UDB VIRTUAL

UNIVERSIDAD DON BOSCO

Asignatura

Diseño y Programación de Software Multiplataforma DPS941 G01T

Catedrático

Ing. Alexander Alberto Sigüenza Campos

Actividad

Trabajo de investigación: proyecto de carrito de compras en JS con facturación

Fecha de entrega

31/08/2025

Estudiante

Cruz Mejía, Josué Esaú CM221973

Índice

1 Introducción

El presente documento describe el proyecto para la tienda ByteBazar, una tienda de artículos informáticos, una aplicación web que simula un carrito de compras, donde se ha usado HTML, CSS y JavaScript, de manera modular. El propósito es presentar un sistema funcional que demuestre buenas prácticas de desarrollo front-end, tales como la buena manipulación del DOM (Modelo de Objetos del Documento), manejo de eventos, validación de entradas, persistencia local (implementación avanzada, de localstorage), control de inventario por producto y separación del comportamiento en módulos reutilizables (catálogo, inventario, carrito, UI, validadores y proceso de checkout). Además, se incluye la funcionalidad de crear una factura, se genera una factura visual al momento de confirmar la compra.

El presente documento explicará la estructura del código y el flujo de ejecución de la aplicación (desde la carga inicial hasta el checkout y la generación de la factura. Dicho documento nace, para que se pueda tener una idea clara de la solución propuesta para ser implementada al momento de trabajar con un carrito de compras.

2 Objetivos

2.1 General

Describir el proceso de ejecución y uso de la aplicación (instalación, puesta en marcha y flujo de compra), de modo que se facilite su despliegue, prueba y evaluación funcional.

2.2 Específicos

- Detallar la estructura de archivos del proyecto, tales como index.html, y carpetas css, js, y con sus respectivos submódulos dentro de estas.
- Describir el flujo de ejecución de la aplicación, la carga del index.html, inicialización en el archivo main.js, carga de productos desde data/products.json o desde el localstorage.
- Mostrar el flujo de compra, al añadir productos al carrito, validaciones de cantidad, verificación de stock antes del pago, procesamiento de la orden, decremento de inventario y generación y visualización de la factura generada.
- Especificar cómo ejecutar y probar la aplicación localmente, indicando los pasos concretos con la herramienta que provee Visual Studio Code (Live Server).

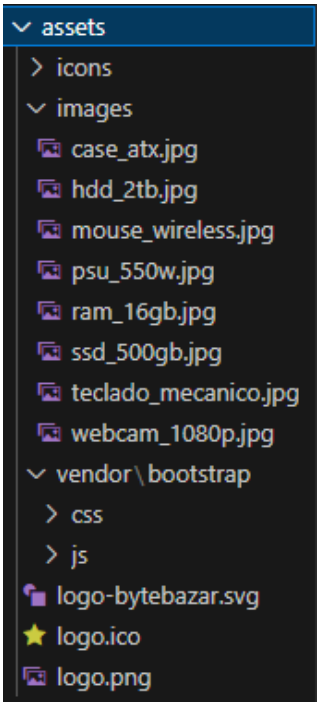
3 Cuerpo del trabajo

3.1 Estructura del código

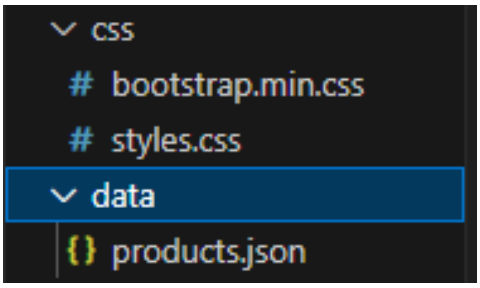
En esta sección se mostrará la estructura de carpetas y archivos que conforman el proyecto del carrito de compra en JavaScript.

El esquema del diagrama principal de carpetas y archivos del proyecto está conformado por: assets, css, data, js y el index

En la carpeta assets, se encuentran las carpetas para los iconos usados como botones en las funciones principales en el sistema, las imágenes mostradas en el catálogo de productos, la carpeta vendor con los archivos de las librerías para las funcionalidades de botstrap, y el documento usado para el logo del proyecto.

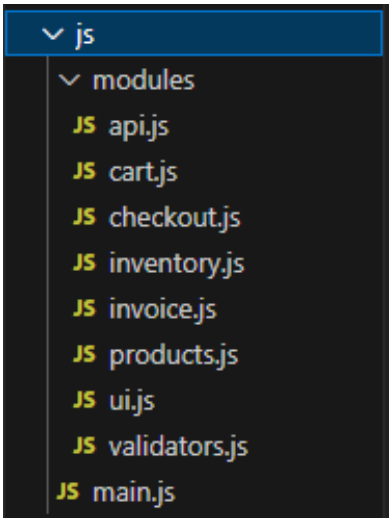


En la carpeta css, se encuentran los archivos propios y personalizados para el diseño del sitio, dentro del documento “styles css”.



En la carpeta data, se encuentra el archivo tipo JSON (products json) utilizado para detallar el catálogo de los productos, tanto aspectos como nombre y descripción, como el precio y la cantidad que se posee de ellos (stock).

