



DPS941

DISEÑO Y PROGRAMACIÓN DE SOFTWARE MULTIPLATAFORMA

Catedrático: Ing. Alexander Alberto Siguenza Campos

Integrantes:

Bolaños Mancía, Manuel Antonio: BM192191

Figuroa Aguilar, José Manuel: FA200209

Hernández Soriano, Rene Alexander: HS191498

Muños Reyes, Christopher Alberto: MR202832

Montoya Reyes, Cristian Alberto: MR211303

Señora Reyes, Jonathan Rafael: SR232918

Trabajo de investigación [10%]

Proyecto de Carrito de Compras en JavaScript con Facturación

Introducción

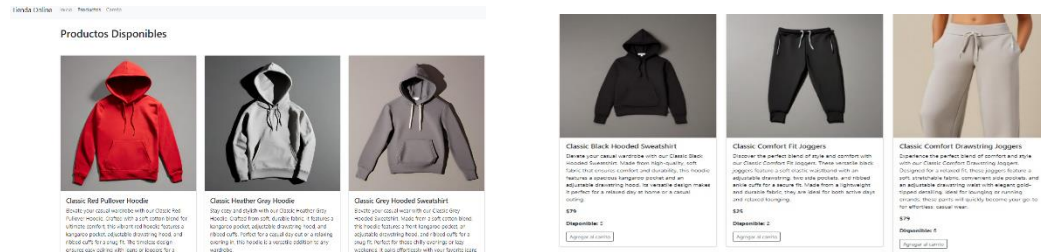
Como equipo de desarrollo de la Universidad Don Bosco se nos encargó la elaboración de un sistema de carrito de compras en JavaScript, el cual posee una muy buena interacción con el usuario final, para poder agregar una herramienta mas a la empresa el cual es las compras en línea y aportar el valor necesario para facilitar el comercio y adaptarse a las tecnologías mas recientes y rentables del mercado.

El tipo de negocio enfocado es una tienda de ropa en línea, la página web mostrara una breve descripción, precio, fotografía de sus producto y opciones de compra al cliente.

