Universidad Don Bosco

Facultad de Ingeniería

Diseño de Software Multiplataforma

UNIVERSIDAD DON BOSCO

Trabajo de Investigación

Docente: Ing. Alexander Alberto Siguenza Campos

Estudiante: González Pérez, Ernesto José

Carné: GP200748

Fecha de entrega: domingo 31 de agosto de 2025

Descripción General

TechStore es una aplicación web de comercio electrónico que permite a los usuarios explorar productos tecnológicos, agregarlos a un carrito de compras, gestionar cantidades y finalizar la compra con generación de factura. La aplicación está desarrollada con HTML, CSS y JavaScript, utilizando principios de Programación Orientada a Objetos (POO).

Repositorio en GitHub: https://github.com/bluezef/bluezef.github.io

Estructura del Código

1. Arquitectura de Clases

El proyecto está organizado en cuatro clases principales:

Clase Product

- Representa los productos disponibles en la tienda
- Propiedades: id, name, price, stock, image
- Responsabilidad: Almacenar la información básica de cada producto

Clase CartItem

- Representa los productos añadidos al carrito
- Propiedades: product, quantity
- Métodos: getTotal() - calcula el precio total del item
- Responsabilidad: Gestionar la relación producto-cantidad en el carrito

Clase Cart

- Gestiona todas las operaciones del carrito
- Propiedades: items[], taxRate
- Métodos:
    - addItem() - agrega productos al carrito
    - removeItem() - elimina productos del carrito

    - updateQuantity() - actualiza cantidades
    - getSubtotal(), getTaxes(), getTotal() - cálculos de precios
    - clear() - vacía el carrito
- Responsabilidad: Gestionar el estado completo del carrito y los cálculos

Clase Store

- Clase principal que coordina toda la aplicación
- Propiedades: products[], cart, currentInvoiceNumber
- Métodos:
    - init() - inicializa la aplicación
    - renderProducts() - muestra productos en la UI
    - addToCart(), removeFromCart(), updateItemQuantity() - gestionan el carrito
    - updateCart() - actualiza la visualización del carrito
    - checkout() - finaliza la compra
    - generateInvoice() - genera la factura
    - hideInvoice() - oculta la factura
- Responsabilidad: Coordinar toda la lógica de la aplicación y la interacción con la UI

2. Flujo de la Aplicación

1. **Inicialización**: Se crea una instancia de Store que inicializa los productos y configura los event listeners.
2. **Visualización de productos**: Los productos se renderizan en la interfaz con sus detalles.
3. **Interacción del usuario**:
    - El usuario selecciona cantidades y agrega productos al carrito
    - Puede modificar cantidades o eliminar productos del carrito

4. **Cálculos**: El carrito actualiza automáticamente los subtotales, impuestos y total.
5. **Finalización de compra**: Al hacer checkout, se genera una factura con todos los detalles.
6. **Continuar comprando**: El usuario puede volver a la tienda después de ver la factura.

3. Gestión de Estado

- **Inventario**: El stock se actualiza en tiempo real al agregar/eliminar productos del carrito.
- **Carrito**: Persiste durante la sesión del usuario en la aplicación.
- **Facturación**: Se genera un número de factura incremental por cada compra.

4. Validaciones

- Cantidades deben ser números válidos mayores a 0
- No se puede exceder el stock disponible
- Interfaces evitan entradas incorrectas con mensajes de error

Cómo Ejecutar la Aplicación

Opción 1: Ejecución Local

1. Descarga o clona los archivos del proyecto
2. Guarda el código HTML en un archivo con extensión .html (ej. techstore.html)
3. Abre el archivo en cualquier navegador web moderno haciendo doble clic sobre él

Opción 2: Alojamiento en GitHub Pages

1. Accede a la aplicación mediante la URL https://bluezef.github.io/

Requisitos del Sistema

- Navegador web moderno (Chrome 60+, Firefox 55+, Safari 12+, Edge 79+)
- JavaScript habilitado en el navegador
- Conexión a Internet (solo para fuentes y si se usan CDNs en el futuro)

Personalización

Agregar Nuevos Productos

Modifica el array products en el constructor de la clase Store:

javascript

```
this.products = [
  new Product(1, "Nombre del Producto", 100, 10),
  // ... más productos
];
```

Modificar Impuestos

Cambia la propiedad taxRate en el constructor de la clase Cart:

javascript

```
this.taxRate = 0.16; // Cambiar por el valor deseado
```

Personalizar Estilos

Los estilos están definidos en la sección <style> del documento y pueden modificarse fácilmente cambiando las variables CSS o las reglas específicas.

Notas Técnicas

- Los datos persisten solo durante la sesión actual (se reinician al recargar la página)

- No requiere dependencias externas (JavaScript)
- Diseño responsive que se adapta a dispositivos móviles y desktop
- Código modular que facilita el mantenimiento y expansión

Posibles Mejoras Futuras

1. Persistencia de datos usando localStorage o una base de datos
2. Implementación de un backend para procesamiento de órdenes reales
3. Sistema de autenticación de usuarios
4. Historial de pedidos
5. Integración con pasarelas de pago
6. Sistema de categorías de productos
7. Funcionalidad de búsqueda y filtrado