En la carpeta js, se encuentra el archivo principal (main js) y una carpeta para los módulos, que contiene todos los demás archivos de Javascript que proporcionan las funcionalidades necesarias para el correcto funcionamiento del proyecto, esto es así, dado que, el proyecto se ha trabajado con modularidad, para tener las funciones separadas y que puedan ser reutilizables tantas veces como sea necesario.



El archivo api js se utiliza para inicializar persistencia. Si no existen productos en localstorage, los carga desde data/products.json mediante fetch (asíncrono) y los guarda. Crea arreglos vacíos para "orders" y "carrito" si no llegasen a existir

El archivo products js se usa para mantener un cache, en memoria, de los productos cargados desde el archivo anterior api.js que deposita todo en localstorage, provee funciones para leer, filtrar y actualizar productos en memoria. Este módulo asume que api.init() ya fue ejecutado y que api.getProducts() devolverá datos.

El archivo cart js encapsula la lógica del carrito de compras: agregar, actualizar cantidades, eliminar, calcular totales, la persistencia en el estado del carrito mediante el módulo de api (localstorage). También, emite eventos del DOM cuando el carrito cambia, para que la UI pueda reaccionar. Esta clase asume que hay otros archivos modulares disponibles: api (persistencia), inventory (stock), validators (validaciones). Emite un evento cart:updated con el detalle del carrito tras cada cambio hecho por el usuario, devuelve objetos para reportar fallos en operaciones.

El archivo inventory js provee utilidades para consultar y modificar el stock de productos, trabaja con lo que ha sido localizado en api.getProducts()/saveProducts (localstorage). Las funciones que posee se persisten con api.saveProducts.

El archivo validators js posee las utilidades para la validación de datos que son usados en la app, en el carrito, checkout. Las funciones devuelven valores booleanos, es corto, pero ayuda a controlar los datos que son ingresados por el usuario de la aplicación.

El archivo checkout js ayuda a revalidar stock antes de procesar la orden, decrementar stock y persistir la orden que ha sido recibida por parte del usuario de la aplicación.

El archivo invoice js es el módulo encargado de generar y mostrar la factura (invoice) en un modal de Bootstrap. Este recibe una orden con una forma o formato esperado:

id: 'ORD-...',

createdAt: 'ISO date',

customer: { name, email, address },

items: [{ id, qty, price, lineTotal}],

totals: { subtotal, tax, total }

En este archivo, se formatean valores monetarios relacionados al dinero que conlleva la compra, y genera el HTML de una factura o recibo a partir de un objeto de tipo order (por ejemplo, lo producido por o lo que trae la función cart.toOrder()). Se muestra la factura en el modal #invoiceModal que se encuentra en el archivo index.html usando Bootstrap. El módulo usa innerHTML para inyectar el HTML generado.

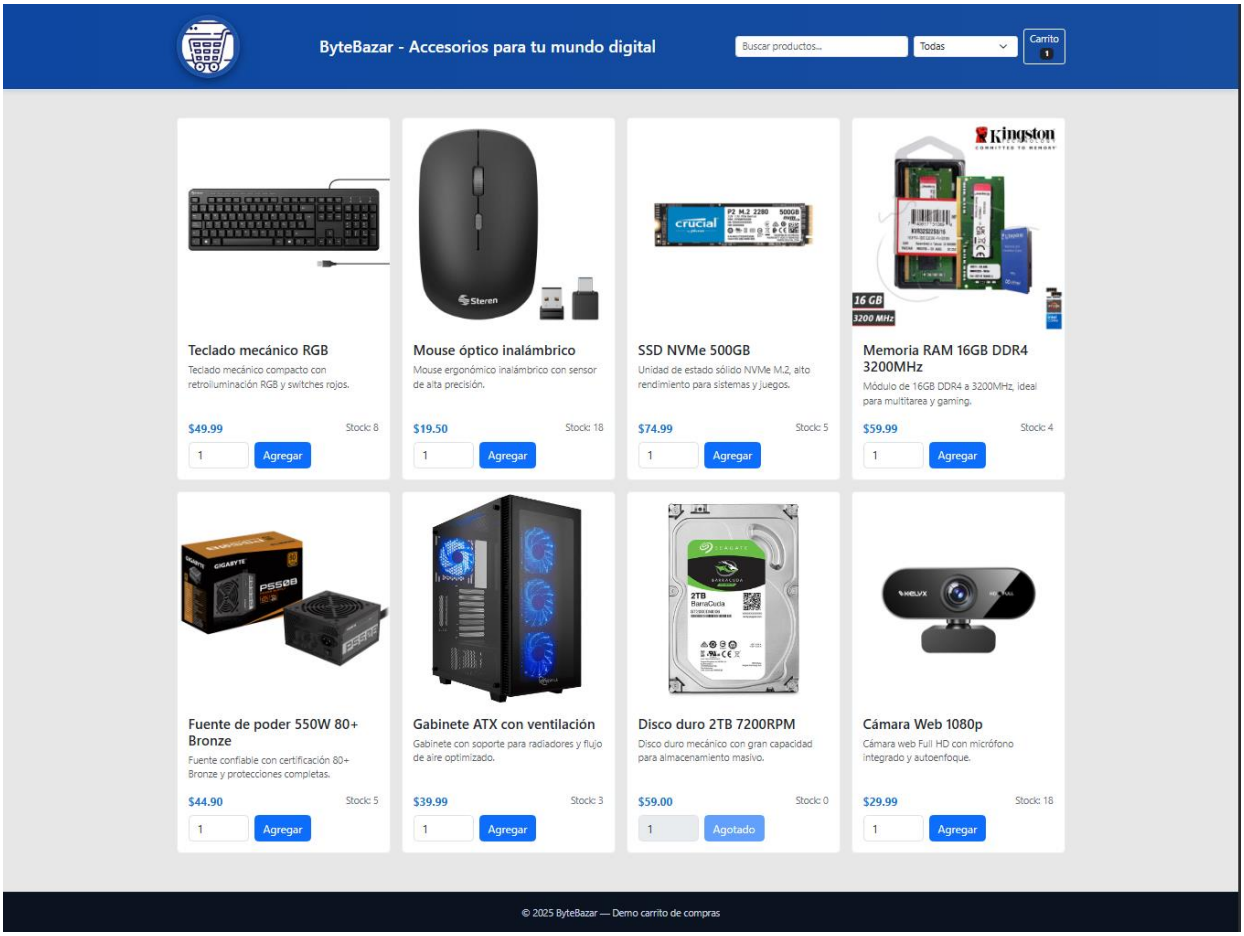
Para evitar problemas, cada campo dinámico pasa o es evaluado por la función escapeHtml() antes de insertarse, para validar el contenido. La función formatCurrency usa Intl.NumberFormat, el cual es configurado con 'en-US' y USD, para que sea acorde a la moneado que se utiliza en nuestro país.

