasa el puntero del mouse sobre "Hombre" o "Mujer".

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 Firebase Firestore** 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 Firebase Firestore** 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.
- Muestra una notificación `Toastify` cuando un producto es agregado 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 auxiliar para la aplicación, aunque no se usa en la versión actual con Firebase.
- Podría definirse como una API de respaldo para obtener todos los productos, obtener un producto por ID, y obtener productos por categoría.
- **No es necesario** cuando Firebase Firestore es la principal fuente de datos.

Configuración de Firebase

firebaseConfig.js

- Configura **Firebase** como la base de datos principal de la aplicación.
- Define la configuración de Firebase, inicializa la app y los servicios como **Firestore** y **Analytics**.
- Proporciona acceso a la base de datos para todos los componentes que necesitan interactuar con los datos.

Interacción con Firebase

- **Firestore** es utilizado para almacenar y recuperar datos de productos.
- Los componentes como `ItemListContainer` y `ItemDetailContainer` realizan consultas directamente a Firestore para obtener y mostrar los productos.

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/`, asegurando que todas las rutas y recursos estén correctamente alineados con la base de la URL del proyecto.

Interacción entre el Frontend y Firebase

- El frontend, desarrollado con **React**, hace **consultas a Firebase Firestore** para obtener los datos de productos.

- Los componentes del frontend utilizan estos datos 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**, manteniendo el estado sincronizado y proporcionando una experiencia de usuario fluida y coherente.