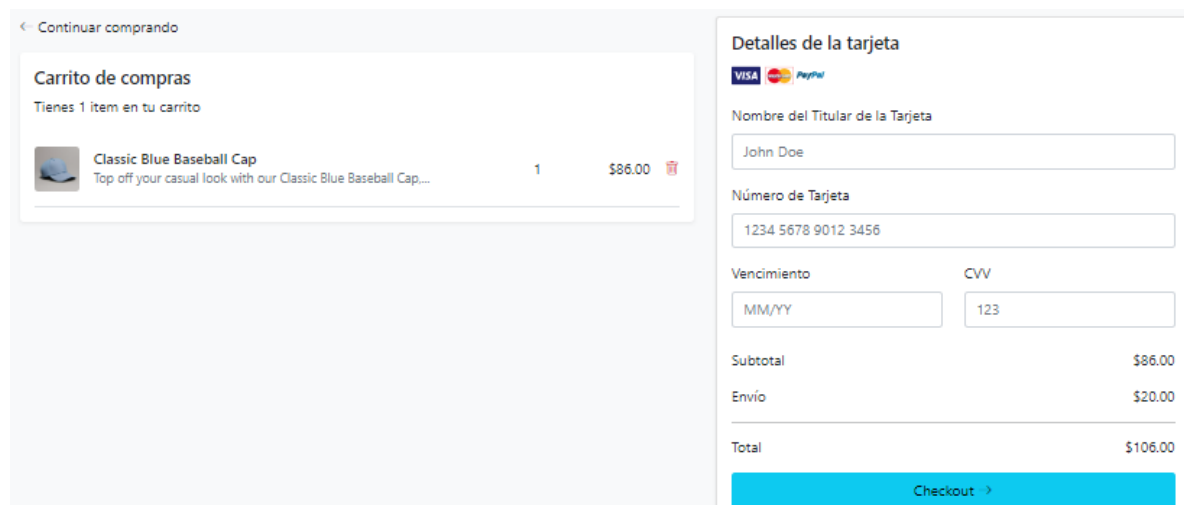
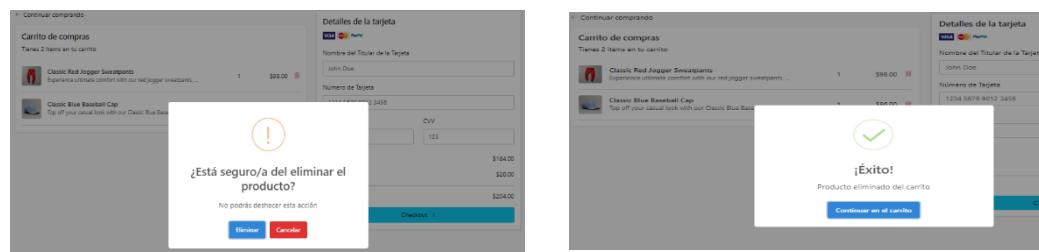
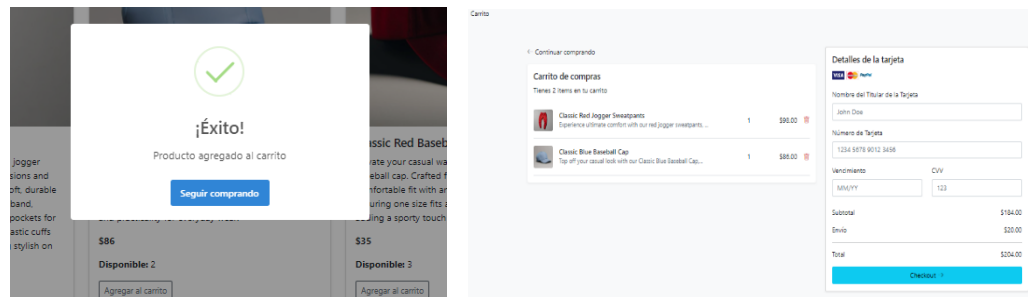
El sistema es totalmente intuitivo y va dirigido a los clientes actuales que por motivo de tiempo efectúan sus compras en línea, brindándoles accesibilidad y facilidad en la adquisición de sus productos favoritos.

Requerimientos Funcionales:

Selección de Productos: • Mostrar una lista de productos con al menos los siguientes atributos: nombre del producto, precio y cantidad disponible. •




Permitir al usuario seleccionar productos e ingresar la cantidad deseada. Agregar y Eliminar Productos: • Agregar los productos seleccionados al carrito de compras. •



Facturación: • Al confirmar la compra, generar y mostrar una factura en pantalla que incluya la lista de productos comprados, la cantidad, el precio unitario, el precio total por producto y el total general de la compra. • Considerar tasas o impuestos si es necesario y mostrarlos en la factura.


Interactividad: • Permitir al usuario seguir comprando después de ver la factura


En la factura esta el acceso para poder seguir comprando en la tienda en línea.



Orden confirmada!

A continuación podrás ver a detalle tu factura:

Fecha de compra	Orden N.	Pago	Tiempo estimado de entrega
29/8/2024	MT22166039		5-7 días hábiles