El archivo ui js es el más extenso de todos los archivos modulares utilizados, ya que, es el que almacena todas las utilidades que permiten operar o que hacen que sea funcional la aplicación Principalmente, se encarga del render del DOM y el manejo de UI, usando Bootstrap Offcanvas, Modal y Toast. Recibe en init un objeto con dependencias, tales como productsModule, cart, inventory, api, validators, checkout, etc. Posee las inicializaciones de los componentes de Bootstrap que se utilizan en la aplicación. Aquí, se implementa la funcionalidad que permite que, al cerrar la factura se cierre el offcanvas que contiene al carrito de compras y el modal de checkout, además, permite que se carguen las disminuciones correspondientes a la compra, en el stock de productos.

También, se configuran las acciones que corresponden a los botones principales de la aplicación, y de los campos que son utilizados para buscar el producto por nombre y para el filtrado.

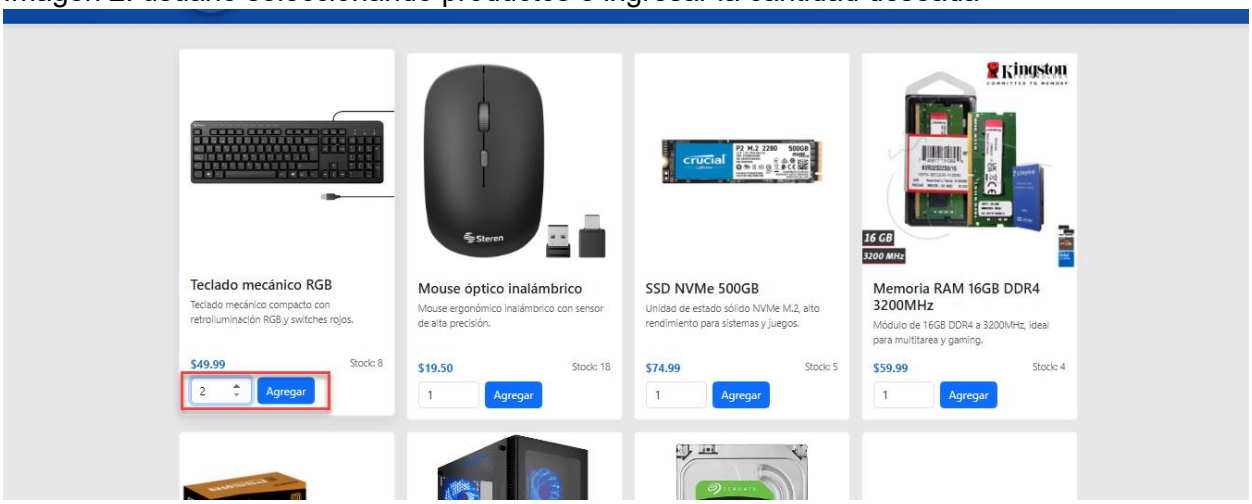
A continuación, se colocan los resultados obtenidos para los criterios de evaluación:

Imagen 1. mostrando el listado de productos



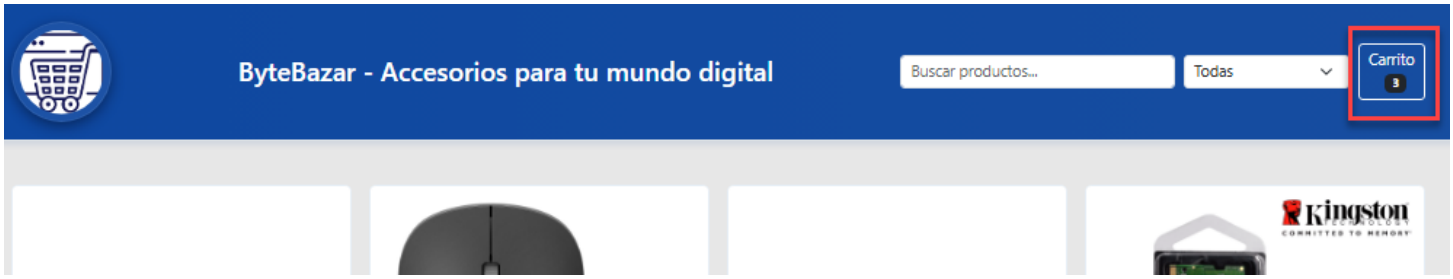
Fuente. Nota. Elaboración propia.

Imagen 2. usuario seleccionando productos e ingresar la cantidad deseada



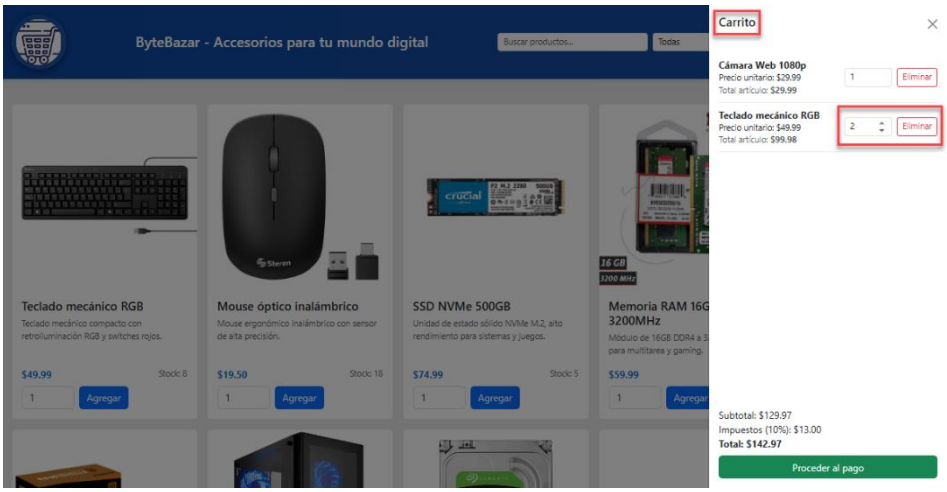
Fuente. Nota. Elaboración propia.

Imagen 3. agregando producto seleccionado al carrito de compras



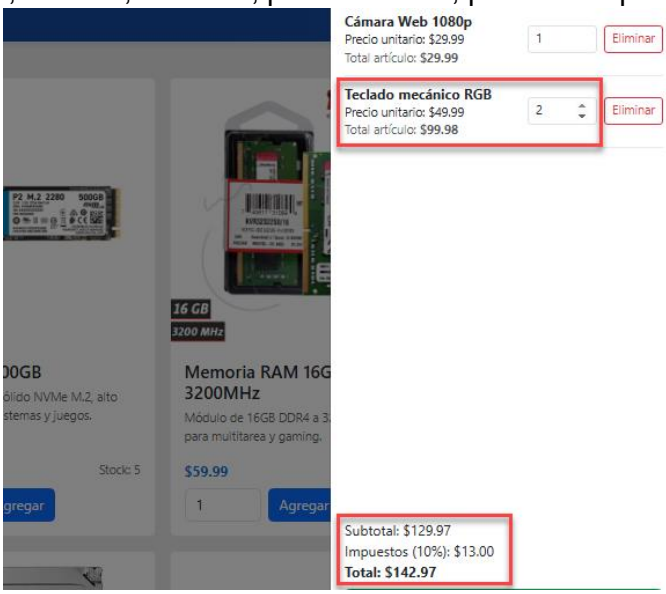
Fuente. Nota. Elaboración propia.

Imagen 4. permitir al usuario eliminar productos del carrito



Fuente. Nota. Elaboración propia.

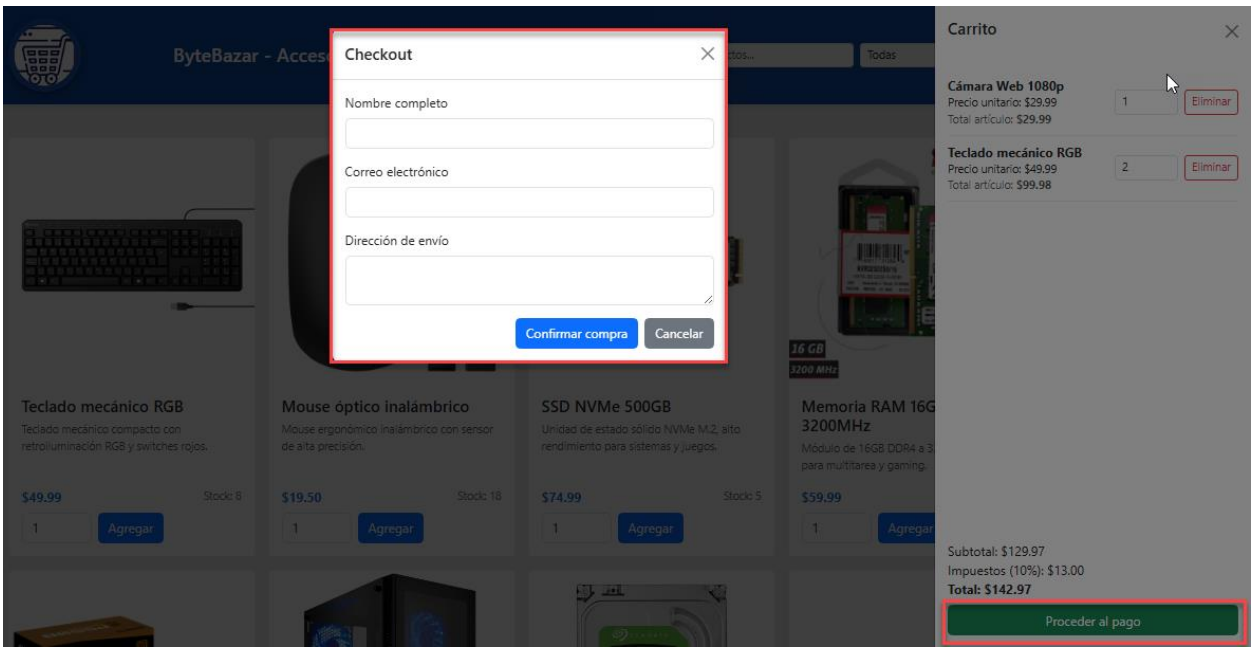
Imagen 5. en el carrito, nombre, cantidad, precio unitario, precio total por producto, total carrito



Fuente. Nota. Elaboración propia.

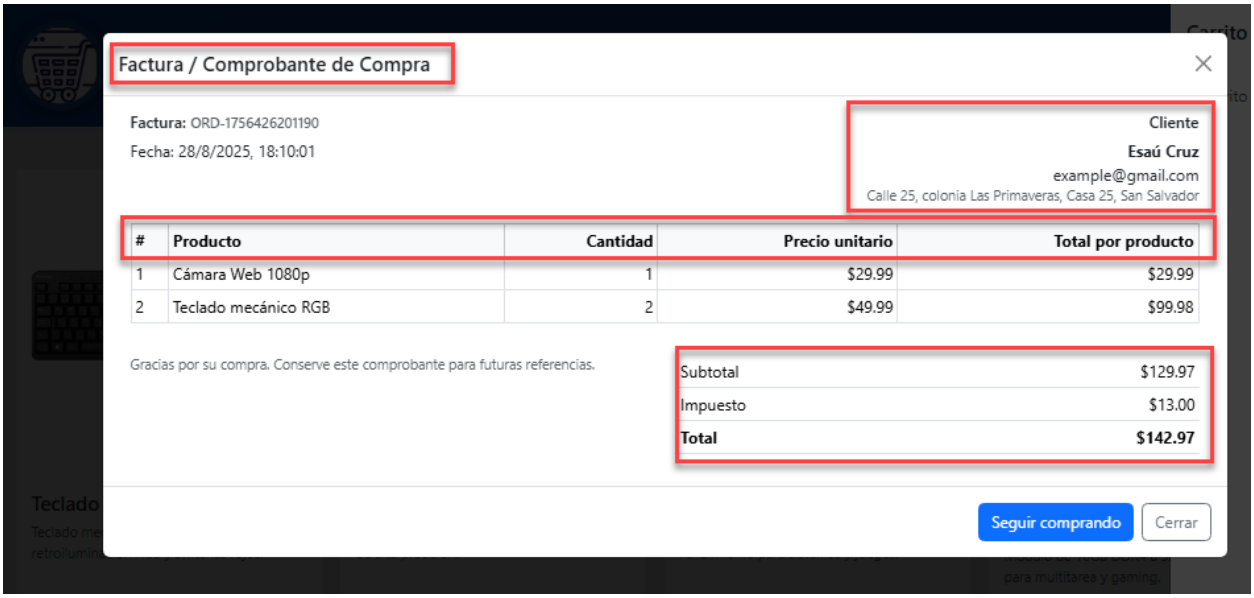
Al confirmar la compra (imagen 6), dando clic al botón “Proceder al pago”, se genera y muestra una factura (imagen 7) en pantalla, la cual incluye la lista de productos comprados, la cantidad, el precio unitario, el precio total por producto y el total general de la compra.