Classic Blue Baseball Cap

Cantidad: 1

\$86.00

Subtotal	\$86.00
Tarifa de envío	\$20.00
IVA 13%	\$11.18
Total	\$117.18

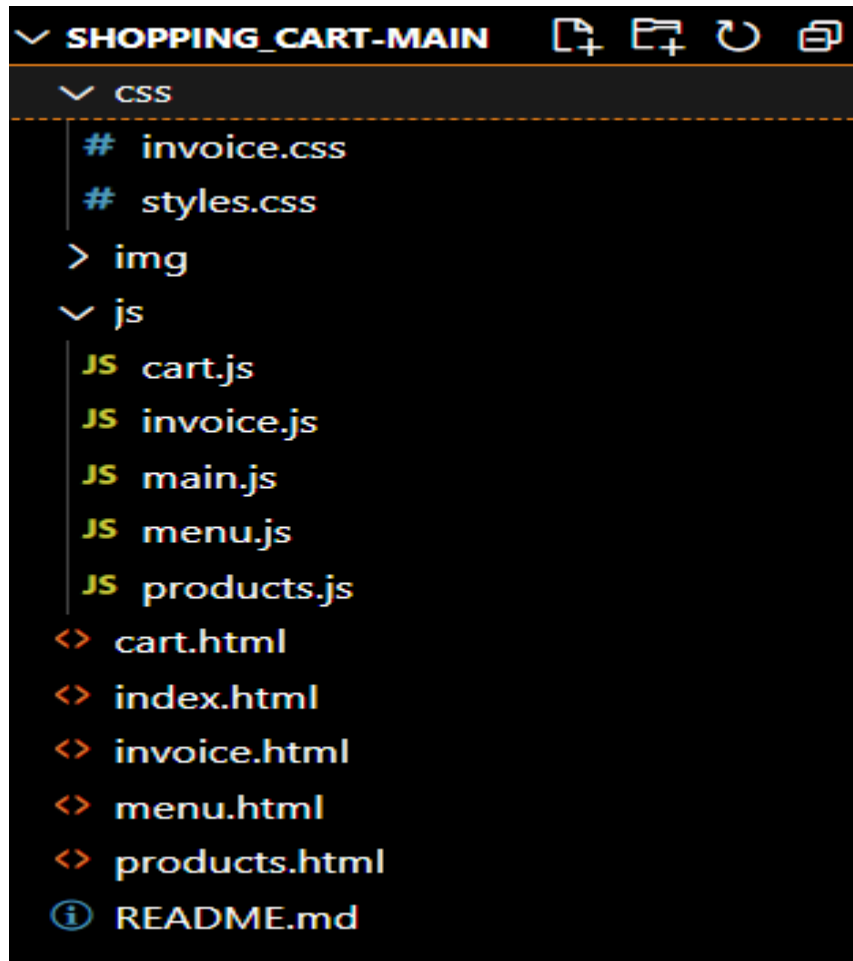
¡Enviaremos el correo electrónico de confirmación de envío cuando el artículo se envíe con éxito!

¡Gracias por comprar con nosotros!

Visítenos nuevamente [click acá](#)

Apartado técnico

Este es el apartado del código el cual esta compuesta por nuestra parte visual , y nuestra área de lógica para las funcionalidades de la página.



En el apartado visual tenemos un menú de inicio (Index), la cual será la presentación inicial al cliente y en la que tendremos las distintas secciones del menú

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Inicio</title>
7   <!-- Call CSS files -->
8   <link rel="stylesheet" href="css/styles.css">
9   <!-- Bootstrap CSS -->
10  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yv1ztcQTwfspd3yD6"
11 </head>
12 <body>
13   <!-- Menu -->
14   <nav class="navbar navbar-expand-lg navbar-light">
15     <div class="container-fluid">
16       <a class="navbar-brand" href="#">Tienda Online</a>
17       <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="T"
18       <span class="navbar-toggler-icon"></span>
19     </button>
20     <div class="collapse navbar-collapse" id="navbarNav">
21       <ul class="navbar-nav">
22         <li class="nav-item">
23           <a class="nav-link active" aria-current="page" href="index.html">Inicio</a>
24         </li>
25         <li class="nav-item">
26           <a class="nav-link" href="products.html">Productos</a>
27         </li>
28         <li class="nav-item">
29           <a class="nav-link" href="cart.html">Carrito</a>
30         </li>
31       </ul>
32     </div>
33   </div>
34 </nav>
35
36 <div class="content">
37   <h1>Bienvenido</h1>
38   <br>Descubre la mejor selección de productos que se adaptan a tu estilo. Compra fácil, rápido y con confianza. </br> ¡Bienvenido a la experiencia de compras en línea
39   <a href="products.html" class="btn btn-custom">Comprar</a>
40 </div>
41
42 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
43 </body>
44 </html>
```

En el uso de hojas de estilo, se configuraron para dos secciones, una que abarca el apartado principal del html index el cual configura los estilos generales del cuerpo y barra de navegación donde tendremos el menú principal y la hoja invoice tendrá las configuraciones visuales de la factura .