Imagen 6. modulo donde se capturan valores importantes del cliente para la factura



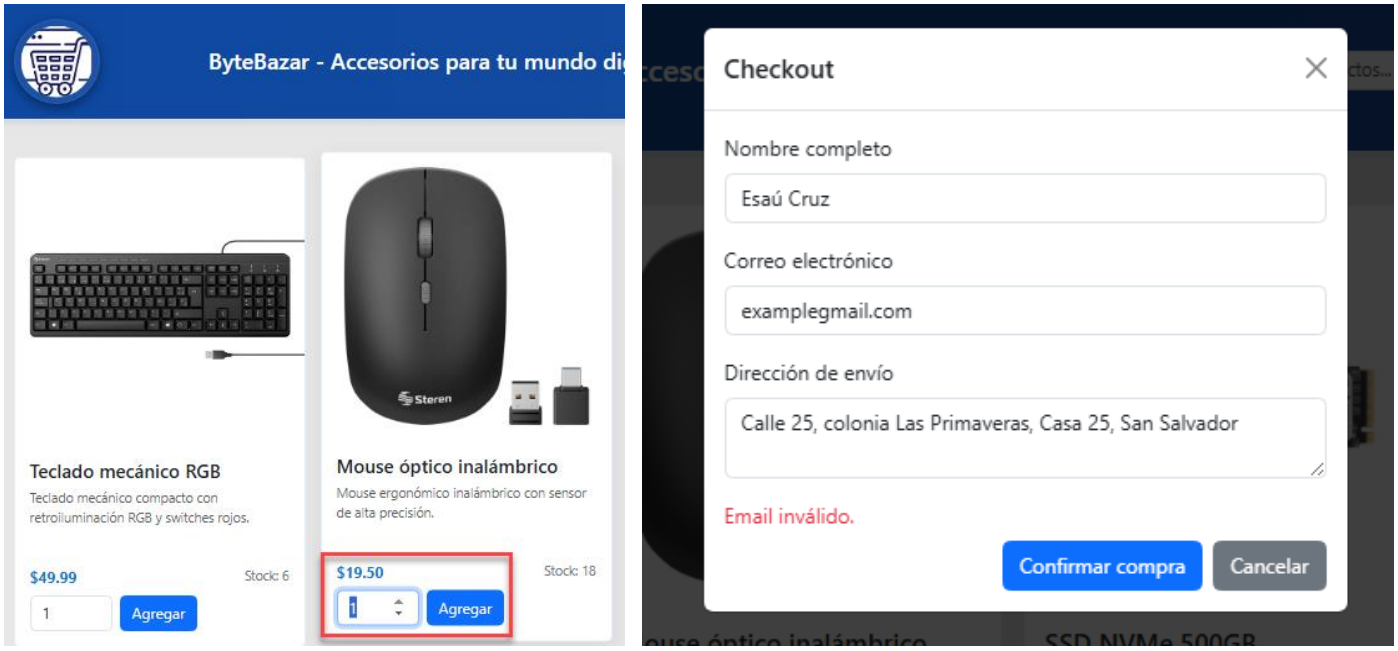
Fuente. Nota. Elaboración propia.

Imagen 7. factura donde se resume la compra del usuario, con montos y descuentos



Fuente. Nota. Elaboración propia.

Imagen 8. validaciones en la entrada del usuario para manejar los errores

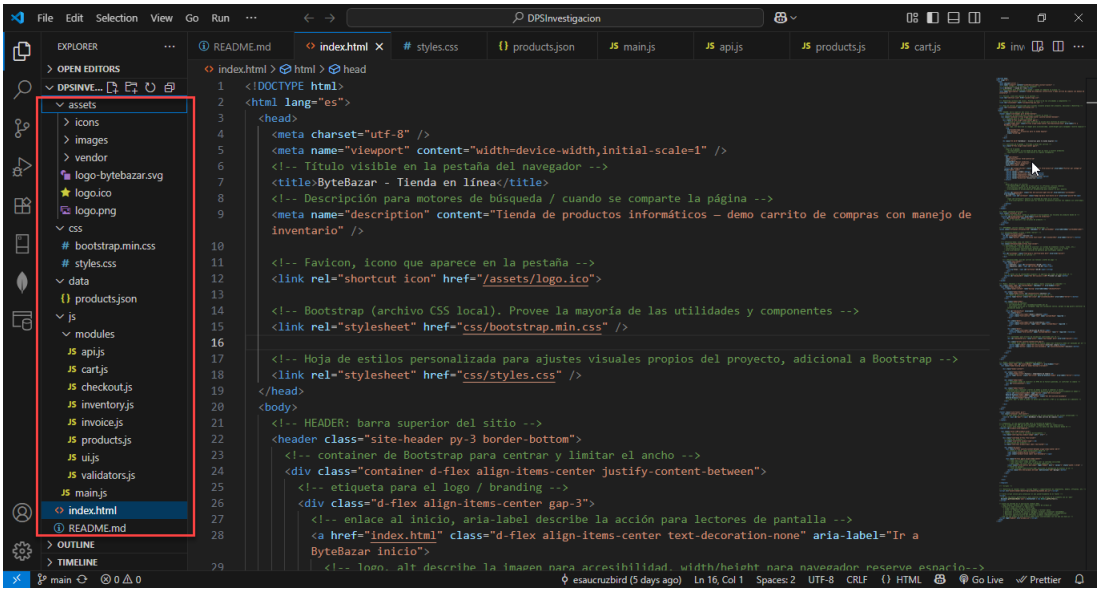


Fuente. Nota. Elaboración propia.

3.2 Cómo ejecutar la aplicación del proyecto del carrito de compras

El proyecto completo está alojado en un repositorio con la tecnología de Git, la plataforma especifica que se ha utilizado es GitHub. Esto, permitió un ágil desarrollo a nivel local, como para hacer una copia remota, directamente desde el editor de texto Visual Studio Code.