```
css > # styles.css > ...
1  /* styles.css */
2
3  body {
4      background-image: url('../img/landingpage.jpg');
5      background-size: cover;
6      background-position: center;
7      height: 100vh;
8      display: flex;
9      flex-direction: column;
10     justify-content: space-between;
11     color: white;
12     text-shadow: 1px 1px 5px rgba(0, 0, 0, 0.7);
13 }
14
15 .navbar {
16     background-color: transparent !important;
17 }
18
19 nav div a{
20     color: white !important;
21 }
22
23 .content {
24     flex: 1;
25     display: flex;
26     flex-direction: column;
27     justify-content: center;
28     align-items: flex-start;
29     padding: 20px;
30     font-size: 20px;
31     margin-left: 20px;
32     color: white;
33 }
34
35 h1 {
36     font-size: 3rem;
37     margin-bottom: 20px;
38 }
39
40 .btn-custom {
41     background-color: #334ee7 !important;
42     border: none;
43     padding: 10px 20px;
44     font-size: 1.2rem;
45     text-transform: uppercase;
46     border-radius: 30px;
47     box-shadow: 2px 2px 10px rgba(0, 0, 0, 0.3);
48 }
49
```

El siguiente código JavaScript se encarga de la funcionalidad del carrito de compras:

1. Inicialización:

- Carga las máscaras para los campos del formulario de pago (tarjeta de crédito).
- Recupera el contenido del carrito de compras almacenado en el navegador (localStorage).
- Si existe un contenedor para el carrito, muestra los productos agregados.
- Agrega funcionalidad para eliminar productos del carrito.
- Agrega validación al formulario de pago.

2. Funcionalidades del Carrito:

- `addProduct(id)`: Agrega un producto al carrito actualizando el stock del producto y la cantidad en el carrito.
- `deleteProduct(id)`: Elimina un producto del carrito y actualiza el stock del producto.
- `clearCart()`: Vacía el carrito por completo.
- `printCart(cart, container)`: Muestra los productos del carrito en un contenedor específico, incluyendo cantidad, subtotal y total (con envío).

3. Validación del Formulario de Pago:

- `validatePaymentForm()`: Valida la información del formulario de pago (nombre del titular, número de tarjeta, fecha de vencimiento y CVV).
- Verifica que el carrito tenga productos.
- Valida el formato del número de tarjeta.
- Valida el formato de la fecha de vencimiento.
- Valida el formato del CVV.
- Muestra mensajes de error en caso de que la validación falle.

4. Funcionalidades Adicionales:

- Utilizamos librerías como SweetAlert para mostrar mensajes de confirmación, alertas y carga.
- Utilizamos funciones como `formatDescription` y `formatTotal` para procesar la descripción y el precio total de los productos.