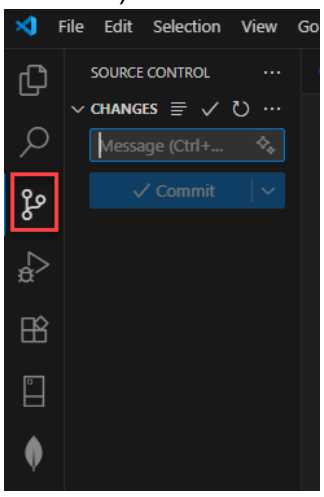
Imagen 9. proyecto en desarrollo local con Visual Studio Code



Fuente. Nota. Elaboración propia.

Enlace del repositorio remoto del proyecto: https://github.com/esaucruzbird/DPSInvestigacion

Imagen 10. extensión de GIT (source control) usada en Visual Studio Code

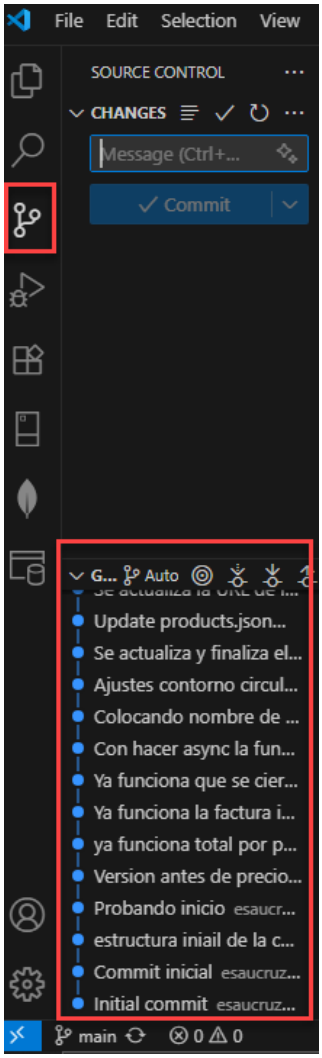


Fuente. Nota. Elaboración propia.

Al utilizar dicha extensión permitió tener un pleno control de los cambios hechos en el repositorio local y remoto, mostrando la descripción de los commits hechos con anterioridad.

Usar esta extensión permitió inicializar el proyecto localmente con Git, y poder subir los cambios fácilmente, a través de commits a GitHub.

Imagen 11. extensión en Visual Studio Code donde se observa el historial de commits

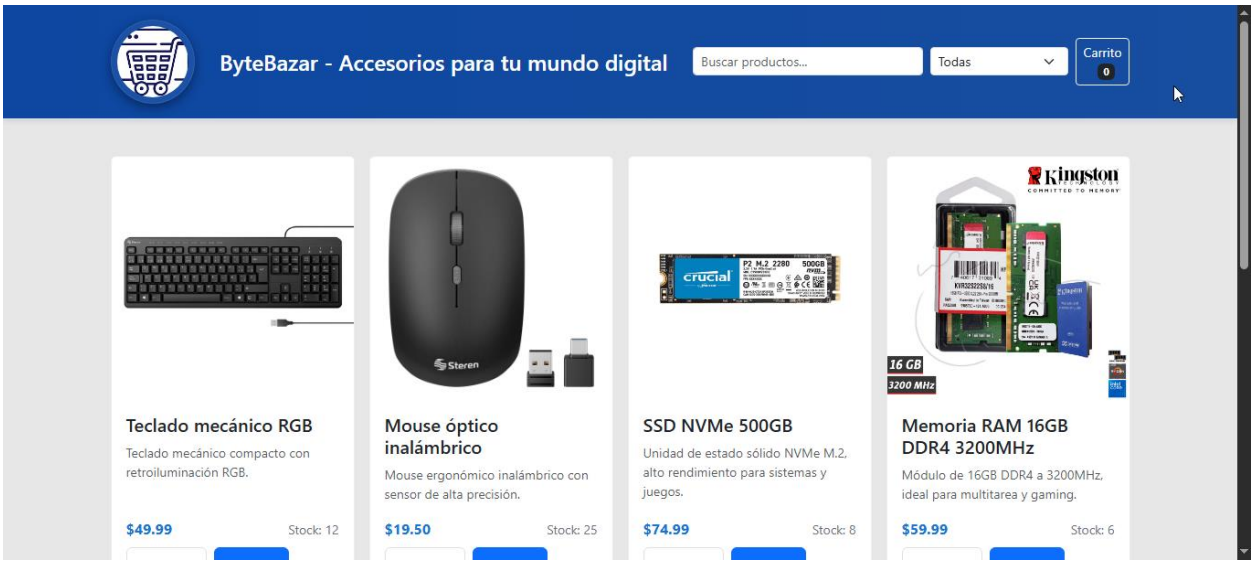


Fuente. Nota. Elaboración propia.

Utilizar dicho repositorio remoto tiene muchas ventajas entre las principales, permite verificar los resultados del código, a medida se van subiendo los cambios, esto, a través del uso de la tecnología de GitHub Pages.

Enlace de GitHub Pages del proyecto: https://esaucruzbird.github.io/DPSInvestigacion/

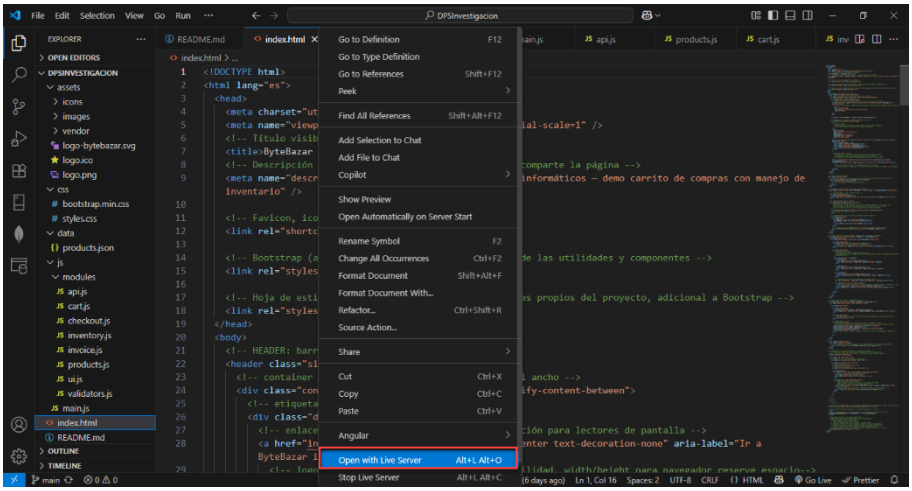
Imagen 12. imagen principal (index) del proyecto en el navegador web de Chrome



Fuente. Nota. Elaboración propia.

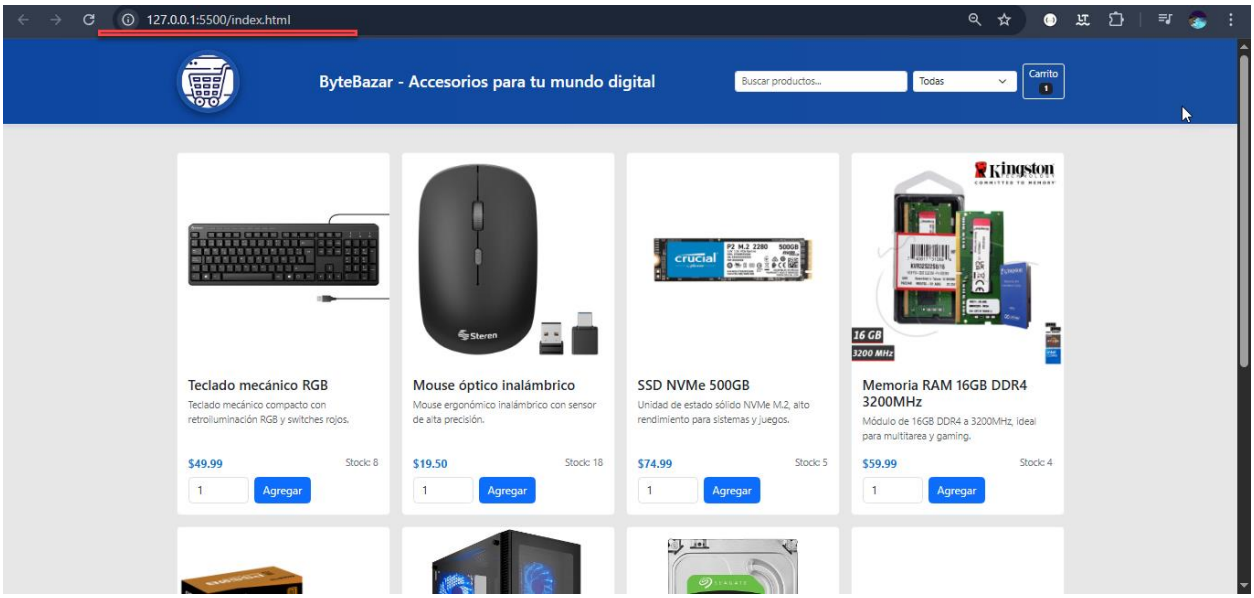
VSC, tiene la ventaja de que, simula un servidor local, para poder verificar los resultados que se obtienen con el código que se tiene en ese momento, esto, a través de la opción “Open with Live Server” ejecutada desde el archivo index html, siendo el archivo principal del proyecto, pudiendo agilizar el desarrollo local, viendo el resultado desde el servidor local de Visual Studio Code.

Imagen 13. opción Open with Live Server en Visual Studio Code para simular un servidor local



Fuente. Nota. Elaboración propia.

Imagen 14. vista en Google Chrome cuando se ejecuta el Live Server en VSC



Fuente. Nota. Elaboración propia.

4 Bibliografía

Haverbeke, M. (2018). Eloquent JavaScript. https://eloquentjs-es.thedojo.mx/Eloquent_JavaScript.pdf

Mdn Contributors (1998-2022), La Guía de JavaScript. Mozilla. https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/About

Perez, J (2008). Introducción a JavaScript. https://www.jesusda.com/docs/ebooks/introduccion_javascript.pdf

Texcel, J. (07 April 2004). Qué es el Modelo de Objetos del Documento. World Wide Web Consortium. https://www.w3.org/2005/03/DOM3Core-es/introduccion.html