```

js > JS cartjs > ...
1 document.addEventListener("DOMContentLoaded", () => {
2     // Implementando mascaras para los campos del formulario
3     importMasks();
4     // Elementos del carrito de compras
5     const cart = JSON.parse(localStorage.getItem("cart")) || [];
6     const containerCart = document.getElementById("container-cart");
7
8     if (containerCart) {
9         printCart(cart, containerCart);
10    }
11
12    // Evento click para eliminar productos del carrito
13    const btnDeleteProduct = document.querySelectorAll("#btn-delete-product");
14    if (btnDeleteProduct) {
15        btnDeleteProduct.forEach((btn) => {
16            btn.addEventListener("click", (e) => {
17                const id = e.target.getAttribute("data-id");
18                deleteProduct(id);
19            });
20        });
21    }
22
23    // Agregar validación al formulario de pago
24    const checkoutButton = document.querySelector(".btn-info");
25    if (checkoutButton) {
26        checkoutButton.addEventListener("click", (e) => {
27            e.preventDefault(); // Prevenir que el formulario se envíe automáticamente
28            validatePaymentForm();
29        });
30    }
31 });
32
33 // Validar formulario de pago
34 const validatePaymentForm = () => {
35     const cardholderName = document.getElementById("cardholderName").value.trim();
36     const cardNumber = document.getElementById("cardNumber").value.trim();
37     const expiration = document.getElementById("expiration").value.trim();
38     const cvv = document.getElementById("cvv").value.trim();
39     const cart = JSON.parse(localStorage.getItem("cart")) || [];
40
41     if (cart.length == 0) {
42         showAlert(
43             "Información",
44             "No hay productos en el carrito",
45             "info",
46             "Entendido"
47         );
48         return;
49     }

```

El siguiente código JavaScript se encarga de gestionar la visualización de la lista de productos en la página web, y proporciona la funcionalidad de agregar productos al carrito de compras.

Funcionamiento General:

1. Carga Inicial:

- Una vez que la página se carga completamente (evento DOMContentLoaded), el código busca el contenedor HTML donde se mostrarán los productos.
- Obtiene los datos de los productos, ya sea desde el almacenamiento local del navegador (localStorage) o haciendo una petición a una API externa (en este caso, a la API de Escuelajs).
- Muestra los productos en el contenedor HTML correspondiente.
- Agrega un evento de clic a los botones "Agregar al carrito" para cada producto.

2. Obtención de Datos de Productos:

- La función getProducts se encarga de obtener los datos de los productos.
- Primero verifica si los datos ya están almacenados en el localStorage.
- Si los datos están en el localStorage, los carga directamente.
- Si no hay datos en el localStorage, realiza una petición a la API para obtener los productos.
- Almacena los datos en el localStorage para futuras cargas.

3. Visualización de Productos:

- La función printProducts se encarga de generar el HTML para mostrar los productos en la página.
- Recorre la lista de productos y crea una tarjeta HTML para cada uno.
- Incluye información como la imagen, el título, la descripción, el precio y la disponibilidad (stock).
- Agrega un botón "Agregar al carrito" a cada tarjeta.

4. Agregar al Carrito:

- Cuando se hace clic en el botón "Agregar al carrito", se llama a la función addProduct.
- Esta función se encargaría de actualizar el estado del carrito, almacenando los productos seleccionados en el localStorage.

```

1 > products > printProducts > products.forEach() callback
2 document.addEventListener("DOMContentLoaded", async () => {
3   const productsContainer = document.getElementById("products-container");
4   if (productsContainer) {
5     const products = await getProducts();
6     localStorage.setItem("products", JSON.stringify(products));
7     printProducts(products, productsContainer);
8   }
9
10  const btnAddProduct = document.querySelector("#btn-add-product");
11
12  if (btnAddProduct) {
13    btnAddProduct.forEach((btn) => {
14      btn.addEventListener("click", (e) => {
15        const id = e.target.getAttribute("data-id");
16        addProduct(id);
17      });
18    });
19  }
20 });
21
22 // Fetch de API de productos
23 const getProducts = async () => {
24   try {
25     let products = localStorage.getItem("products") || [];
26     let data = [];
27
28     if (products.length === 0) {
29       const response = await fetch("https://api.escola.js.co/api/v1/products");
30       data = await response.json();
31       products = data.slice(1, 21).map((product) => {
32         return {
33           ...product,
34           stock: Math.floor(Math.random() * 10),
35         };
36       });
37     } else {
38       products = JSON.parse(products);
39     }
40
41     return products;
42   } catch (error) {
43     console.log("Error fetching the products:", error);
44   }
45 };
46
47 // Imprimir productos
48 printProducts = (products, container) => {
49   let productCard = "";
50   container.innerHTML = "";
51   products.forEach((product) => {
52     productCard += `
53     <div class="col-md-4">
54       <div class="card mb-4 shadow-sm">
55         
56       </div>
57       <div class="card-body">
58         <h5 class="card-title">${product.title}</h5>
59         <p class="card-text">${product.description}</p>
60         <p class="card-text"><strong>${product.price}</strong></p>
61         <p class="card-text"><strong>Disponible: </strong>${product.stock}</p>
62         <div class="d-flex justify-content-between align-items-center">
63           <button type="button" ${product.stock === 0 ? "disabled" : ""}
64             class="btn btn-sm btn-outline-secondary" data-id="${product.id}"
65             id="btn-add-product">
66             Agregar al carrito
67           </button>
68         </div>
69       </div>
70     </div>
71   `;
72   container.innerHTML = productCard;
73 };

```

Este código JavaScript se encarga de mostrar la información de una factura basada en el contenido del carrito de compras almacenado en el navegador.

1. Obtención de Datos:

- Recupera el carrito de compras del localStorage.
- Define el costo de envío fijo y la tasa de IVA .
- Obtiene referencias a elementos HTML de la factura (fecha, número de pedido, método de pago, tabla de productos y totales).

2. Generación de Factura:

- Crea una fecha de compra actual y un número de pedido aleatorio.
- Asigna estos valores y un ícono de pago predeterminado a los elementos del HTML.

3. Cálculo de Montos:

- Recorre el carrito y por cada producto:
 - Calcula el precio total multiplicando la cantidad por el precio unitario.
 - Agrega este total a un subtotal acumulado.
- Calcula el IVA como un porcentaje del subtotal.
- Calcula el monto total sumando subtotal, envío e IVA.

4. Mostrar Factura en Pantalla:

- Recorre el carrito nuevamente y por cada producto:
 - Crea una fila en la tabla de productos.
 - Agrega imagen, título, cantidad y precio total a la fila de la tabla.
- Actualiza los elementos HTML para mostrar subtotal, envío, IVA y total de la factura.

5. Botón de Reiniciar Carrito:

- El elemento con el ID "reset-cart" , agrega un evento de clic.
- Al hacer clic en el botón, elimina el contenido del carrito de localStorage.

```

1 document.addEventListener("DOMContentLoaded", () => {
2     // Obtener la información del carrito desde el localStorage
3     const cart = JSON.parse(localStorage.getItem("cart")) || [];
4     const shippingCost = 20.00; // Costo de envío fijo
5     const taxRate = 0.13; // 13% de IVA
6
7     // Obtener los elementos del DOM
8     const invoiceDate = document.getElementById("invoice-date");
9     const orderNumber = document.getElementById("order-number");
10    const paymentMethod = document.getElementById("payment-method");
11    const productTable = document.getElementById("product-table");
12    const subtotalElement = document.getElementById("subtotal");
13    const shippingElement = document.getElementById("shipping");
14    const taxElement = document.getElementById("tax");
15    const totalElement = document.getElementById("total");
16
17    // Generar una fecha de compra actual y un número de orden aleatorio
18    const date = new Date().toLocaleDateString();
19    const orderNum = `M${Math.floor(Math.random() * 100000000)}`;
20
21    // Asignar valores a los campos
22    invoiceDate.textContent = date;
23    orderNumber.textContent = orderNum;
24    paymentMethod.innerHTML = ``; // Puedes ajustar el método de pago según sea necesario
25
26    // Calcular subtotal, IVA y total
27    let subtotal = 0;
28    let tax = 0;
29
30    // Agregar productos a la tabla
31    cart.forEach(product => {
32        const totalPrice = product.price * product.quantity;
33        subtotal += totalPrice;
34
35        // Crear fila de producto
36        const row = document.createElement("tr");
37        row.innerHTML = `
38            <td width="20%"></td>
39            <td width="60%">
40                <span class="font-weight-bold">${product.title}</span>
41                <div class="product-qty">
42                    <span class="d-block">Cantidad: ${product.quantity}</span>
43                </div>
44            </td>
45            <td width="20%">
46                <div class="text-right">
47                    <span class="font-weight-bold">${totalPrice.toFixed(2)}</span>
48                </div>
49            </td>
50        `;
51        productTable.appendChild(row);
52    });
53
54    tax = subtotal * taxRate;
55    const total = subtotal + shippingCost + tax;
56
57    // Asignar valores a los elementos de totales
58    subtotalElement.textContent = `${subtotal.toFixed(2)}`;
59    shippingElement.textContent = `${shippingCost.toFixed(2)}`;
60    taxElement.textContent = `${tax.toFixed(2)}`;
61    totalElement.textContent = `${total.toFixed(2)}`;
62
63    const resetCartLink = document.getElementById("reset-cart");
64
65    if (resetCartLink) {
66        resetCartLink.addEventListener("click", () => {
67            // Borra el carrito del localStorage
68            localStorage.removeItem("cart");
69        });
70    }
71 }
72

```

Anexos

